Mathematical Problems of Computer Science 46, 66-72, 2016.

Development and Implementation of Some Advanced Web Server Protection Methods

Arthur S. Petrosyan and Gurgen S. Petrosyan

Institute for Informatics and Automation Problems of NAS RA
e-mail: arthur@sci.am, gurgen@sci.am

Abstract

Article describes the development work done in the Academic Scientific Research Computer Network of Armenia (ASNET-AM) managed by the Institute for Informatics and Automation Problems (IIAP) of the National Academy of Sciences of the Republic of Armenia (NAS RA) regarding the implementation of web server protection and hardening. Some advanced methods and solutions implemented recently within ASNET-AM Web Hosting Services are being described. Based on the experience of ASNET-AM Web Hosting Services the best practice recommendations about secure web hosting environment implementation are given.

Keywords: WWW, Web Hosting Environment, Web Server, Apache, Fail2ban

1. Introduction

Increased capabilities in the age of modern-day information technology progress, makes the web server protection more and more urgent. New types of exploits, bots and brute force tools are being introduced. This article is a logical continuation of the previous research and deployment of the improved methods for web server protection [1] done in Academic Scientific Research Computer Network of Armenia (ASNET-AM) managed by the Institute for Informatics and Automation Problems (IIAP) of the National Academy of Sciences of the Republic of Armenia (NAS RA). Work described in [1] was mainly focused on the methods of shared web server internal hardening and shared environment isolation. This paper is focused on several advanced methods to protect web server from external attacks. In this paper Apache is considered as a web server example, because of being the most used web server on the Internet [2].

2. Protection of Writable Directories

Web sites may require some files to be uploaded from web browsers. It means that a write permission to the user running the web server should be provided to be able to put the uploaded file into the destination directory. The common solution is to give all the users 777 (read/write/execute) access to the destination directory. Such common solution becomes an easy way to upload a malicious program to the web server for later harmful execution.

In case of Apache 2 ITK MPM [3] package, which is used within ASNET-AM Web Hosting Services to deploy the improved web server protection methods, such write permissions are being limited only to a group level, without need of giving the 777 write access to all users. Additionally it is recommended to put such writable directories out of virtual host area, i.e., on the upper level, than web site's document root. This way no direct URL-based access will be possible. Example of such configuration follows:

Document root directory of 'somesite.am' website - */home/somesite.am/WWW*
Writable directory for uploaded files - */home/somesite.am/uploadfiles*
(not */home/somesite.am/WWW/uploadfiles*)

Although the above configuration additionally hardens the web site, it is sometimes difficult to accomplish because of the specific web site structure.

Any writable directory should always be configured to prohibit execution of any scripts. The best way to do that is making appropriate static <Directory> configuration for that web site in the web server's main configuration [4]. If not possible (web site owner may not be granted permission to operate web server's configuration) it should be done via the *.htaccess* file in that directory [5]. Example of such configuration follows:

```
<IfModule mod_php4.c>
 php_flag engine 0
</IfModule>
<IfModule mod_php5.c>
 php_flag engine 0
</IfModule>
<IfModule mod_php7.c>
 php_flag engine 0
</IfModule>
```

Next protection method to be implemented is the limitation of file types and max file size to be allowed for uploading. This can be done by means of specific checks. Following is the code example on PHP language to allow uploading only PDF files:

```
if ($_FILES[file][tmp_name] <> "" && $_FILES[ file][tmp_name] <> "none" &&
$_FILES[file][size] <= $FILE_MAX_UPL_SIZE &&
is_uploaded_file($_FILES[file][tmp_name]) &&
$_FILES[file][type] == "application/pdf" ||
$_FILES[file][type] == "application/acrobat" ||
$_FILES[file][type] == "application/x-pdf" ||
$_FILES[file][type] == "applications/vnd.pdf" ||
$_FILES[file ][type] == "text/pdf" ||
$_FILES[file][type] == "text/x-pdf") { }
```

The above solutions are not complete panacea, because many web sites are based on the opensource solutions (like Joomla, Wordpress, Drupal, etc.) and if not updated regularly, may contain known vulnerability/exploit. Thus, an additional mechanism for writable directory contents checking is recommended to be used. It could be implemented by means of a specific shell script, regularly parsing writable directory contents in search of pre-defined prohibited file types (such as .php files). In case such file found the script moves it to quarantine location for further analysis and logs the incident.

If such writable directory is protected with *.htaccess* file (since it is always located in the same directory) it could be replaced with some other one to allow the execution vulnerable scripts. Thus, the script also checks for the presence of the *.htaccess* file, and if it does not find it, it recovers it from the backup copy and logs the incident.

It also checks the contents of *.htaccess* file by comparing it with the backup version and if there is a difference detected recovers it from the backup copy and logs the incident.

Following is the shell script code example:

```
function moveandlog {
mv $1 $PGS_WRONG_FILES_FOLDER_PATH
echo "$(date) PHPMOVE::: $1" >> $PGS_LOG_FILE_PATH
}

export -f moveandlog
/var/WrongFiles

find $i -iregex '.*\(php\)' -exec bash -c 'moveandlog "{}"' \;
find $i -iregex '.*\(php3\)' -exec bash -c 'moveandlog "{}"' \;
find $i -iregex '.*\(phtml\)' -exec bash -c 'moveandlog "{}"' \;
find $i -iregex '.*\(phps\)' -exec bash -c 'moveandlog "{}"' \;

find $i -name ".htaccess" | while read fname; do

if ! cmp -s $PGS_HTACCESS_PATH $fname
then
        cp $PGS_HTACCESS_PATH $fname
        chown $PGS_CHOWN_USER:$PGS_CHOWN_GROUP $fname
        echo "$(date) OVERWRHT:: $fname" >> $PGS_LOG_FILE_PATH
fi
done
```

The script permanently runs as a daemon and makes checks every 3 seconds (time can be changed).

Thus, even if some web site vulnerability would enable an attacker to upload some malicious file to the writable directory, it will be detected and immediately moved out, and the incident will be logged.

3. Proper Fail2ban Configuration

Fail2ban [6] utility is available in most Linux-based web server solutions. It can be used to detect and block certain IP addresses from which the attempts of unauthorized access to the server is done. These IP addresses are determined by the results of the monitoring of log files - log-files (for example, /var/log/auth.log, /var/log/apache/error.log etc.). If any IP address in a certain period of time, makes too many unsuccessful log in attempts or any other suspicious activity, the host with the IP address is blocked (by adding iptables firewall rules) for a certain time interval specified by fail2ban configuration.
The above-described default functionality of fail2ban allows to protect website not only from the so-called "brute force" attacks, but also from automatic scanning of web site by means of some bot scripts.
Below are given some best practice recommendations to protect web site using fail2ban, in order to increase the security of the web server.

By scanning means of *apache-auth* filter unsuccessful attempts to gain access to the Web server directories, that are protected by username and password (.htaccess / .htpasswd) can be identified. Using *apache-nohome* filter attempts to scan the list of site scripting files can be detected.

Additional strict filter that scans and detects *all* types of files and folders was written by ASNET-AM team:

```
failregex = ^%(_apache_error_client)s File does not exist
```

While it adds web site protection, it should be used very carefully, because the web site developers or content editors may include broken links to different files, pictures, etc. And there is a chance to block normal users who just browse the site and occasionally click on the broken link, thus unintentionally giving the impression of improper access. It is probably better not to use this fail2ban filter for a web site that is not completely ready and tested.

Using *apache-badbots* filter enables to identify and prevent different bots crawl our web site to find email addresses to spam databases.

Several other filters (*apache-botsearch*, *apache-fakegooglebot*, *apache-overflows*) help to identify fake googlebot-s, long suspicious requests, etc.

php-url-fopen filter can detect attempts to run php injection.
apache-shellshock filter can detect attempts to exploit shellshock vulnerability.

In spite of the above there are more intelligent bots that can identify the period of their blocking and produce scan or brute force in such a way, so as not to be blocked (slow brute force). To prevent such attacks a specific fajl2ban setting of different block time use can be configured. This is done by having a random time to be added for each blocking incident to some predetermined period of time. Example of such settings follows:

```
bantime  = 86400
bantime.increment = true
bantime.rndtime = 79m
bantime.factor = 1
```

Fail2ban also has useful filter *recidive* that allows to scan own fail2ban's log to identify the frequency of a particular IP address blocking (recidive), so as to re-block it for a longer period of time.

It is advisable to set up fail2ban to all the services that will be active on the web server, such as ssh (after changing to a non-standard port), mail server, etc.

4. Some Additional Techniques to Increase Web Server Security

Following additional methods of protecting web servers and web sites are also recommended for implementation as much as possible.

Many complex database-driven web sites have separate *frontend* (the site itself) and *backend* (content management system (CMS)). In that case best practice would be to use different database users for frontend and backend. The frontend database user should have minimum rights (the best possible option is only to give read rights), while backend database user should have full rights to modify the database on which the web site is based.

Following are examples of MySQL permissions for frontend and backend database users

Frontend user rights:
GRANT SELECT ON somesitedb. TO somesitedbfrontenduser@localhost IDENTIFIED BY 'somefrontendpassword';*

Backend user rights:
GRANT ALL PRIVILEGES ON somesitedb. TO somesitedbbackenduser@localhost IDENTIFIED BY 'somebackendpassword' WITH GRANT OPTION;*

Additionally it is best to separate the backend interface, by configuring it as a separate virtualhost and even putting it on a non-standard application port (different from 80). This will add a security layers to the backend.

For some web sites based on the ready opensource solutions (like Joomla, Wordpress, Drupal, etc.) it requires some tricks to separate backend and frontend parts. When the code is written the way that ties backend and frontend parts together, it could be easier, just to create two similar copies of website, but use different database users as shown above, so as to eliminate possibility of SQL-injections or other database-related exploits to be run via public frontend. This is especially important for the ready opensource solutions, where some vulnerability may be found and not fixed yet, because of not being updated regularly. Although present such vulnerability will have no effect, because database user for frontend doesn't have enough privileges to modify the database.

In case the web site backend is being used only from specific workstations, additional protection could be made by not registering the separate name of that backend virtual host in DNS, but only statically adding that name in the 'hosts' file of such workstations. Alternatively such backend protection could also be implemented through the special reverse proxy configuration.

Finally backend is good to have .htaccess / .htpasswd user/password protection in addition to any other protection methods. And of course backend connection should be made through the HTTPS protocol only.

5. Conclusion

The use of the described methods to protect writable directories can prevent running the malicious scripts to harm the web site and the web server as a whole, as well as will help in timely detection and fixing of any vulnerabilities in the web site code, that could allow an attacker to upload undesirable files to the web server. Using fal2ban utility can greatly increase the security of the web server and web sites, as it can identify and block attacks on services and sites such as brute force, vulnerability scans, etc. If also some additional non-standard protection methods described above are used, maximum protection could be achieved or at least we would make it more difficult to attack our web server.

References

[1] A. Petrosyan and G. Petrosyan, “Research and deployment of improved web server protection methods”, *Transactions of IIAP of NAS RA Mathematical Problems of Computer Science*, vol. 42, pp. 81-84, 2014.
[2] Wikipedia, the free encyclopedia, Web server, Market share, [Online]. Available: https://en.wikipedia.org/wiki/Web_server#Market_share
[3] Apache 2 ITK MPM, [Online]. Available: http://mpm-itk.sesse.net/
[4] Apache HTTP Server Version 2.4, Apache Core Features, <Directory> Directive, [Online]. Available: http://httpd.apache.org/docs/current/mod/core.html#directory
[5] Apache HTTP Server Version 2.4, Apache HTTP Server Tutorial: .htaccess files, [Online]. Available: http://httpd.apache.org/docs/current/howto/htaccess.html
[6] Fail2ban. [Online]. Available: http://www.fail2ban.org/

Submitted 19.07.2016, accepted 03.11.2016.

Վեբ սերվերների պաշտպանության որոշ առաջադեմ մեթոդների մշակումը և կիրառումը

Գ. Պետրոսյան և Ա. Պետրոսյան

Ամփոփում

Հոդվածը նկարագրում է վեբ-սերվերների պաշտպանության բարձրացմանն ուղված հետազոտական աշխատանք, որը կատարվել է Հայաստանի ակադեմիական գիտահետազոտական կոմպյուտերային ցանցում (ASNET-AM), որը ղեկավարում է Հայաստանի Հանրապետության գիտությունների ազգային ակադեմիայի (ՀՀ ԳԱԱ) Ինֆորմատիկայի և ավտոմատացման պրոբլեմների ինստիտուտը (ԻԱՊԻ):

Նկարագրված են որոշ առաջադեմ մեթոդներ և լուծումներ, որոնք մշակվել և իրականացվել են վերջին ժամանակներում ASNET-AM ցանցի վեբ-հոսթինգի ծառայության համար: Բերված են նաև ASNET-AM ցանցի վեբ-հոսթինգի ծառայության փորձի հիման վրա անվտանգ միջավայրի ստեղծման գործնական առաջարկություններ:

Разработка и реализация некоторых передовых методов защиты веб-серверов

Г. Петросян и А. Петросян

Аннотация

Статья описывает исследовательскую работу по усилению защиты веб-серверов, проведенную в академической научной исследовательской компьютерной сети Армении (ASNET-AM), действующей под управлением Института проблем информатики и автоматизации (ИПИА) Национальной академии наук Республики Армения (НАН РА). Описаны некоторые передовые методы и решения, разработанные и реализованные в последнее время для службы веб-хостинга сети ASNET-AM. Также приведены практические рекомендации о реализации безопасной среды, на основе опыта работы службы веб-хостинга сети ASNET-AM.