2011

Unrestricted File Download V1.0 – Windows Server

Let’s bypass blacklist protection methods once again

Author: Soroush Dalili
Website: www.SecProject.com
Email: irsdl at yahoo dot com

Jan 2011

Unrestricted File Download V1.0 – Windows Server

I do not want to talk about Insecure Direct Object References without any protection as they are obviously exploitable; Instead, I want to talk about bypassing the protected ones! The problem that I want to explain here is how hard it is to protect a system that uses Insecure Direct Object References by using black-list technique.

Whenever penetration testers see a website which accepts a path as an input, they think about these questions:

1- Can I have access to the secret files?

2- Can I do directory traversal?

3- Can I modify another file?

4- Can I do race condition?

And so on.

The answer from programming point of view is: “it depends!”:

1- If they have no protection in-place: “Yes. Yay!”

2- If they are using black-list method: “Think about a bypass now! There should be a way and I just need to find it! Think about encodings, decoding, effective characters, behaviour of the system against special characters, and so on.”

3- If they are using white-list method: “Is there anything on the list that can be misused? Can I stick some of them together to make another character or change the behaviour of the system?”

My point is that there is often a way to bypass a black-list. However, it is not the same for white-list if you do it correctly.

Let’s Bypass a Blacklist Method

Now, I want to use a case to show an example of using black-list, and methods of bypass.

Assume we have “www.vulnerable.com/download.aspx” which accepts a file path as an input and reads it and loads it into the output. (To make it easier, “/upload” folder is on the root of the website)

For example: “/download.aspx?file=/upload/document.doc”

Now, if you try the following inputs, you will receive an “access denied” error from the page:

“/download.aspx?file=web.config”

“/download.aspx?file=download.aspx”

“/download.aspx?file=/download.aspx”

But, if you try the following inputs, you will receive a “file not found” error or a blank-page from the page:

“/download.aspx?file=test.doc”

“/download.aspx?file=/upload/../test.txt”

“/download.aspx?file=/test.f0ob4r”

According to the response of the page, obviously, it is using a black-list method.

These are the first things that I can think about (my pre-test-cases):

0- Use uppercase, lowercase, and Unicode in the extension. For ex: “/download.aspx?file=/wEB.CoNfiG” and so on.

1- As you might know, there are some characters after the filename that will be ignored by Windows. So, I should try something like “/download.aspx?file=/web.config.” or “/download.aspx?file=/web.config... ..”

2- Using short filename format of the file: “/download.aspx?file=/web~1.con”

3- Using null character: “/download.aspx?file=/web.config%00.txt”

4- Using another extension in the path: “/download.aspx?file=/test.txt/../web.config”

5- Using different space characters in the path: “/download.aspx?file=/web.config%09”, “/download.aspx?file=/web.config%0a”, “/download.aspx?file=/web.config%0b”, “/download.aspx?file=/web.config%0c”, “/download.aspx?file=/web.config%0d”, “/download.aspx?file=/web.config%20”, and so on (similar to 1).

6- Finding a character that is removed by the web application automatically before loading a file to put it in the extension and bypass the black-list protection.

7- Try alternate data stream to read the files: “/download.aspx?file=/web.config::$Data”

8- Try to use direct path and share path. Ex: “/download.aspx?file=c:\\windows\\win.ini”, “/download.aspx?file=\\?\c:\\windows\\win.ini”, or “/download.aspx?file=\\127.0.0.1\c$\WINDOWS\\win.ini”

9- Try to do directory traversal. Ex: “/download.aspx?file=../../../../../../../../../../boot.ini”

10- Try other file-system understandable vectors. Ex: “/download.aspx?file=web.config/.”, “/download.aspx?file=web.config\.”, and so on (similar to 1).

And combination of the above solutions to create more complicated test cases!

What do you think? Please let me know if you know any other interesting test case. This is the result:

0	Successful: Web.config was downloaded
1,2	Failed: Show the source code in error message. Error: “Failed to map the path”
3,7,8	Failed: Show the source code in error message. Error: “is not a valid virtual path”
4	Failed: Access Denied
5	Successful: Web.config was downloaded
6	Failed: No character was found
9	Failed: Show the source code in error message. Error: “Cannot use a leading .. to exit above the top directory”
10	Successful: Web.config was downloaded. Some new vectors were found: “?file=\.”, “?file=/.”, “?file=/\./\.”

Each of the above vectors could lead to bypassing the protection. Now, I can tell you that the actual vulnerable source code of the page was:

```
10   string fileName = Request.Params["File"];

20   if (ForbidenExtentions.Contains(fileName.Substring(fileName.LastIndexOf("."))))
30   {
40      HttpContext.Current.Response.Redirect("~/CustomError.aspx?msg=ForbidenFileDownload");
50   }

60   if((fileName != null) && (fileName != ""))
70   {
80      string strPath = Server.MapPath("/" + fileName);
90      if(System.IO.File.Exists(strPath))
100    { …
```

And, we can download the confidential files with different vectors (see number 0, 5, and 10 on the table above). Now, an attacker can download the entire website and look for the credentials, hidden files and folders, and find any other vulnerability such as SQL Injection by having the source code.

Secure and Effective Solution

Now, what can we do to stop this attack? These are the general solutions:

1- Do not use direct object references when it is possible:

For indirect references, use something random, hard to guess, and meaningless such as GUIDs. You need to implement more functions and invest more time on programming and debugging. However, your achievements are:

1.1- Increasing the Security by using strong random pointers such as GUIDs

1.2- Easier asset managing and have different access controls

2- Force yourself to always use white-lists:

It is very rare that you have to only use a black-list for an input! If an input is random and unpredictable, you may need to redesign that input. Write down the input purpose(s) and do whatever you can to restrict it to a range of characters. Now, think about this range and review the characters one by one. Is there anything in the list which can cause an issue? Do you need to allow any other characters besides [a-zA-Z0-9]? Why? Think about it and follow the best security practices.

Sometimes you need to use blacklist after passing the input from a white-list to have more security. For example: an input can contain a file path. Therefore, we should allow dot "." character. However, we should not allow any double dot ".." as it can cause directory traversal.

If you are designing a system, look for the vulnerabilities which have been reported on the similar systems in Internet. You may find something that you had not had any knowledge about it before! Do not think that you know everything! Even a semi-colon or colon can compromise your system sometimes.

Talk about your system with the security people; with experts (not script kiddies). You can ask your questions in different security forums to find a clue. Ask them to break your protection to improve the security.

Note 1: a bad implementation is worse than not having any implementation! When you don't have any protection, at least you know you do not have anything to protect yourself and the system is unsafe!!! However, when you have an insecure/bad implementation, you think the system is safe enough but it is not, and attackers will find this out – trust me!

Note 2: If you are putting different inputs next to each other, it is better to pass them at least through a black-list protection after concatenation.

Now, without using an indirect reference, two solutions for our vulnerable example ("www.vulnerable.com/download.aspx") can be:

Solution 1 (More White-list – more restricted):

1- Replace all the "/" with "\" character in order to make the validation easier (for Windows OS). (Black-List)

2- Replace all the dot characters before the backslash character (".\") with a single "\" character in order to make the validation easier. (Black-List)

3- Only accept limited characters as an input: RegEx: (([a-zA-Z0-9][\.]{1})|[a-zA-Z0-9\\])*

4- File name should start with: RegEx: ^[a-zA-Z0-9\\] (White-list)

5- File name should end with: RegEx: [a-zA-Z0-9]$ (White-list)

Then a general ReGex will be (include 3, 4, and 5): ^([a-zA-Z0-9\\]{1})(([a-zA-Z0-9][\.]{1})|[a-zA-Z0-9\\])*([a-zA-Z0-9])$ (White-list)

6- Find the file extension by using the last dot "." character of the file. This extension should be in the list of allowed extensions such as "gif", "jpg", "doc", "docx", "pdf", "rtf", and so on. (White-List)

Limitation: It is not possible to use Unicode or special characters in the file or the directory name.

Solution 2 (More Black-List – less restricted):

1- Trim the input to remove unnecessary spaces (Black-List)

2- Replace all the "/" with "\" character in order to make the validation easier (for Windows OS). (Black-List)

3- Replace all the ".." with "." character in a loop till you cannot find any ".." anymore. (Black-List)

4- Replace all the space and dot characters before and after the "\" character with a single "\" character in order to make the validation easier. (Black-List)

5- Replace all the "\\" with "\" character in order to make the validation easier. (Black-List)

6- Path should not contain these characters: RegEx: [^:*?"<>|;~] - (for Windows OS)

7- Find the file extension by using the last dot "." character of the file. This extension should be in the list of allowed extensions such as "gif", "jpg", "doc", "docx", "pdf", "rtf", and so on. (White-List)

Quick Conclusion:

Stop using blacklist protections for direct object references if you cannot use indirect ones. Moreover, do not forget to talk to the specialists to implement it correctly.

Final Words

Please send me your feedbacks via my email address (irsdl at yahoo dot com) to improve this white-paper. You can use whole or part of this document by putting a reference to the author (Soroush Dalili) and link of the main document.

Currently just by using Google, a lot of vulnerable websites and Content Management Systems (CMS) can be found. If you find an issue based on the content/idea of this paper in a permitted system (such as your website CMS), please report it to its legal authority to patch the system as soon as possible; and I would be thankful if you put a link to this document as a reference in your advisory.

However, please do not use this knowledge against any website or system without having a legal permission. And, I do not accept any responsibility for any usage from this white-paper and its content/idea.

Reference(s):

- OWASP, Unrestricted File Upload:
http://www.owasp.org/index.php/Unrestricted_File_Upload