Web Application Security Assessment Report

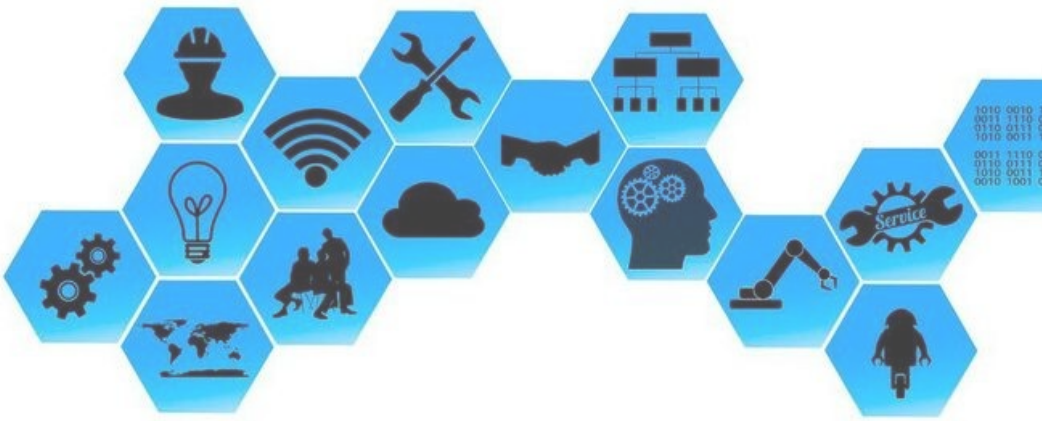
Costumer: *ITHS*

Subject: *Metasploitable 2 (10.2.10.154)*

Timeline: *2024-08-30 – 2024-09-06*

Version: *1.0*

Created: *2024-09-06*



Metasploitable 2 – Web Application
Date: 2024-09-06

Table of Contents

1 Results and Recommendations

1.1 Severity ratings

Severity	Description
High	Security vulnerabilities that can give an attacker total or partial control over a system or allow access to or manipulation of sensitive data
Medium	Security vulnerabilities that can give an attacker access to sensitive data, but require special circumstances or social methods to fully succeed.
Low	Security vulnerabilities that can have a negative impact on some aspects of the security or credibility of the system or increase the severity of other vulnerabilities, but which do not by themselves directly compromise the integrity of the system
Info	Informational findings are observations that were made during the assessment that could have an impact on some aspects of security but in themselves do not classify as security vulnerabilities.

Table 1: Severity ratings.

1.2 Outline of identified vulnerabilities

Vulnerability	High	Medium	Low	Info.
PHP-CGI Argument Injection	✓			

Table 2: Identified vulnerabilities.

1.3 Technical description of findings

1.3.1 PHP-CGI Argument Injection (CVE-2012-1823)

CVSS: 7.5
Severity: High

Background

Argument Injection refers to an attack where an attacker will try to manipulate arguments passed to an application or script. This occurs when an application or service improperly handles user input, allowing attackers to inject additional arguments into a command or script execution to change it's behavior or configuration. The exploitation of this vulnerability could be leading to stolen data, deployed malware, remote code execution or remote access to a system and network.

Description

During the security assessment multiple web applications were found to be vulnerable to argument injections by not sanitizing the query strings sent in the requests. This is due to PHP running in a CGI setup where the web server passes the PHP script requested to a PHP handler for execution and returning the results of the script. The handler used, accepts arguments sent from the web server but is inaptly configured to interpret specially crafted query strings as command-line flags or options for the PHP interpreter. An attacker could thereby apply arguments found in PHP's Server API to view source code, override configuration and execute malicious code remotely.

This argument injection vulnerability do not require any authentication or privileges to be exploited. A remote attacker with access to the web application could exploit this vulnerability by using a web browser or other tools for manipulating the query string or the request sent to the web server.

Shown in Image 1, by not including an equal sign ("=") and adding a Server API flag to the query string, the query is interpreted as an argument when executing the PHP script, returning the source code of the PHP script that was executed for that route/endpoint.

Image 1: Adding "?-s" as query string returns source code.

When using more powerful tools to further manipulate and send requests to the server an attacker will be able to inject system commands to steal information, manipulate the functionality of the web application and system, execute malicious code and establish a reverse shell.

In another instance, shown by Image 2, multiple POST request were sent to the web server without an equal sign (“=”) and containing multiple “define” flags (“-d key=value”). This temporarily overrides the configuration for PHP CLI to accept and execute scripts sent in the request body.

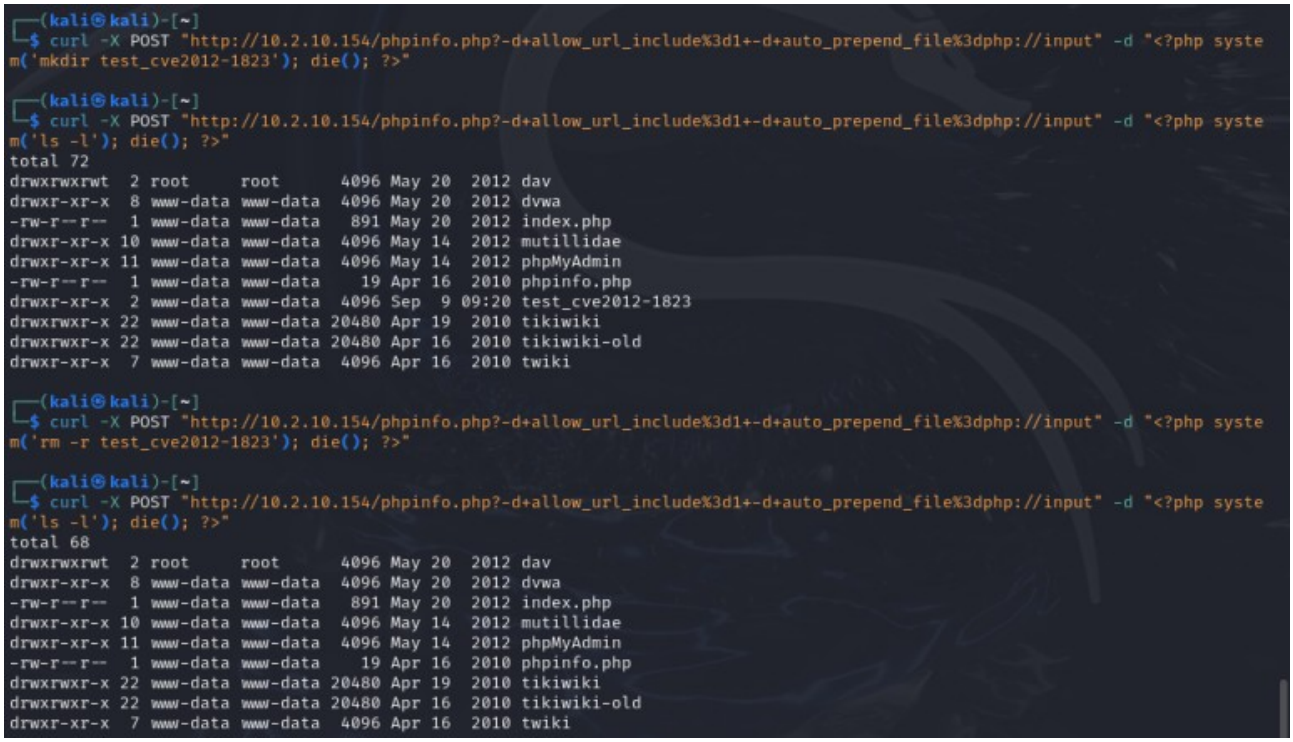


Image 2: Overriding configuration to create and delete a directory (test_cve2012-1823) as the user running the service.

By researching version numbers found in /phpinfo.php, a nmap version scan and exploiting CVE-2012-1823 connected to those versions, multiple web applications were found in a vulnerable state. The results are listed in Table 3 below.

Routes and Endpoints tested	Vulnerable to argument injections
/	Yes
/phpMyAdmin	Yes
/mutillidae	Yes
/dvwa	Yes
/phpinfo.php	Yes
/twiki	No

Table 3: List of tested routes and endpoints.

Recommendations

Upgrading to later version of PHP is strongly recommended, at least version 5.3.12 or 5.4.2 where this vulnerability is mitigated.
If CGI mode is unnecessary for the functionality of the web applications, it could be disabled.
Apply mod_rewrite rules in Apache to not accepting query strings starting with a hyphen (“-”) and not containing an equal sign (“=”). This change shouldn't break any applications.

And as a general rule, user controlled data should never be trusted and should thereby be properly sanitized and validated. For more information:
EsecurityPlanet - Prevent web attacks using input sanitization[1]

1 https://www.esecurityplanet.com/endpoint/prevent-web-attacks-using-input-sanitization/