CSSS 3510 Writing Secure Code

Spring 2 2020

Author: Jose Pablo Murillo

Hands-on lab report

1.) Lab settings:
    a. PC Hardware settings and Software Configuration
        i. See phpinfo() output in Annex 1
        ii. In order for Apache to allow PHP to create files, a command was used to allow the Apache user (usually www-data) to create files in the directory where the lab is. This was accomplished using the command: **sudo chown www-data labDirectory**. Alternatively to allow changes in all folders and files of the directory, use the command **sudo chown -R www-data labDirectory**

    b. Virtual machine settings
        i. Operating system: Ubuntu 64-bit
        ii. Base memory: 4096 MB
        iii. Chipset: PIIX3
        iv. Number processors: 2
        v. Network: Bridged

2.) Lab objectives:
    a. This lab is a demonstration of command injection, where the attacker can execute arbitrary commands, in a vulnerable PHP form and then present recommended corrections to the vulnerability.

3.) Lab description:
    1. A simple form is designed with one textbox input. The textbox input is used to specify the name of a file to be created. The form sends the input variable to the PHP scripts that calls the system() function to create a file with the file name and extension.

    2. The vulnerable version of the PHP script takes the user input as it is sent and then attempts to create the file using the system() function.

    3. **Attack under MITRE ATT&CK Framework**

        a. **Discovery**
            i. Account discovery
                1. An attacker can write within the input commands that output a list of domain accounts. This is turn exposes the user accounts in the system and the attacker can attempt to steal the information or impersonate a user

(especially one with admin privileges). In this demo I put the value of "**; cat /etc/passwd**" to list each user or account on the system running the web application.

ii. File and directory discover
    1. An attacker can write within the input commands that allow him to navigate through the filesystem. They can list the files and directories in specific locations of the machine. This can allow the attacker to access personal files or other restricted files. I put the value of "**; cd /home; ls**" to see the users in the system. I then used "**; cd/home/username/Documents; ls"** to list documents from the user.

iii. System information discovery
    1. An attacker can write with the input commands that attempt to get detailed information about the operating system and hardware. Using this information they can then elaborate a more sophisticated attack that target a vulnerability of the current configuration settings. I used the value of "**; uname -v;"** to see the current kernel version of the host machine.

4. The secure version PHP script avoids arbitrary code execution using the escapeshellcmd() function. This function escapes characters in a string that might trick a shell command into executing arbitrary commands. As a result, a file with the input specified will be created and no other command will be executed.

5. Code can be found in Annex 2

4.) Conclusion: Server-side scripts should now allow the possibility for arbitrary command execution as may alter the behavior of the web application and an attacker can discover more information that required.