Whitepaper

Source Code Disclosure over HTTP

Exploiting the Download Page

Anant Kochar, SecurEyes, December 2007

TABLE OF CONTENTS

Abstract

Full source code disclosure is any website owner's worst nightmare and any hacker's dream. This paper explores a widely prevalent coding flaw in web applications which hackers can exploit to extract source code and configuration files over HTTP.

Introduction

Many websites offer files for download to their users through specialized dynamic pages. If this download page is insecurely coded, an attacker can exploit it to download the source code files and even the configuration files.

This insecurity is widely prevalent on websites on the internet as well as on the intranets. Not much has been written on this technique but we feel that those vulnerable to this attack cannot afford to ignore it: *there are few things worse for a website owner than a full source code disclosure of web pages or a compromise of the configuration file.*

A Likely Scenario

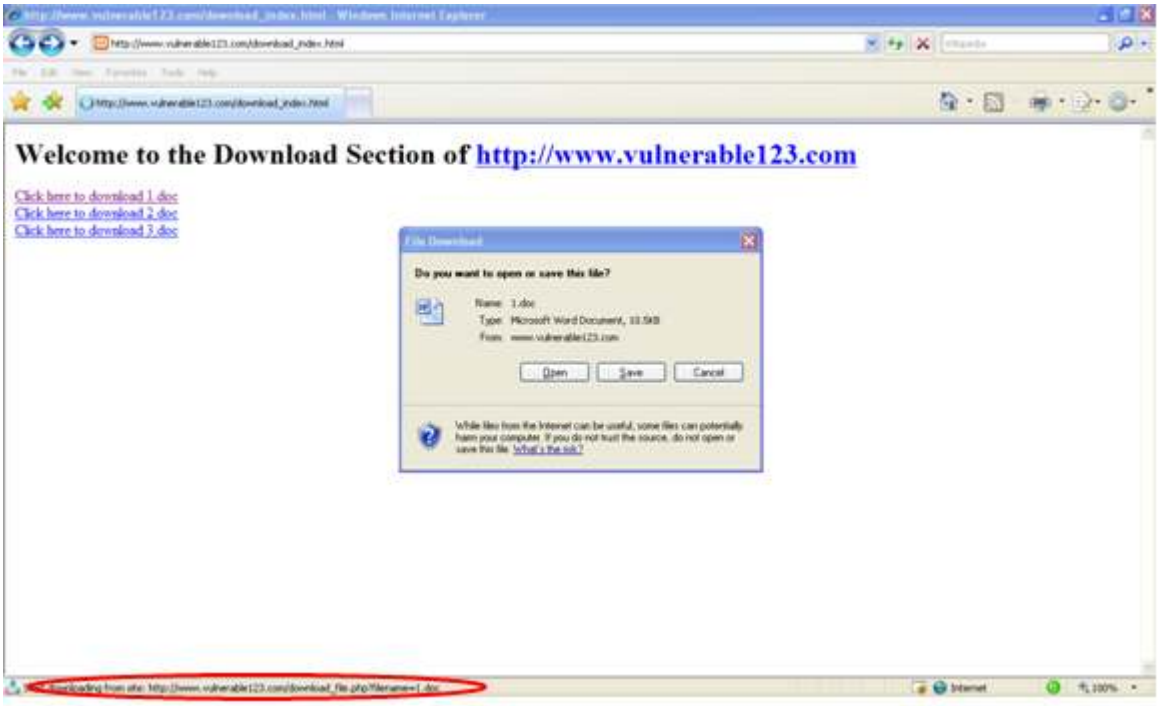
An administrator of a website made on PHP (www.vulnerable123.com) wants to offer non-HTML files for download to the website users. Since he wants to keep a track of things like which file is the most popular download, he does not offer users direct URLs of these files. Instead, these files are offered for download through a dynamic page on which some logic can be written to keep a track of file downloads. In this application, this dynamic page is named 'download_file.php' and its URL is http://www.vulnerable123.com/download_file.php

In our vulnerable site, the downloadable documents are stored in the website root directory. A user can download a file when a URL parameter 'filename', containing the path of the file to be downloaded, is passed to the 'download_file.php' page.
For instance, a user who needs to download the '1.doc' goes to the 'Download Index' page at http://www.vulnerable123.com/download_index.html

The user then clicks on the appropriate link (Click here to download 1.doc) which submits the following URL to the server:

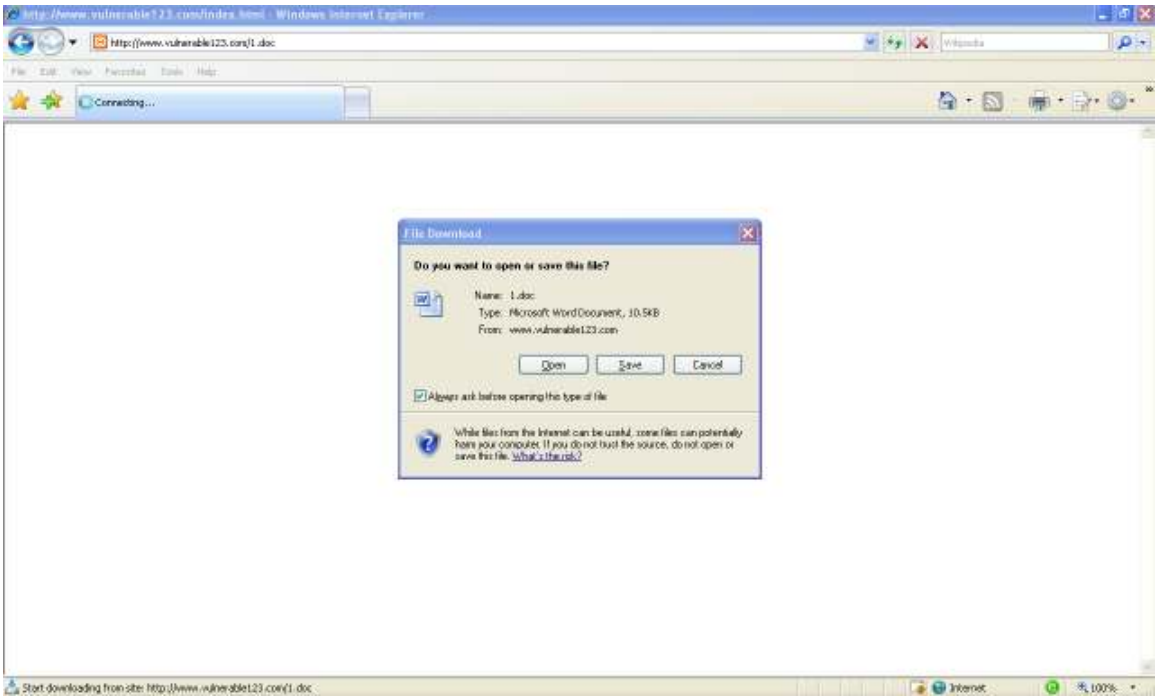
http://www.vulnerable123.com/download_file.php?filename=1.doc

This causes the download page to begin downloading of the desired document:

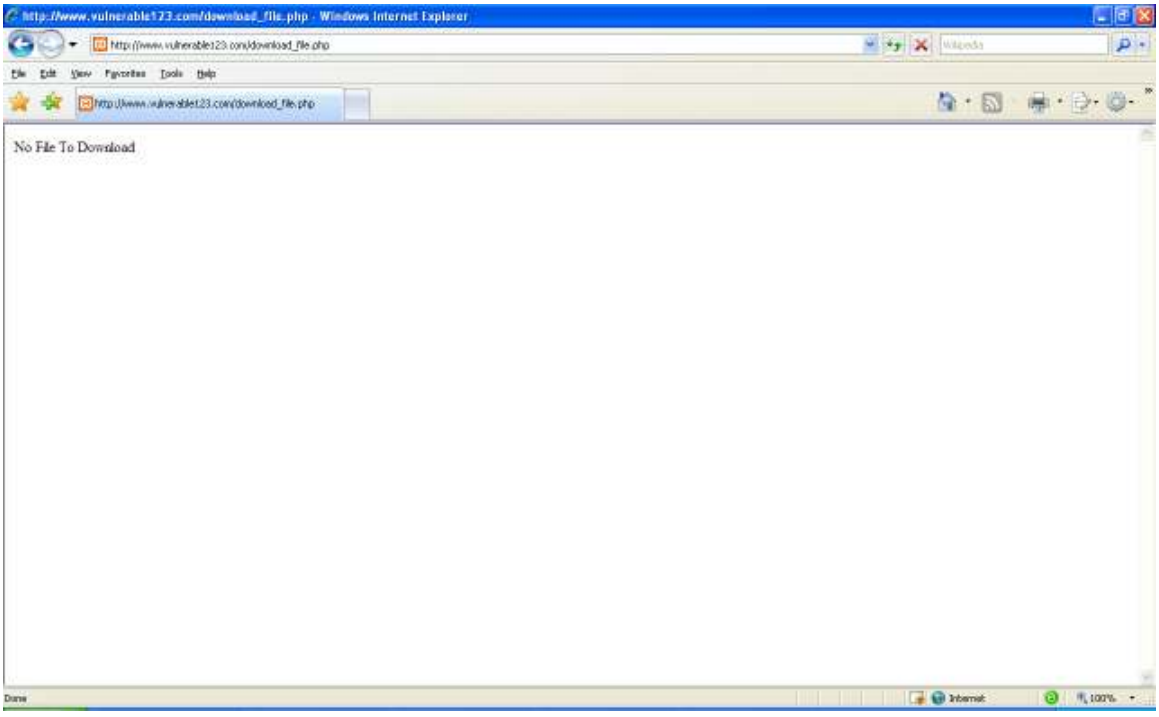


Simply supplying the URL or direct path of the file may or may not download it, based on the specific application implementation. In certain cases, the user will be

able to download the file just by submitting the following URL: http://www.vulnerable123.com/1.doc



Can the source code file of the 'download_file.php' page also be accessed by direct reference? That is, will the submission of the following URL download the 'download_file.php' PHP code file: http://www.vulnerable123.com/download_file.php

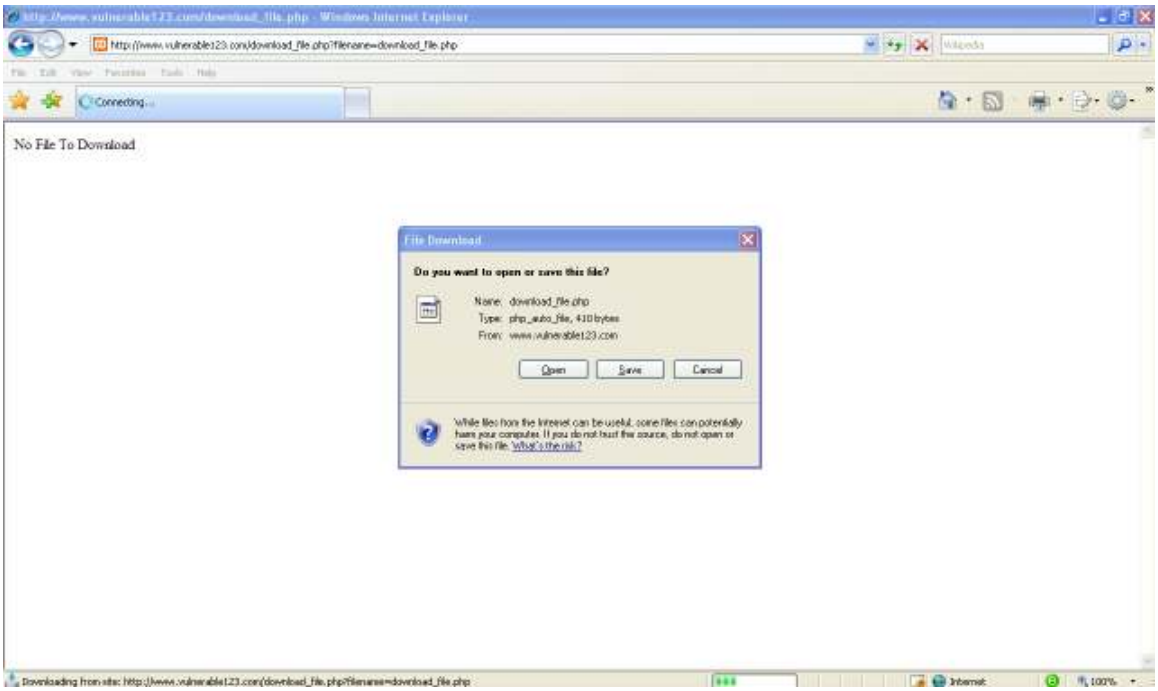


No. The PHP engine will not render this file because it is not permitted to do so due to the built-in security features. It will only give the HTML portion of the PHP pages and not the source behind them. (The engine will also render HTML pages and other common non-HTML file types like .zip and .doc.)

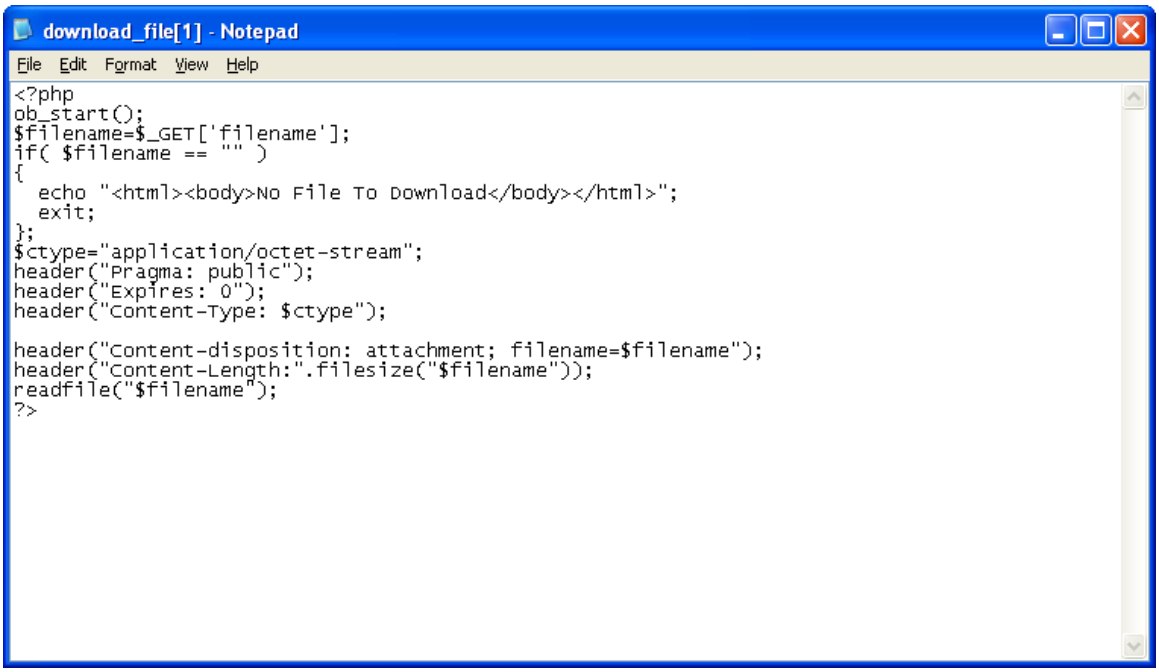
The Exploit

Does the PHP engine also keep a tab on what type of file is being rendered through the download page? In other words, is there any in-built security feature which stops the user from accessing sensitive files *through* a dynamic page? Apparently not!

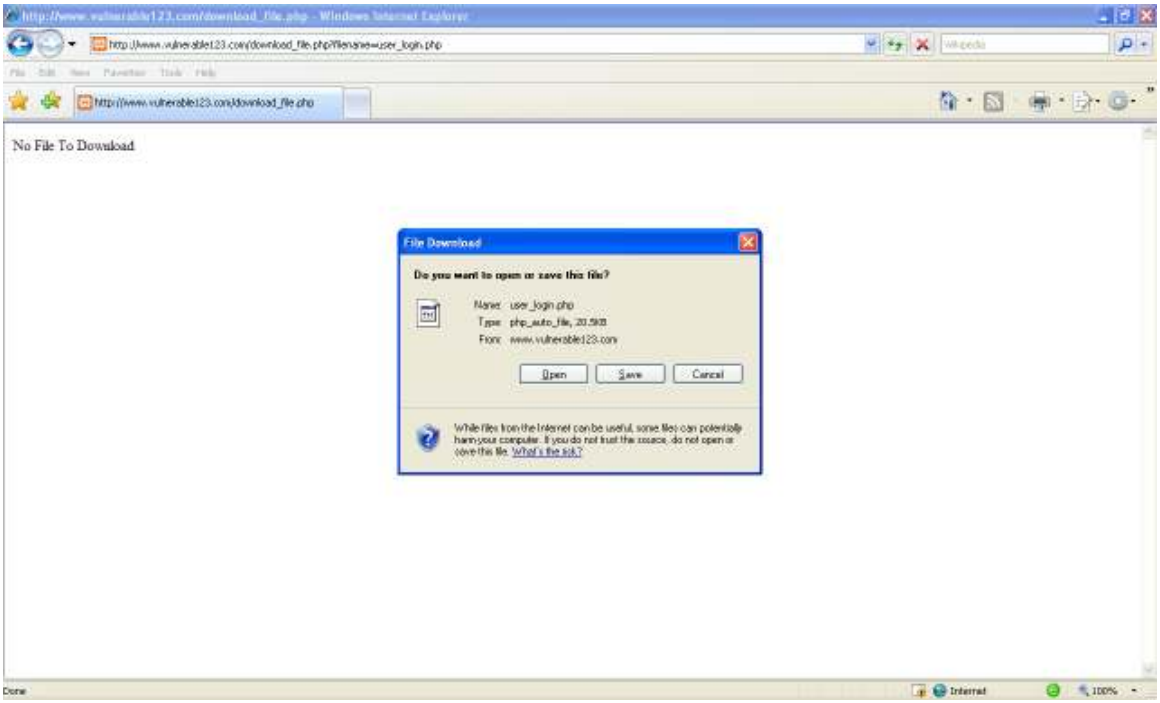
It was observed that though the 'download_file.php' file containing the source is not directly accessible, it can be accessed *through* a PHP page designed to let users download files from the server. Yes! The 'download_file.php' file can be accessed via the 'download_file.php' page! In fact, an attacker can use the 'download_file.php' page to download any file from the web root folder. The attacker simply has to provide the path of the file to the 'filename' parameter in the URL:
http://www.vulnerable123.com/download_file.php?filename=download_file.php
Lo and behold- the PHP source code file of the 'download_file.php' page is presented for download.



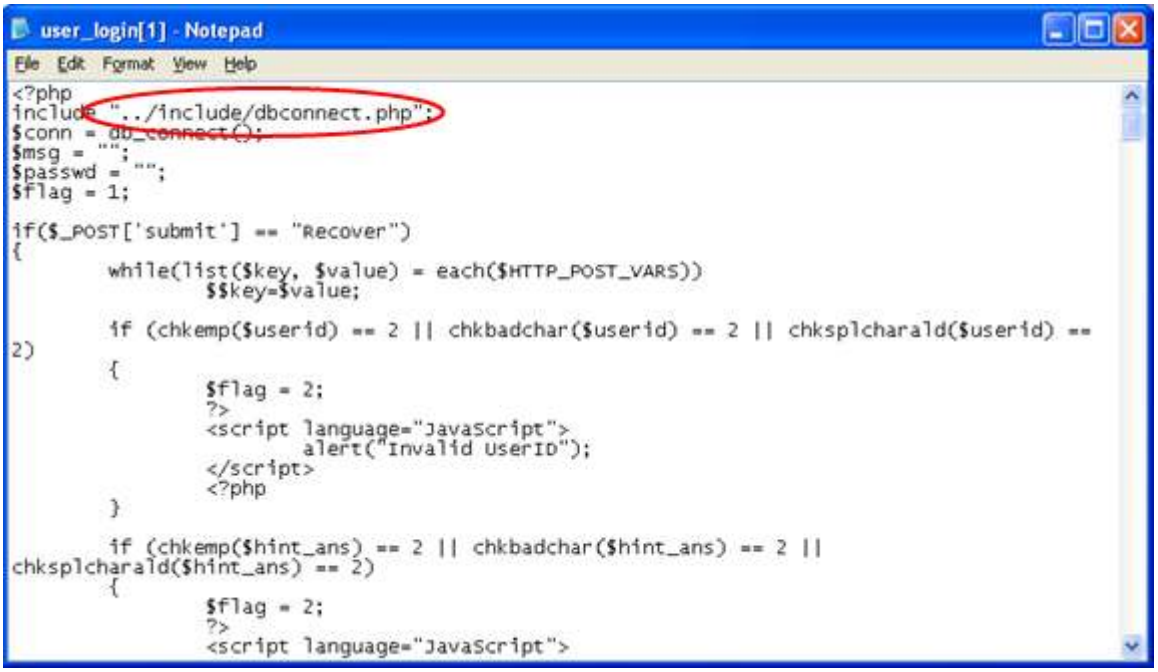
All that the attacker has to do is click on 'Open' and voila:



The attacker can also access other interesting source code files, like that of the login page. For this website, the name of the login page is 'user_login.php'. The attacker repeats the above process by submitting the following link to the server:
http://www.vulnerable123.com/download_file.php?filename=user_login.php

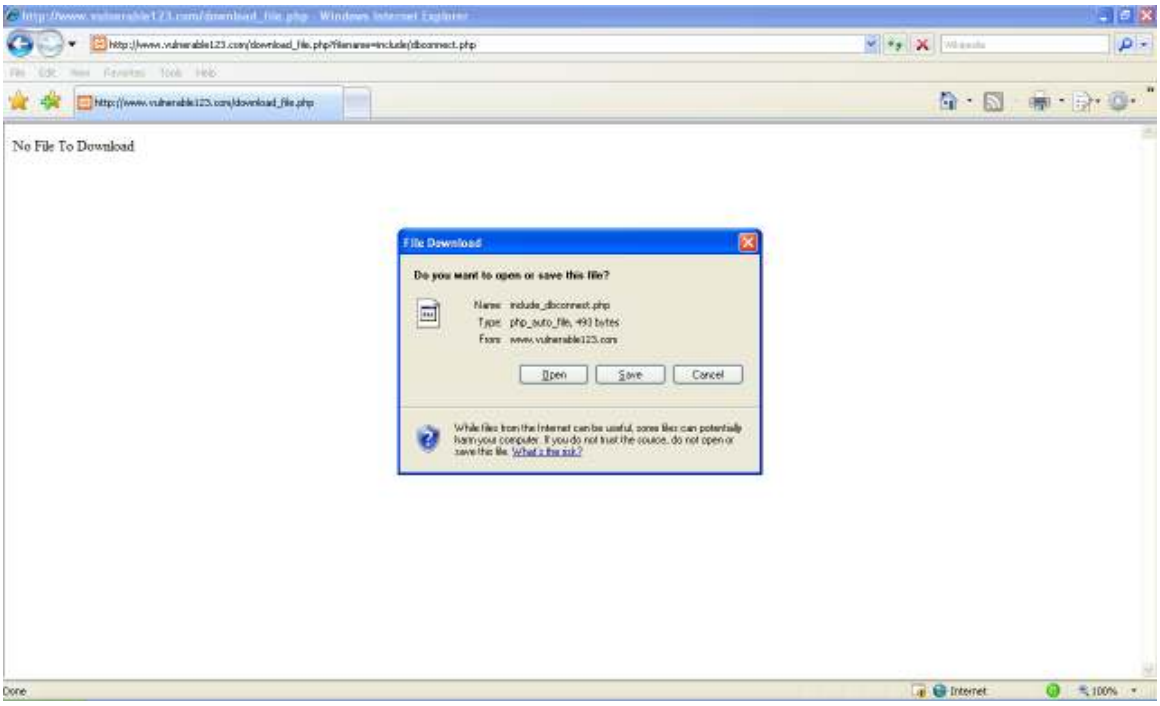


The attacker clicks on 'Open' to view the source code of the login page:

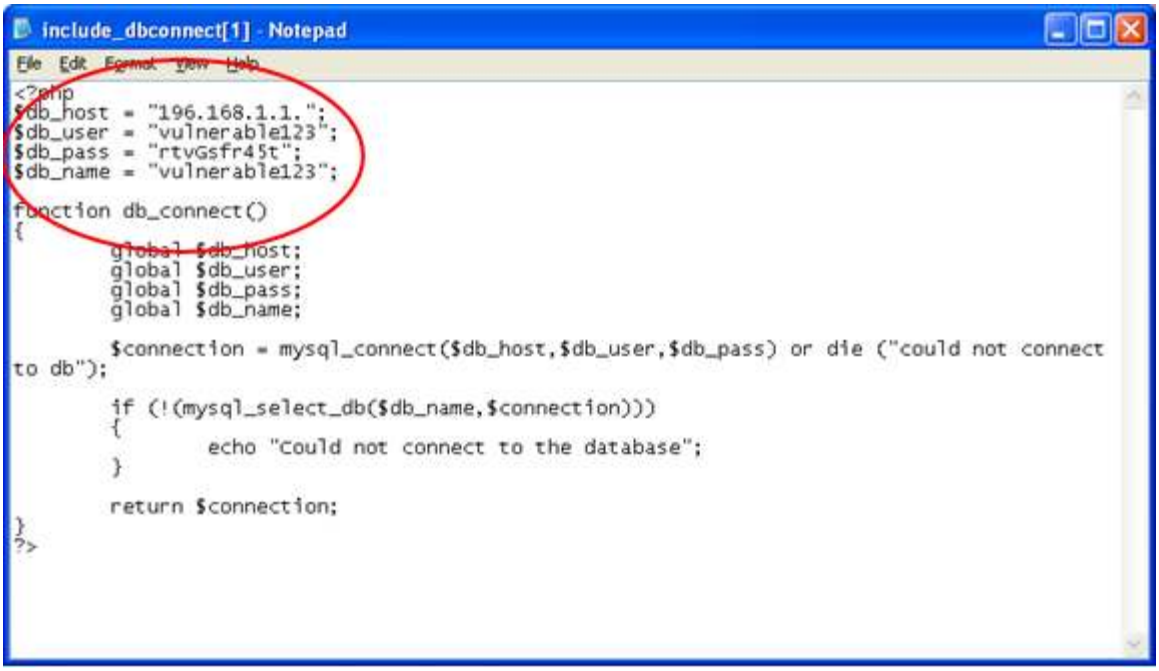


The path of a very interesting file has been revealed through the above source code. The attacker enters the path encircled above as the new value of the 'filename' parameter:

http://www.vulnerable123.com/download_file.php?filename=include/dbconnect.php



Again, the attacker clicks on 'Open' to view the source code of this PHP file:



As can be deciphered from the text encircled in the above screenshot, the source code of this file reveals all the information an attacker will need to take over the database of this website.

The Impact

Disclosure of source code and configuration files can be devastating for a web application. They usually contain database connection information like IP address, port number and valid credentials. In certain cases, application test users' login names and passwords may also be stored in these files.

What makes this attack even more dangerous is that it will go completely unnoticed because it only exploits a *functionality* of the page! It will leave no unusual trail like an error log.

For intranet applications, disclosure of the database connection information can be even more devastating since, in most cases, the databases are directly accessible on the intranet. Users can then directly connect to the database using a client and gain complete control over it.

An administrator, on discovering the changes on his database, will check the logs. He is not likely to see the attack since all the malicious database transactions were performed using legitimate credentials. The administrator will instead position his resources on hardening the network and database as attacks of this nature usually occur via that route. As can be deduced, no amount of hardening can prevent this attack from recurring.

Recommended Resolutions

The following are the recommended solutions for thwarting this attack:

- Validate the folder from where the file to be downloaded is being requested (maintain a white list of directory names from where files are allowed to be downloaded and validate the requests based on this)
- Validate the file types that are requested by users.
- Index files to be downloaded and pass only their index numbers as the URL parameter values.

About The Author

Anant Kochar is a senior IT Security Consultant at SecurEyes. He has led many application security projects. He can be reached at anant.kochar@secureyes.net.

About SecurEyes

SecurEyes is a Bangalore, India, based firm specializing in IT security. SecurEyes offers a wide range of security services and products to its clients. For more information, please visit our website: http://www.secureyes.net/.