

# SIL765 – Project 3

## Securing a Website

Nipun Shrivastava

2011cs50288

## 1 INTRODUCTION

---

In this assignment we were asked to discover vulnerabilities in a website, which ran on port 80 at 10.208.21.112. Since it was a website internal to IIT Delhi network, we were allowed to use all kinds of penetration/security testing tools and scanners that we can lay our hands on.

## 2 VULNERABILITIES DISCOVERED

---

To take advantage of this template's design, use the Styles gallery on the Home tab. You can format your headings by using heading styles, or highlight important text using other styles, like Emphasis and Intense Quote. These styles come in formatted to look great and work together to help communicate your ideas.

### ● COMMAND INJECTION

---

- The site was vulnerable to the technique involving an OS command injection. This technique allows an attacker to execute system commands by abusing an application feature. The injection typically occurs when the developer is using user input to construct an executable command specific to the pseudo system shell in use.  
Process.php has a user defined parameter which passes through without sanitization. This user input then is executed on server's shell. This provides an attacker a lot of commands that he/she can execute on victim's side like **ls**, **cd**, **pwd**, **cat** etc. Though commands like **cp** or **rm -r** didn't work suggesting that the way we sneaked in didn't provide us administrator level privilege on the victim's side.
- We were able to see all folders and files on the server side using this. This also enabled to see the file **points.htm** which was residing in the **pointsHere** folder. Upon opening it, it said you have gotten 100 marks if you are student suggesting that the attack was a successful one.
- We were also able to see **bankInfo.htm** which had some information on the amount of money in the account holder's name. This file was in **webhacking/script-attacks/confidential** folder. It

also contained the message – “If you can see this, the web site is pretty much entirely screwed up”.

- During the attack we were also able to view comments in process.php which read “if you are student and can see this, you gained 10 extra points”
- It was a very serious vulnerability as I was able to see the tree structure of directories, sub-directories and files and read them with a cat command. We were also able to see internal directories like **home**, **etc**, **bin**, etc. which had absolutely nothing to do with anyone only there to access the site.

## Finish

### Points for the project

**Note:** If you are a student and managed get until here, you made **100** points and you are done. Good job!

*What did you learn? -- Web hackers constantly are on the look for insecure CGI and PHP Scripts together with other files and directories that are set with risky permissions. Using them, they manage to get access to critical files and learn crucial information about an organization. Therefore, it is important to design web applications carefully by assigning proper permissions to the file system and by practicing secure scripting techniques.*

Supply the secret "hacking is fun!!" to your instructor by email to claim your 100 points for the project!

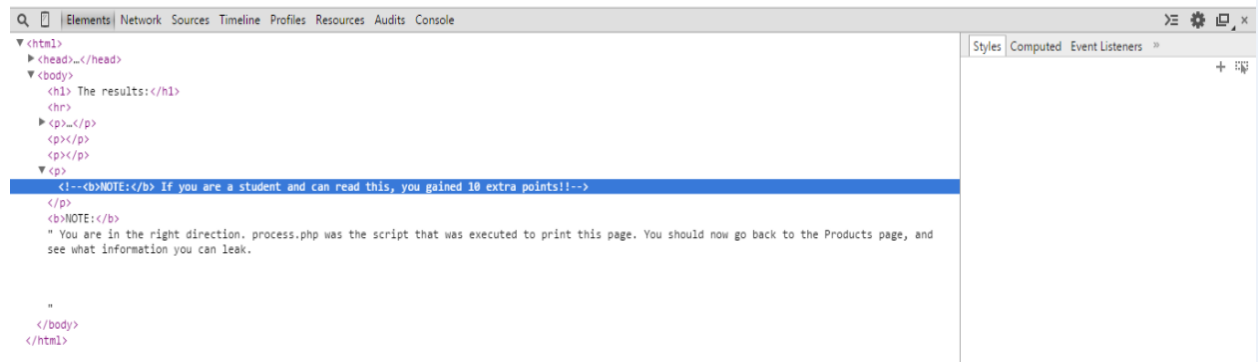
## If you can see this, the web site is pretty much entirely screwed up!!

I have \$1,00,000 in my bank account. I am so happy because NO ONE can know this. Yay!!

### The results:

The files in the tool are: Blue hills.jpg \_vti\_cnf descr.htm

NOTE: You are in the right direction. process.php was the script that was executed to print this page. You should now go back to the Products page, and see what information you can leak.



```
<html>
  <head>...</head>
  <body>
    <h1> The results:</h1>
    <hr>
    <p>...</p>
    <p></p>
    <p></p>
    <p></p>
    <p><!--<b>NOTE:</b> If you are a student and can read this, you gained 10 extra points!-->
    </p>
    <b>NOTE:</b>
    " You are in the right direction. process.php was the script that was executed to print this page. You should now go back to the Products page, and see what information you can leak.

    "
  </body>
</html>
```

## • DIRECTORY BROWSING

- Directory browsing was also enabled on the server's side. As the name suggests, using this vulnerability we were able to reveal directory listings (even **hidden files**, scripts, recovery files, etc.).
- By exploiting such a vulnerability, we can gain information about the web application by browsing directory listings that reveal files and folder hierarchy in the application. This information can be used to exploit vulnerabilities in the web application. In worst-case Scenario

attacker may be able to access useful configuration information or view the contents of files that should be restricted.

- **MULTI VIEWS**

---

- When the multi view is enabled then the server when asked for a file, say “**filename**” then it will then return best match from all the files named foo.\* This allows the attacker to build the whole map of files using brute force.

- **CLICKJACKING**

---

- Since **X-Frame-Options Headers was Not Set** in the HTTP response provided by the server, it can be used to execute “ClickJacking” attacks which can be used to trick a user to click on a button or link on another page when they were intending to click on the top level page.

- **X-CONTENT-TYPE-OPTIONS**

---

- Since Sniffing header in X-content-Type was not set to ‘nosniff’, attackers can perform MIME-sniff in older browser versions. This will cause the response body to allow the display of non-declared-content

- **APACHE 2.2.22 IS OUTDATED**

---

- The server was using an outdated version of APACHE which was riddled with a lot of vulnerabilities. The hacker can use any of them to gain critical information from the server’s side which was never intended to be shared.

### 3 INFORMATION LEAKED

---

The first two vulnerabilities provided significant information leak from the server’s side. While both the vulnerabilities enabled us to re-create the file structure of server on the attacker’s side, the first vulnerability even allowed us to access those files. Even the hidden files or the operating system files of the server were also exposed to the attacker.

### 4 FIX FOR THE FOUND VULNERABILITIES

---

- **Command Injection**  
Check for the contents coming from user side in the post token and raise a red flag if you see illegal shell commands.
- **Directory Browsing**  
If the directory listing is exploitable in a custom application then review the code and prevent

malformed strings or tricked URI's from bypassing the filters or input validation you are applying to directory GET requests.

- **Multi Views**

Disable the multiView on the server side.

Change httpd.conf file. A recommended configuration for the requested directory should be in the following format: `<Directory/{YOUR DIRECTORY}>Options FollowSymLinks </Directory>`

Multiviews option should be removed even from the configuration file.

- **Apache 2.2.22**

One can apply the necessary update patches to the APACHE in order to fix the vulnerabilities related to it. The newer version has fixed all of the known vulnerabilities.

- **Other Vulnerabilities**

They can be simply avoided by specifying the corresponding headers with appropriate values to avoid any funny business.