SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
Volume 8 No 5
ISSN 2054-7390

WEB CIBERSECURITY

Danger of PHP Code Injection Vulnerability

Pedro Ramos Brandao
Coordinator Professor, Instituto Superior de Tecnologias Avançadas.
Integrated Researcher, Universidade de Évora
ORCIC: https://orcid.org/0000-0001-6351-6272
pedro.brandao@istec.pt

ABSTRACT

This work demonstrates that the use of laboratories for the development of curricular work in the area of information technology exclusively supported by cloud computing technology does not decrease the level of learning and assessment objects on the part of students. This scenario arose due to the need to interrupt face-to-face classes in physical laboratories due to the COVID-19 Pandemic. The study had as universe a master's degree in Computer Science.
Keywords: Cloud Computing; Azure; Distance learning; WEB App, Virtual Labs.

1 Introduction

Currently, cybersecurity is an extremely important topic and covers all professional areas. However, there are areas that are much more at risk than others. All companies or organizations that are permanently connected to the Internet through their activities are obviously more exposed to risks. Within this group there is also a subgroup more exposed to risks, it is the organizations that, in addition to being permanently or systematically connected to the Internet, have public web servers. This type of infrastructures is subject to permanent attacks within the scope of Cybersecurity.

This work focused more specifically on the analysis of one of the most common and most dangerous vulnerabilities of web servers, which by default use the HTTP protocol, programs or scripts that run on these platforms. We do not focus on relevant issues such as the technological protection of web servers, but more on how programs that run on these servers may have errors that facilitate the existence of vulnerabilities.

The aim of this paper is to demonstrate that inadequate programming for the Web can result in serious vulnerabilities. We have analysed in detail the issue of code injection in PHP files that are executed on web servers and that, due to their characteristics, become vulnerabilities by themselves.

2 PHP Scripts and possible vulnerabilities (the problem)

The PHP language is currently used for the development of web-related applications, and has a wide range of possibilities for using scripts, the so-called PHP Scripts, which are usually executed at the server level.

DOI: 10.14738/tmlai.85.9051
Publication Date: 01st Nov, 2020
URL: http://dx.doi.org/10.14738/tmlai.85.9051

This language has its own characteristics, it is a language of interpretation, that is, it is executed without the need for compilation (1).

As we said, this language is totally devoted to being used on the Internet and with its own and unique configurations, however, PHP Scripts can perform actions that almost all other Web applications execute normally, for example they execute any command coming from another GET type application and POST or REQUEST, execute cookies at all levels, and can create active online variables sent via line commands.

One of the most common and most dangerous vulnerabilities that can be found in PHP Scripts is the Source Code Injection. A malicious user who in any way can access a machine in a DMZ or a network can run Scripts on one of the servers on that network, as long as they operate in PHP or through PHP. If the PHP parameters are not strongly configured, the malicious user can, through the Request command, inject the execution of something into any PHP file.

3 The PHP Possible Source Code Injection

The conde injection is the most common problem of this kind. When the administrators do not configure correctly the system, it's very possible open a door to an attack. Then the attacker creates a Specifique request to interfere in the server code execution, and the PHP interpreter include that code.

Example: htpps://serer1/node1/file1.php

If we analyse the source file, we find in the script

<? If (!empty ($page)) include ($page) >

We can say this variable is danger because is a global one and is hallways automatically registered.

So, if this value of variable is received is certainly the external PHP file be included and totally executed in the PHP script code inside the server.

Many mistakes are like this one:

<? Include (“file1.txt”) ‘>

Many developer's cam believed that file cannot be adulterer, but the *file1* parameter for shore contain the path txt and this one can contained PHP instructions and executed like a script.

The attack type depends waylays is the vulnerability type is local or global kind. The global type is extremally dangerous because that allows any kind of attack in the server, the intruder can execute anything in the server side.

To be possible an external attack of this kind three conditions is necessary exist.

First: inside the *include ()* function, must exist a string in the begin of the script.

Correct example: include (“$data12.txt”)

Dangerous example: include (/$data12.txt)

Second: the connections HTTP out are not with restrictions.

Third: if in the default configurations of the PHP config there are any kind of remote access enable.

4 The research methods

The main techniques and tools used for gathering research data include quantitative technique:

We put a PHP site in one Web Server, the configurations are divers, then we make exploiting tests to the site in the Web Server, then we note de reaction to our actions.

5 The analysis – Exploiting with code injection

Created a web application with pHP on a WEB server, it was assumed that a scenario prone to cybercriminal attacks was created.

We tried to find vulnerabilities, note that the team that created the site is not the same team that tested the site.

The type of attack possible or feasible is directly related to the type of villability found.

As we already mentioned, Global PHP source code injection is one of the most dangerous attacks and is allowed by a common vulnerability to find. This type of vulnerability allows files to be run remotely on a server.

We tried the following code and found a vulnerability:

http: //localhost/66/23.php

you get:

<?

include ("./ files / $. php");

?>

It was the detection and use and execution of a soruce code injection.

Programming languages with interpretation and using strings that contain all types of characters including null characters, this can be a problem.

We tried to run an include script ("./$name/file.php"), we acted like a normal programmer and it was believed that the interpreter included and ran only FILE.PHP. But what actually happened was the script found the null terminator as a string name: name = temp.txt% 00?

The PHP interpreter opened and executed the ./temp.txt\0/file.pnp file in which \ o is the null character, this string has been changed to include (), and this is a very serious problem in terms of security. The result was to include in the ./temp.txt execution, the code injection was done.

Finally, we demonstrate how the PHP interpreter can easily execute all types of scripts:

Consider http: //localhost/45/22.php

File: <? echo file ("a-b-c x: y: w");

$ e = "test";

Copyright © Society for Science and Education United Kingdom

?>

 this is a text test in which no variable like $ a will be replaced by any value.

<?

echo "this test text will appear.

the variables will be replaced by values, example: a = $ a ";

?>

This is facilitated by the fact that many configuration files do not have to have <? ?>.

6 Conclusion

We assume that if there was no care on the part of programmers in the configuration of PHP files on a server, this could enable and facilitate attacks of a certain type.

The most dangerous type of attack was specified.

A PHP site was simulated on a WEB Server with a common pattern of configurations, without taking into account the specific care of security.

An injection attack was attempted.

The attack was successful, and it proved why.

So, we can conclude that if the settings of a PHP site have not been improved and very well implemented, it is easy to attack by injection a PHP site.

REFERENCES

[1] NIXON, R.. (2018). Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (Learning PHP, MYSQL, Javascript, CSS & HTML5. O'Reilly Media

[2] TATROE, K. (2020). Programming PHP: Creating Dynamic Web Page. O'Reilly Media.

[3] WELLING, L. (2016). PHP and MySQL Web Development. Addison-Wesley Professional.

[4] LOCKHART, J. (2015). Modern PHP: New Features and Good Practices. O'Reilly Media.

[5] AJZELE, A. (2017). Mastering PHP 7: Design, configure, build, and test professional web applications. Packt Publishing.

[6] POWERS, D. (2019). PHP 7 Solutions: Dynamic Web Design Made Easy. Apress.

[7] SNYDER, C. (2010). Pro PHP Security: From Application Security Principles to the Implementation of XSS Defenses. Apress.

[8] EDMUNDS, B. (2016). Securing PHP Apps. Apress.