Whitepaper

Exploiting Local File Inclusion in A Co-Hosting Environment

A Proof-of-Concept

Utkarsh Bhatt

Anant Kochhar

Exploiting LFI in a Co-Hosting Environment

TABLE OF CONTENTS

Exploiting LFI in a Co-Hosting Environment

Abstract

Local File Inclusion (LFI) vulnerability in a PHP web application can be exploited to the fullest only when it is possible to upload files into the web server. This paper explores a technique through which a properly implemented file upload module in a co-hosted website can be used for full exploitation of the LFI vulnerability, which can lead to full web server control.

Introduction

Web hosting companies and organizations often host several sites in the same web server container as a way to economize resources. This is commonly referred to as 'co-hosting'. In this paper, we demonstrate a technique through which an attacker can fully exploit an LFI in one co-hosted site through the upload module (even those which are securely implemented) in another co-hosted web site.

Upload Modules

Web applications often implement a file upload functionality allowing users to upload files, like documents etc, directly to the file system. Despite proper validations for the uploaded files, malicious code can still be appended to a file, which is then accepted as valid file content. This malicious code can then be executed by exploiting an existing LFI (Local File Inclusion) vulnerability in a co-hosted website.

Local File Inclusion

In PHP, one page can make calls to or 'include' the code written in other pages or files by using 'require', 'require_once', 'include' or 'include_once' core PHP functions. Pages vulnerable to LFI accept file path as user input to directly reference to files which have to 'included' during the run-time of the page execution.

Exploiting LFI in a Co-Hosting Environment

Attackers manipulate this user input to reference to files which are not intended by the original developers of the PHP application. For example, attackers typically will try and reference to '**etc/passwd**' file on a UNIX system.

However, attackers are more interested in referencing to files containing malicious code and try to discover vulnerabilities through which they can upload malicious files to the server.

The Attack Scenario

LFI in One Co-Hosted Site

An attacker discovers LFI vulnerability in a website. The URL of the vulnerable website page is http://www.vul456.com/index.php?file=test.php


The 'index.php' page accepts the file path as user input for the variable 'file'. It then includes the file during run-time as shown above.

The attacker can thus traverse to all files on the server. However, this is of not much use to the attacker, since all files are valid and do not contain any malicious code:
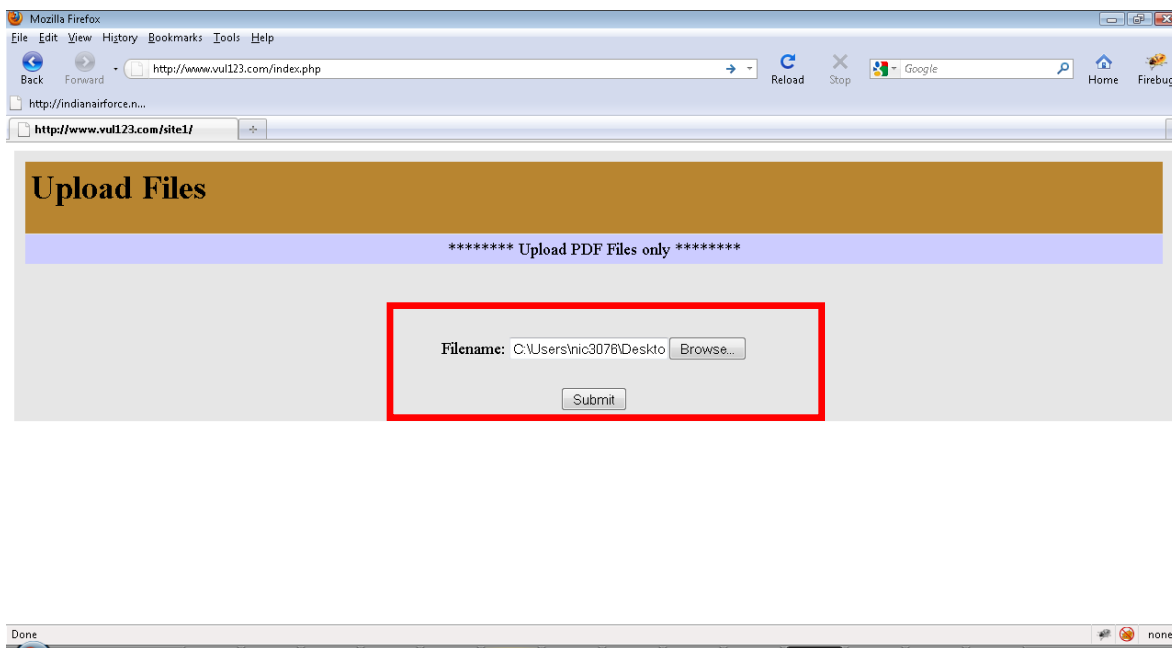URL: http://www.vul456.com/index.php?file=../site3/index.php

Exploiting LFI in a Co-Hosting Environment
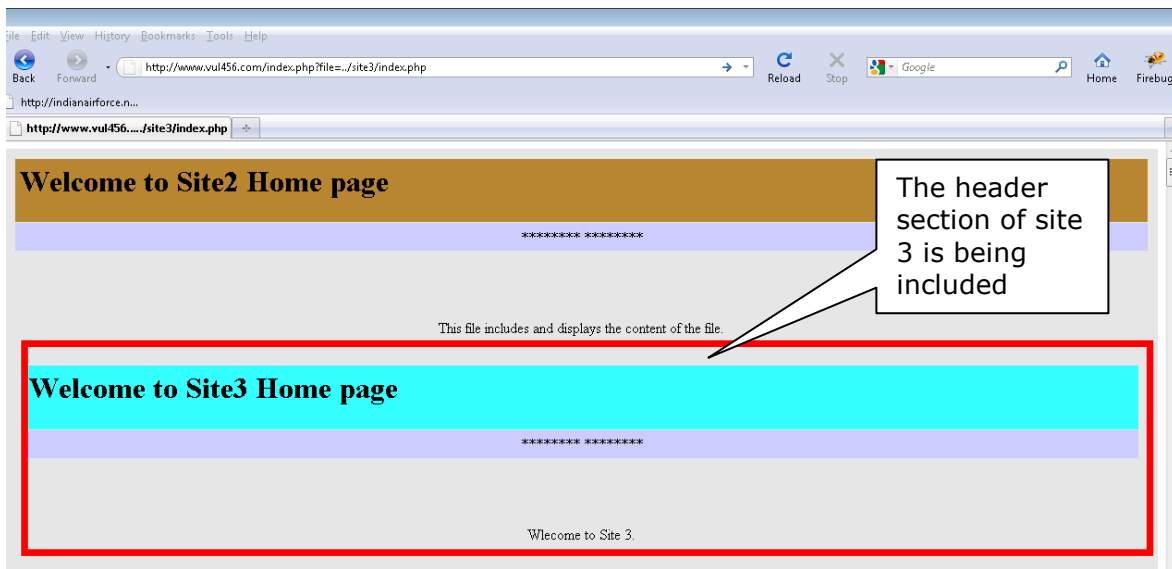
Upload Module in another Co-Hosted Website

The attacker discovers an upload module in a public page of a co-hosted website. Note that the upload module checks for valid file type and content:
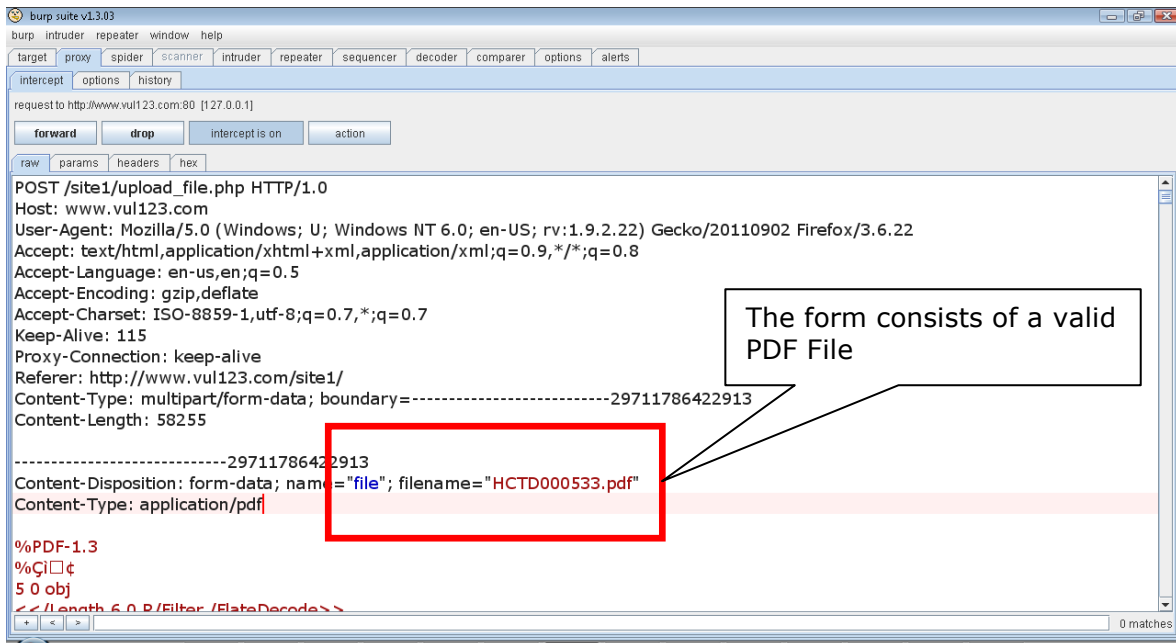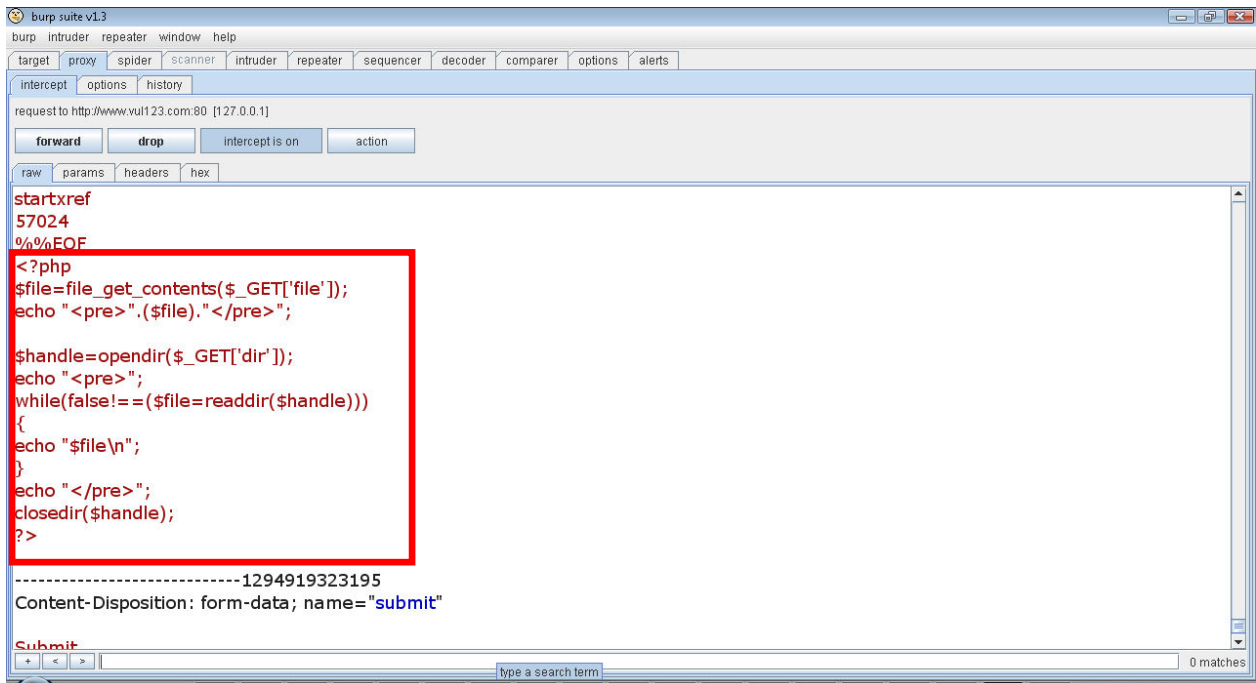
URL : http://www.vul123.com/index.php

The Exploit

The attacker captures the HTTP request to the server in an HTTP proxy, like Burp:
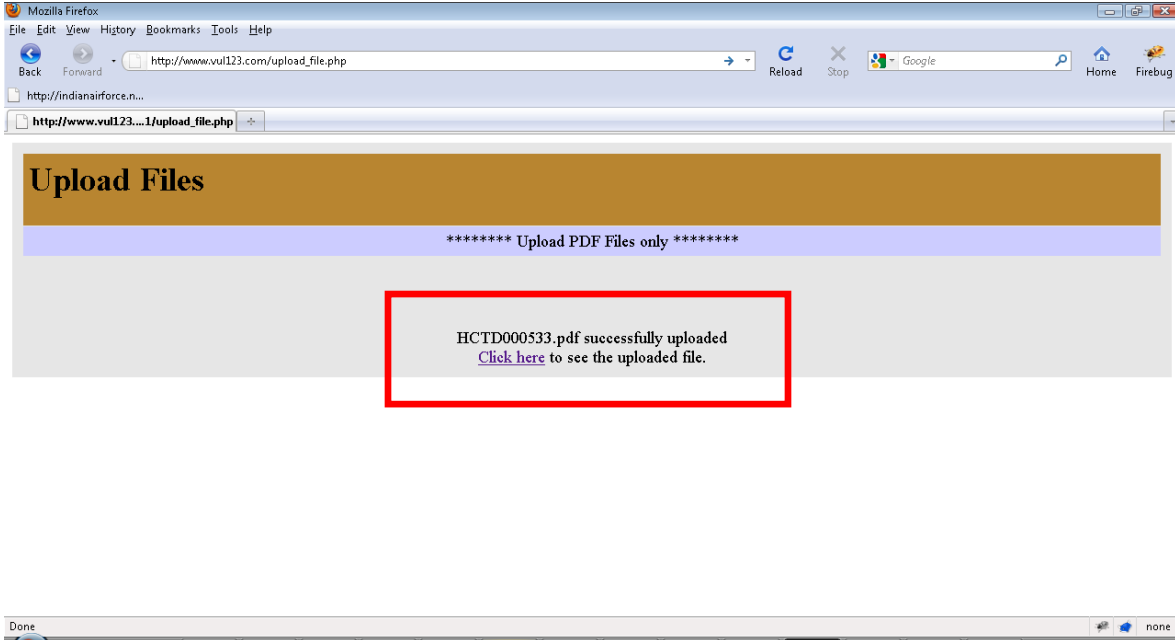

The attacker appends the malicious PHP code at the end of the PDF content.
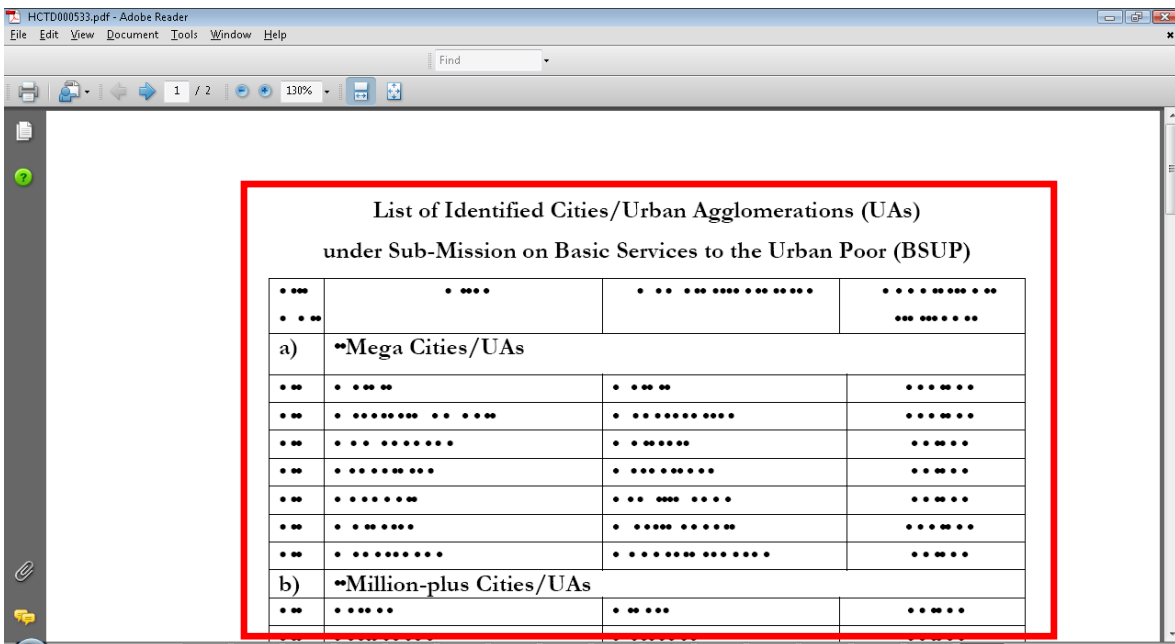
Exploiting LFI in a Co-Hosting Environment

The attacker submits the above request and successfully uploads the PDF file into the server. Note that the native PHP file validation function… is implemented in the page.
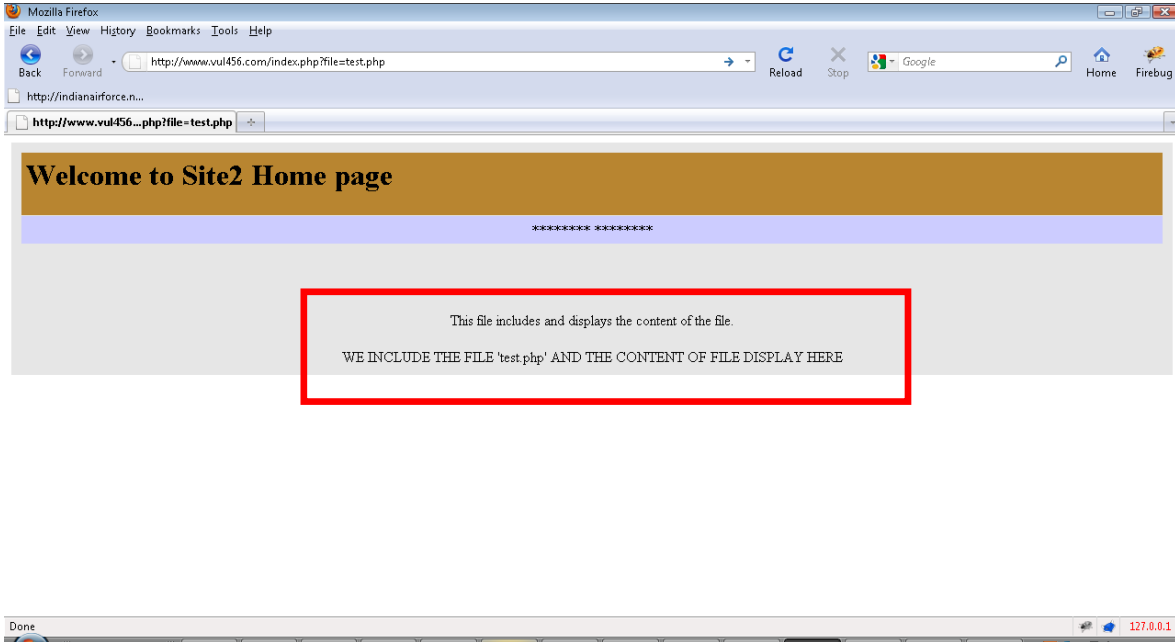

As shown below, the uploaded file is a valid PDF file which opens in the Adobe Acrobat reader:
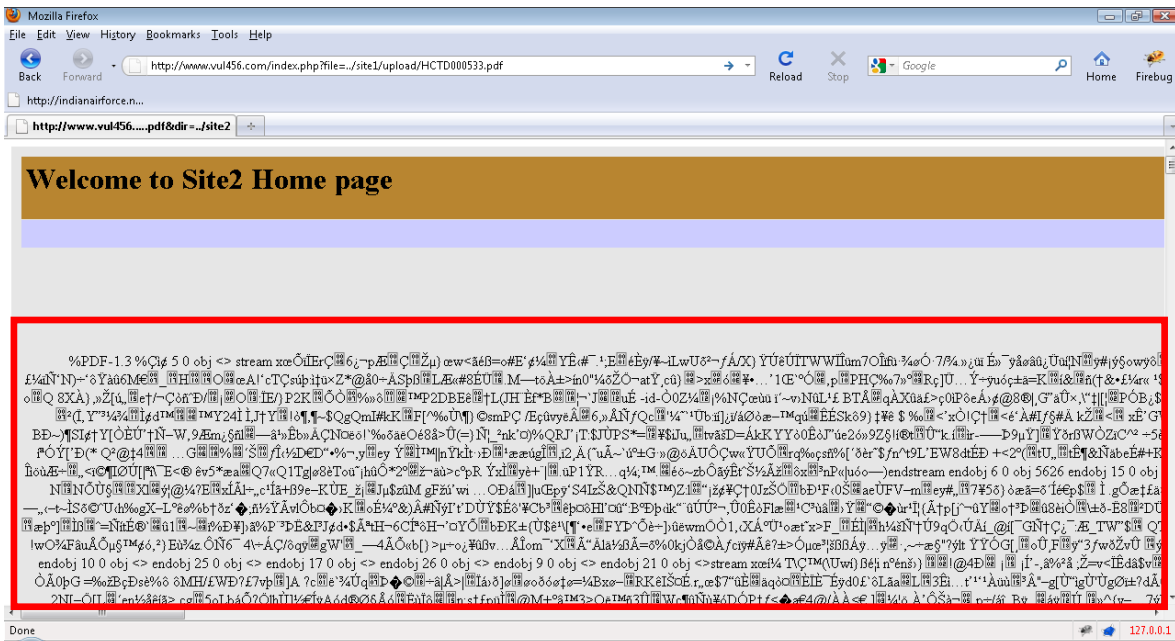
Exploiting LFI in a Co-Hosting Environment

The attacker can now fully exploit the previously discovered Local File inclusion vulnerability. The URL of the vulnerable website is

 http://www.vul456.com/index.php?file=test.php

The attacker references to the uploaded PDF file by entering its path, using directory traversal, in the 'file' GET parameter:

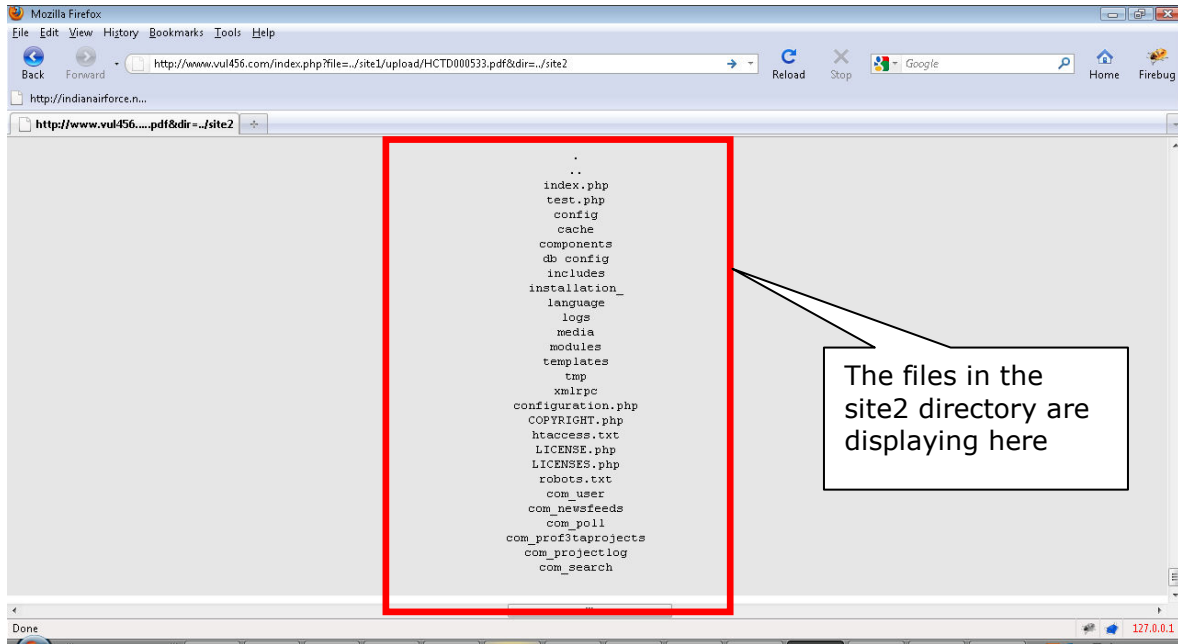http://www.vul456.com/index.php?file=../site1/upload/HCTD000533.pdf

Exploiting LFI in a Co-Hosting Environment

An attacker can pass input parameters to the malicious PDF file code by sending them as GET parameters:

http://www.vul456.com/index.php?file=../site1/upload/HCTD000533.pdf&dir=../site2

As can be seen above, an attacker can list all directories on the server. He can also read the source of any file on the server:

http://www.vul456.com/index.php?file=../site1/upload/HCTD000533.pdf&file=../site2/configuration.php
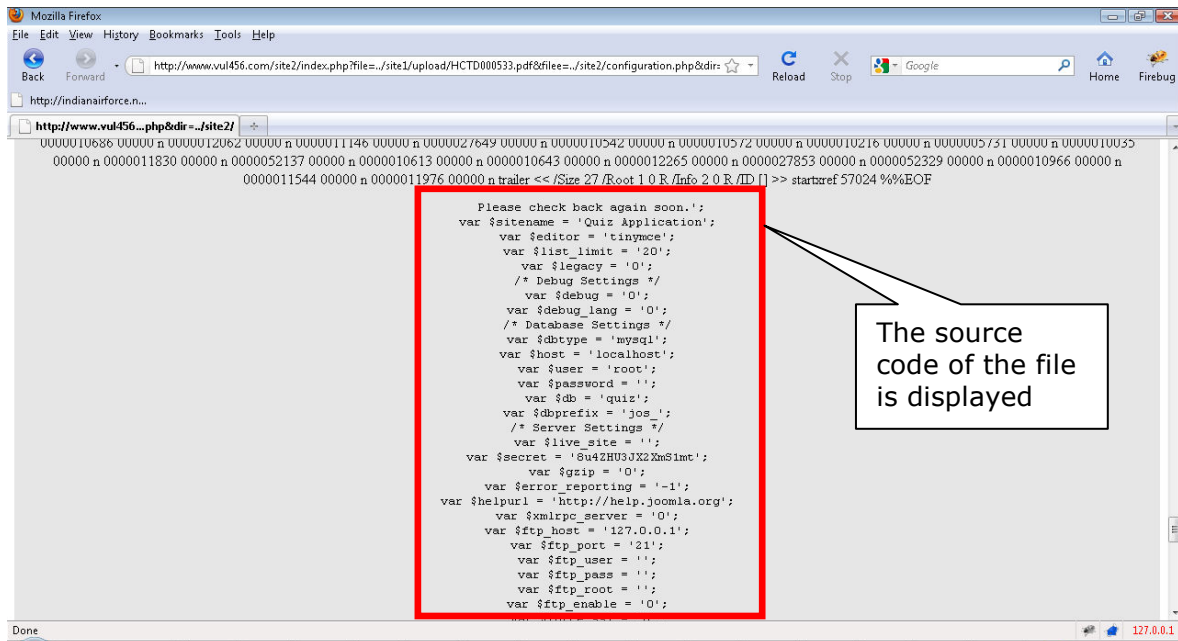
The Impact

The impact of this vulnerability is that the attacker can execute any arbitrary code on the server potentially taking the complete control of the web server.

Recommended Resolutions

Secure web hosting settings should be implemented to disallow pages/ scripts in one website to reference/ include files of other co-hosted websites. Such settings will vary according to web server type and other web hosting environment conditions. However, the following core vulnerabilities can be resolved by using secure coding techniques:

Public 'File Upload' issue:

1. File upload module should not be publically accessible. If a public file upload module is required, then the uploaded files should be stored in a database and not in the file system.

LFI vulnerability:

1. Filenames/ file paths of files to be included during execution should not be passed as a user input from the client side.
2. Proper indexing should be maintained for the files and the file id's can be passed instead of the file names. For example, 'id=1' instead of 'file=change.php'.

About The Authors

Utkarsh Bhatt and Anant Kochhar are Information Security Consultants and they have, between them, secured several web applications. They can be reached at utkarsh.bhatt@secureyes.net and anant.kochhar@secureyes.net respectively.

About SecurEyes

SecurEyes is a Bangalore 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/.