Honey Logger

1. PROJECT DESCRIPTION

Team Members: Aashish Radhakrishnan (Team Leader)
Mrunmayee Shinde
Riya Bulia
Shashwath Hosur

a) **Abstract:**

Honey logger is a project based on honey-pots. Honey-pots are an excellent tool to improve security since it make use of decoy machines to lure the attackers and drive their attention away from the protected website. It uses the five listed elements to secure the system.

1. Harden security as much as possible
2. Prepare for being attacked
3. If attacks occur, then detect them
4. If detection is successful, then respond appropriately
5. Take all the information and use it to harden the system and repeat all five steps again.

Honeypot is a very interesting concept since it increase the security by adding an extra layer to it. **Network Security Building Blocks**: We have made mid-interaction honeypot which is used to read the IP addresses of the attacker and storing it in the database.

Functionalities of Honey pots:

Data capture: This component is used to capture the intrusion activities, events or attacker
Data Control: It is used to slow down the intrusion activities or defuse attackers
Interfaces: Provides the API, a network or non-network IF (Implementation Facilitator) using the interface component.
In our project we have implemented the first functionality of honeypots which is to capture the data.

b & c) Project Description: (ACHIEVEMENT)

Basically to attack or hack a system, the attacker makes uses of Bot. A bot is a type of malware that allows an attacker to take control of website or a computer. Bots often spread by searching for vulnerable, unprotected computers. And when they find an expose computer they infect the machines. In our project, we create a website which distinguishes human from the bots. Initially, we ask the user to enter the name and the email id to which the user wants to sends the message. But here the catch is we have the toggle honeypot button which says that the field has to left blank if they are human. The honeypot works by, hiding the input from human visitors, but not the spambots. The bots fill it in just like any other input which tells the PHP or JS Validation that it's spam and then we direct them to new website which tells that the page is not found and make them aware that it has been detected that they are spammers and their IP addresses has been logged into Honey_log.txt thus giving us a simple honeypot which we call the honey logger.

We are using PHP language to illustrate the Honey logger. Following files are used to support our project:

1. Honeypot.php - This is the homepage of our website and introduction to what our project does. This contains the link to the demo.
2. Honeypot-demo.php - This is the login page with username, email and toggle honey pot on/off button. This button is usually unchecked so that the human user enter any details by mistake. If toggle honeypot contains text then it is directed to the backdoor.php otherwise to the honey-confirmation.php.
3. Honey-confirmation.php - This shows that the user is a human and an email is sent.
4. Backdoor.php - Only the spammers are directed to this page. In this we display message saying that it has been detected that they are spammers and their IP addresses has been logged.
5. Honey_log.txt - This text file contains the IP addresses of the spammers.

d) System Framework and Programming Platform:

Apache: It is the server for running PHP
PHP: We are using php language to build our simple honeypot application to misguide the attacker so that we can protect our targeted application

e) Each Member's Contribution:

Honeypot.php - Shashwath Hosur
Honeypot-demo.php - Aashish Radhakrishnan
Honey-confirmation.php - Riya Buliya
Backdoor.php - Mrunmayee Shinde

f) Sample code

1). Honeypot.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html><head><meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Simple PHP HoneyPot | Tutorial | { visibility: inherit; }</title>
```

```
<link rel="shortcut icon" href="/favicon.ico" type="image/x-icon">
<link href="/styles.css" rel="stylesheet" type="text/css" media="screen">
<link href="/styles-alt.css" rel="alternate stylesheet" type="text/css" media="screen"
title="styles-alt">
</head>

<body id="code">
<div id="wrapper">

<div id="nav">
<?php include("../includes/nav.php"); ?>
</div>
<div id="content">

<h1 align="center">Honey Logger</h1>
<h2 align="center"> Welcome to our CNS Project</h2>
<h3 align="center">Done by:
  Shashwath HA
  Aashish R
  Mrunmayee S
  Riya B</h3>
  <h4 align="center">Under the guidance of Prof. Jie Wang</h4>

<p><a href="honeypot-demo.php">Here is a working Demo <b>&raquo;</b></a></p>
<p>
The honeypot works by, hiding the input from human visitors, but not the spambots.
The spambots fill it in just like any other input which tells the PHP or JS Validation that it's spam
and prevents the email from sending.
So we would call the input something unsuspecting so the bot fills it in and does not become
wise to our ways.
Placeholder is in place so that if CSS is off the human user knows not to fill it in (so again may
want to say something more general here).
And autocomplete off is in place so that the human users browser does not by mistake fill in the
hidden input.</p>
</div>
<div id="clearfooter"></div>

</div>
<div id="footer">
```

```
<div id="footerborder"></div>

</div>

</body>
</html>
```

2). Honeypot-demo.php

```
!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Simple PHP HoneyPot | Demo | { visibility: inherit; }</title>
<style type="text/css">
#honeypot {
visibility:hidden;
}
input:checked ~ br + #honeypot {
visibility:visible;
}
</style>
</head>
<body>
<h1>Simple PHP HoneyPot</h1>
<p><a href="honeypot.php">&laquo; Back To Tutorial</a></p>

<form method="post" action="honeypot-formmail.php">
	<label for="name">Name</label>
	<input type="text" id="name" name="name" placeholder="Name">
	<br>
	<label for="email">Email</label>
	<input type="text" maxlength="50" id="email" name="email" placeholder="Email">
	<br>
	<br>
	<input type="text" id="honeypot" name="honeypot" placeholder="Leave Blank If
Human" autocomplete="off">
	<br>
	<input type="submit" name="submit" value="Submit" id="submit">
</form>

</body></html>
```

3). Honeypot-confirmation.php

```
<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Simple Confirmation | { visibility: inherit; }</title>
</head>
<body>
<p>Email Sent!</p>
<p><a href="honeypot-demo.php">&laquo; Back To HoneyPot Demo</a></p>
</body></html>
```

4). Backdoor.php

```
<html>
<head> <title>404 Not found</title></head>
<body bgcolor=white>
<h1>404 not found</h1>


The requested mail could not be sent as it has been detected that you are a spammer.
You are being reported and your ip address has been logged.
Thank You have a bad day.
Bye Bye
<img src="trollface.jpg"/>
<?php
if (isset($_SERVER["HTTP_X_FORWARDED_FOR"]))
{
   $ip = $_SERVER["HTTP_X_FORWARDED_FOR"];
}
else {
  $ip = $_SERVER["REMOTE_ADDR"];
}
if(isset($ip))
{
  $myFile = '/var/www/html/website/honey_log.txt';
  if (!file_exists($myFile)) {
  }
  else if(!$fh = fopen($myFile, 'a')) {
  }
  else {
   $output = $ip."--".gethostbyaddr($ip)."--".date("Y:m:d:H:i:s");
```

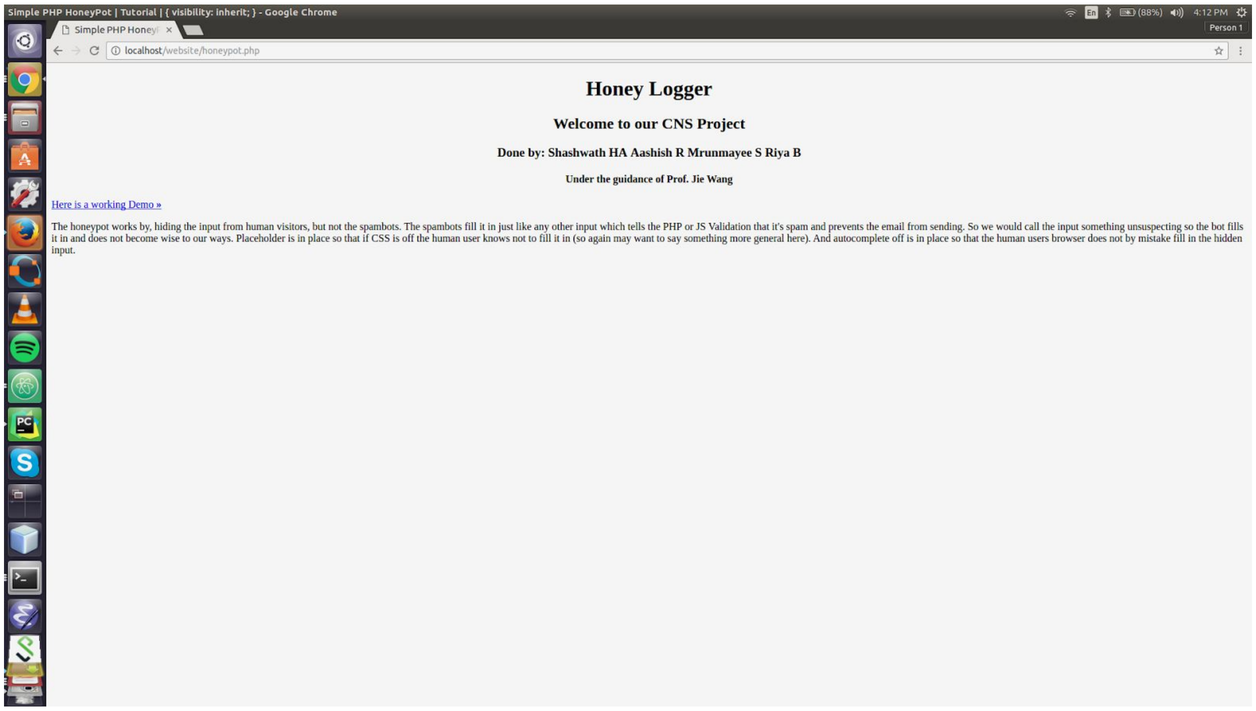
```
    fwrite($fh, $output);
    fclose($fh);


  }


}
?>
</body>
</html>
```

g) Snapshots of Running Samples:

1). The welcome page of our Honey Logger website which introduces what the honeypot does



Simple PHP HoneyPot | Tutorial | { visibility: inherit; } - Google Chrome

localhost/website/honeypot.php

Honey Logger

Welcome to our CNS Project

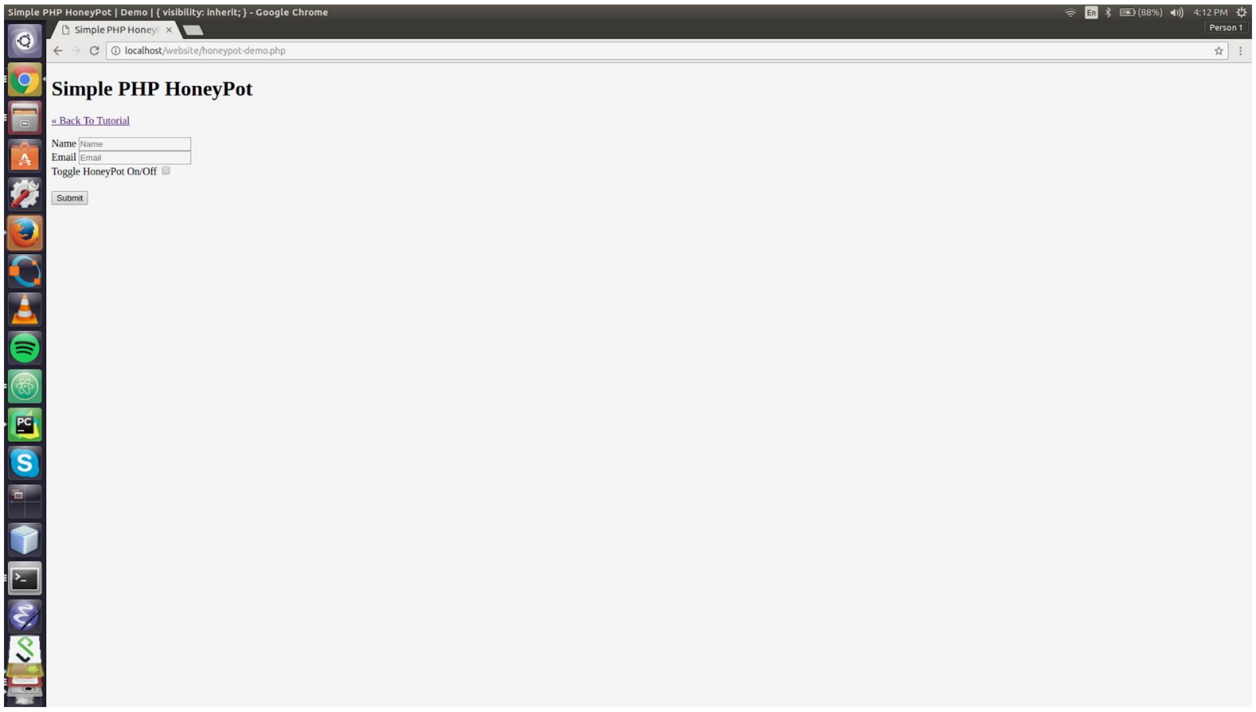
Done by: Shashwath HA Aashish R Mrunmayee S Riya B

Under the guidance of Prof. Jie Wang

Here is a working Demo »

The honeypot works by, hiding the input from human visitors, but not the spambots. The spambots fill it in just like any other input which tells the PHP or JS Validation that it's spam and prevents the email from sending. So we would call the input something unsuspecting so the bot fills it in and does not become wise to our ways. Placeholder is in place so that if CSS is off the human user knows not to fill it in (so again may want to say something more general here). And autocomplete off is in place so that the human users browser does not by mistake fill in the hidden input.

2). A simple form asking the user to enter the name and the email id


Simple PHP HoneyPot | Demo | { visibility: inherit; } - Google Chrome
localhost/website/honeypot-demo.php

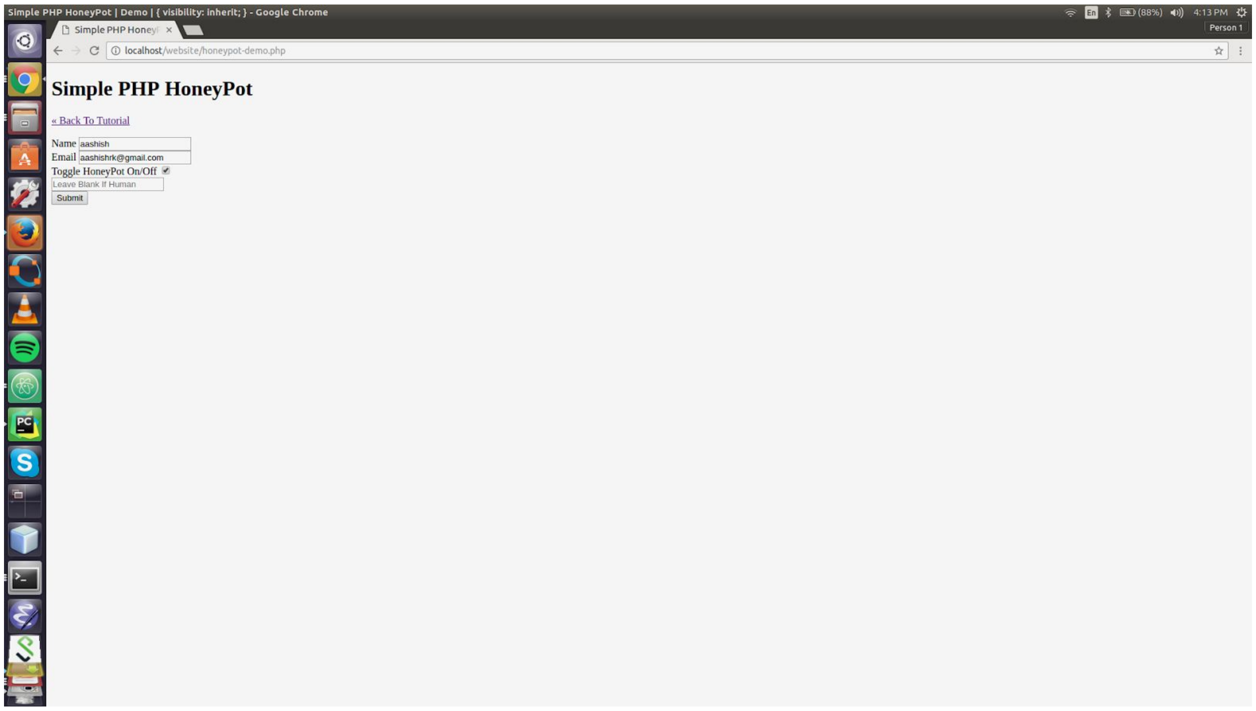
Simple PHP HoneyPot

« Back To Tutorial

Name
Email
Toggle HoneyPot On/Off

Submit

3). The user enters the name and the email but leaves the toggle Honeypot field empty



Simple PHP HoneyPot | Demo | { visibility: inherit; } - Google Chrome

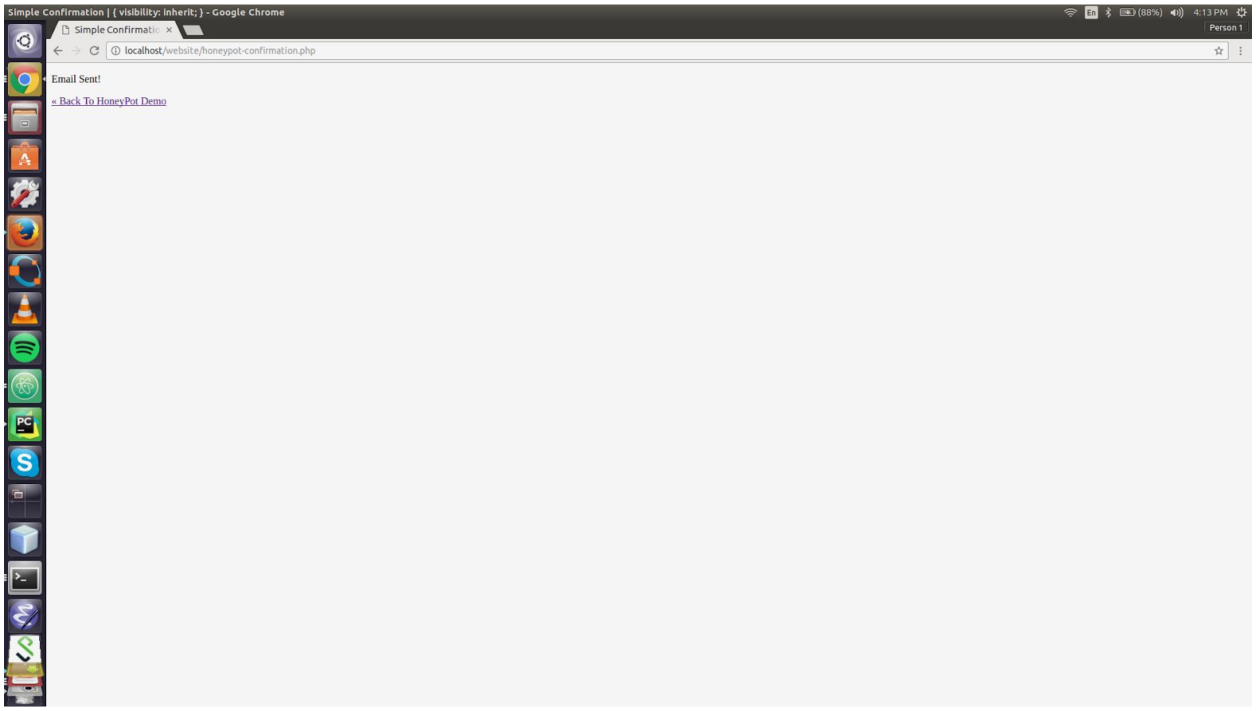
localhost/website/honeypot-demo.php

Simple PHP HoneyPot

« Back To Tutorial

Name aashish
Email aashishrk@gmail.com
Toggle HoneyPot On/Off ✓
Leave Blank If Human
Submit

4). Since the user left the toggle honeypot button empty we display the email sent message



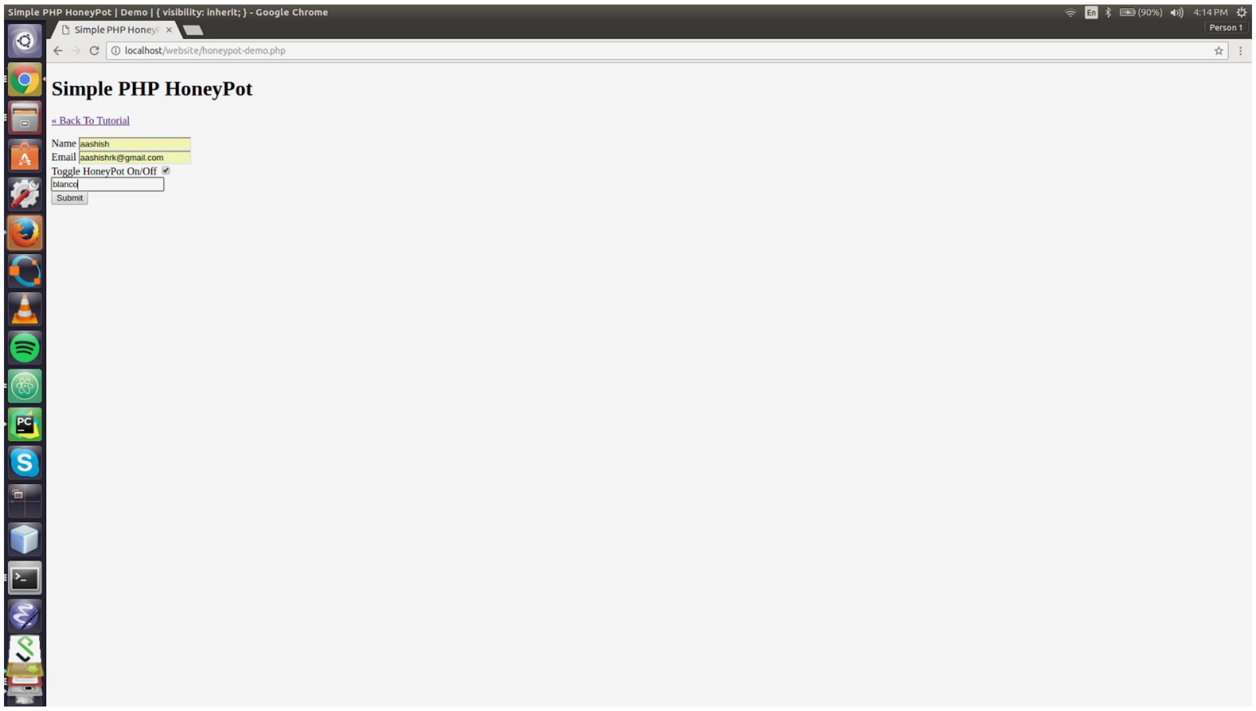
Simple Confirmation | { visibility: inherit; } - Google Chrome

localhost/website/honeypot-confirmation.php

Email Sent!

« Back To HoneyPot Demo

5). In this screenshot the user enters in the toggle Honeypot field.



Simple PHP HoneyPot | Demo | { visibility: inherit; } - Google Chrome

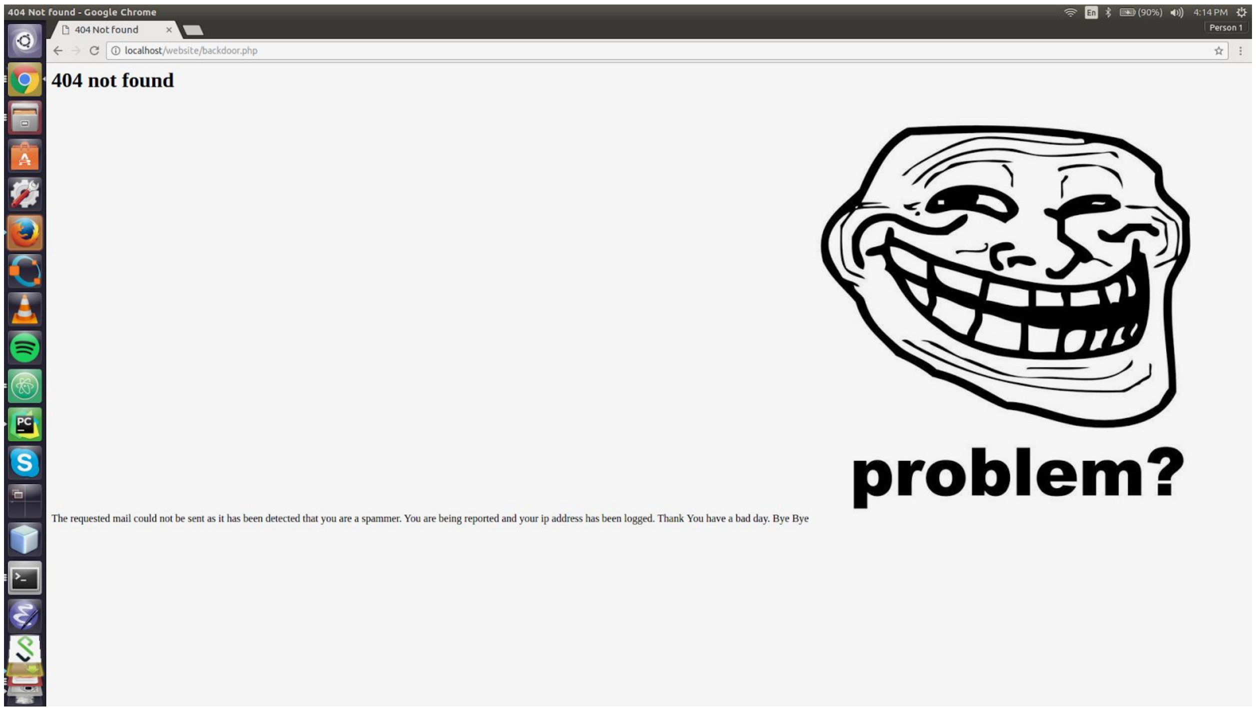
localhost/website/honeypot-demo.php

Simple PHP HoneyPot

« Back To Tutorial

Name: aashish
Email: aashishrk@gmail.com
Toggle HoneyPot On/Off ✓
blanco
Submit

6). Here the user is directed to backdoor.php where we display a message saying that the spammer has been detected and that it's ip address has been logged



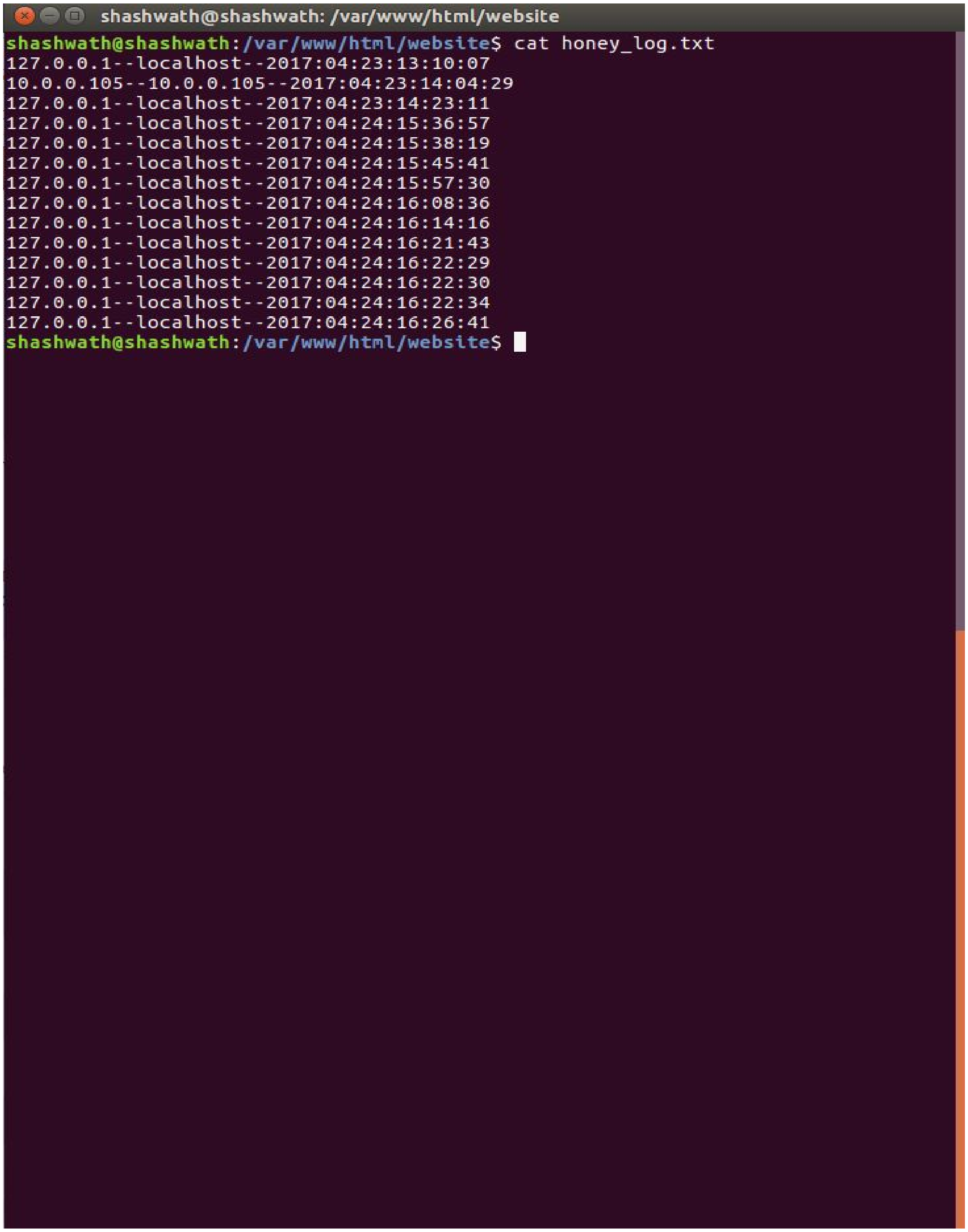
404 Not found - Google Chrome

localhost/website/backdoor.php

404 not found

The requested mail could not be sent as it has been detected that you are a spammer. You are being reported and your ip address has been logged. Thank You have a bad day. Bye Bye

7). This shows the content of the honey_log.txt which shows the attacker's ip address



shashwath@shashwath: /var/www/html/website

```
shashwath@shashwath:/var/www/html/website$ cat honey_log.txt
127.0.0.1--localhost--2017:04:23:13:10:07
10.0.0.105--10.0.0.105--2017:04:23:14:04:29
127.0.0.1--localhost--2017:04:23:14:23:11
127.0.0.1--localhost--2017:04:24:15:36:57
127.0.0.1--localhost--2017:04:24:15:38:19
127.0.0.1--localhost--2017:04:24:15:45:41
127.0.0.1--localhost--2017:04:24:15:57:30
127.0.0.1--localhost--2017:04:24:16:08:36
127.0.0.1--localhost--2017:04:24:16:14:16
127.0.0.1--localhost--2017:04:24:16:21:43
127.0.0.1--localhost--2017:04:24:16:22:29
127.0.0.1--localhost--2017:04:24:16:22:30
127.0.0.1--localhost--2017:04:24:16:22:34
127.0.0.1--localhost--2017:04:24:16:26:41
shashwath@shashwath:/var/www/html/website$
```

REFERENCES:

1. http://www.diva-portal.org/smash/get/diva2:327476/fulltext01
2. https://www.youtube.com/watch?v=-Zo-P_M4dOk
3. https://www.youtube.com/watch?v=o6dtMdIKZSk&t=245s
4. Wang, J. (2008). *Computer network security: Theory and practice*. Beijing: Higher Education Press.