Web Security and Malware Analysis

Assignment 4 – 12/12/2019

Goal: Solve each task by providing a brief explanation of the solution that you adopted. You must use BURP or one of the indicated tools to solve the tasks (when required). Explanations should mainly include screenshots and some brief comments.

Note: Although you may find the solutions to many of the proposed tasks on the web, *try to solve them on your own and do not immediately give up*. This training will be very useful for the practical question that will be asked during the exam. Also, remember to give **brief written** answers (avoid copy-and-paste from other tutorials and try to write your own answers) and to use screenshots to describe what you do. You are also recommended not to use copy-and-paste texts from tutorials, but to use your own words.

Deadline: Jan 7th,2020

What to do: Send the report in the PDF format to davide.maiorca@unica.it with the subject "[WEBSEC] - Report For Assignment 4 – NAME SURNAME" and name the file "websec_report_4_name_surname.pdf". Remember to include your name, surname, and matriculation number in your report!

Starting Notes: most of the tasks of this assignment will be done by attacking the Damn Vulnerable Web Application (DVWA). To access the machine, you have to follow these steps:

1. Connect to https://app.cyberranges.com/
2. Create an account
3. Click on "Play," log in with your account
    a. Click on "Library," then "Category," then "Web."

   b. Select DVWA-Reflected XSS #1, then load the Virtual Machine
   c. Click on VPN and download the related VPN file (it should have a .ovpn extension)
4. You have to open this file with OpenVPN. If you do not have it, download it from https://openvpn.net/community-downloads/ and install it.
5. In Windows, open the command prompt *with administrator privileges* (VERY IMPORTANT, otherwise the VPN will not work).
6. If you installed in the default path, give this command: **"C:\Program Files\OpenVPN\bin\openvpn.exe" "your_ovpn_path"**
7. If the VPN is working, you should get the message "Initialization Sequence Completed."
8. With the VPN working and the virtual machine started in app.cyberranges, connect to http://192.168.125.150/dvwa/, and put as username: "admin" and as a password: "password."

Note that the virtual machine will be reset after 1-2 hours, so you have to reload it again from the Cyberranges web app.

You will be asked to solve tasks by changing their difficulty progressively. To change it, go to DVWA and select the difficulty by clicking on DVWA Security on the menu (you will be asked to solve tasks from low to high).

Task 1 – Reflected XSS

You have to solve the Reflected XSS tasks of the Damn Vulnerable Web Application at the low, medium, and high difficulty. The final goal is stealing your own session cookie and intercepting it using a Webhook.

As a webhook, you can use this service: https://requestcatcher.com

To bypass filters, you can check these payloads:

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XSS%20Injection#xss-in-htmlapplications

For the high level, the payload will be a bit different. You will need the concept of the document.location property (google it).

In addition, please describe the following:

1. How the server-side codes evolve from the low difficulty to the high difficulty.
2. The reason why you need a substantially different payload to solve the high level.

Task 2 – Stored XSS

You have to solve the Stored XSS tasks of the Damn Vulnerable Web Application at the low, medium, and high difficulty. The final goal is redirecting the user to the page https://www.unica.it. You are required to discuss how the codes evolve from low to high difficulty and how security is progressively enforced.

Hint 1: Solve this task after Reflected XSS, use all you learned.

Hint 2: BURP will be very useful here. Can you imagine why?

Hint 3: Once you complete every difficulty level, reset the VM.

Task 3 – (Evil) JavaScript Obfuscation

Consider the following obfuscated JavaScript code:

```
var a = [
    'length',
    'match',
    'fromCharCode',
    'uqiweqv',
    'You\x20win!',
    'Try\x20again\x20:-('
];
var b = function (c, d) {
    c = c - 0x0;
    var e = a[c];
    return e;
};
function authenticate_bau(c) {
    var d = function (e, f) {
        if (f < 0x0)
            return d(e, f + 0x1a);
        var g = '';
        for (var h = 0x0; h < e[b('0x0')]; h++) {
            var i = e[h];
            if (i[b('0x1')](/[a-z]/i)) {
                var j = e['charCodeAt'](h);
                if (j >= 0x41 && j <= 0x5a)
                    i = String[b('0x2')]((j - 0x41 + f) % 0x1a + 0x41);
                else if (j >= 0x61 && j <= 0x7a)
                    i = String[b('0x2')]((j - 0x61 + f) % 0x1a + 0x61);
```

```
            }
            g += i;
        }
        return g;
    };
    if (d(c, 0x8) === b('0x3')) {
        alert(b('0x4'));
    } else
        alert(b('0x5'));
}
```

Answer the following questions:

1. What does the code do? What is its functionality?
2. What is the input of the *authenticate_bau* function that allows generating the alert “You win”?
3. Analyze each part of the code and describe how it has been obfuscated.
4. BONUS Question: is there a way to answer question number 2 without analyzing the code?

Hint: The ASCII table will be very useful here.