1 - INTRODUCTION

1.1-BACKGROUND

According to Bruce Schneider "*Security is a process, not a product*." This famous quote is well echoed by the phenomenon that, although there exist umpteen numbers of security tools that are available today (either as commercial or open-source solutions), none of these tools can single-handedly address all of the security goals of an organization. The security professionals are thus looking for more advanced tools which are effective in detecting and recovering from security breaches. In order to monitor the activities of hackers, the methodology adopted is to deceive, by giving them some emulated set of services on a system which appears to be legitimate. The hackers' activities are then logged and monitored to gain insight into their employed tactics. This idea is adopted in honey-pots, a system whose value lies in being probed, attacked and/or compromised.

Honey-pots have received a lot of attention lately from the research community, owing to its use in capturing and logging suspicious networking activities. Apart from its use as a research tool, it has also been deployed in educational institutions as a study tool.

For example, the Honey net Project at Georgia Tech has been used in network security classes in order to teach students how to use tools such as ethereal in order to analyze attack traffic. However, the problem with deploying such a honey-pot inside a campus network is twofold. First, the installation of honey-pot is quite risky and must be carried out with utmost precision and care, so that the campus network is not intruded. Secondly, legal and ethical issues (such as the Wiretap Act 18 U.S.C. 2511) must be taken care of before such a network is deployed.

Honeypots can be classified based on their deployment and based on their level of involvement. Based on deployment, honeypots may be classified as:

1. production honeypots
2. research honeypots

Production honeypots are easy to use, capture only limited information, and are used primarily by companies or corporations; Production honeypots are placed inside the production network with other production servers by an organization to improve their overall state of security. Normally, production honeypots are low-interaction honeypots, which are easier to deploy. They give less information about the attacks or attackers than research honeypots do.

Research honeypots are run to gather information about the motives and tactics of the Black hat community targeting different networks. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats organizations face and to learn how to better protect against those threats. Research honeypots are complex to deploy and maintain, capture extensive information, and are used primarily by research, military, or government organizations.

Based on design criteria, honeypots can be classified as

1. pure honeypots
2. high-interaction honeypots
3. low-interaction honeypots

Pure honeypots are full-fledged production systems. The activities of the attacker are monitored using a casual tap that has been installed on the honeypot's link to the network. No other software needs to be installed. Even though a pure honeypot is useful, stealthiest of the defense mechanisms can be ensured by a more controlled mechanism.

High-interaction honeypots imitate the activities of the real systems that host a variety of services and, therefore, an attacker may be allowed a lot of services to waste his time. According to recent researches in high interaction honeypot technology, by employing virtual machines, multiple honeypots can be hosted on a single physical machine. Therefore, even if the honeypot is compromised, it can be restored more quickly. In general, high interaction honeypots provide more security by being difficult to detect, but they are highly expensive to maintain. If virtual machines are not available, one honeypot must be maintained for each physical computer, which can be exorbitantly expensive. Example: Honeynet.

Low-interaction honeypots simulate only the services frequently requested by attackers. Since they consume relatively few resources, multiple virtual machines can easily be hosted on one physical system, the virtual systems have a short response time, and less code is required, reducing the complexity of the security of the virtual systems. Example: Honeyd.

The project will be focused on the following aspects:

- This project will be focusing on server side programming language specifically PHP which are mainly used for development of CMS – content management systems, e – commerce, social networking site.

- The project acts as a Service Provider for Honey-pot security to various websites.

- The project is proposed to trace the user activities on the client site – demo websites.

- Various attacks performed on the demo sites will be traced within the tool. Thus providing a competitive analytical & statistical data so as to find the loop holes of the demo sites.

- This project can be deployed at on the same server as in where the demo website is located or can also work with remote web server.

- This project will be using PHP, JavaScript, & my SQL. Our main aim is to emphasize more on the LAMP framework – LINUX – APACHE-MY SQL-PHP

- With a user-friendly environment, it will diminish the problem of reading long and unstructured log files and efficiently captures what has happened inside the virtual honey-pots.

Current System & their drawbacks:

The current honey pot tools available are moreover network specific and don't focus more on attacks the Web based attacks. None of the present honey pot tools are designed to work with website & determine the attacks performed on the web based applications.

- **The following are some famous Honey-pots available in the market:**

- **Back Officer Friendly (BOF):** Back Officer Friendly is a spoofing server application that runs on your Windows or UNIX system, and notifies you whenever someone attempts to remote control your system using Back Officer. Basically, it pretends to be a

Back Orifice server. Back Officer Friendly gives the attacker false answers that look like they came from Back Orifice, while logging the attackers IP address and the operations they attempted to perform.

- **Specter:**Specter is a commercial honey-pot supported by NetSec, a network security company. Specter is a smart honey-pot or deception system. It simulates a complete machine, providing an interesting target to lure hackers away from the real machines. Specter offers common Internet services such as SMTP and FTP which appear perfectly normal to the attackers but in fact are traps for them to mess around and leave traces without even knowing that they are connected to a fake system which does none of the things it appears to do it instead logs everything and notifies the appropriate people. Furthermore, Specter automatically investigates the attackers while they are still trying to break-in.

- **Honeyd:** Honeyd is a pre-packaged Open Source honey-pot designed for the UNIX platform by Neils Provos. It is a low interaction honey-pot; therefore, there is no operating system to interact with and it is designed primarily to detect attacks or unauthorized activity. Since it is an Open Source solution and highly customizable, the user may configure it to listen on any port he/she wants and to adjust the level of emulation to meet his/her specifications.

1.2-PROBLEM STATEMENT

The system acts as a service provider for Honey-pot Security to various websites. It acts as a framework to implement honey-pot which can be used by any organization to test their website applications / portals.

1-Our clients would consist of websites of PHP format.

2-It plan to trace characteristics of hackers like

3-The browser they use

4-Their IP address from the IP header

5-The files accessed

6-The loopholes they discover

7-Various inputs that are used for various input fields

8-Script Injection

Websites & CMS developed in technologies such as PHP, ASP, CGI, JavaScript, and Ajax have made it much easier for people to build and deploy services on the Internet. Unfortunately, this has opened a wide possibility for new attacks since it is accidentally introduce new vulnerabilities into it. Therefore, websites have increasingly been the focus of attackers. Although a lot of web administrator has a lot of choice to choose the more stable and secure open source content management system & websites as their favorite instant deployment medium, they still need to monitor for vulnerabilities and threats that have been occurred on the web servers. For that reason, the web administrator needs an easier way on how to analyze the long and unstructured log file for every server. The best way is to pass on the threat and monitor it on single point like having a proxy within the network to log any HTTP request.

This project will propose a proper way on how to help the web administrator monitor their entire web-server HTTP request by looking at the log server only instead of having to read every each server log file.

Hence there has been always a need for website owners & web development companies for high interaction website monitoring tools for the attacks.

1.3-SCOPE OF THE PROJECT

The main aim of the system is to keep a track of the attacker entering the database. The banking site consists of many databases including dummy databases, used to attract the hacker. There is a need to keep a close watch on the data files that have dummy database stored in them. If the dummy database is attacked there is no harm to the original databases. The LOG files will give the administrator any information about the modified files. In this project the user will give a random port range to test. The port scanner will take the IP address of the server and range of ports as input. It will display the open ports. The request will be saved in the LOG file. Honeypot setup must be regularly monitored, if an intruder gets into a honeypot, whole system is exposed.

Plus, it can launch new attacks to other networks from the honeypot.

1-DataCollection

Honeypots collect very little data, and what they do collect is normally of high value. This cuts the noise level down, make it much easier to collect and archive data.

2-Resources

Many security tools can be overwhelmed by bandwidth or activity. Network Intrusion Detection Devices may not be able to keep up with network activity, dropping packets, and potentially attacks. Centralized log servers may not be able to collect all the system events, potentially dropping some events. Honeypots do not have this problem, they only capture that which comes to them.

3-SingleDataPoint

Honeypots all share one huge drawback; they are worthless if no one attacks them. Yes, they can accomplish wonderful things, but if the attacker does not send any packets to the honeypot, the honeypot will be blissfully unaware of any unauthorized activity.

4-Risk

Honeypots can introduce risk to your environment. It is because of these disadvantages that honeypots do not replace any security mechanisms. One of the greatest problems in security is wading through gigabytes of data to find the data you need. Honeypots can give you the exactly the information you need in a quick and easy to understand format.

1.4-RELEVENCE OF THE PROJECT

The current honey pot tools available are moreover network specific and don’t focus more on attacks the Web based attacks. None of the present honey pot tools are
Designed to work with website & determine the attacks performed on the web based applications.

The following are some famous Honey-pots available in the market:

1-Back Officer Friendly (BOF):

Back Officer Friendly is a spoofing server application that runs on your Windows or UNIX system, and notifies you whenever someone attempts to remote control your system using Back Officer. Basically, it pretends to be a Back Orifice server. Back Officer Friendly gives the attacker false answers that look like they came from Back Orifice, while logging the attackers IP address and the operations they attempted to perform.

2-Specter:

Specter is a commercial honey-pot supported by NetSec, a network security company. Specter is a smart honey-pot or deception system. It simulates a complete machine, providing an interesting target to lure hackers away from the real machines. Specter offers common Internet services such as SMTP and FTP which appear perfectly normal to the attackers but in fact are traps for them to mess around and leave traces without even knowing that they are connected to a fake system which does none of the things it appears to do it instead logs everything and notifies the appropriate people. Furthermore, Specter automatically investigates the attackers while they are still trying to break-in.

3-Honeyd:

Honeyd is a prepackaged Open Source honey-pot designed for the UNIX platform by Neils Provos. It is a low interaction honey-pot; therefore, there is no operating system to interact with and it is designed primarily to detect attacks or unauthorized activity. Since it is an Open Source solution and highly customizable, the user may configure it to listen on any port he/she wants and to adjust the level of emulation to meet his/her specifications.

2- LITERATURE SURVEY

There are a lot of Honey-pot tools has been around such as Project honey-net, Spectra, Honeyed, back officer and many more. The only reason the existence of Honey-pot is because the information beneath it that can help web administrator to understand any kind of attack and how to countermeasure it.

According to Lance Spitzner(2002):A Honey-pot as "a security resource whose value lies in being probed, attacked or compromised ".This means that whatever we designate as a Honey-pot, it is our expectation and goal to have the system probed, attacked, and potentially exploited. Keep in mind; Honey-pots are not a solution.They do not 'fix' anything. Instead, Honey-pots are a tool. How you use that tool is up to you and depends on what you are attempting to achieve. A Honey-pot may be a system that merely emulates other systems or applications, creates a jailed environment, or may be a standard built system. Regardless of how you build and use the Honey-pot, its value lies in the fact that it is attacked'. Research Honey-pots are Honey-pots designed to gain information on the blackhat community. These Honey-pots do not add direct value to a specific organization. Instead they are used to research the threats organizations face, and how to better protect against those threats. Think of them as 'counter-intelligence', their job is to gain information on the bad guys. This information is then used to protect against those threats.

Honey-pots are decoy computer resources set up for the purpose of monitoring and logging the activities of entities that probe, attack or compromise them. Activities on Honey-pots can be considered suspicious by definition,as there is no point for benign users to interact with these systems.Honey-pots come in many shapes and sizes;examples include dummy items in a database,low-interaction network components like preconfigured traffic sinks, or full interaction hosts with real operating systems and services.Such a system is called high-interaction honey-pots because the attacker is able to fully interact with the honey-pot just like a real system.This offers the best potential for analyzing all aspects of an attack, but also introduces risk that the attacker will use the capabilities of the system to attack others.A high interaction honey-pot must disguise itself as a real machine, hiding its surveillance methods to all users even if they have root privileges.A physical honey-pot is a real machine with its own IP address.This can be a disaster if the machine can be compromise there is high opportunity that it is going to break.

Honey pots are closely monitored decoys that are employed in a network to study the trail of hackers and to alert network administrators of a possible intrusion. [2]
Following are the proposed honey pots:

Honey tokens: Honey tokens are fake records that are inserted in the database. These fake records are not expected to be used by normal users. If any of these honey tokens are used, they alert us of the database having been compromised. An example of honey tokens are fake username/passwords in the user database. These users do not exist in the real world, and hence are not expected to be logging in to the application. If the application sees these credentials being used, it immediately recognizes that the user database has been compromised.

Honey pages: These are obscure web pages sprinkled in the web site. They have no legitimate purpose, nay they are not even linked from any valid page. Normal users would never reach these pages. However, we drop hints about these pages by embedding their URL as comments or hidden fields in valid pages. While normal users would never see this, an attacker who analyzes the source code, or a vulnerability scanner that spiders the site would see these and follow the link. When the page is accessed, it points us to the intruder.

Remote File Inclusion(RFI) is a technique often used to attack Internet websites from remote computer. With malicious intent, it can be combined with the usage of XSA to harm a web server. Remote File Inclusion attacks allow malicious users to run their own PHP code on a vulnerable website. The attacker is allowed to include his own (malicious) code in the space provided for PHP programs on a web page.

SQL injectionis a code injection technique that exploits a security vulnerabilityoccurring in the database layer of an application. The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed. It is an instance of a more general class of vulnerabilities that can occur whenever one programming or scripting language is embedded inside another. SQL injection attacks are also known as SQL insertion attacks.

Feasibility study

All projects are feasible, given unlimited resources and infinite time. But the development of software is plagued by the scarcity of resources and difficult delivery rates. It is prudent to evaluate the feasibility of the project at the earliest possible time.
Three key considerations are involved in feasibility analysis.

1. <u>Technical Feasibility:</u>

Technical feasibility centers on the existing computer system (Hardware, Software etc.,) and to what extent it can support the proposed addition. If the budget is a serious constraint, then the project is judged not feasible.

2. *<u>Economic Feasibility</u>***:**

This procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. It benefits outweigh costs, and then the decision is made to design and implement the system. Otherwise, further justification or alternations in proposed system will have to be made if it is to have a chance of being approved. This is an ongoing effort that improves in accuracy at each phase of the system lifecycle.

3. <u>Operational Feasibility:</u>

People are inherently resistant to change, and computers have been known to facilitate change. It is understandable that the introduction of a candidate system requires special effort to educate, sell, and train the staff on new ways of conducting business.

We have made the following estimations

TIME IN DAYS			
Activities	**Optimistic**	**Most Likely**	**Pessimistic**
Study a Networks & Security	20	23	28
Study on various types Honey-pots	50	53	58
Analysis on Web Based Honey-pot & various web attacks	10	10	11
Existing System Study	5	7	8
Fact finding & Interview	2	3	3
Module specification of proposed system	10	12	14
Hardware and Software requirement.	5	5	5
Feasibility Study	5	6	7
System Design	10	11	14
Development	20	28	34
Testing	10	10	12
Implementation	2	4	5
Final Documentation	8	10	12

Table 2.1 operational feasibility

Predecessor Table:

This is the table that involves the various activities involved and the time estimated per activity.

Activity Number.	**Activities**	**Must be preceded by**	**Estimated Time Days**
A	Study about Security concept	-	23
B	Study about various honeypots	A	53
C	Literature surveyed	-	10
D	Existing System Study	B,C	8
E	Fact finding & interview	D	4
F	Module specification of proposed System	E	14
G	Hardware and Software requirement.	F	6
H	Feasibility Study	G	6
I	System Design	H	14

Table 2.2 Processor table

3-PROJECT DESIGN

3.1-UseCase Diagram:

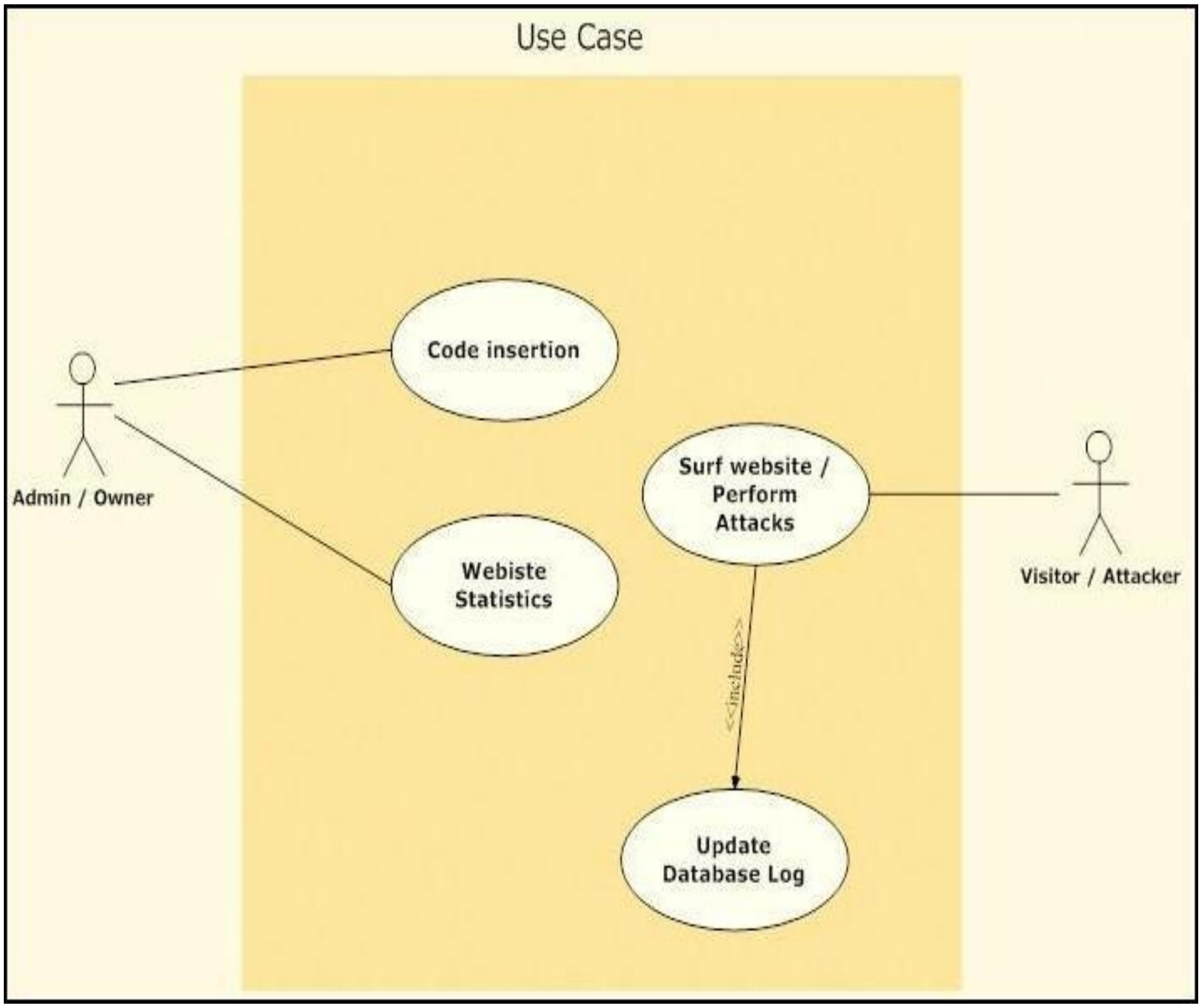


Fig 3.1 Use case Diagram

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

3.2- Data Flow Diagram:

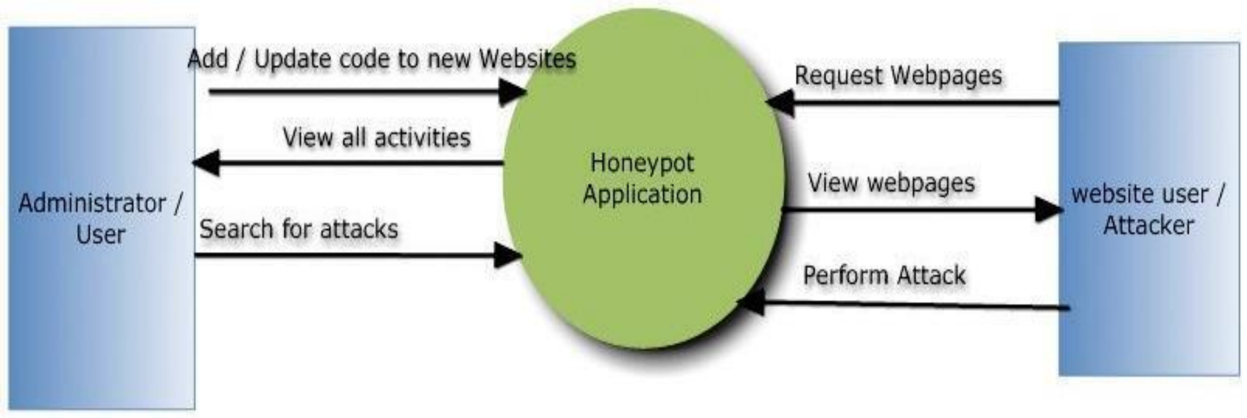


Fig 3.2 DFD level 0 for HPSS

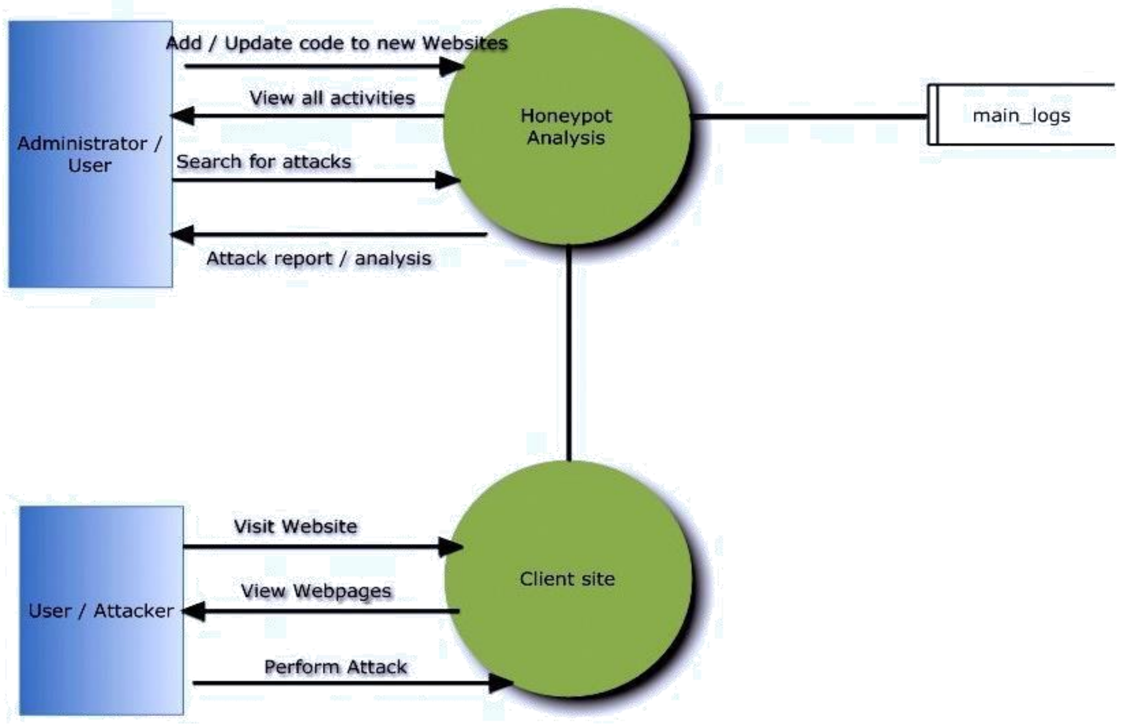


Fig 3.3DFD level1 for HPSS

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFD can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

3.3-System Architecture:

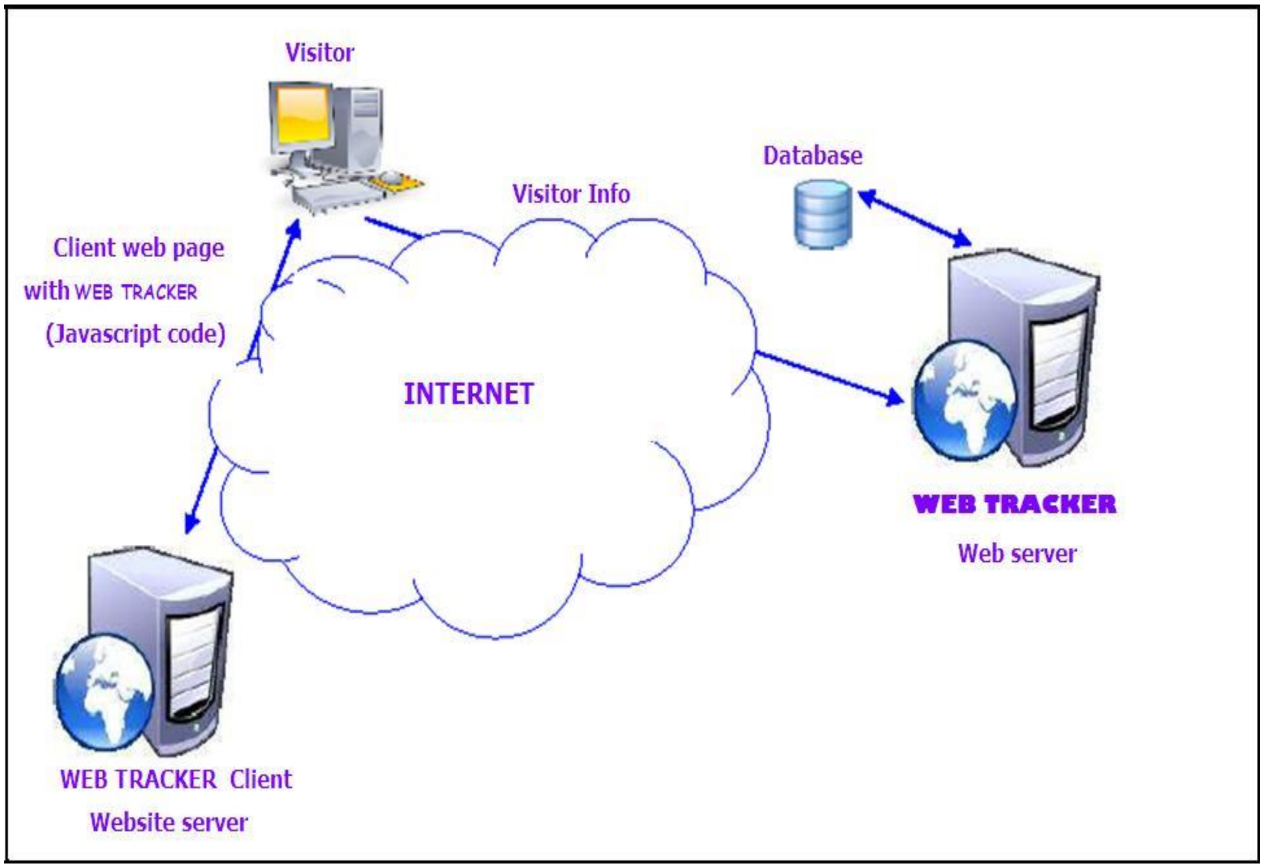


Fig 3.4 System Architecture

System Architectureis a diagram of a system, in which the principal parts or functionsare represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in the engineering world in hardware design, electronic design, software design, and process flow diagram

3.4-Sequence Diagram:

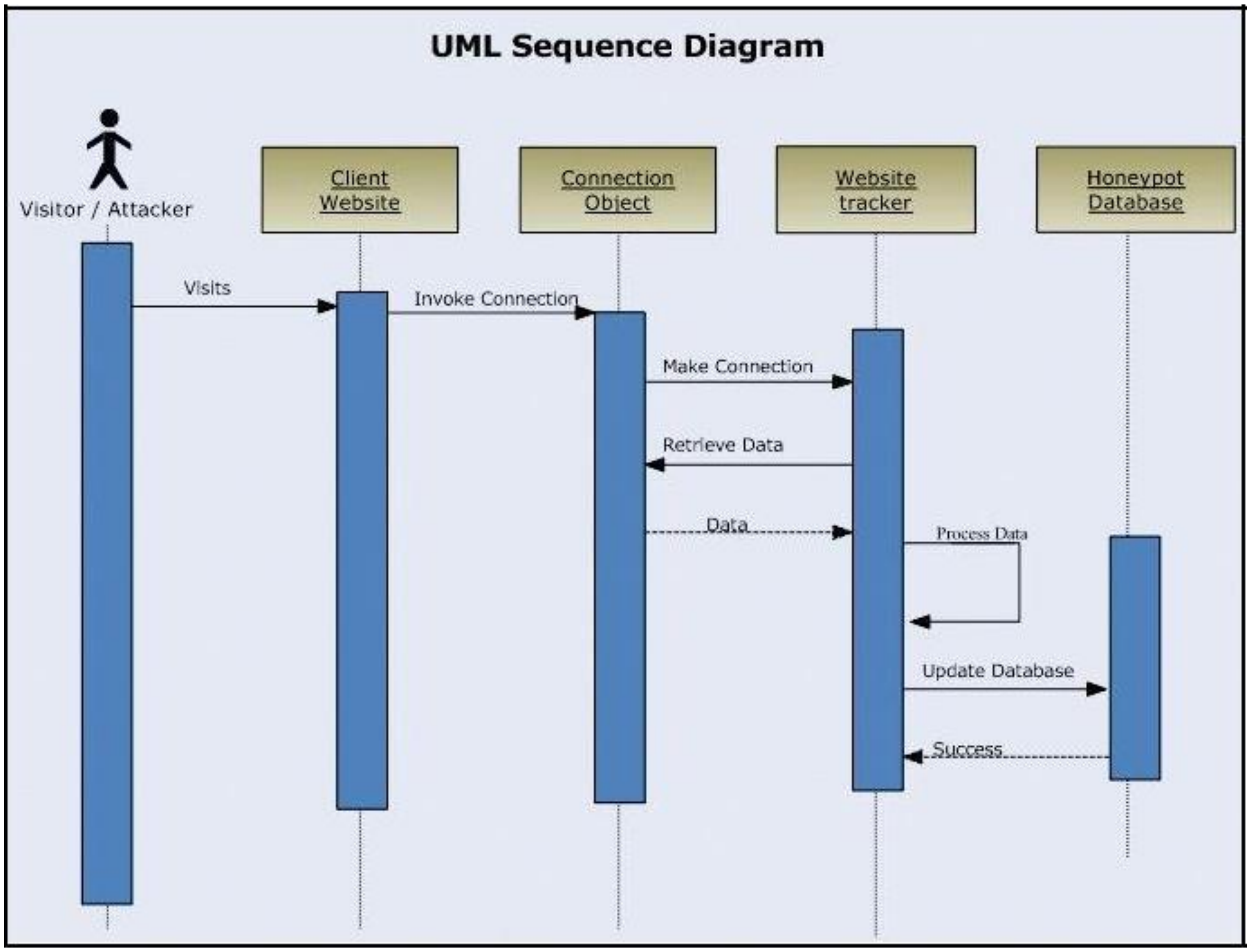


Fig 3.5 Sequence Diagram

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and **timing diagrams**.

3.5-Activity Diagram:

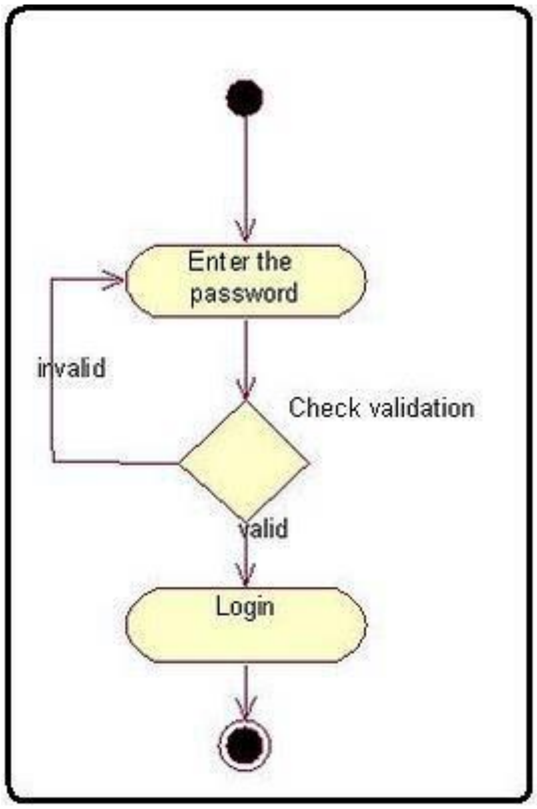


Fig 3.6 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities andactions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

3.6-Network Diagram:

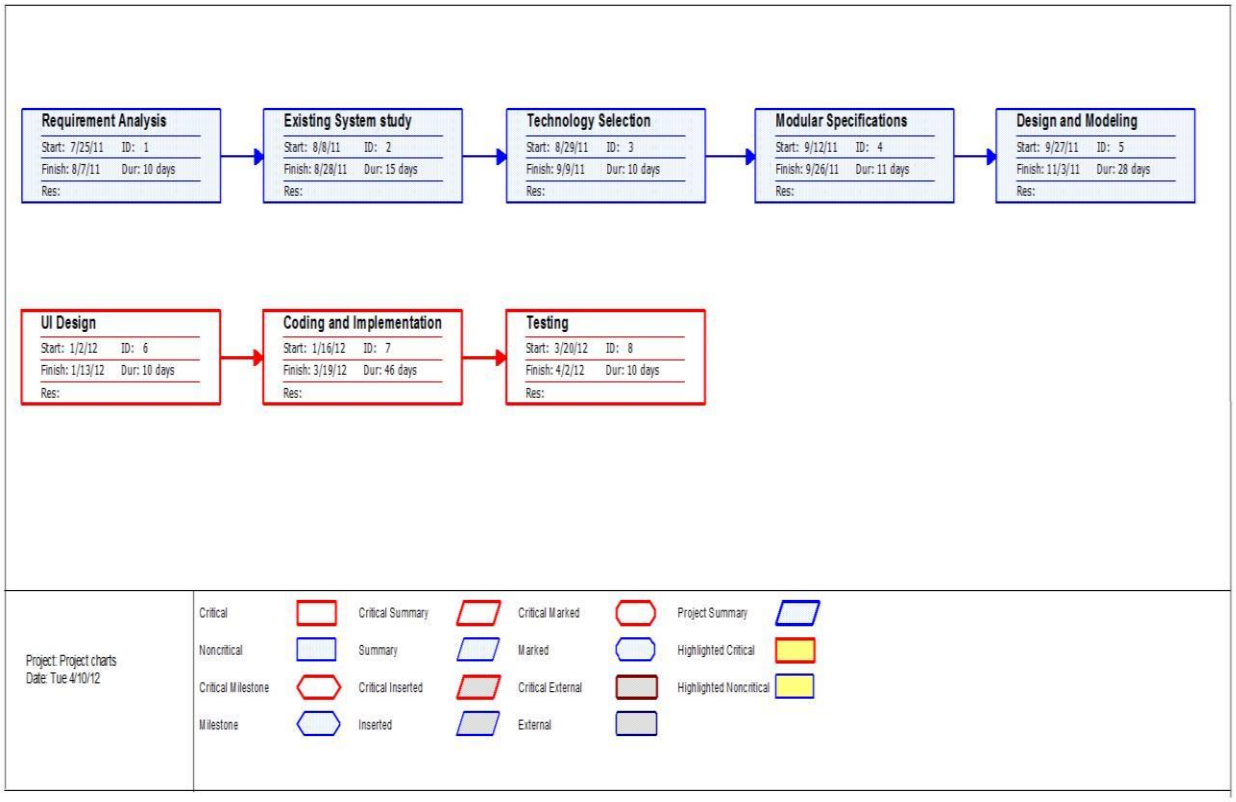


Fig 3.7 Network Diagram

A **network diagram** may refer to:

- Computer network diagram, a depiction of nodes and connections in a computer or telecommunications network
- Graph drawing, methods for visualizing graphs and networks regardless of their application
- Project network, a flow chart showing the sequence of a project's tasks and their dependencies
- Social network, a social structure of individuals or organizations

4 - IMPLEMENTATION

4.1-METHODOLOGIES

Process Model:

Software process model deals with the model which we are going to use for the development of the project. There are many software process models available but while choosing it we should choose it according to the project size that is whether it is industry scale project or big scale project or medium scale project.

Accordingly the model which we choose should be suitable for the project as the software process model changes the cost of the project also changes because the steps in each software process model varies.

This software is build using the waterfall mode. This model suggests work cascading from step to step like a series of waterfalls. It consists of the following steps in the following manner

4.1.1-Waterfall model

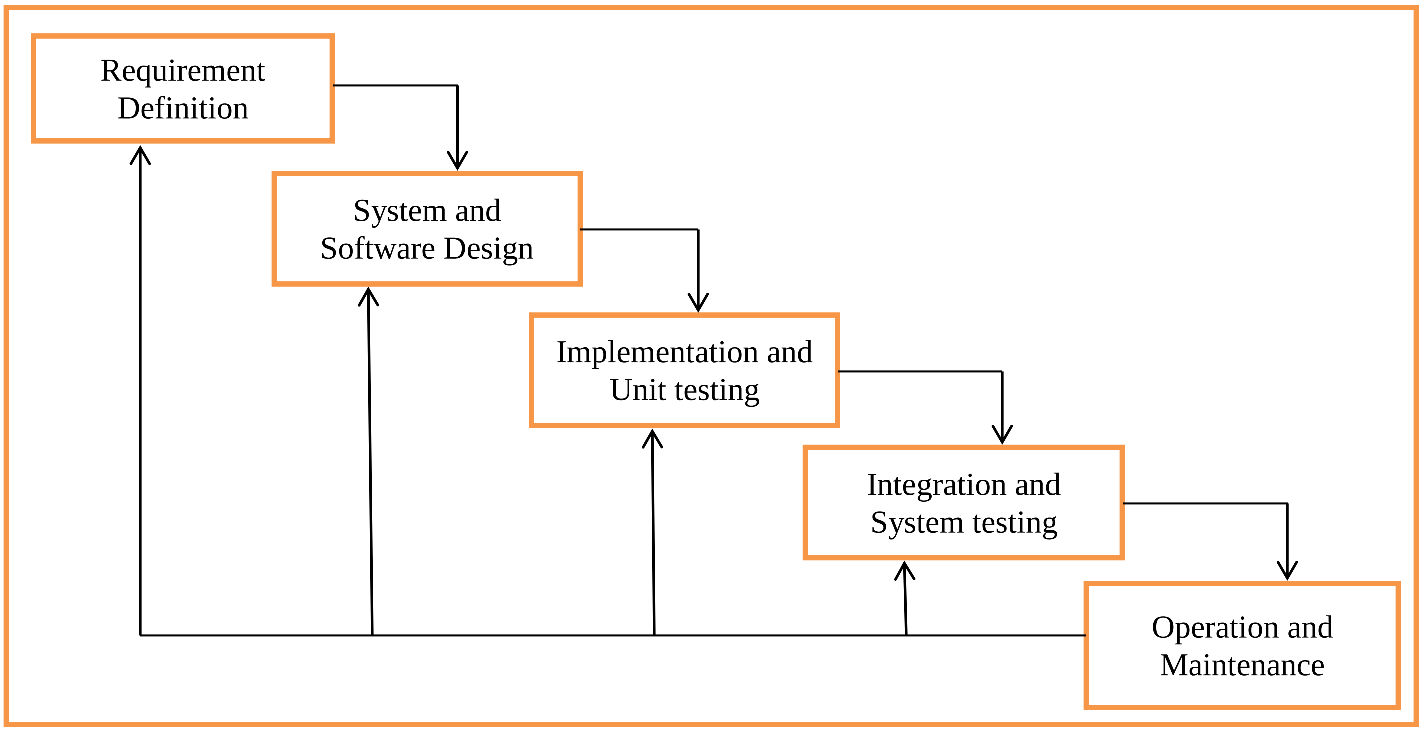

Requirement Definition
System and Software Design
Implementation and Unit testing
Integration and System testing
Operation and Maintenance

Fig 4.1.1 Waterfall Diagram

*Requirement Definition and Analysis Phase:*To attack a problem by breaking it into sub-problems. The objective of analysis is to determine exactly *what* must be done to solve the problem. Typically, the system's *logical* elements (its boundaries, processes, and data) are defined during analysis.

*System Software and Design Phase:*The objective of design is to determine *how* the problem will be solved. During design the analyst's focus shifts from the logical to the *physical.* DataElements are grouped to form physical data structures, screens, reports, files, and databases.

*Implementation Phase:*The system is created during this phase. Programs are coded, debugged, documented, and tested. New hardware is selected and ordered. Procedures are written and tested. End-user documentation is prepared. Databases and files are initialized. Users are trained.

*Integration and System Testing Phase:*Once the system is developed, it is tested to ensure that it does what it was designed to do. After the system passes its final test and any remaining problems are corrected, the system is implemented and released to the user.

The system acts as a service provider for Honey-pot Security to various websites. It acts as a framework to implement honey-pot which can be used by any organization to test their website applications / portals.
Our clients would consist of websites of PHP format.
We plan to trace characteristics of hackers like
The browser they use
The loopholes they discover
Various inputs that are used for various input fields
Script Injection
We will be tracing the entire request made by the client to the server from the demo site. All the GET-POST values sent & received from the client will be logged in to the Honey-pot database for analysis purpose.
We will broadly use the G-P-S-C [GET POST SERVER COOKIES] data variables which are sent & received from the client to the server.

There are two ways the browser client can send information to the web server.

- The GET Method
- The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand. Example: name1=value1&name2=value2&name3=value3

Spaces are removed and replaced with the + character and any other no alphanumeric characters are replaced with a hexadecimal values. After the information is encoded it is sent to the server.

4.1.2-The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the**?** character.

Example:

1. The GET method produces a long string that appears in your server logs, in the browser's Location: box.
2. The GET method is restricted to send up to 1024 characters only.
3. Never use GET method if you have password or other sensitive information to be sent to the server.
4. GET can't be used to send binary data, like images or word documents, to the server.
5. The data sent by GET method can be accessed using QUERY_STRING environment variable.
6. The PHP provides *$_GET* associative array to access all the sent information using GET method.

4.1.3-The POST Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

1. The POST method does not have any restriction on data size to be sent.

2. The POST method can be used to send ASCII as well as binary data.

3. The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.

4. The PHP provides **$_POST** associative array to access all the sent information using GET method.

4.1.3-Cookies

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.

There are three steps involved in identifying returning users:

1. Server script sends a set of cookies to the browser. For example name, age, or identification number etc.

2. Browser stores this information on local machine for future use

3.When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

We will be tracing the entire request made by the client to the server from the demo site. All the GET-POST values sent & received from the client will be logged in to the Honey-pot database for analysis purpose.We will broadly use the G-P-S-C [GET POST SERVER COOKIES] data variables which are sent & received from the client to the server.

There are two ways the browser client can send information to the web server.

- The GET Method
- The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

Example: name1=value1&name2=value2&name3=value3

Spaces are removed and replaced with the + character and any other no alphanumeric characters are replaced with a hexadecimal values. After the information is encoded it is sent to the server.

The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.

Example: http://www.test.com/index.htm?name1=value1&name2=value2

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send up to 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides *$_GET* associative array to access all the sent information using GET method.

```
<?php

   if( $_GET["name"] || $_GET["age"] )

   {

         echo "Welcome ". $_GET['name']. "<br />"; echo "You
         are ". $_GET['age']. " years old."; exit();
```

```
    }

?>

<html>

<body>

    <form action="<?php $_PHP_SELF ?>" method="GET"> Name:
    <input type="text" name="name" />

    Age: <input type="text" name="age" /><input
    type="submit" />

    </form>

</body>

</html>
```

The POST Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using GET method.

```
<?php

    if( $_POST["name"] || $_POST["age"] )

    {

          echo "Welcome ". $_POST['name']. "<br />"; echo "You are
          ". $_POST['age']. " years old."; exit();

    }

?>

<html>

<body>

    <form action="<?php $_PHP_SELF ?>" method="POST">



    Name: <input type="text" name="name" />

    Age: <input type="text" name="age" />



    <input          type="submit"
    /></form>

</body>

</html>
```

The $_REQUEST variable

The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE. The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

```
<?php

   if( $_REQUEST["name"] || $_REQUEST["age"] )

   {

         echo "Welcome ". $_REQUEST['name']. "<br />"; echo "You are
         ". $_REQUEST['age']. " years old."; exit();

   }

?>

<html>

<body>

   <form action="<?php $_PHP_SELF ?>" method="POST">



   Name: <input type="text" name="name" /> Age:
   <input      type="text"      name="age"      /><input
   type="submit" />

   </form>

</body>

</html>
```

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.

There are three steps involved in identifying returning users:

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

Accessing Cookies with PHP:
PHP provides many ways to access cookies. Simplest way is to use either $_COOKIE or $HTTP_COOKIE_VARS variables. Following example will access all the cookies set in above example.

4.1.4-SQL Injection

SQL injection is a code injection technique that exploits a security vulnerabilityoccurring in the database layer of an application. The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed. It is an instance of a more general class of vulnerabilities that can occur whenever one programming or scripting language is embedded inside another. SQL injection attacks are also known as SQL insertion attacks.

SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker.

The primary form of SQL injection consists of direct insertion of code into user-input variables that are concatenated with SQL commands and executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed.

The injection process works by prematurely terminating a text string and appending a new command. Because the inserted command may have additional strings appended to it before it is executed, the malefactor terminates the injected string with a comment mark "--". Subsequent text is ignored at execution time.

The following script shows a simple SQL injection. The script builds an SQL query by concatenating hard-coded strings together with a string entered by the user:

1. **Select Query:**

A sample string that has been gathered from a normal user and a bad user trying to use SQL Injection. We asked the users for their login, which will be used to run a **SELECT** statement to get their information.

Code:

```
//   a  good  user's  name
$name = "timmy";

$query  =  "SELECT  *  FROM  customers  WHERE  username  =  '$name'";  echo
"Normal: “. $query . "<br />";

//   user input that uses SQL Injection
$name_bad = "' OR 1'";
```

```
// our MySQL query builder, however, not a very safe one
$query_bad = "SELECT * FROM customers WHERE username = '$name_bad'";

// display what the new query will look like, with injection echo "Injection: " .
$query_bad;
```

Screen output:

```
Normal: SELECT * FROM customers WHERE username = 'timmy' Injection: SELECT *
FROM customers WHERE username = '' OR 1''
```

Working of above example:

The normal query is no problem, as our MySQL statement will just select everything from customers that has a username equal to *timmy*.

However, the injection attack has actually made our query behave differently than weintended. By using a single quote (') they have ended the string part of our MySQL query

username = ' '

and then added on to our WHERE statement with an OR clause of 1 (always true).

username = ' ' **OR 1**

This OR clause of 1 will always be *true* and so **every single entry** in the "customers" table would be selected by this statement!

2. **Deletion of data:**

Although the above example displayed a situation where an attacker could possibly get access to a lot of information they shouldn't have, the attacks can be a lot worse. For example an attacker could empty out a table by executing a ***DELETE*** statement.

CODE:

```
$name_evil = "'; DELETE FROM customers WHERE 1 or username = '";

// our MySQL query builder really should check for injection $query_evil = "SELECT * FROM
customers WHERE username = '$name_evil'";

// the new evil injection query would include a DELETE statement echo "Injection: " .
$query_evil;



resultant query output:

SELECT * FROM customers WHERE username = ' '; DELETE FROM customers
WHERE 1 or username = ' '
```

If you were run this query,then the injected **DELETE** statement would completely empty your "customers" table.

3. **Insert & POST data:**

Assume that query is creating a new account. The user provides a desired username and an email address. The registration application generates a temporary password and emails it to the user to verify the email address.

Code:

```
<?php

$sql = "INSERT

INTO users
(reg_username, reg_password,
                         reg_email,


              VALUES              ('{$_POST['reg_username']}',
                       '$reg_password',


                         '{$_POST['reg_email']}')";
?>
```

This query is constructed with **$_POST**, which should immediately look suspicious.

User Input:

```
<?php


$sql = "INSERT


INTO      users      (reg_username,
                          reg_passwo
                          rd,
                          reg_email)


              VALUES ('bad_guy', 'mypass', ''), ('good_guy', '1234',


                          'shiflett@php.net')";
?>
```

Rather than the intended action of creating a single account (good_guy) with a valid email address, the application has been tricked into creating two accounts, and the user supplied every detail of the bad_guy account.

4. **<u>Alteration to Database Structure:</u>**

SQL uses the semicolon for statement termination, and if the input is not sanitized properly, there may be nothing that prevents us from stringing our own unrelated command at the end of the query.

The most drastic example if the Database is not **Read-Only** is:

```
SELECT email, passwd, login_id, full_name


   FROM members
WHERE email = 'x'; DROP TABLE members; --';
```

This one attempts to drop (delete) the entire **members** table

4.1.6-XSS ATTACKS

Cross-site scripting (**XSS**) is a type of computer security vulnerability typically found inweb applications that enables malicious attackers to inject client-side script into web pages viewed by other users. An exploited cross-site scripting vulnerability can be used by attackers to bypass access controls such as the same origin policy. Cross-site scripting carried out on websites were roughly 80% of all security vulnerabilities documented by Symantec as of 2007. Cross-site scripting attacks are therefore a special case of code injection.

The expression "cross-site scripting" originally referred to the act of loading the attacked, third-party web application from an unrelated attack site, in a manner that

executes a fragment of JavaScript prepared by the attacker in the security context of the targeted domain (a *reflected* or *non-persistent* XSS vulnerability). The definition gradually expanded to encompass other modes of code injection, including persistent and non-JavaScript vectors (including Java, ActiveX, VBScript, Flash, or even pure HTML), causing some confusion to newcomers to the field of information security.

Some prominent sites that have been affected in the past are the social networking sites Twitter, Face book, MySpace, and Orkut. The developers of Media-Wiki have fixed at least 26 XSS holes in order to protect Wikipedia and other wiki users.[7] In recent years, cross-site scripting flaws surpassed buffer overflows to become the most common publicly-reported security vulnerability, with some researchers claiming that as many as 68% of websites are likely open to XSS attacks.

There is no single, standardized classification of cross-site scripting flaws, but most experts distinguish between at least two primary flavors of XSS: *non-persistent* and *persistent*.

Non-persistent

The *non-persistent* (or *reflected*) cross-site scripting vulnerability is by far the most common type. These holes show up when the data provided by a web client, most commonly in HTTP query parameters or in HTML form submissions, is used immediately by server-side scripts to generate a page of results for that user, without properly sanitizing the response. This does not appear to be a serious problem: by submitting a malicious input to the web site, the user would only be able to compromise their own security context—that is, their own browser cookies, cache objects, and so forth. It is important to realize, however, that a third-party attacker may easily place hidden frames or deceptive links on unrelated sites and cause victims' browsers to navigate to URLs on the vulnerable site automatically—often completely in the background—and in such a case, the attacker can intrude into the security context that rightfully belonged to the victim.

Persistent

The *persistent* (or *stored*) XSS vulnerability is a more devastating variant of a cross-site scripting flaw: it occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping. A classic example of this iswith online message boards where users are allowed to post HTML formatted messages for other users to read.

4.1.7-DIRECTORY CHANGE ATTACK

A **directory traversal** (or **path traversal**) consists in exploiting insufficient security validation / sanitization of user-supplied input file names, so that characters representing "traverse to parent directory" are passed through to the file APIs.

The goal of this attack is to order an application to access a computer file that is not intended to be accessible. This attack exploits a lack of security (the software is acting exactly as it is supposed to) as opposed to exploiting a bug in the code.

Directory traversal is Also Known as the ../ (dot dot slash) attack, directory climbing, and backtracking. Some forms of this attack are also canonicalization attacks.

A typical example of vulnerable application in PHP code is:

```
<?php

$template = 'red.php';

if    (isset($_COOKIE['TEMPLATE']))
       $template = $_COOKIE['TEMPLATE'];

include ("/home/users/phpguru/templates/" . $template);

?>
```

An attack against this system could be to send the following HTTP request:

```
GET /vulnerable.php HTTP/1.0

Cookie: TEMPLATE=../../../../../../../../../etc/password
```

Generating a server response such as:

```
HTTP/1.0 200 OK

Content-Type: text/html

Server: Apache


root:fi3sED95ibqR6:0:1:System                    Operator:/:/bin/ksh
daemon:*:1:1::/tmp:

phpguru:f8fk3j1OIf31.:182:100:Developer:/home/users/phpguru/:/bin/csh
```

The repeated ../ characters after /home/users/phpguru/templates/ has caused include() to traverse to the root directory, and then include the Unix password file /etc/password.

Unix /etc/password is a common file used to demonstrate **directory traversal**, as it is often used by crackers to try cracking the passwords.

However, in more recent Unix systems, the password file does not contain the hashed passwords. They are, instead, located in the shadow file which cannot be read by unprivileged users on the machine. It is however, still useful for account enumeration on the machine, as it still displays the user accounts on the system.

4.2 -TEST CASES

System testing is a critical phase implementation. Testing of the system involves hardware devise and debugging of the computer programs and testing information processing procedures. Testing can be done with text data, which attempts to stimulate all possible conditions that may arise during processing. If structured programming Methodologies have been adopted during coding the testing proceeds from higher level to lower level of program module until the entire program is tested as unit. The testing methods adopted during the testing of the system were unit testing and integrated testing.

4.2.1 Unit testing:-

Unit testing focuses first on the modules, independently of one another, to locate errors. This enables the tester to detect errors in coding and logical errors that is contained within that module alone. Those resulting from the interaction between modules are initially avoided.

4.2.2 Integration testing:-

Integration testing is a systematic technique for constructing the program structure while at the same time to uncover the errors associated with interfacing. The objective is to take unit-tested module and build a program structure that has been detected by designing. It also tests to find the discrepancies between the system and its original objectives. Subordinate stubs are replaced one at time actual module. Tests were conducted at each module was integrated. On completion of each set another stub was replaced with the real module.

The following are the test cases in Honeypots

4.2.3 Test Objective: To test the Login Module for the application

Item No	**Test Condition**	**Operator Action**	**Input Specification**	**Output Specification (Expected Results)**	**Pass Or Fail**
1	Successful Insertion Of Tracking script	Copying the Script in the End File That Needs To be traced	Script code	If the scriptis successfully copied, the file should execute properly	Pass

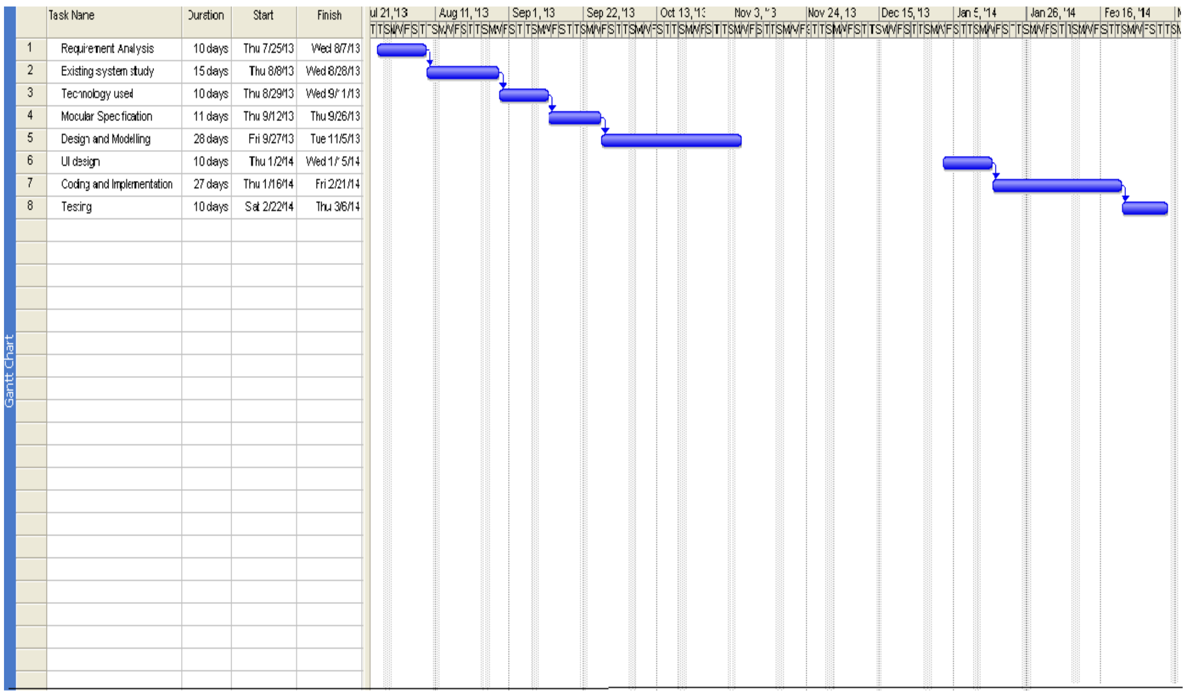
2	Successful tracing of The client Web page	Copying the Script in the End file That Needs To be traced. Clayin g The Webpage URL	Surfing through the Page	Successful tracing of data to the Honeypot server	Pass
3	Successful analyzing The results	If the Data is Tracked properly in to The Honeypot database.	Request Analysis / overview	Show the results of various attacks / non attacks found. Determining the type of attacks.	Pass
4	Successful generation Of analysis report	Analysis report for all the Attacks Between Certain period	Select Analysis / Statistics & provide from & to date	Successfully generate the report if any records exist or show No records found.	Pass

Table 4.2 Test Case

4.3- PROJECT TIMELINE AND TASK DISTRIBUTION

The Gantt Chart Shows planned and actual progress for a number of tasks displayed against a horizontal time scale.

It is effective and easy-to-read method of indicating the actual current status for each of set of tasks compared to planned progress for each activity of the set. Gantt Charts provide a clear picture of the current state of the project

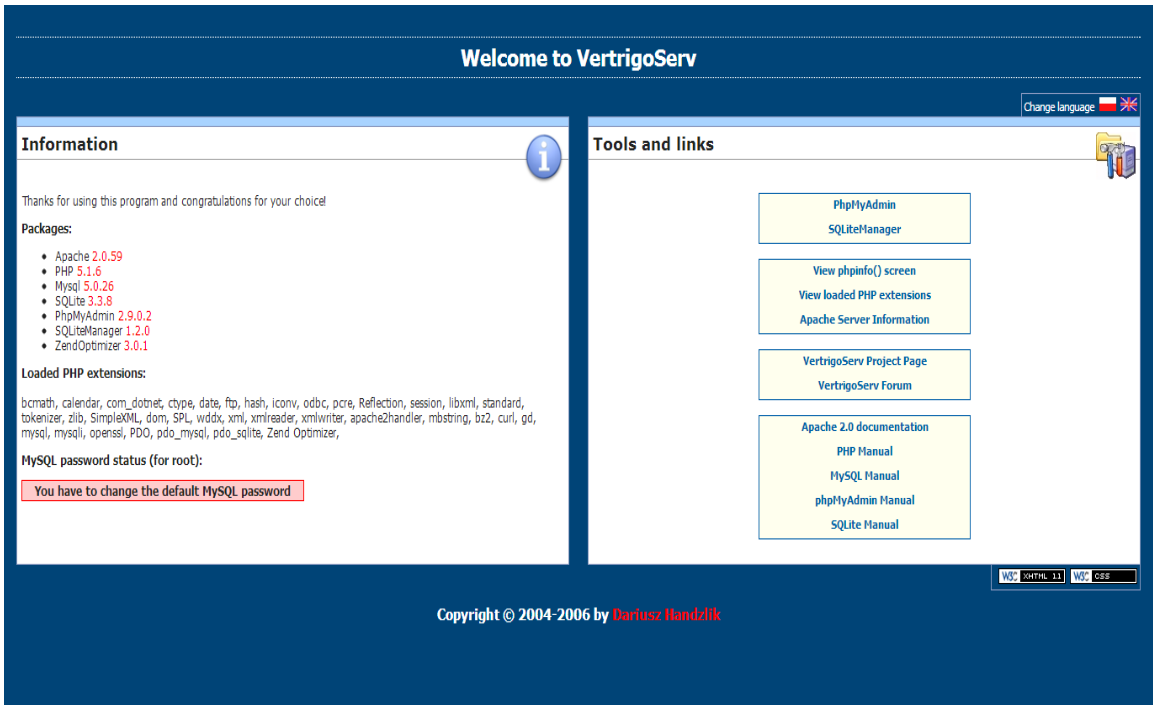


	Task Name	Duration	Start	Finish
1	Requirement Analysis	10 days	Thu 7/25/13	Wed 8/7/13
2	Existing system study	15 days	Thu 8/8/13	Wed 8/28/13
3	Technology used	10 days	Thu 8/29/13	Wed 9/11/13
4	Mocular Specification	11 days	Thu 9/12/13	Thu 9/26/13
5	Design and Modelling	28 days	Fri 9/27/13	Tue 11/5/13
6	UI design	10 days	Thu 1/2/14	Wed 1/15/14
7	Coding and Implementation	27 days	Thu 1/16/14	Fri 2/21/14
8	Testing	10 days	Sat 2/22/14	Thu 3/6/14

Fig 4.3 Gantt Chart

.

5 - RESULTS



Welcome to VertrigoServ

Change language

Information

Thanks for using this program and congratulations for your choice!

Packages:

- Apache 2.0.59
- PHP 5.1.6
- Mysql 5.0.26
- SQLite 3.3.8
- PhpMyAdmin 2.9.0.2
- SQLiteManager 1.2.0
- ZendOptimizer 3.0.1

Loaded PHP extensions:

bcmath, calendar, com_dotnet, ctype, date, ftp, hash, iconv, odbc, pcre, Reflection, session, libxml, standard, tokenizer, zlib, SimpleXML, dom, SPL, wddx, xml, xmlreader, xmlwriter, apache2handler, mbstring, bz2, curl, gd, mysql, mysqli, openssl, PDO, pdo_mysql, pdo_sqlite, Zend Optimizer,

MySQL password status (for root):

You have to change the default MySQL password

Tools and links

PhpMyAdmin
SQLiteManager

View phpinfo() screen
View loaded PHP extensions
Apache Server Information

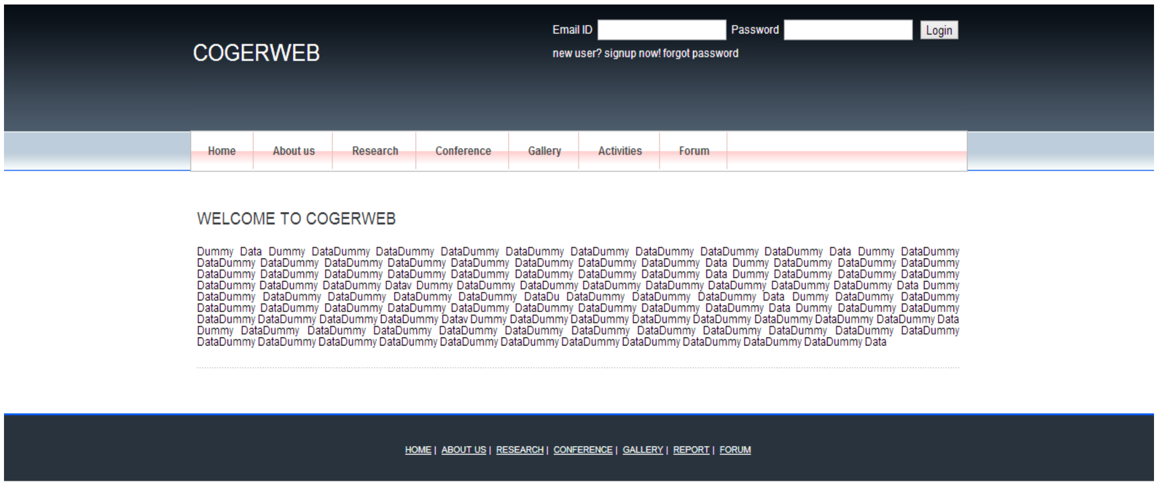
VertrigoServ Project Page
VertrigoServ Forum

Apache 2.0 documentation
PHP Manual
MySQL Manual
phpMyAdmin Manual
SQLite Manual

Copyright © 2004-2006 by Dariusz Handzlik

Fig 5.1 Vertrigo server

The vertrigo server 2.13 is first downloaded from the internet and installed in the computer and honeyweb database is stored in the database section.



COGERWEB

Email ID Password Login

new user? signup now! forgot password

WELCOME TO COGERWEB

Dummy Data Dummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy Data Dummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy Data Dummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy Data Dummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy Datav Dummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy Data Dummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDu DataDummy DataDummy DataDummy Data Dummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy Data Dummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy Datav Dummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy DataDummy Data Dummy Data

HOME | ABOUT US | RESEARCH | CONFERENCE | GALLERY | REPORT | FORUM

Fig 5.2 Cogerweb login website

We login using the Email id and password in the Cogerweb website which shows the result of the attack in the database.

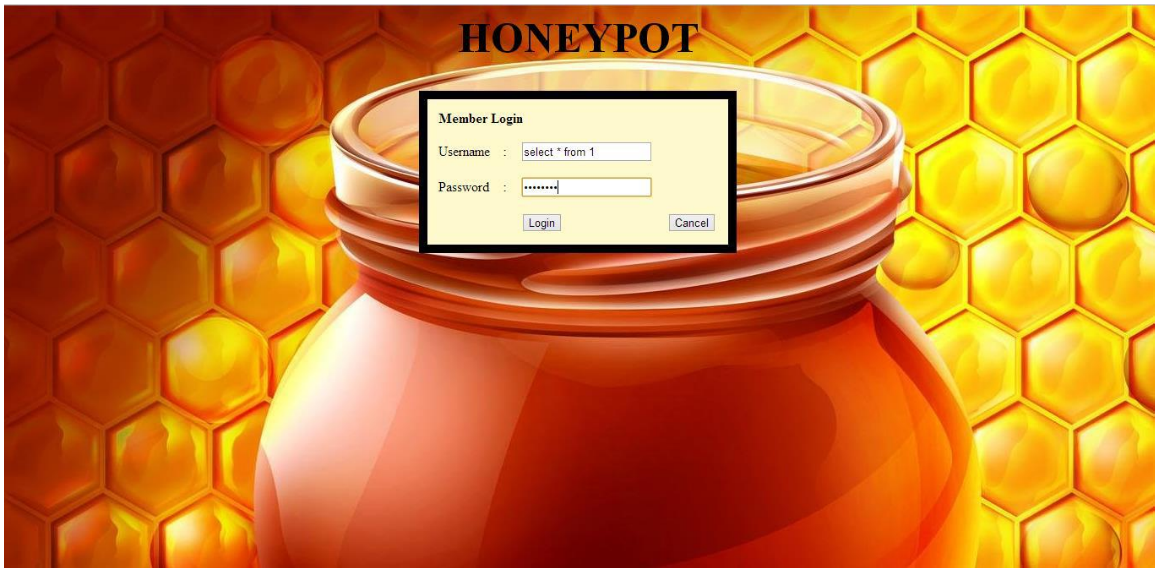
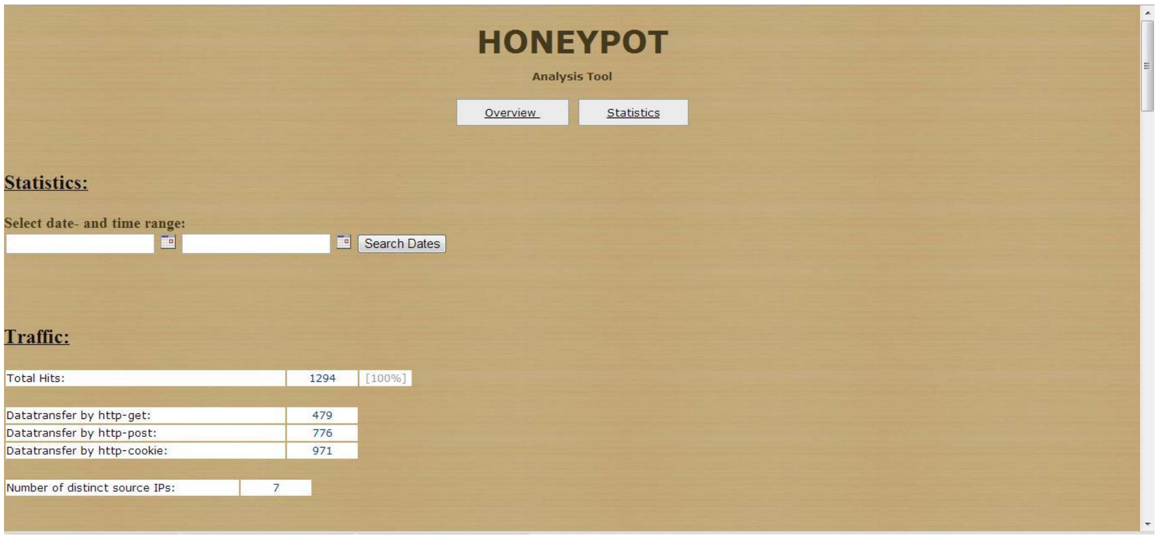


Fig 5.3 Honeypot login website

Fig 5.4 Honeypot website page

Another website honeypot is created. We enter the login and password and open the website page. If the attacker tries to open the website, the honeypot warns the user about the attack and updates the database.


HONEYPOT

Analysis Tool

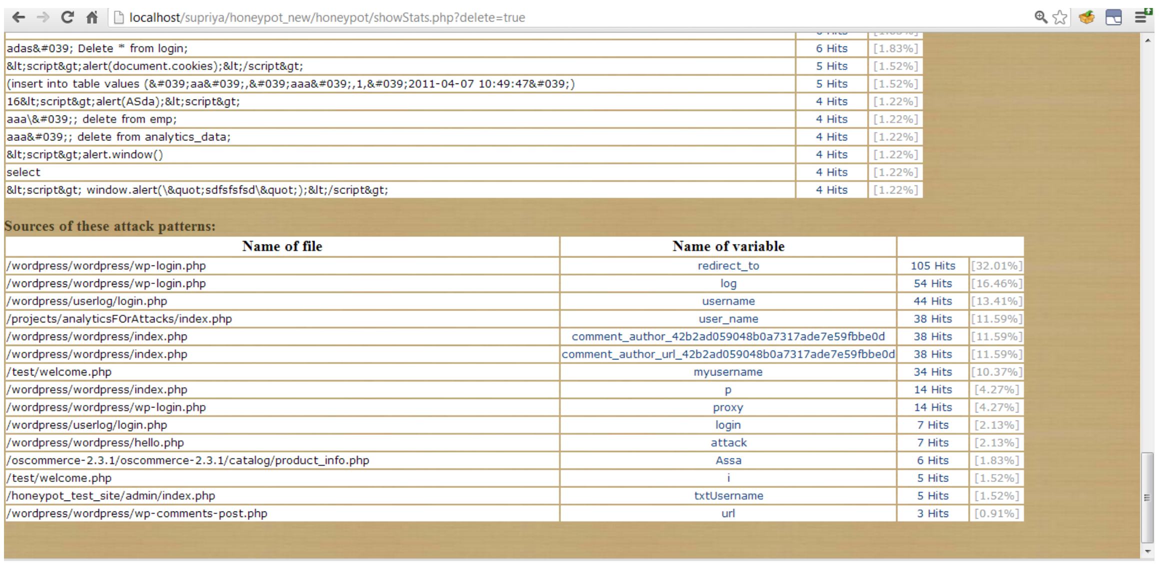
Statistics:

Select date- and time range:

Traffic:

Total Hits:	1294	[100%]
Datatransfer by http-get:	479	
Datatransfer by http-post:	776	
Datatransfer by http-cookie:	971	
Number of distinct source IPs:	7	

Fig 5.5 Honeypot Database statistics 1


localhost/supriya/honeypot_new/honeypot/showStats.php?delete=true

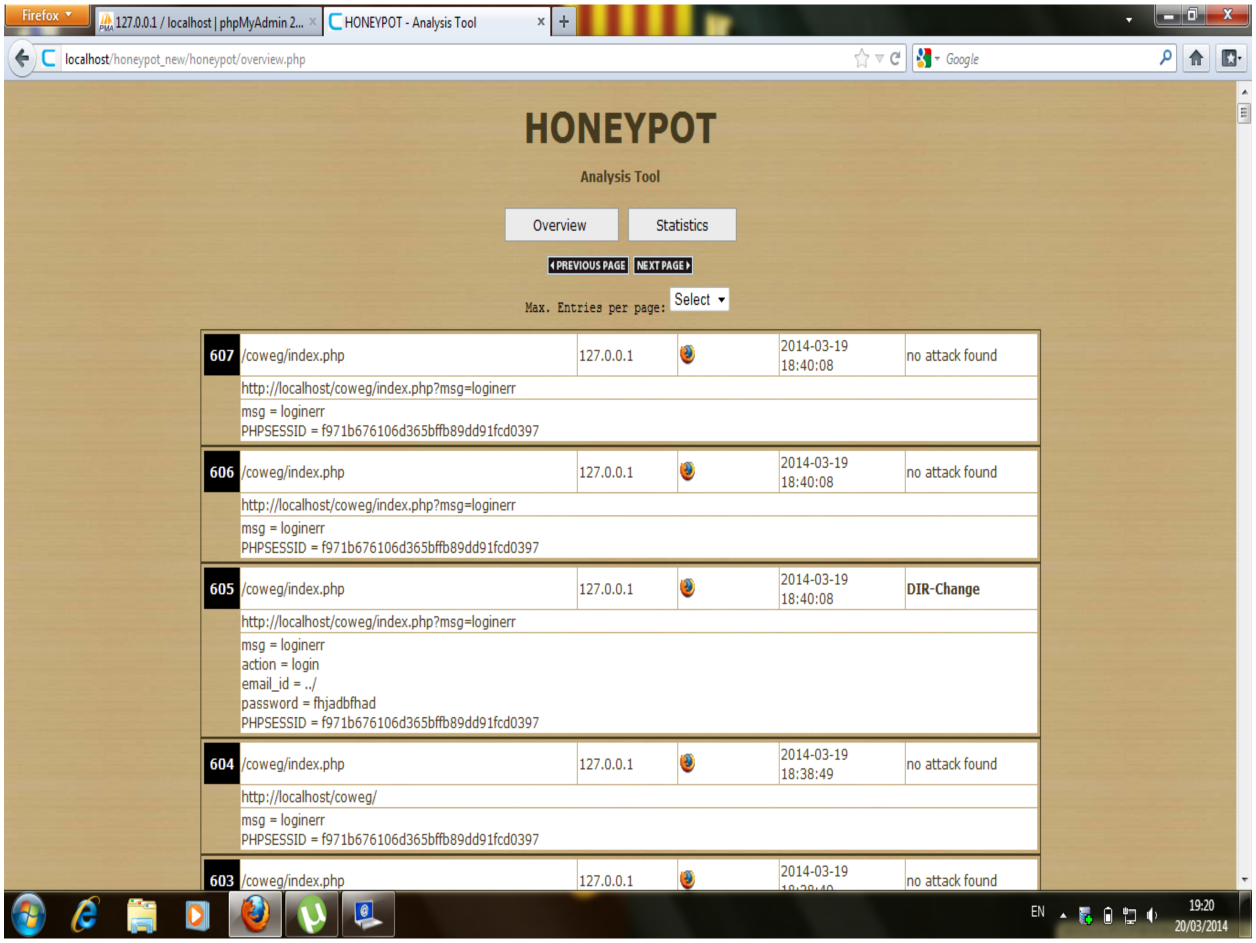
adas&#039; Delete * from login;	6 Hits	[1.83%]
<script>alert(document.cookies);</script>	5 Hits	[1.52%]
(insert into table values (&#039;aa&#039;,&#039;aaa&#039;,1,&#039;2011-04-07 10:49:47&#039;)	5 Hits	[1.52%]
16<script>alert(ASda);<script>	4 Hits	[1.22%]
aaa\&#039;; delete from emp;	4 Hits	[1.22%]
aaa&#039;; delete from analytics_data;	4 Hits	[1.22%]
<script>alert.window()	4 Hits	[1.22%]
select	4 Hits	[1.22%]
<script> window.alert(\&quot;sdfsfsfsd\&quot;);</script>	4 Hits	[1.22%]

Sources of these attack patterns:

Name of file	Name of variable		
/wordpress/wordpress/wp-login.php	redirect_to	105 Hits	[32.01%]
/wordpress/wordpress/wp-login.php	log	54 Hits	[16.46%]
/wordpress/userlog/login.php	username	44 Hits	[13.41%]
/projects/analyticsFOrAttacks/index.php	user_name	38 Hits	[11.59%]
/wordpress/wordpress/index.php	comment_author_42b2ad059048b0a7317ade7e59fbbe0d	38 Hits	[11.59%]
/wordpress/wordpress/index.php	comment_author_url_42b2ad059048b0a7317ade7e59fbbe0d	38 Hits	[11.59%]
/test/welcome.php	myusername	34 Hits	[10.37%]
/wordpress/wordpress/index.php	p	14 Hits	[4.27%]
/wordpress/wordpress/wp-login.php	proxy	14 Hits	[4.27%]
/wordpress/userlog/login.php	login	7 Hits	[2.13%]
/wordpress/wordpress/hello.php	attack	7 Hits	[2.13%]
/oscommerce-2.3.1/oscommerce-2.3.1/catalog/product_info.php	Assa	6 Hits	[1.83%]
/test/welcome.php	i	5 Hits	[1.52%]
/honeypot_test_site/admin/index.php	txtUsername	5 Hits	[1.52%]
/wordpress/wordpress/wp-comments-post.php	url	3 Hits	[0.91%]

Fig 5.6 Honeypot Database Statistics 2

The Statistic page gives the details about the honeypot database the number of attacks and the overview of the database.



HONEYPOT

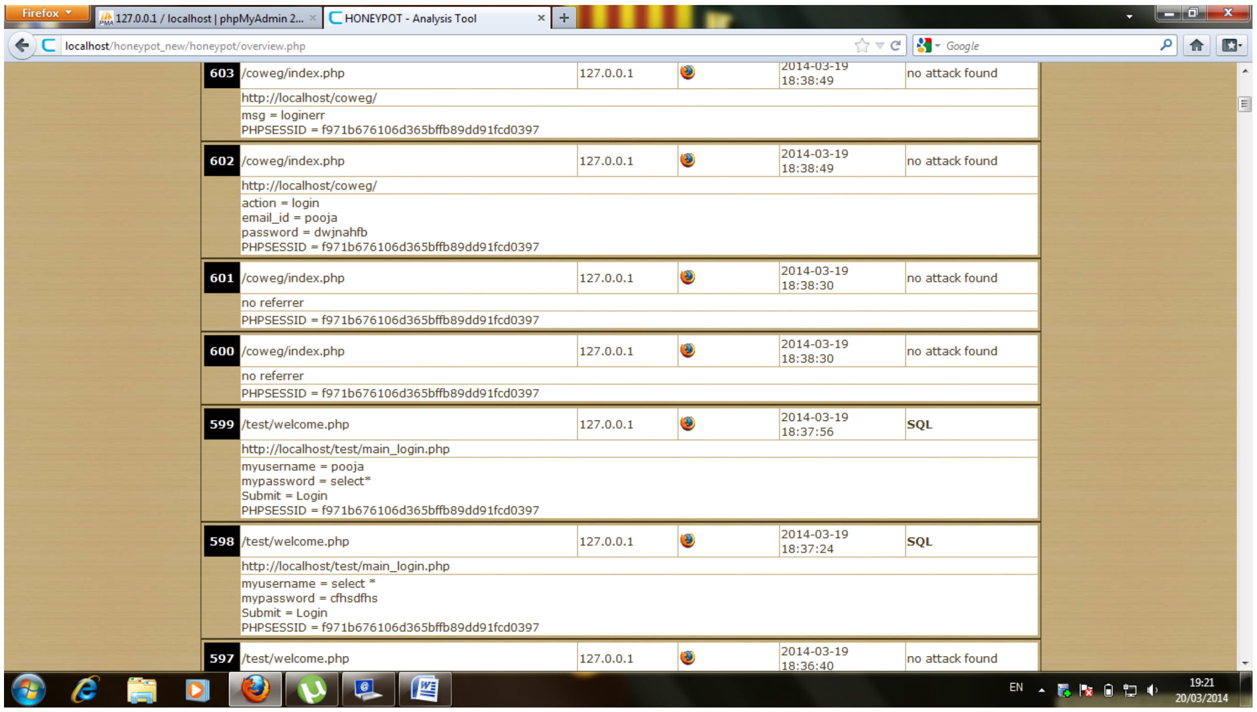
Analysis Tool

Max. Entries per page: Select

607	/coweg/index.php	127.0.0.1		2014-03-19 18:40:08	no attack found
	http://localhost/coweg/index.php?msg=loginerr				
	msg = loginerr PHPSESSID = f971b676106d365bffb89dd91fcd0397				
606	/coweg/index.php	127.0.0.1		2014-03-19 18:40:08	no attack found
	http://localhost/coweg/index.php?msg=loginerr				
	msg = loginerr PHPSESSID = f971b676106d365bffb89dd91fcd0397				
605	/coweg/index.php	127.0.0.1		2014-03-19 18:40:08	**DIR-Change**
	http://localhost/coweg/index.php?msg=loginerr				
	msg = loginerr action = login email_id = ../ password = fhjadbfhad PHPSESSID = f971b676106d365bffb89dd91fcd0397				
604	/coweg/index.php	127.0.0.1		2014-03-19 18:38:49	no attack found
	http://localhost/coweg/				
	msg = loginerr PHPSESSID = f971b676106d365bffb89dd91fcd0397				
603	/coweg/index.php	127.0.0.1		2014-03-19	no attack found

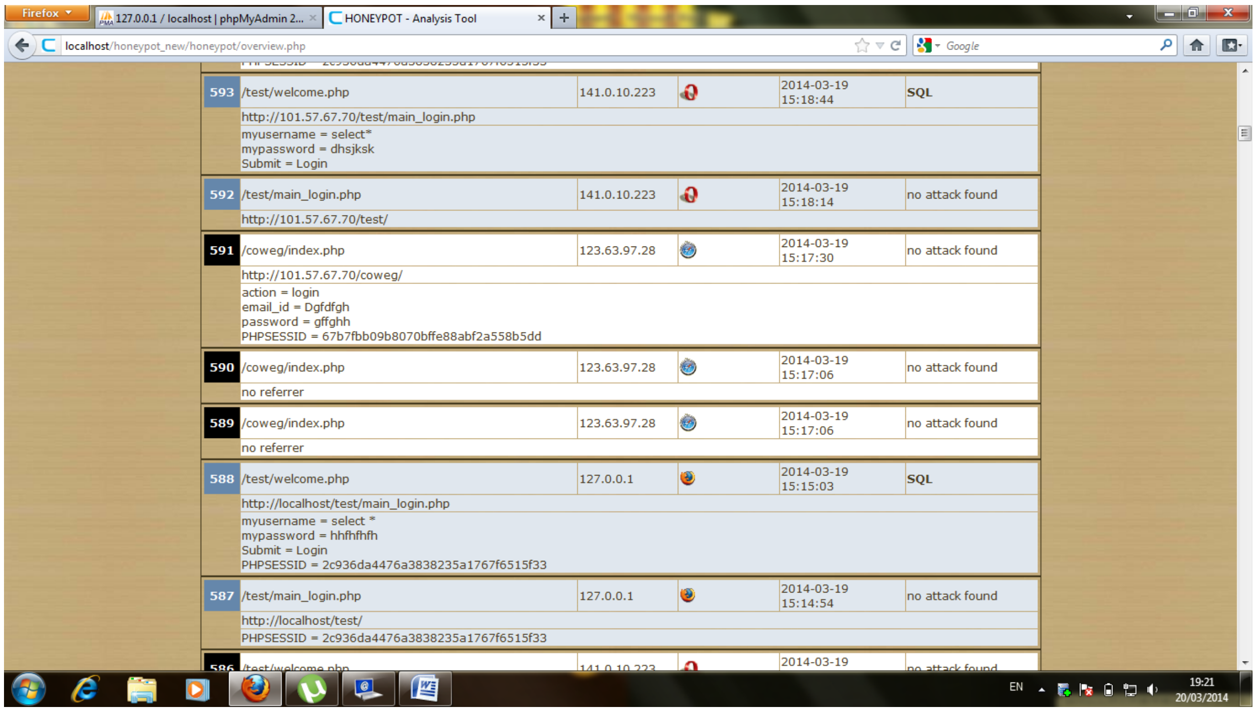
Fig 5.7 Honeypot Database Overview 1

The overview page gives the detail of every attack that takes place. It gives user the knowledge from which website the attack has taken place and at what time. It even tells the user which type of attack has taken place on the website.


Firefox
127.0.0.1 / localhost | phpMyAdmin 2...
HONEYPOT - Analysis Tool
localhost/honeypot_new/honeypot/overview.php
Google
EN
19:21
20/03/2014

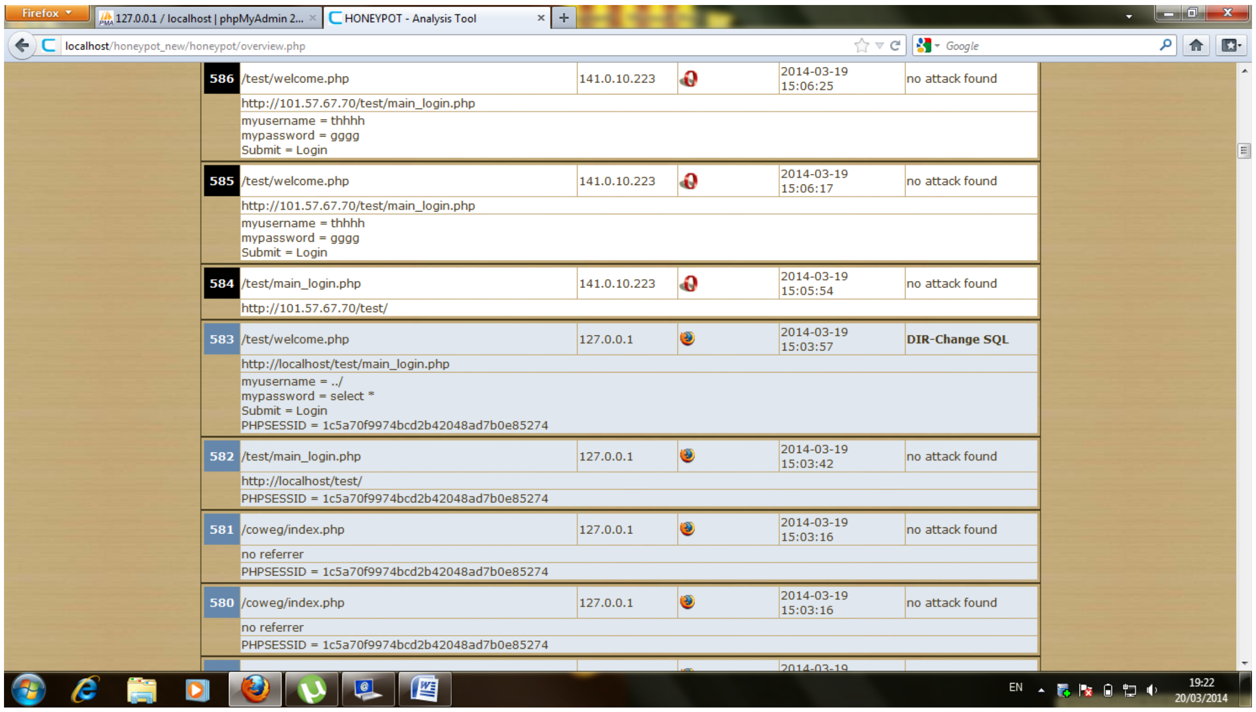
603	/coweg/index.php	127.0.0.1	●	2014-03-19 18:38:49	no attack found
	http://localhost/coweg/ msg = loginerr PHPSESSID = f971b676106d365bffb89dd91fcd0397				
602	/coweg/index.php	127.0.0.1	●	2014-03-19 18:38:49	no attack found
	http://localhost/coweg/ action = login email_id = pooja password = dwjnahfb PHPSESSID = f971b676106d365bffb89dd91fcd0397				
601	/coweg/index.php	127.0.0.1	●	2014-03-19 18:38:30	no attack found
	no referrer PHPSESSID = f971b676106d365bffb89dd91fcd0397				
600	/coweg/index.php	127.0.0.1	●	2014-03-19 18:38:30	no attack found
	no referrer PHPSESSID = f971b676106d365bffb89dd91fcd0397				
599	/test/welcome.php	127.0.0.1	●	2014-03-19 18:37:56	**SQL**
	http://localhost/test/main_login.php myusername = pooja mypassword = select* Submit = Login PHPSESSID = f971b676106d365bffb89dd91fcd0397				
598	/test/welcome.php	127.0.0.1	●	2014-03-19 18:37:24	**SQL**
	http://localhost/test/main_login.php myusername = select * mypassword = cfhsdfhs Submit = Login PHPSESSID = f971b676106d365bffb89dd91fcd0397				
597	/test/welcome.php	127.0.0.1	●	2014-03-19 18:36:40	no attack found

Fig 5.8 Honeypot Database Overview 2


Firefox
127.0.0.1 / localhost | phpMyAdmin 2...
HONEYPOT - Analysis Tool
localhost/honeypot_new/honeypot/overview.php
Google
EN
19:21
20/03/2014

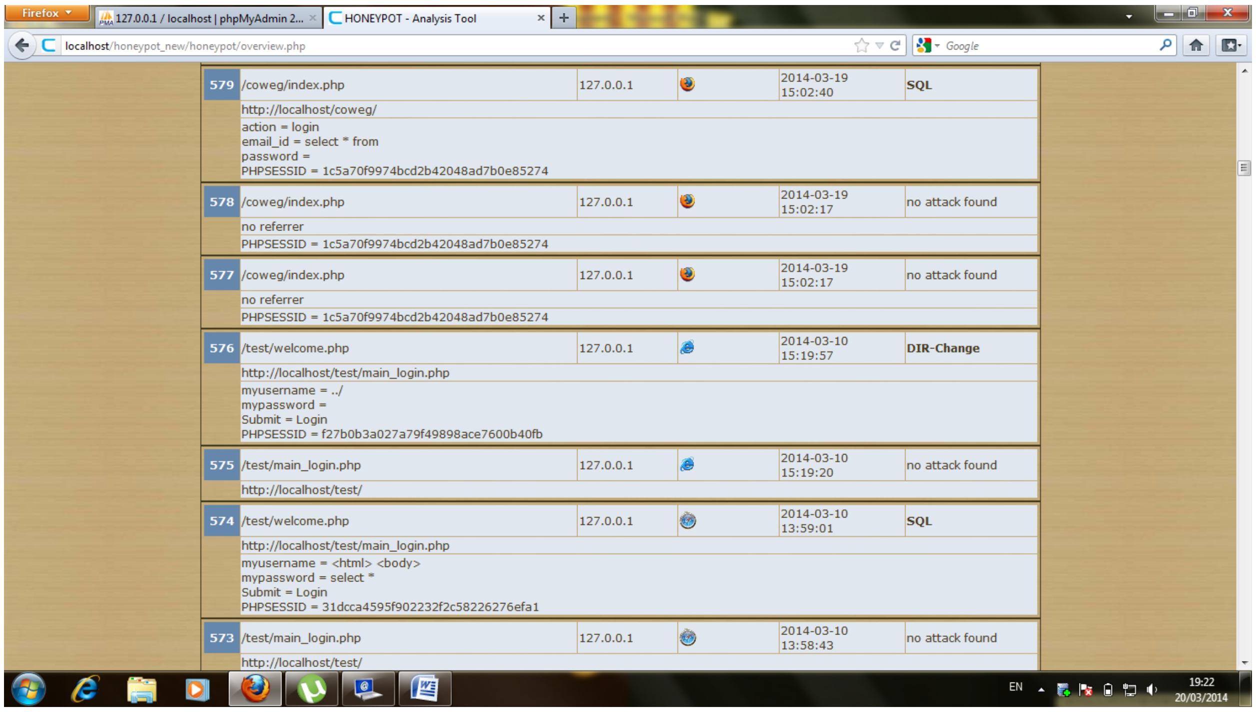
593	/test/welcome.php	141.0.10.223	●	2014-03-19 15:18:44	**SQL**
	http://101.57.67.70/test/main_login.php myusername = select* mypassword = dhsjksk Submit = Login				
592	/test/main_login.php	141.0.10.223	●	2014-03-19 15:18:14	no attack found
	http://101.57.67.70/test/				
591	/coweg/index.php	123.63.97.28	●	2014-03-19 15:17:30	no attack found
	http://101.57.67.70/coweg/ action = login email_id = Dgfdfgh password = gffghh PHPSESSID = 67b7fbb09b8070bffe88abf2a558b5dd				
590	/coweg/index.php	123.63.97.28	●	2014-03-19 15:17:06	no attack found
	no referrer				
589	/coweg/index.php	123.63.97.28	●	2014-03-19 15:17:06	no attack found
	no referrer				
588	/test/welcome.php	127.0.0.1	●	2014-03-19 15:15:03	**SQL**
	http://localhost/test/main_login.php myusername = select * mypassword = hhfhfhfh Submit = Login PHPSESSID = 2c936da4476a3838235a1767f6515f33				
587	/test/main_login.php	127.0.0.1	●	2014-03-19 15:14:54	no attack found
	http://localhost/test/ PHPSESSID = 2c936da4476a3838235a1767f6515f33				
586	/test/welcome.php	141.0.10.223	●	2014-03-19	no attack found

Fig 5.9 Honeypot Database Overview 3


127.0.0.1 / localhost | phpMyAdmin 2...
HONEYPOT - Analysis Tool
localhost/honeypot_new/honeypot/overview.php

586	/test/welcome.php	141.0.10.223		2014-03-19 15:06:25	no attack found
	http://101.57.67.70/test/main_login.php myusername = thhhh mypassword = gggg Submit = Login				
585	/test/welcome.php	141.0.10.223		2014-03-19 15:06:17	no attack found
	http://101.57.67.70/test/main_login.php myusername = thhhh mypassword = gggg Submit = Login				
584	/test/main_login.php	141.0.10.223		2014-03-19 15:05:54	no attack found
	http://101.57.67.70/test/				
583	/test/welcome.php	127.0.0.1		2014-03-19 15:03:57	**DIR-Change SQL**
	http://localhost/test/main_login.php myusername = ../ mypassword = select * Submit = Login PHPSESSID = 1c5a70f9974bcd2b42048ad7b0e85274				
582	/test/main_login.php	127.0.0.1		2014-03-19 15:03:42	no attack found
	http://localhost/test/ PHPSESSID = 1c5a70f9974bcd2b42048ad7b0e85274				
581	/coweg/index.php	127.0.0.1		2014-03-19 15:03:16	no attack found
	no referrer PHPSESSID = 1c5a70f9974bcd2b42048ad7b0e85274				
580	/coweg/index.php	127.0.0.1		2014-03-19 15:03:16	no attack found
	no referrer PHPSESSID = 1c5a70f9974bcd2b42048ad7b0e85274				

Fig 5.10 Honeypot Database Overview 4


127.0.0.1 / localhost | phpMyAdmin 2...
HONEYPOT - Analysis Tool
localhost/honeypot_new/honeypot/overview.php

579	/coweg/index.php	127.0.0.1		2014-03-19 15:02:40	**SQL**
	http://localhost/coweg/ action = login email_id = select * from password = PHPSESSID = 1c5a70f9974bcd2b42048ad7b0e85274				
578	/coweg/index.php	127.0.0.1		2014-03-19 15:02:17	no attack found
	no referrer PHPSESSID = 1c5a70f9974bcd2b42048ad7b0e85274				
577	/coweg/index.php	127.0.0.1		2014-03-19 15:02:17	no attack found
	no referrer PHPSESSID = 1c5a70f9974bcd2b42048ad7b0e85274				
576	/test/welcome.php	127.0.0.1		2014-03-10 15:19:57	**DIR-Change**
	http://localhost/test/main_login.php myusername = ../ mypassword = Submit = Login PHPSESSID = f27b0b3a027a79f49898ace7600b40fb				
575	/test/main_login.php	127.0.0.1		2014-03-10 15:19:20	no attack found
	http://localhost/test/				
574	/test/welcome.php	127.0.0.1		2014-03-10 13:59:01	**SQL**
	http://localhost/test/main_login.php myusername = <html> <body> mypassword = select * Submit = Login PHPSESSID = 31dcca4595f902232f2c58226276efa1				
573	/test/main_login.php	127.0.0.1		2014-03-10 13:58:43	no attack found
	http://localhost/test/				

Fig 5.11 Honeypot Database Overview 5

6- CONCLUSION

Finally the project is concluded with following features

- Automatically scans for known attacks.
- Detects SLQ-Injections, (Remote) File-Inclusions, Cross-Site Scripting (XSS), Download attempts for malicious files e.g. with WGET or CURL, Command-Injections, etc.
- Provides an overview mode which allows you to look and scan for new incidents quickly (semi-automatic mode).
- Supports detailed information about all data correlated with every access to the honeypot. This includes but is not limited to HTTP-GET, HTTP-POST and COOKIE data.
- Saves copies of malicious tools in a secured place for later analysis.
- Provides a geographical, IP-based mapping about the attack sources. The generated map shows the origin of the attacks and offers additional details for each location.
- Generates numerous statistics about all traffic recognized at the system.
- Supports Android as well as Apple Technology also.

Assumptions of project:

- We assume that our system will better work with PHP based web applications. So we conclude our host Tests are performed on PHP sites.
- We assume the client side scripting language on the Host sites is JavaScript.
- Injecting scripts and other features on a client website would require Folder Access and FTP rights.

7 - FUTURE WORK

Our project can be enhanced with many features in future. Let us have a look at the enhancements that can be done in future.

In addition, we plan to enrich our system by providing it with more powerful algorithms that try to predict the future interests of a user, and by designing and implementing security modules for supporting companies looking for candidates.

Finally, we plan to study the possibility to integrate our system with a latest framework for realizing a system capable of blocking an attacker so that web-site can be more secure.

REFERENCES

[1] - “Honeypot available today”. [*Online*]. Available at:-
http://www.cs.wustl.edu/~jain/cse571-09/ftp/honey/ [29 july2013].

[2] - Lance Spitzner“Honeypot applications and reviews” Department of Computer Science University of California, 2002.

[3]“Honeypot decoy”.**[***Online***].**Available at:-
http://en.wikipedia.org/wiki/Honeypot_(computing)[24 August 2013].

APPENDIX

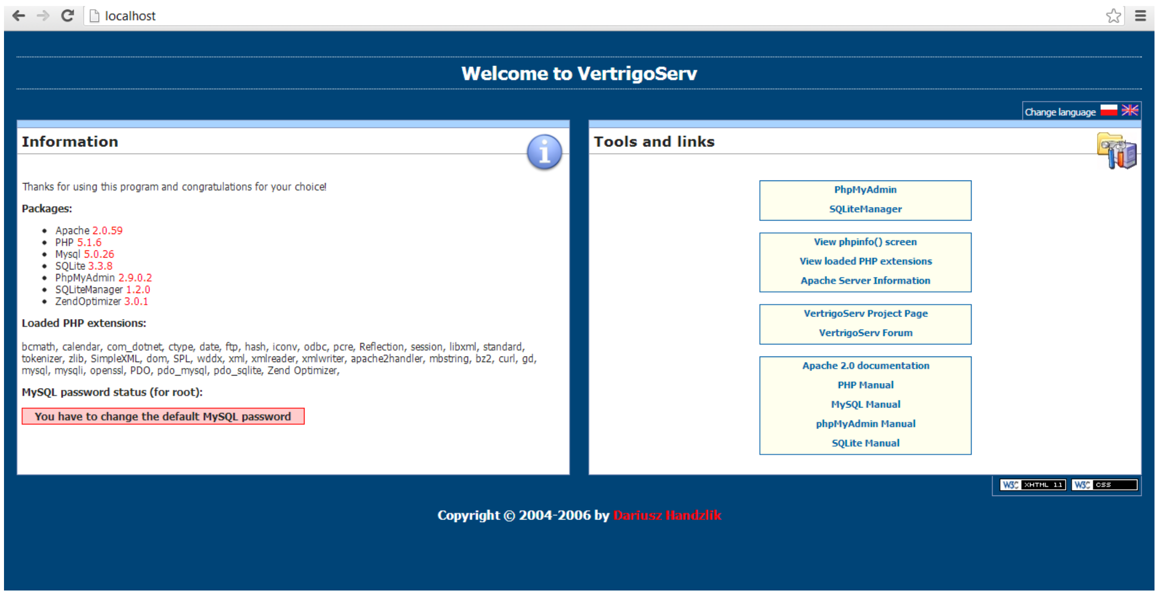
STEP 1: INSTALLATION OF VERTRIGO SERVER:

Install the vertrigo server with the file name as "vertrserv_213.exe"

Save all downloaded content to C drive (System drive)

STEP 2: RUN THE SERVER:

To run the server go to any browser, and type Localhost in address bar of browser. The server will look like an entire web-page with few useful links



Welcome to VertrigoServ

Change language

Information

Thanks for using this program and congratulations for your choice!

Packages:

- Apache 2.0.59
- PHP 5.1.6
- Mysql 5.0.26
- SQLite 3.3.8
- PhpMyAdmin 2.9.0.2
- SQLiteManager 1.2.0
- ZendOptimizer 3.0.1

Loaded PHP extensions:

bcmath, calendar, com_dotnet, ctype, date, ftp, hash, iconv, odbc, pcre, Reflection, session, libxml, standard, tokenizer, zlib, SimpleXML, dom, SPL, wddx, xml, xmlreader, xmlwriter, apache2handler, mbstring, bz2, curl, gd, mysql, mysqli, openssl, PDO, pdo_mysql, pdo_sqlite, Zend Optimizer,

MySQL password status (for root):

You have to change the default MySQL password

Tools and links

PhpMyAdmin
SQLiteManager

View phpinfo() screen
View loaded PHP extensions
Apache Server Information

VertrigoServ Project Page
VertrigoServ Forum

Apache 2.0 documentation
PHP Manual
MySQL Manual
phpMyAdmin Manual
SQLite Manual

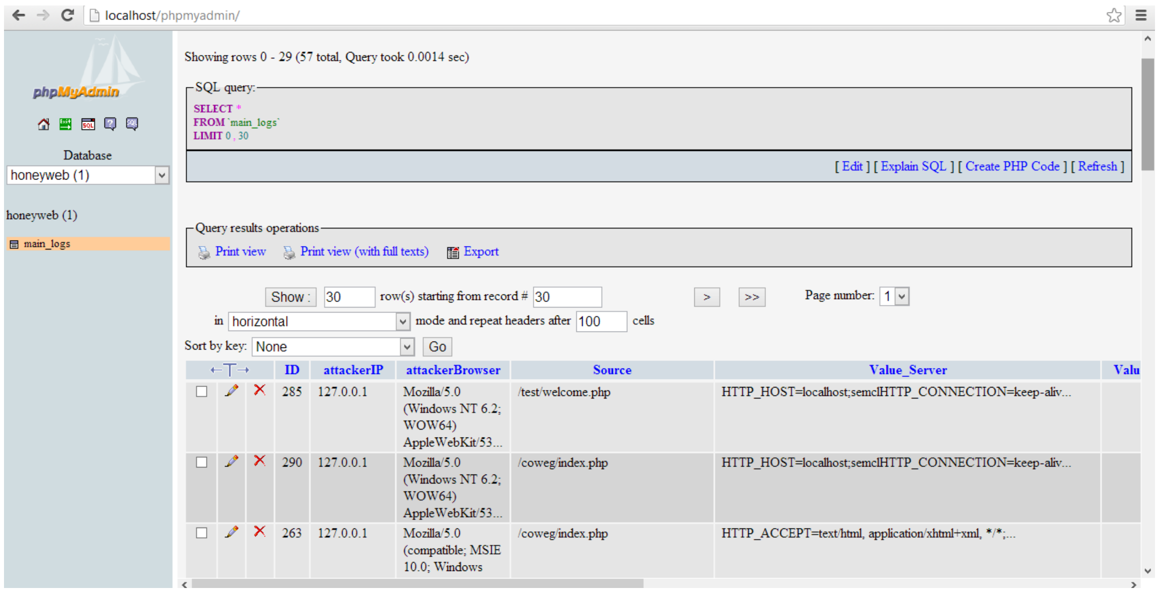
W3C XHTML 1.1 W3C CSS

Copyright © 2004-2006 by Dariusz Handzlik

STEP 3: ACESS TO DATABASE OF MYSQL:

- To view the actual database, user must click on PhpMYadmin link
- The dialogue box which contains user name and password field should be entered with user name as "root" and password as "vertrigo".
- Select appropriate tables from database and click on view button

Entire database is visible to user. You can also delete the records, change the database structure of your database.



localhost/phpmyadmin/

Database: honeyweb (1)

honeyweb (1)

main_logs

Showing rows 0 - 29 (57 total, Query took 0.0014 sec)

SQL query:

```
SELECT *
FROM `main_logs`
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

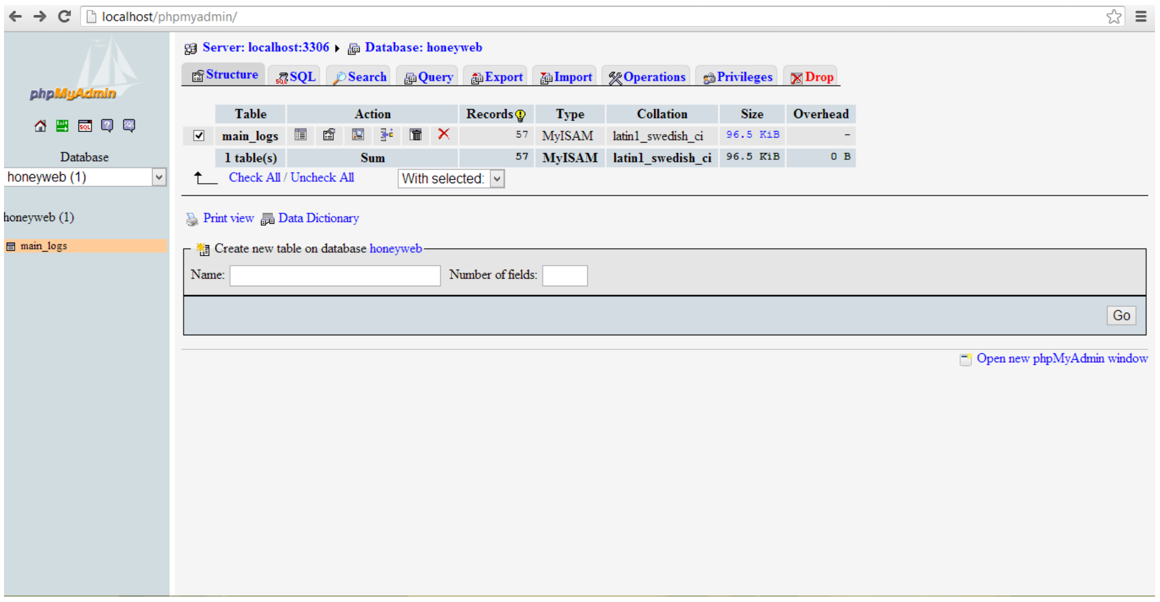
Query results operations: Print view · Print view (with full texts) · Export

Show : 30 row(s) starting from record # 30 · Page number: 1

in horizontal mode and repeat headers after 100 cells

Sort by key: None · Go

			ID	attackerIP	attackerBrowser	Source	Value_Server	Valu
☐	✎	✗	285	127.0.0.1	Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/53...	/test/welcome.php	HTTP_HOST=localhost;semclHTTP_CONNECTION=keep-aliv...	
☐	✎	✗	290	127.0.0.1	Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/53...	/coweg/index.php	HTTP_HOST=localhost;semclHTTP_CONNECTION=keep-aliv...	
☐	✎	✗	263	127.0.0.1	Mozilla/5.0 (compatible; MSIE 10.0; Windows	/coweg/index.php	HTTP_ACCEPT=text/html, application/xhtml+xml, */*;...	



localhost/phpmyadmin/

Database: honeyweb (1)

honeyweb (1)

main_logs

Server: localhost:3306 ▸ Database: honeyweb

Structure · SQL · Search · Query · Export · Import · Operations · Privileges · Drop

	Table	Action	Records	Type	Collation	Size	Overhead
☑	main_logs	✗	57	MyISAM	latin1_swedish_ci	96.5 KiB	-
	1 table(s)	Sum	57	MyISAM	latin1_swedish_ci	96.5 KiB	0 B

Check All / Uncheck All · With selected:

Print view · Data Dictionary

Create new table on database honeyweb

Name: · Number of fields: · Go

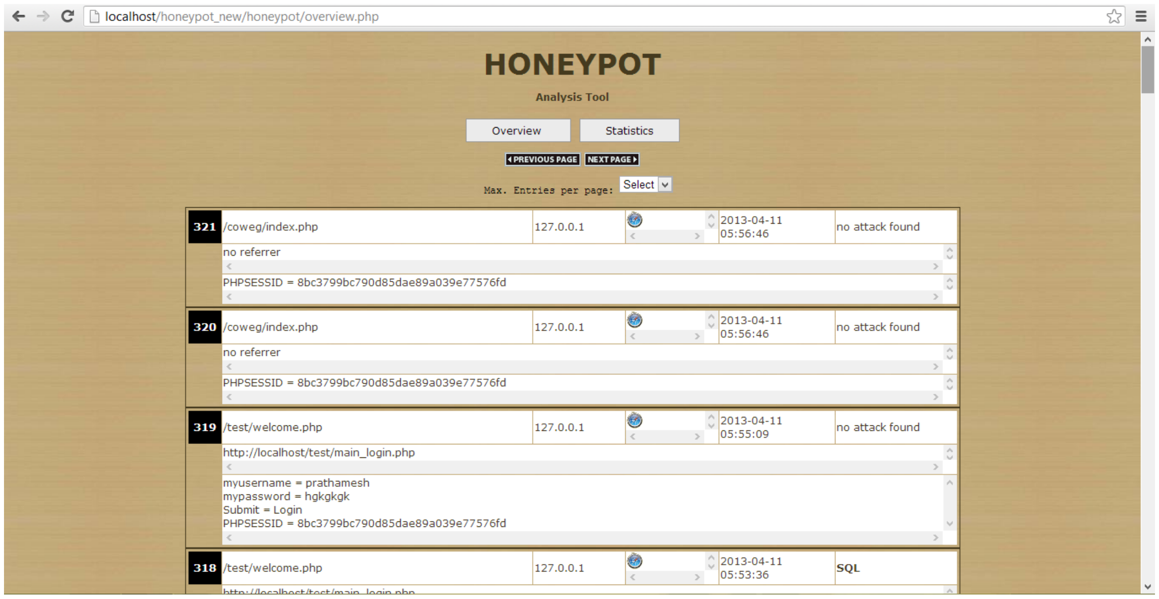
Open new phpMyAdmin window

STEP 4: OPEN HONEY-POT TOOL:

- To open actual honeypot tool type localhost/honeypot_new in address bar of local browser.

- Go to parent directory and click on honeypot folder

- Entire tool is visible with user access entries and website statistics.



321	/coweg/index.php	127.0.0.1	2013-04-11 05:56:46	no attack found
	no referrer			
	PHPSESSID = 8bc3799bc790d85dae89a039e77576fd			
320	/coweg/index.php	127.0.0.1	2013-04-11 05:56:46	no attack found
	no referrer			
	PHPSESSID = 8bc3799bc790d85dae89a039e77576fd			
319	/test/welcome.php	127.0.0.1	2013-04-11 05:55:09	no attack found
	http://localhost/test/main_login.php			

	myusername = prathamesh
	mypassword = hgkgkgk
	Submit = Login
	PHPSESSID = 8bc3799bc790d85dae89a039e77576fd			

318	/test/welcome.php	127.0.0.1	2013-04-11 05:53:36	**SQL**
	http://localhost/test/main_login.php			