Penetration Testing Auditability

Alexandros Tsiridis, Stamatios Maritsas

Master of System and Network Engineering
The University of Amsterdam
January 2016

Abstract

The purpose of this project is to figure out a method to improve penetration testing auditability and collaboration through it. In order to achieve that, we answered the research questions of what are the sources of auditability data, what methods can be used to audit these sources, how to store efficiently these data and how to enforce collaboration during the penetration testing procedure. Using the answers from the research questions and the knowledge obtained from the literature study, a framework was designed and proposed to improve penetration testing auditability process and teamwork. A prototype was implemented, based on the framework, that was later on tested and proved that the framework achieved its goals.

Contents

List of Figures

Acknowledgements

We would like to thank our supervisors, Christopher Mills and Rick van Galen, for their support, co-operation, help and providing us with all the needed resources. We would also like to thank KPMG company for giving us the opportunity to work on this research project as well as accepting us in their building, in order to work close to the security team. We would also like to thank the team of pen testers for letting us observe how they work, give us time for interviews as well as helping us test the system and provide feedback for it. Last but not least, we would like to thank the whole team of SNE for guiding us during the whole procedure of our research project.

Chapter 1

Introduction

1.1 Problem Description

By the term "penetration test", or just pen test is meant an effort to evaluate the security of a IT infrastructure using several methods to detect and exploit vulnerabilities. A penetration test can help determine the security profile of a target, such as whether it is possible to gain unauthorised access to a system[4]. Also, pen test reveals in what kind of attacks a system's defences were sufficient, and which defences (if any) the test defeated. This process is usually performed within teams of penetration testers[27]. Penetration testing itself is unstructured, due to the mass volume of data and logging that the pen testers need to have in order to track their steps. Pen testers usually use numerous command line tools for that purpose as well as GUI tools. Whilst there is a high level structure to a penetration test, as the focus is on making systems work in situations, developers do not anticipate. The process of finding these unanticipated situations can be quite ad hoc, and consequently difficult to log consistently.

Due to the above mentioned problems we can clearly see that there is a need for better penetration testing auditability. By the term "auditability" is meant that we have to establish a way to track a pen tester's actions in order to be able to show what triggered a finding of a vulnerability or a negative result such as impacting the availability of the targeted system (e.g. crashing it). A proper auditability that is not intrusive to the pen tester will lead not to forget steps of the penetration process that should be tracked. It will also increase his productivity and will help in collaboration within a team since a member of the team can see what another member did.

1.2 Aims

The aim of this project is the improvement of penetration testing auditability. The following research questions had been established in order to guide us

to the solution of penetration testing auditability problem.

1. What are the sources of penetration testing auditability data?
2. What methods can be used to effectively audit these sources?
   (a) What are the ways to capture command line streams?
   (b) How to gather information from pen testing tools?
   (c) How to capture manual actions?
3. What methods can be used to store these data efficiently and practically?
4. How can penetration testing auditability enhance collaboration during penetration testing procedure?

1.3 Objectives

In this section we are going to describe the several milestones that our research project will be based on.

To begin with, a deep research was conducted using a variety of methods such as reading articles, distributing questionnaires among the security team and arranging short-time interviews with pen testers of the team.

Later on, we collected and analysed all the data that we obtained from the research and moved on to the design of a framework that could solve the problems that arise from previous analysis.

Finally, we implemented a prototype as a proof of concept. The prototype was meant to show that the framework can solve the problems specified during the analysis.

Chapter 2

Literature Survey

2.1 Introduction

This chapter is dedicated to our actual conducted research, which will be based on four major parts. First of all, we are going to describe and analyse what we mean by penetration testing procedure and what are the phases that a penetration test should be consisted of. Moreover, what types of attacks can be deployed during a penetration test and also investigate if a penetration testing procedure can actually be standardized or not. This part of the research will show how the penetration testing is done, what data it produces and this will lead on the second part of the literature survey.

Second of all, we are going to describe, based on the findings, how penetration testing auditability should be performed. This includes the sources of penetration testing data, the goals of auditability and lastly how to perform this auditability procedure efficiently and not intrusively to the pen tester.

In the third part, we will discuss about existing work and how they try to solve those issues. Finally, the fourth part will introduce a better solution based on the comparison of existing work and the knowledge obtained from our general research.

2.2 Penetration Testing Procedure

2.2.1 Definition

By the term "penetration test", or just pen test is meant an effort to evaluate the security of a IT infrastructure using several methods to detect and exploit vulnerabilities [4]. In other words, penetration testing should, therefore, model the attack profiles of potential threat sources in order to accurately determine the level of success of a deployed attack [3]. In penetration testing, both the actions of authorized user and the unauthorized perpetrator have to be considered. Finally, penetration tests should be designed carefully to

evaluate the ways in which security flaws can be exploited [19].

2.2.2 Phases of penetration testing procedure

A penetration testing model consists, according to the articles [17, 3, 11], of the following six phases:

1. Introductory Planning and Preparation
2. Discovery and Investigation
3. Assessment and Strategy
4. Exploitation / Invasion
5. Maintaining Access
6. Reporting / Documenting

Introductory Planning and Preparation

During this phase of the penetration testing procedure, a settlement on the objectives and scope of the pen test is established between the client and the pen testers. This has to be done by taking into consideration the criminal and civil law. In other words, a contract is prepared and then signed by a client, that specifies what the pen testers are allowed to test and what they are not [17].

In addition, during this phase, one of the following penetration testing types is chosen:

1. **Black box testing** can replicate the scenario of an outside pen tester, who has no knowledge of the system that he is trying to penetrate, but only publicly available information about the target (e.g. IP addresses, public information about the company, etc.) [5]. In other words, the pen testers have no prior knowledge of the network that they are targeting and any information that they obtain are through public sources and investigation [11].
2. **White box testing** can replicate the scenario of an inside attacker, who has inside knowledge of the system that he/she is trying to attack. In other words, the pen testers are given near to full information about the target and sometimes collaborate with the staff of the company [5, 11].
3. **Grey box testing** is a combination of the black and white box penetration testing. The pen tester is given some information about the targeted system. This type of testing is performed when time and technology are limited [22].

Discovery and Investigation

According to this phase which is the start of actual testing, the main purpose is the information gathering of the targeted system. That purpose is often being succeeded using methods such as [23]:

- Network port and service identification
- DNS interrogation
- Network sniffing
- Searching the organization's web servers or directory servers
- etc.

Assessment and Strategy

Assessment and Strategy stage usually happens to be undetectable, as it does not involve any contact with the targeted system. In this stage, the above findings are assessed in order to determine any possible vulnerabilities. Additionally, this phase is the sorting of the gathered data to piece together an idea of what a pen tester is trying to penetrate. After this analysis of the gathered data, a strategy is devised for the upcoming attack(s) [3].

Exploitation / Invasion

Once a pen tester had collected a reasonable amount of information about the targeted system, it is then possible to begin Exploitation / Invasion phase and perform the actual planned attacks [3]. This stage usually consists of [23]:

- Gaining access
- Escalating privileges
- System browsing
- Installation of additional tools

Maintaining Access

In order for this stage to take place, success of previously deployed Exploitation / Invasion phase is taken for granted. During this phase, steps are usually taken to make future accesses to compromised system easier to deploy. Usual ways for conducting this stage are the following [3]:

- Installation of a back-door program

- Setting up a home base under a seldom-used account name
- Usage of a misconfigured user account with suitable permissions

Reporting / Documenting

This is the last phase of penetration testing procedure, where the final report is produced by the pen tester which documents the findings and it is delivered to the client. This stage occurs simultaneously with the other five phases of the pen test and should contain the following [17, 23]:

- Evaluation of vulnerabilities found in the form of potential risk
- Recommendations for eliminating found vulnerabilities
- Transparency of the process

2.2.3 Manual and automated penetration testing

Penetration testing consists of two forms [2]:

1. **Manual penetration testing**, which is performed without the aid of automated tools. A pen tester will perform manual actions such as typing commands to a command line or interacting with the environment. Manual penetration testing is divided into the following two categories:
   (a) **Exploratory manual penetration testing**. In this category, the pen tester performs actions based on his/her own experience and instincts without the use of a test plan. An example of such actions is testing the default username and password of routers or other devices.
   (b) **Systematic manual penetration testing**. In contrary to the manual penetration testing, this category involves the following of a predefined test plan. This is done in order to reduce time, take advantage of repeated errors and experience of the whole pen testers team and make the process more systematic, in order not to forget any test cases.
2. **Automated penetration testing** is taking the form of automated testing tools, that were created from pen testers in order to do repeated tasks that appears to be usually common during the penetration testing process. In addition, pen testing tools that are home made and encapsulate the experience of their creators are not uncommon.

2.2.4 Standardized procedure or not?

According to the article [5], penetration testing is an art. This is based on the reasoning that a scientific process depends on falsifiable hypotheses, where penetration testing is impossible to have such feature. We cannot create an known-to-be-complete list of vulnerabilities that a hypotheses can be built upon them, therefore penetration testing is not science. Vulnerabilities are closely coupled with the implementation of the targeted system and sometimes unique to itself. Penetration testers have to use their own creativity, experience, knowledge and instincts to find and exploit security flaws.

2.3 Penetration Testing Auditability

By the term "penetration testing auditability" we mean the process of recording the actions that a pen tester performed during the process of penetration testing. The goal of auditability is to keep track of the actions performed by a penetration tester in order to be documented properly and used as evidence to an exploited security flaw of the system and how it can be reproduced. During the penetration testing procedure, the system might be affected making it unavailable, produce wrong results or even damaged. The auditability has also the goal to prove if the actual pen testers are responsible of damaging the system or if the damage has occurred due to other reasons.

2.3.1 Sources of auditability data

As mentioned at previous sections, penetration testing has two forms of actions, manual and automated. The automated actions involve the use of automated tools. Those tools can be command line based or have a graphical user interface. In addition, they produce log files and reports, which hold the data needed for the auditability. Moreover, the initial settings and the commands that where executed by the automated tools are also data that can be used for auditability.

In the manual actions, we have two different types of data sources. One of them is the command line where a user can simply type commands. This means that the command line streams is a source of the data needed to be captured for auditability purposes. The second type of data that we need to be captured, is when pen testers deploy actions using their environment such as typing the default username and password of a router's page or navigating through the browser in order to get access to restricted areas.

These lead us to the conclusion, that we have three types of sources, which are needed to be captured. The first source is the command line streams that can capture the command line manual actions as well as the command line based automated tools. The second is the screen of the machine, that can be used to audit GUI automated tools and manual actions of a pen tester

that uses his/her environment. The last source is the produced log files and reports of automated tools.

2.3.2 Current methods

In the case of manual actions we have two types of sources as we have described above, which have different methods of capturing. The command line can be captured using two different methods. The simplest of the two, is just copy pasting the commands output from the terminal interface into a different file. The second method is by using tools that can capture the streams of the command line and store them into files.

In the case of manual actions that pen testers use just their environment, they can log their process using tools for screen shots and screen recording.

The third case, is when a pen tester uses automated tools. As mentioned above, these tools are generating their own log files and reports, which a pen tester can simply copy and use.

The final method can actually log all of the above, but it is the most intrusive and time consuming. This method involves the penetration tester to take manual notes of what he/she did and when using text editing tools.

2.3.3 Problems

The penetration testing auditability process comes with three serious problems that need to be stated. The first problem is that it is highly intrusive to the user, especially when it comes down to manual actions, as he/she needs to stop after each action to document it.

The second problem is that it is time consuming. The reason of this problem is that the user needs to take time to well document his/her actions and that he/she will need some time to go back to the flaw of penetration testing procedure.

The third and last problem is that due to the volume and variety of data and actions that the pen tester needs to record, the process becomes chaotic. All these data are difficult to be well organized.

Due to the above problems, usually the pen testers do not perform this process to the extent that they should. This leads to the problem that some actions will be left to be recorded later on collectively, which contains the risk that the pen tester might eventually forget to record some of those actions.

2.4 Existing Work

2.4.1 Command line capturing tools

In this subsection, we are going to talk about capturing one of the sources of auditability data that we mentioned in previous sections. In the command

line session a pen tester can perform manual actions as well as automated ones using command line tools (e.g. Nmap, Nessus, Metasploit, etc.). This makes command line a valuable source of auditability data. In order to log this source of data, we are going to describe some of the most used tools such as `script`, `scriptreplay` and `asciinema`.

First of all, `script` helps a user to make a copy of a terminal session. Specifically, what this tool does, is capturing the I/O of a command line session and stores it into a file. In order to use `script` a user can call it at his/her command line session at the time that he/she wants to start the capturing and stop it at any time using the `exit` command. The logged session will be stored into a file with the user's chosen name or the default name `typescript`. In addition, `script` tool provides the parameter `-t`, which allows the user to record the timing between each typed letter, command line output and stores it into a separate file or outputs it to the standard error [14].

A relative tool of `script`, is `scriptreplay`. This tool take a captured file using the `script` tool and replays it using timing information. Note that `scriptreplay` does not reproduce the commands by actually calling them, but it only creates a video-like output of the script captured and shows it to the user [15].

As a third tool for capturing the command line I/O, we have the `asciinema`. This tool is inspired by `script` and `scriptreplay`. It consists of three components, which are a command line session recorder, a website with an API at asciinema.org and a javascript player. Asciinema can capture the command line session, store it into a file or upload it to the website asciinema.org, in order to use javascript player to play the recorded session like a video. It can also play the recorded session as a video using the command line just like `scriptreplay` [1].

To sum up, the above tools can capture the command line I/O, helping the penetration tester to log his/her actions. They can capture both manual actions and automated tools and place the results into a file that pen testers can use for further analysis. In addition, the user can selectively start and stop the capture in order to record valuable actions. The only thing that those tools do not do is add proper timestamps at the commands and tools executed as well as to the results of those executions. As mentioned in previous research, timestamps are valuable as they allow a client to correlate the actions that pen testers did with the effect at their systems and also give pen testers the capability of tracking their steps in chronological order.

2.4.2 Auditability data management tools

In this subsection, we are going to talk about tools that they were designed and implemented for purposes of gathering and managing the data produced by the penetration testing procedure, in order to allow the user to organize

them and analyse them.

One of these tools, is called MagicTree. It is designed to allow easy and straightforward data consolidation, querying, external command execution and report generation [8]. In addition, it can import data from XML files and query them using XPath. MagicTree stores its data into an XML format and presents them in a tree structured way. It does not collect those data automatically and needs the penetration tester to input them manually, either using XML files or simply typing them [7].

The second tool that it is related to our topic is Kvasir, an open source Cisco Systems product, which represents a complete vulnerability /penetration testing data management system designed, especially, to improve collaboration within penetration testing teams [10]. Kvasir does this by homogenizing data sources into a pre-defined database structure. Currently, it supports many tools such as [9]:

- Rapid 7 NeXpose
- Metasploit / Metasploit Pro
- Tennable Nessus
- ShodanHQ
- Nmap
- THC-Hydra
- Medusa
- John The Ripper

The above discussed tools can manage and organize penetration testing data produced by automated pen testing tools. This is also their disadvantage. They do not provide any method of gathering and presenting any data produced by manual actions of the pen testers. Additionally, as stated at previous research, penetration testing was characterized as an art, meaning that the pen testers might use their own created tools or tools that they are not supported by those applications. Kvasir, for instance, uses add-ons, in order to support well known tools, which makes it difficult to maintain as well as impossible to cover all the automated tools, leading the pen testers, that use Kvasir, to limit their selves only to the supported tools.

2.4.3 Manual notes

Pen testers also need to keep notes of their performed actions during a penetration test. Moreover, they might need to wirte down their thoughts, explain discoveries or keep notes on what they want to do later on.

As a first way to accomplish this goal, we are going to discuss text editing tools. Tools like Notepad++, gedit, sublime and etc. supply the user with the capability to write down their own notes, thoughts and simultaneously use various fonts and styles.

A second method is to use tools that are specialized on keeping notes instead of documents. Tools like KeepNote and Evernote allow the user to keep both their thoughts and findings, while performing a pen test, in a more structured and organized way.

As a conclusion, the above tools provide the user with the freedom of writing and organizing his/her actions exactly in the way that he/she wants. Although this need seems to be very important, it can also be very intrusive, in case that a pen tester uses it, in order to log all of his/her actions and findings. We suggest the use of those tools to be limited on simply taking notes and organizing their thoughts, instead of using them for logging their actions.

2.4.4 Collaborative tools

The pen testing procedure is most often performed by a team of pen testers. During this period, collaboration among the team is an essential part of the procedure. With collaboration pen testers can achieve more in less time and can even exchange ideas, share knowledge and increase their productivity. There are a few tools that enforce collaboration and we are going to describe them in this subsection.

One of such collaborative tools is the open source project called LAIR. It uses the Mongo database, in order to normalize, centralize and manage data from a number of automated pen testing tools such as Nmap, Nessus, Nexpose and Burp. It also allows real-time updates of the database and merges data from various tools for a host. Furthermore, it allows multiple pen testers to use it at the same time by the use of accounts. Users can create projects, which they have a unique identifiers and later on they can add pen testers as contributors to a project. LAIR makes use of five different colours, in order to identify if a host is under investigation by a contributor, if it needs further testing or if it completed. As this tool uses Meteor, the web browser becomes a database client, allowing the user to run Javascript scripts to directly modify the data. Finally, it hosts a chatting environment that pen testers can use to communicate [25].

The second tool that actually created to enhance collaboration among a penetration testing team is so-called Collaborative Penetration-testing and Analysis Toolkit (CPAT). The main two fundamental goals of this project are the ease of collaboration and the real-time data management. It somewhat has the same architecture as LAIR, which means it uses the Meteor framework and Mongo database. Its features a centralized data storage which uses parsers in order to extract data from automated tools reports and log

files to import them into itself. This CPAT tool allows live data visualization from the attached database as well as a chat application for providing communication among the team. Finally, it provides a list of recent activity. [22].

Both of the above tools cover almost all issues that collaborative work may have. They remove data duplication and they provide a well organization of data as well as query of the stored data inside their database. They also designed to allow a method of communication (chatting) and cover most of the common pen testing tools. The problem that arises with these tools, is that they do not have a method of inserting data from manual actions as well as uncommon and home made tools. LAIR has a functionality of adding data manually, but only if they are related to a host. Furthermore, none of the tools has a way of organizing the project into sub-tasks that users might share. LAIR, for example, has a way to show that a certain host or hosts are under investigation from someone, but not under which type of investigation. In other words, those tools lack any functionality of planning and task sharing.

2.5 Conclusion

This section will conclude the most valuable information that we obtained through the Literature Survey. First of all, as we discussed, penetration testing is an art, meaning that the pen testers are mostly dependent on their creativity, experience and knowledge to exploit possible vulnerabilities and security flaws. As those pen testers are characterized as artists, means that a system that will help them at the auditability of their actions needs to allow and not restrict their freedom. We need a system that will not limit pen testers to specified penetration testing tools and allow auditability of manual actions.

As we discussed, we have two types of penetration testing actions, manual and automated. From those actions we have various sources of data. These sources are the command line, the log files and reports from the automated tools and finally the environment that a pen tester is using. The command line can be logged using a tool that can capture the I/O along with timestamps. The automated tools produced their own log files and reports, meaning that there is no actual need to audit them using another tool, but it would be good to have a tool that manages their data. Lastly, by using screen shots and screen casting we can audit the manual of the pen tester when uses his/her own environment.

Moreover, we saw that collaboration is an essential part of penetration testing procedure as it is usually performed within a team. The problems of collaboration is planning, task sharing and duplicated actions. A system that addresses those problems will increase the productivity of the team as

well as their performance. In order to achieve that, a centralized storage that can store any type of files is needed. This will let the pen testers use any type of tool, so that they will be creative and not restricted.

Chapter 3

Requirements

3.1 Introduction

In this section, we will establish the requirements of the system, which describe the exact functionality of it in order to solve the problems that were recognised. This is a needed phase, as these requirements will provide a roadmap for the design of the system.

3.2 Methods of gathering requirements

In order to formalize and obtain the system requirements, we used a variety of methods to gather the needed data. The analysis of those data will extract the requirements of the system. A list of those methods is the following:

- Interviews
- Questionnaires
- Literature study

Interviews were the first approach that we used for the sake of obtaining the initial information that guided our research. This method was also performed after the literature study to see the correlation between the research existing and the actual practice. The personnel that was interviewed varied from penetration testers that were recently employed staff to senior staff and team leaders. The goal of this targeted variety was to extract knowledge from experienced penetration testers, as well as new innovative ideas from the younger personnel.

Questionnaires were used to gather quantitative data and answer questions that were formed after the analysis of the interviews. The literature study was performed to obtain knowledge from the security community and provide us with a more scientific view.

3.3 Gathering and analysis of requirements

According to our research, the system will be based on the sources of auditability data, the methods of gathering those sources and functionality that increases teamwork. As we identified form the Literature Survey Chapter 2, the sources of auditability data are the manual actions and automated tools. Those actions involve the command line (manual commands, command line tools), GUI tools and actions that use the environment (e.g. browser navigation, router login etc.). In order to capture the identified sources we need the following methods:

- Capturing of the command line streams (`stdin, stdout, stderr`).
- Log files and reports of automated tools (both command line based and GUI).
- Screen shots and screen casting for auditing manual actions that use the environment and GUI tools.

In addition, collaboration plays a valuable role to penetration testing as each job requires a team of pen testers. In order to increase their teamwork, our research showed that they need planning, task sharing, communication and collaborative documenting. The system needs to have functionality that will cover the above mentioned aspects.

3.4 Functional Requirements

3.4.1 Action Recording

1. Capturing of the command line streams.

   Description: Capturing input and output of the command line interface and store it into a file.

   Priority Level: High

2. Add timestamps to the command line capture.

   Description: Add timestamps to the command line commands, as well as their output and store them into a file.

   Priority Level: High

3. Image capturing.

   Description: Allow a user to take screen shots of his/her actions.

   Priority Level: Medium

4. Screen casting.

   Description: Allow a user to record screen videos of his/her actions.

   Priority Level: Medium

5. Create / Edit / Delete Notes.

   Description: Allow a user to take manual notes of his/her actions, ideas, goals and organize them.

   Priority Level: Medium

3.4.2 Planning

1. Create a plan.

   Description: Allow users to create a plan for the penetration testing procedure.

   Priority Level: High

2. Modify a plan.

   Description: Allow users to modify a plan for the penetration testing procedure.

   Priority Level: High

3. Delete a plan.

   Description: Allow users to delete a plan for the penetration testing procedure.

   Priority Level: High

4. Add / Edit / Delete task to / from a plan.

   Description: Allow users to add / delete / modify a task of / to the penetration testing procedure plan.

   Priority Level: High

5. Allow access to a plan and its tasks to the pen testers.

   Description: Make the plan and its tasks available to the users related to the penetration testing procedure.

   Priority Level: High

6. Associate findings to a task and users.

   Description: Show the data that has been collected by a certain user related to a task.

   Priority Level: High

7. Show current status of a task.

   Description: Show if a task is under progress, completed or if it is new.

   Priority Level: Low

3.4.3 Data Sharing / Storage

1. Upload files to a centralized store.

   Description: Allow a user to upload any type of files to a centralized storage space.

   Priority Level: High

2. Allow users to have access to the uploaded files.

   Description: Allow users to view and download all files that have been uploaded from the other users to the centralized storage space.

   Priority Level: High

3. Associate files to the users that had uploaded them.

   Description: Show which user uploaded a certain file.

   Priority Level: Medium

3.4.4 Chatting

1. Global chat.

   Description: Allow communication between users using a global chat.

   Priority Level: Medium

2. One-to-one chat.

   Description: Allow one-to-one communication between users.

   Priority Level: Low

3. Store chats.

   Description: Allow users to see the messages that have been exchanged during previous conversations.

   Priority Level: Low

3.4.5 Collaborative Documenting

1. Create a collaborative document.

   Description: Allow users to create shared collaborative documents.

   Priority Level: High

2. Modify a collaborative document in real time.

   Description: Allow multiple users to modify existing shared collaborative documents and show the changes that are being made in real time.

   Priority Level: High

3. Delete a collaborative document.

   Description: Allow users to delete existing shared collaborative documents.

   Priority Level: High

3.5 Non Functional Requirements

1. The system should be responsive.

   Description: Commands should take place, when users request them, meaning that they should start and stop immediately when a user asks for it. This has the exception of network bandwidth and latency.

2. The system should be available for as long as the penetration testing procedure takes place.

   Description: The system should be available at least during working hours and time line of the pen testing procedure. The machine, where the centralized storage space is placed (server) should be running and be accessible during this period.

3. The backup procedure should be easy and performed regularly.

   Description: The state of the centralized storage space should be able to be backup at any time without much effort.

4. User friendly interface.

   Description: The system should have a user interface, that is easy to learn and use.

5. Quick access to the uploaded files.

   Description: A user should be able to access the files in less than four actions.

Chapter 4

Design

4.1 Introduction

This section will be dedicated to the design of our proposed system. The design constitutes an important and absolutely necessary phase, as it describes how the system should actually work. This phase follows the establishment of the functional and non-functional requirements, as the design presents how these requirements are being satisfied by the system. The design phase will provide us with a roadmap for the implementation of the system.

4.2 Framework

In general, a framework is a real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful [21]. This framework is our proposed methodology of how penetration testing auditability should be done among a team.

Our framework is based on the following categories:

- Action recording
- Centralized storage
- Communication
- Collaboration

By analysing the above categories, we can conclude that the framework will be divided into five sub-systems:

1. Action Recording
2. Planning
3. Data Sharing / Storage

4. Chatting

5. Collaborative Documenting

4.2.1 Action Recording

In this sub-system, the functional requirements 1-5 of action recording, section 3.4.1, will be satisfied.

To be more precise, this sub-system should have a functionality of capturing in real time the I/O of a command line session and input timestamps for each command along with its output. Moreover, it will need to be able to offer image capturing and screen casting. As a final requirement, this sub-system will include a functionality that will allow a user to take notes during the penetration testing procedure.

4.2.2 Planning

This sub-system will cover the functional requirements 1-7 of planning section 3.4.2.

Specifically, the user should be given the capability of creating, modifying and deleting a plan of action for the penetration testing procedure. The idea is to create multiple tasks, that users will be able to share among themselves. This sub-system should provide a method of adding, editing and deleting tasks, as well as showing which tasks were performed by one or more users and what data they obtained from those tasks.

Finally, this sub-system should present a complete image of a plan and its tasks. It should also present the information that is relevant to the status of a task (e.g. new, under progress, completed, etc.).

4.2.3 Data Sharing / Storage

In this subsection, will be covered the functional requirements 1-3 of the data sharing / storage sub-system.

First of all, a user should be able to upload any type of files to the centralized storage space, but also be able to download and read any other files that other users uploaded, in order to enforce collaboration. Another responsibility of this sub-system will be to show which files are uploaded by a certain user.

4.2.4 Chatting

Chatting sub-system will be implemented in a way that provide communication between users, in order to avoid using any public communication method that might lead to leakage of information. It should allow global

communication among the users of the system. It may also provide functionality for one-to-one communication, as well as storing the messages that had been exchanged. These functionalities are the requirements 1-3 of the section 3.4.4.

4.2.5 Collaborative Documenting

Collaborative documenting that covers the requirements 1-3 of section 3.4.5, needs to be a sub-system of its own.

This sub-system will hold features for creating, editing and deleting collaborative documents, that will be updated in real time. In addition, those documents should be accessible by all users of the system.

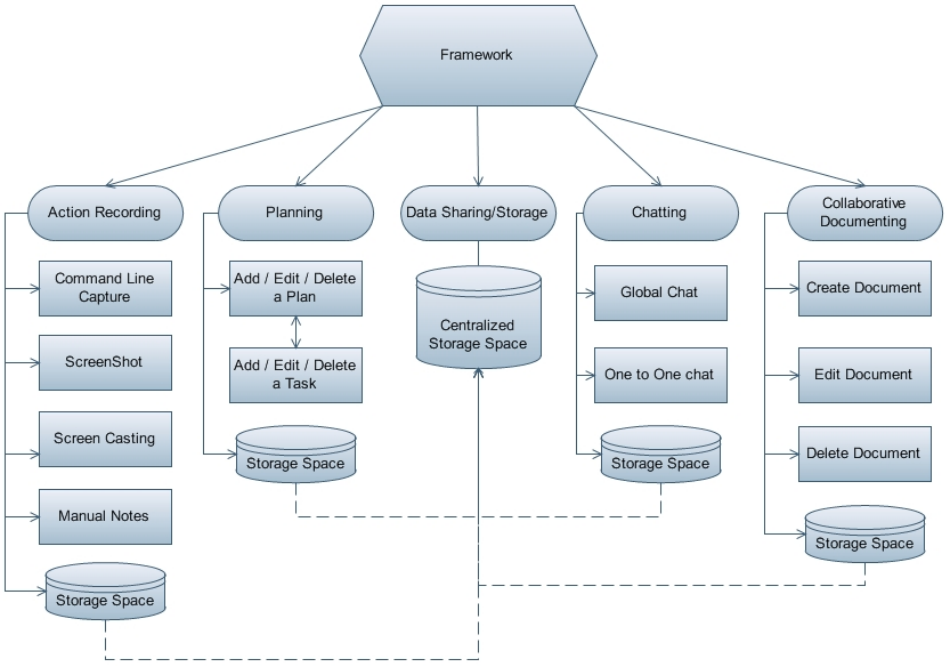

Figure 4.1: Proposed framework.

4.3 Prototype

This section describes the design of a prototype that was created as an implementation of the proposed framework. This prototype will be used as a proof of concept and evaluate the improvements that the framework can establish for the penetration testing auditability.

4.3.1 System Architecture

Our prototype is based on the scenario that pen testers have machines on which they perform the pen testing procedure and a server that will host a centralized storage space. In our perspective, the sub-systems of data sharing / storage and planning can be combined and implemented using a web application, which will be hosted on the server. The rest of the sub-systems, will be placed on the pen testers machines. There are already existing open source tools, that can satisfy those sub-systems and will be integrated to form the implementation of the framework.

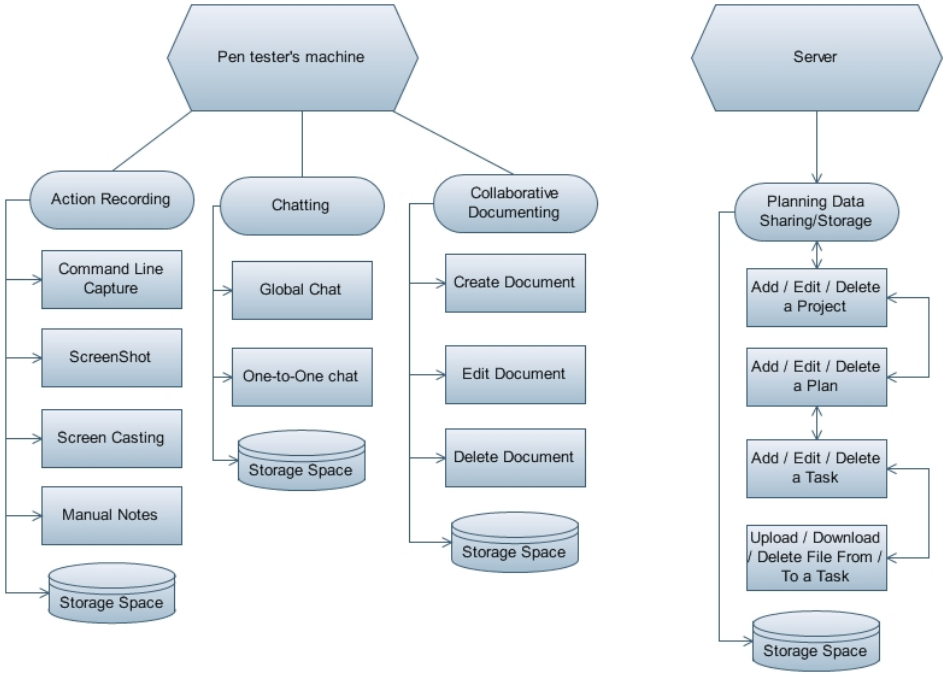

Figure 4.2: Prototype architecture.

Web Application

The architecture of the web application will be formed by three layers, which are the back-end, the middleware and the front-end. In the back-end, a database which will hold the accounts of the pen testers and a file directory tree, which will hold the files and the planning of the procedure. Middleware will be responsible for the connection of the database and the file directory tree with the front-end. Lastly, the front-end is the user interface of the web application.

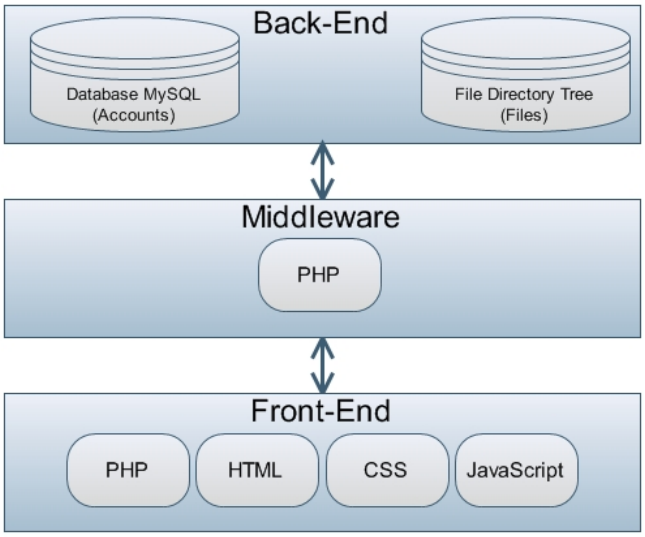

Figure 4.3: Web application architecture.

Command Line Capture

During our research, we found out that command line capture should also include timestamps for security purposes and correlation between the actions that pen testers performed and the behaviour of the targeted system. Although, we found many tools that can capture the command line streams, we did not discover any tool that can also add timestamps to the capture.

For this reason, we needed to create one that should be able to capture the command line streams and add timestamps after each called command as well as the output of it. Additionally, this tool should work on the command line interface that it was called from and allow all the functionality that the command line provides. It will also allow the user to specify the name of each log file that will contain the capture. Finally, due to fact that pen testers might perform testing for a foreign company or they can be geographically distributed, the timestamps should be in the coordinated universal time (UTC).

Chapter 5

Implementation

5.1 Introduction

This chapter will be devoted to the implementation of a prototype that is based on the already designed framework. The prototype is implemented, in order to test whether our proposed framework increases the quality of penetration testing auditability as well as the collaboration among the pen testers. The general overview of the implementation is based on two virtual machines, one that would be the server and another which is going to be the pen tester's machine.

5.2 Server

The server is simulated using a virtual machine that runs `Kali Linux 2.0` as its operating system. This virtual machine will be used for the hosting of our web application as well as for the storage, planning and collaboration among a team of pen testers.

5.2.1 XAMPP

Taking into consideration the requirements that were established, while describing the functionality of the server machine we came to the conclusion that, first, a web sever is needed to host the web application and, second, a database to manage the accounts of the pen testers.

For the above mentioned reasons, we choose the XAMPP package (version 5.5.30) that includes, among several tools, an Apache HTTP server 2.4.17, PHP 5.5.30, MariaDB 10.1.9 and Perl 5.16.3. Using this package we can satisfy the needed requirements to start implementing the web application and host it to our server.

5.2.2 Back-end

This layer of our implemented prototype is responsible for the centralized storage. It is divided into a database and a file directory tree.

Database MySQL

This database is only responsible for managing of the pen testers accounts. The database is named `cpto`. The following table describes the design of a single table, called `pentesters` that our database holds.

Table 5.1: Design of pentesters table.

Field Name	**Type**	**Primary Key**	**Extra**
id_user	int(11)	Yes	AUTO_INCREMENT
name	varchar(128)	No	None
email	varchar(64)	No	None
username	varchar(16)	No	None
password	varchar(32)	No	None

File Directory Tree

The file directory tree is responsible for storing the projects, plans of these projects and files as well as relate a file to its task and its uploader. The design in Figure 5.1 was chosen, in order to organize the mass of data that a pen testing procedure creates. Using this directory tree, a user can easily find a file without the use of the web application. On the other hand, the web application makes accessing and presenting of those files easier and faster.

In addition, backup of those files is as simple as copying the directory "Projects". Let us consider the scenario that, this web application is going to be used into a smaller scale, for instance, if a small server is established to serve a small team of pen testers for a single job. The directories of the Projects can then be copied from the smaller scale server into the main server. This will have the result, that the web application will instantly show from the main server, the Projects that came from the smaller server without any implications.

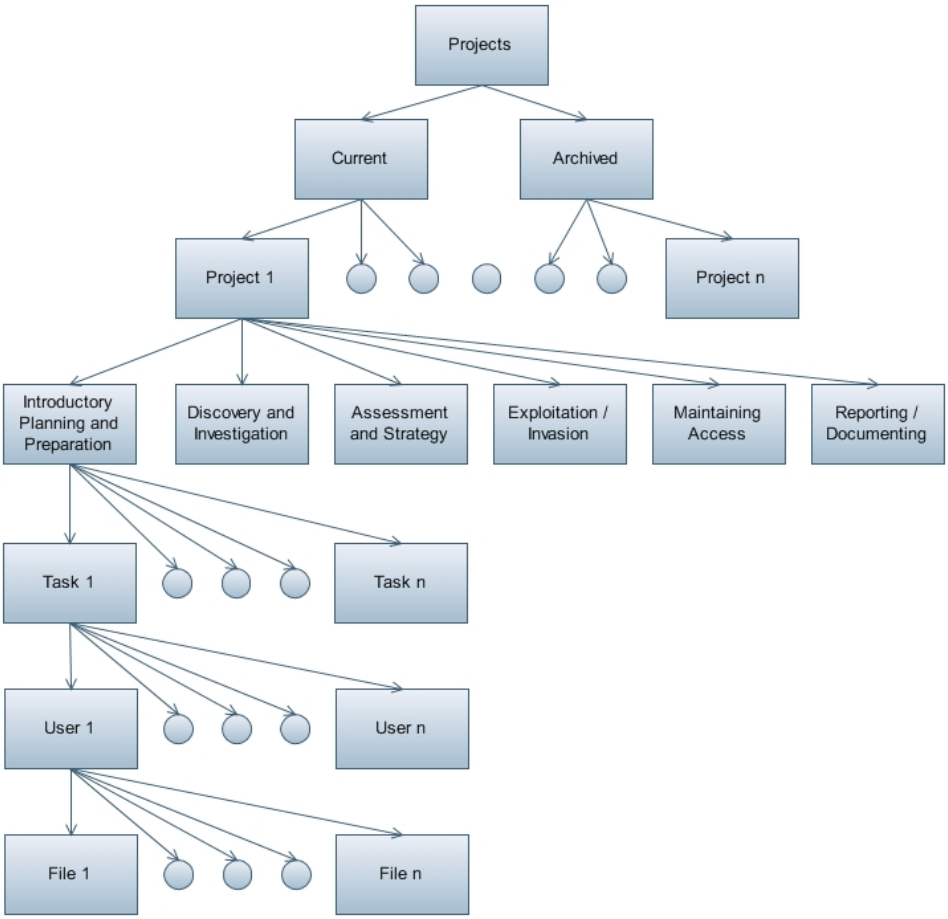

Figure 5.1: File directory tree.

5.2.3 Middleware

Middleware by its definition, is responsible for the connection and the interaction of the front-end with the back-end. In our case, while a user uses the middleware can communicate (send requests / receive responses) with the database, in order to manage his/her account, as well as with the file directory tree.

Database Management

The database management is done by three PHP files, namely `fg_membersite.php`, `formvalidator.php`, `membersite_config.php`. These files were obtained from [12] and modified according to our needs.

First of all, we have the `membersite_config.php`, which is responsible for the initial connection to the database. Using this file we can configure

the name of the database that is going to be used and the needed credentials in order to connect to it. Additionally, this PHP file calls functions from `fg_membersite.php` in order to initialize the connected database. The code is available at Appendix section A.20.

The PHP file `fg_membersite.php` is responsible for querying the database and present the results to the front-end. It contains functions such as `Login(), CheckLogin(), LogOut(), ChangePassword(), RegisterUser()`, etc. For example, `Login()` is responsible to check the credentials of the user and if they are correct creates a session for that user. The `CheckLogin()` function, on the other hand, checks which user is using the session and `LogOut()` destroys the session and redirects the user to the home page. The code is available at Appendix section A.18.

Finally, the role of the PHP file `formvalidator.php` is to check the validity of the data submitted to certain forms. For instance, it has the functionality to check whether an email is valid, if the username given already exists in the database, check the strength of the password, etc. The code is available at Appendix section A.19.

File Directory Tree Management

The management of the file directory tree is done by a number of PHP files that are responsible to create / move / delete projects and inside those projects to create / delete tasks and upload / delete files to / from those tasks.

The PHP file `addcurrent.php` is responsible to add a project into the Current directory. It simply creates a directory with the name provided, if that does not exist in Current or Archived directories, along with the directories of the phases namely Introductory, Discovery, Assessment, Exploitation, Maintaining and Reporting. The code is available at Appendix section A.17.

The goal of `moveArchived.php` is to archive the current jobs, in order to mark them as completed. The only functionality it has is to move the existing project directory that its name was given by the user from the Current directory to the Archived one. The code is available at Appendix section A.15.

The functionality that deletes a project from the Archived directory is provided by `deleteFolder.php`. This is done by providing the path to the directory needed to be deleted and the code recursively deletes all the files and directories inside the specified directory and then deletes that directory as well. The code is available at Appendix section A.16.

The PHP file `addtask.php` is responsible for creating the tasks inside a phase of a project. This is done by providing the path of the phase directory in which the task needs to be created. Inside that path it will create a directory with the name of the task and inside that directory the directory CoLoR will be created as well. The code is available at Appendix section

A.6.

The role of the `deletetask.php` file is to delete a task. This is achieved by first giving the path to the phase that the task belongs to as well as the name of the task. The code will recursively delete all the files inside the task and then the directory of the task itself. The code is available at Appendix section A.9.

The next in line is the `upload.php` file, which is responsible to upload a file and relate it to a task and the user that actually upload it. This is done by providing the path to the directory of the task, in which we want to upload the file and get the username from the session. Then, the code will check if a directory with the username exists inside the path and if it exists, it will place the uploaded file inside it or, if it does not, it will create it and then place the file inside it. The code is available at Appendix section A.10.

The role of PHP file `deletefile.php` is to delete a file from within the task. In a few words, the path to the file is provided and then it simply deletes it. The code is available at Appendix section A.8.

Finally, we have the `color.php` file, which is responsible for setting the indicator of the status color. This functionality simply checks if the CoLoR directory is empty, it will create the red.color file inside it, or if the red.color file exists it will delete it and create the green.color file and finally if the green.color file exists, it will delete it. The code is available at Appendix section A.7.

5.2.4 Front-end

Home page

The home page of the web application, that can be seen at Appendix section B.1, allows the user to login or register. The button Login redirects the user to `login.php` file, while the Register button redirects the user to `register.php`. The code of the home page is available at Appendix section A.2.

Register page

The `register.php` allows a user to enter the needed details in order to be enrolled in the database. A screen shot of the page can be seen at Appendix section B.2. The details of the user are checked for their validity by `formvalidator.php`. If the data given are correct, then the Submit button will call the `RegisterUser()` function of the `fg_membersite.php` that actually is responsible for enrolling the user into the database. After registering the user to the database, it will redirect him/her to `thank-you-regd.html` page. The code of register page is available at Appendix section A.3 and the code of `thank-you-regd.html` at section A.4.

Login page

The login page (`login.php`) can be seen at Appendix section B.3 and its code at Appendix section A.12. The code of this file simply calls the `Login()` function of the `fg_membersite.php` that actually checks if the credentials given by the user are correct. If they are valid, then the user will be redirected to the profile page (`login-home.php`). If not, then the user will get an error message.

Profile page

The profile page (`login-home.php`) gives the options of seeing the projects, change his/her password or logout. The Go to Projects button redirects the user to the jobs page. The Change Password button redirects the user to the change password page and the logout destroys the session and redirects the user to the logout page (`logout.php`, see Appendix section A.11). A view of the profile page is available at Appendix section B.5 and its code at section A.13.

Change password page

Next in line is the change password page (see screen shot at Appendix section B.5 and for its code at section A.21). This page prompts the user to give his/her old password and the new one. The Submit button calls the `ChangePasswword()` function of the `fg_membersite.php` that actually checks if the old password is correct and change it with the new one in the database. Then, the `LogOut()` function of the `fg_membersite.php` is called and the user is redirected to `changed-pwd.html` (see Appendix section A.22 for its code).

Jobs page

The page jobs gives an overview of the currently opened jobs and the completed ones (see screen shot at Appendix section B.6). A view of the page is available at Appendix section B.6. A loop is responsible to print a button with each of the current jobs that redirects to its plan page. Next to the job button a save button is printed that calls the `moveArchived.php`. Another loop is responsible to print a button with each of the archived jobs that redirects to its plan page and next to it a delete button that calls the `deleteFolder.php`. In addition, it has the functionality to create a new job, by simply typing the name of the job, at the text box provided, and click on the Create button that calls the `addcurrent.php`. The code of the jobs page is available at Appendix section A.14.

Plan page

The plan page allows a user to see and create a plan to the job chosen (see screen shot at Appendix section B.7). Additionally, it allows the users to share tasks and files among themselves. This page has a panel menu with the phases of the penetration testing procedure. At each phase tab, a function `taskfill($phase, $directory)` is called. This function has nested loops that are responsible to print the tasks, inside of the task will print a panel of the user's name and at its body the files that he/she uploaded. The files are printed as clickable links, that allow the user to download them. Next to the file a delete button is printed that calls the `deletetask.php`. Next to the task name a status button, that changes its color every time it is being clicked, is printed. This button calls the `color.php`. Next to the status button a delete button that calls the `deletetask.php` is printed. At the end of each task panel, a button Choose File allows the user to choose a file that he/she wants to upload and by clicking the Upload button, that calls the `upload.php`, uploads the file to the task under his/her name. Finally, at the end of the phase tab a user can create a task by providing the name of the task inside the a text box and click at the add task button, which calls the `addtask.php`. The code of the plan page is available at Appendix section A.5.

5.3 Pen tester's machine

In order to simulate a pen tester's machine, we created a virtual machine that runs `Kali Linux 2.0` as its operating system. We chose this operating system, because it is designed for advanced penetration testing and security auditing. It contains several pen test tools and by default `root` privileges [13].

In this virtual machine, we are going to add a suite of tools needed, in order to support our proposed framework. As described in Chapter 4: Design, subsection 4.3.1: System Architecture we have three sub-systems in the pen tester's machine namely:

1. Action Recording
2. Chatting
3. Collaborative Documenting

5.3.1 Action Recording

This sub-system has the functionality needed for capturing data needed for the penetration testing auditability procedure. The functionality implemented here has as an aim to keep the freedom of the pen testers. In

addition all those tools used here can easily be replaced with others as long as they follow the suggested framework.

Script capture.py

This Python script is responsible for the capturing of the command line streams (`stdin, stdout, strerr`) and also add timestamps to each command and its output. In order to see an example of how it works, see Appendix B.8 and B.9.

This is done by spawning a process and connect its controlling terminal with the current process's standard I/O. This allows us to capture the standard I/O and place it into a file. The timestamps are using UTC time zone and are placed at each new line that the program detects inside the captured stream. The `capture.py` is based on the code given at [20], which was modified in order to achieve the requested functionality. The code of the `capture.py` script is available at Appendix section A.1.

Screen shots

The chosen tool for this functionality in our prototype is Shutter (version 0.92). This application is free, open-source, and licensed under GPL v3. The reason that we chose this one is because, first it is for a Linux based operating system and it has the functionality to take a screenshot of a specific area, window, or the whole screen [24].

Screen casting

For screen casting the tool Vokoscreen (version 2.1.0) was chosen. It has the functionality to capture videos of the whole screen, specified window or selected area. In addition, it allows voice recording and camera recording. Using this tool pen testers can record their actions and also use the voice recording to record their thoughts. It is available at GitHub repository: `https://github.com/vkohaupt/vokoscreen`

Notes

In order to allow pen testers to keep notes of their actions, thoughts and ideas in an organized way, we chose Tomboy (version 1.14.1). Using this software, a user can create notebooks, where he/she can create multiple notes. Among some of its features is auto-linking web and email addresses, inline spell checking, font styling and sizing as well as it allows a user to export his/her notes in HTML format [26].

5.3.2 Chatting

As discussed in previous sections, communication is essential for the collaboration among a team of pen testers. in order to avoid any public communication method, we decided to add a chatting software that works within a LAN. The software chosen to fulfill these requirements is iptux (version 0.6.1). Iptux supports auto-detection of other users on the intranet, send messages to other users and even send files to other users [16].

5.3.3 Collaborative Documenting

To cover the collaborative documenting we chose Gobby (version 0.5.0), which is a multi-platform collaborative text editor. This software allows multiple users to work on the same document and shows the results in real time. The data transferred are encrypted including perfect forward secrecy (PFS) and the sessions can be password-protected. User can configure a server where the files are going to be hosted. Moreover, it has integrated group chat that allows communication of users while they modify a document. Gobby is licensed under GPLv2+ and ISC [6].

Chapter 6

Testing and Results

6.1 Introduction

This chapter will be dedicated, first, to the testing of our implemented framework along with its prototype and second to the gathered results. To be more precise, during the second section we are going to present all the testing methods that we used in order to gather our results. In the third section, the main goal is to present those results, using diagrams and finally, during the fourth section we are going to present a statistical analysis on those results. This analysis will lead eventually to answer the question if our framework actually improved the penetration testing auditability procedure.

6.2 Testing Methods

In order to rate the usability testing of our framework, we used three different methods. The first method was observation with the think-aloud approach. In more detail, we set up a testing environment, using two laptops connected at the same network and three VMs. The two VMs, a client and a server, respectively, were running on one of the two laptops using the network setting "Bridge" (provided by the VirtualBox virtualization software package, which was developed by Oracle) and the third VM was another client that was running on the other laptop with the same network settings. These network settings were used, in order to make the three VMs appear in the same network. The think-aloud method simply involved asking test participants, in our case 8 out of 16 in total pen testers, to use our system while continuously thinking out loud. This led to the verbalization of their thoughts as they were moving through the user interface [18].

The second method that we used, in order to evaluate the system was a questionnaire. The reason we chose that method was to gather written responses that later can be analysed using statistical methods. This questionnaire was given to pen testers after they finished testing the system's

usability.

The third method that was used, it refers only to the created `capture.py` script. After we contacted our supervisor, Christopher Mills, we agreed to give him the script and let him testing its performance in a real world scenario.

6.3 Results

6.3.1 Questionnaire

The already mentioned questionnaire that was used, in order to gather the results is divided into three parts, the "Action Recording" part, which consists of the first six questions, the "Web Application: Collaborate Pen Testing Organizer (CPTO)" part which includes questions 7 - 16 and finally the "System as a whole" part, which includes questions 17 - 19. The questions 6 and 14 - 19 have a rate scale 1 - 10 and the rest 1 - 5. Questions that have rate scale 1 - 5 represent the options "Strongly Disagree" to "Strongly Agree".

Action Recording

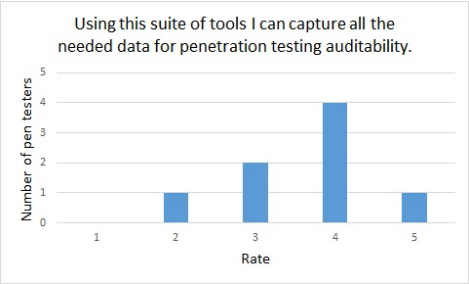

Figure 6.1: Question 1.

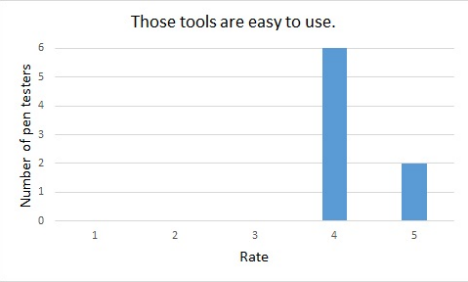

Figure 6.2: Question 2.

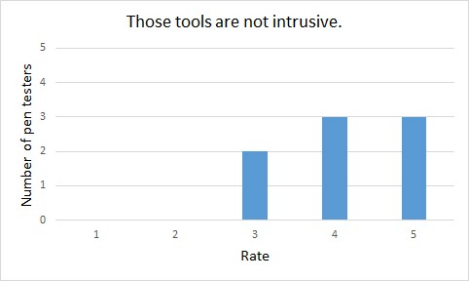

Figure 6.3: Question 3.

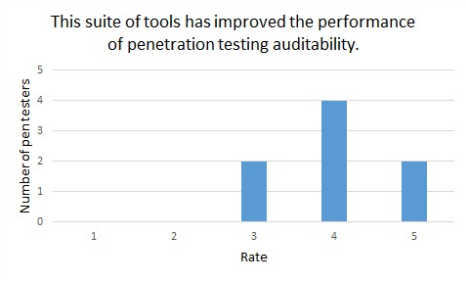

Figure 6.4: Question 4.

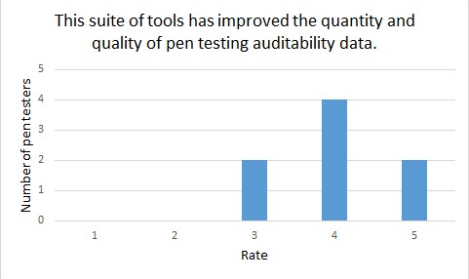

Figure 6.5: Question 5.

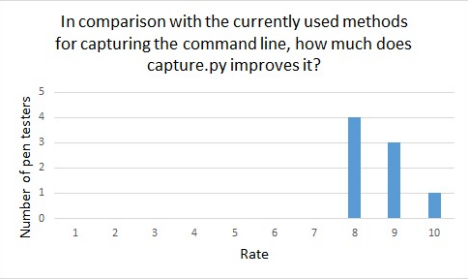

Figure 6.6: Question 6.

Web Application: Collaborate Pen Testing Organizer (CPTO)

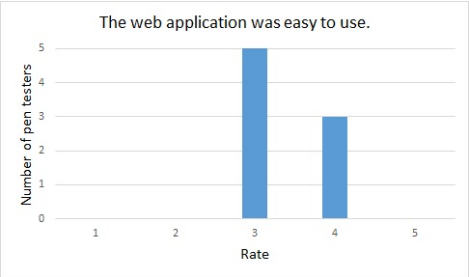

Figure 6.7: Question 7.

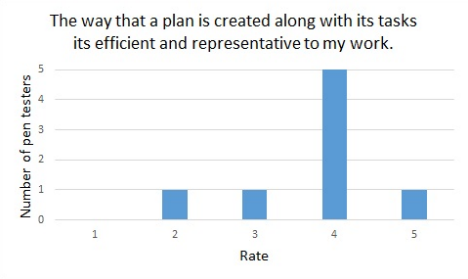

Figure 6.8: Question 8.

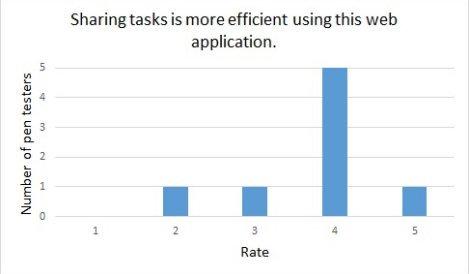

Figure 6.9: Question 9.

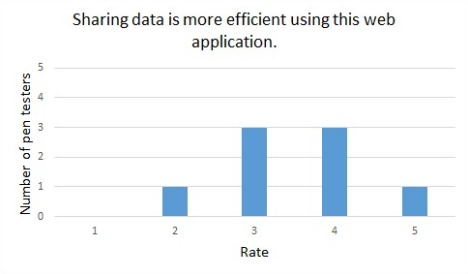

Figure 6.10: Question 10.

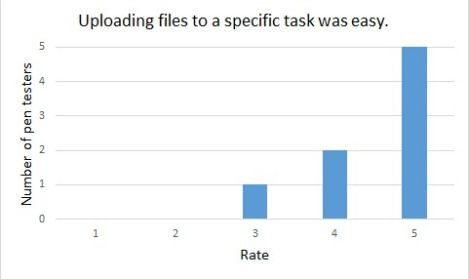

Figure 6.11: Question 11.

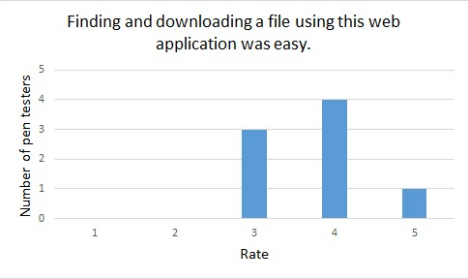

Figure 6.12: Question 12.

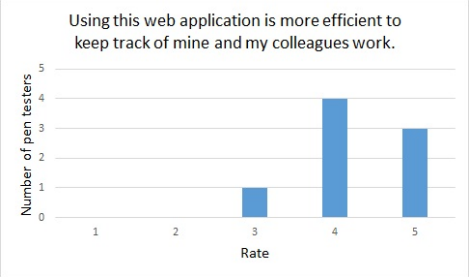

Figure 6.13: Question 13.

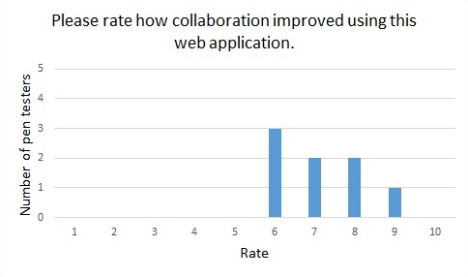

Figure 6.14: Question 14.

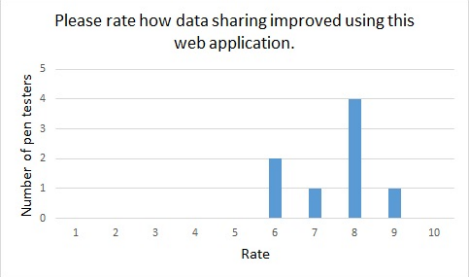

Figure 6.15: Question 15.

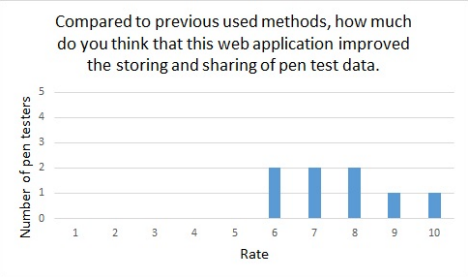

Figure 6.16: Question 16.

System as a whole

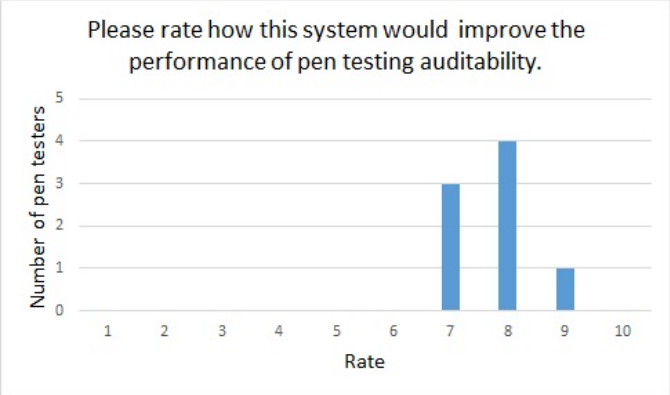

Figure 6.17: Question 17.

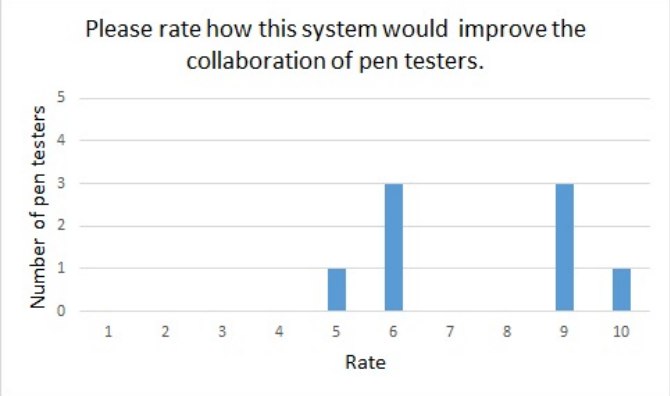

Figure 6.18: Question 18.

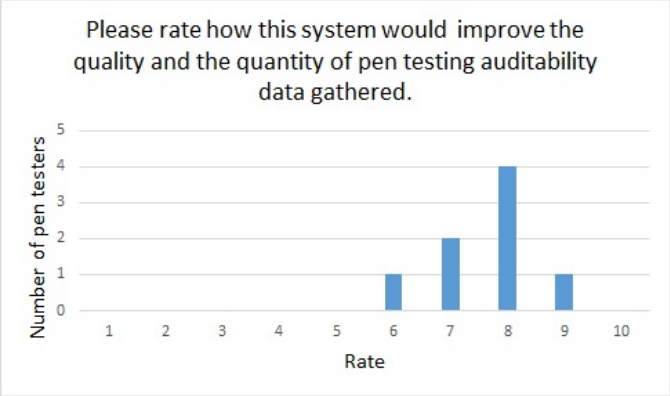

Figure 6.19: Question 19.

6.3.2 Script capture.py

After a week of continuous testing, our supervisor Christopher Mills replied with an important feedback. He stated that while using the script he did not face any errors or unreasonable behaviour. In addition, he managed to capture more actions than previous methods making his log more verbose and furthermore the `capture.py` managed to log failed attempts that might had left corrupted data to the targeted system. Using the timestamps he could easily correlate his actions with reported behaviour from the client of the targeted system.

6.4 Analysis

In this section is going to be performed an analysis on the data that we obtained from pen tester's answers to the mentioned questionnaire. The following table provides statistics about the mean and the median of the above questions answers.

Table 6.1: Statistics.

Question	Rate Scale	Mean	Median
Q1	1 - 5	3.625	4
Q2	1 - 5	4.25	4
Q3	1 - 5	4.125	4
Q4	1 - 5	4.25	4.5
Q5	1 - 5	4	4
Q6	1 - 10	8.625	8.5
Q7	1 - 5	3.375	3
Q8	1 - 5	3.75	4
Q9	1 - 5	3.75	4
Q10	1 - 5	3.5	3.5
Q11	1 - 5	4.5	5
Q12	1 - 5	3.625	3.5
Q13	1 - 5	4.25	4
Q14	1 - 10	7.125	7
Q15	1 - 10	7.5	8
Q16	1 - 10	7.625	7.5
Q17	1 - 10	7.75	8
Q18	1 - 10	7.5	7.5
Q19	1 - 10	7.625	8

Action Recording

Question 1 shows that the proposed suite of tools, indeed captures all the needed data for penetration testing auditability, as its statistics show that users agree with it. The tools chosen were also easy to use as the mean and the median are close to each other at the rate of 4, which means that users agree with it. In addition, those tools were not intrusive to pen testers as the statistics of question 3 show. Furthermore, looking at statistics of question 4, we can conclude that the tools indeed improved the performance of auditability, as the mean and the median show that users agree a bit stronger on that question. They also improve the quality and quantity of the captured data as the mean and the median of question 5, which are both equal to 4, conclude that users agree. The last question of "Action Recording" part (question 6) is devoted to the rating of the `capture.py`. Its statistics of mean and median show that `capture.py` improves significantly the process of auditing the command line.

To sum up, taking into consideration the questions of "Action Recording" part, the prototype managed to improve the process of auditing pen tester's actions during the penetration testing procedure. As the prototype is the implementation of our framework, this leads to the conclusion that the framework's sub-system Action Recording (see Chapter Design sub-section 4.2.1) indeed improves the penetration testing auditability.

Web Application: Collaborate Pen Testing Organizer (CPTO)

Question 7 shows that the implementation of the web application is on the correct path, but it will need some improvements on the human computer interaction as the statistics of mean and median are close to 3 showing that users are neutral on whether the web application is easy to use or not. The planning is representative to the pen tester's work and using the web application makes it more efficient as the statistics of question 8 show.

The process of task sharing is more efficient using the web application as users agree with it based on the statistics of question 9. Also, sharing of data among a pen tester's team (question 10) is a bit more efficient in our implementation as the statistics of both mean and median are equal to 3.5. Question 11 statistics show that the method implemented for uploading a file to a specific task is extremely easy. On the other hand, the process of finding a file using the web application (question 12) will need some improvement as the statistical values of mean and median are both close to 3.5. The web application achieved its goal on planning as question 13 statistics conclude that users agreed to that statement.

As far as collaboration improving rating concerns (question 14), the statistical values of mean and median are close to 7 showing that this web application indeed improves it. Additionally, question 15 proofs that the

process of sharing the penetration testing auditability data among a team of pen tester's was actually enhanced as mean is equal to 7.5 and median is equal to 8. Last but not least, the question 16, which ask whether this system improves data storing and sharing in comparison to previous used methods, the users responses showed that this implementation improves those aspects as both statistical values are close to 7.5.

The conclusion that was drawn based on the statistical analysis of questions 7 - 16 is that the web application indeed improves storing and sharing of penetration testing auditability data as well as enhances collaboration among pen testers. This part of the prototype is the implementation of the framework's sub-systems "Planning" and "Data Sharing / Storage" (see Chapter Design sub-sections 4.2.2 and 4.2.3). This is a proof that the suggested methodology for storing / sharing and planning indeed improves the penetration testing auditability procedure and the collaboration of pen tester's through it.

System as a whole

Questions 17 - 19 are the rating of the whole prototype, in order to see the rate of improvement on penetration testing auditability using it. First, we have question 17 which asks if the performance of penetration testing auditability was actually increased. Judging from the mean being equal to 7.75 and the median being equal to 8, we can conclude that performance is increased significantly.

Next in line is question 18, which helped us understand whether the system as a whole enhanced collaboration in a team of pen testers. With the values of mean and median being both equal to 7.5, we can interpret that the system achieves its goal on teamwork.

Lastly, question 19 asks the users to rate whether our prototype improved the quality and quantity of pen testing auditability data gathered. The responses show that it achieved that aim as the statistical values of mean and median are equal to 7.625 and 8 respectively.

To conclude, our prototype achieved all of its goals, that were based on collaboration, data gathering, data storing and sharing as well as planning and organizing the penetration testing auditability procedure. Although some of the prototype's features implementation need to be improved, the methodology still is in correct path. As the prototype is the implementation of the proposed framework, inevitably the proposed framework can improve penetration testing auditability on those aspects.

Chapter 7

Conclusion and Further work

7.1 Introduction

The aim of this chapter is to describe in both a general and a specific way, what was the outcome of our research as far as concerns the improvement on penetration testing auditability process. We will start by giving an overview of the project, meaning the process that took place in order to complete our research. We will also discuss the contribution of this project and how it can benefit the community. Last but not least, we are going to mention what we believe can be extended as further work.

7.2 Overview of the project

The project started by recognizing the problems that needed to be solved regarding the penetration testing auditability and followed by identifying our scope by establishing the research questions. Next in line, was the beginning of our research by interviews of penetration testers, questionnaires that were given to them and literature study.

After collecting all the needed knowledge, we proceeded with a design of a framework that can solve the identified problems. By the completion of the framework's design, we started the design of a prototype that was aimed as a proof of concept. The next phase in our research project was the implementation of the designed prototype.

The prototype was tested by pen testers, in order to evaluate its performance on penetration testing auditability. The results of these tests were gathered and statistically analysed, so that a conclusion could be drawn on whether the proposed framework was actually solving the identified problems of penetration testing auditability.

7.3 Answering the research questions

During our research project we managed to answer the established research questions. Our first question was related to what are the sources of penetration testing auditability data. The answer to that question consists of two types of actions, manual and automated actions. Manual actions were split into command line actions and other actions that used the environment. This leads to the conclusion, that command line is one source of the data and the other is the screen of the machine. We chose the screen as a source because we wanted to have a generalized source that can present any type of action. The automated actions were based on automated tools that were categorized into command line tools and GUI tools. Command line, again, is a source of pen testing auditability data as well as the screen of the machine, and now, additionally, we have the produced log files and reports of those tools.

The second research question was related to the methods that can be used to effectively audit the identified sources. Our proposed solution was to capture the command line streams and store them into a file along with timestamps of the commands executed and their outputs. An implementation of this solution is our Python script called `capture.py`. To capture the second source of data, the screen of the machine, we proposed a solution that includes taking screen shots as well as screen casting. In order to audit the automated tools, we can use their log files and reports as they are, and additionally the command line tools can be audited using the method of capturing command line streams, something that `capture.py` does.

We believe that a centralized storage is the solution of storing these data, because using this method everyone in a team can access these data. In addition, we recommend that the centralized storage should support any type of files, so that pen testers can keep their freedom and use any tool or method that they want.

Our last research question was referring to how can penetration testing auditability enhance collaboration among pen testers during penetration testing procedure. Our answer begins with using planning and task sharing. Auditability is based on recording actions of a task. In this perspective, planning and sharing of tasks can be included in auditability. Data sharing can also enhance collaboration. These data are the ones that are used in auditability. Our proposed methodology to increase teamwork using auditability is to provide the pen testers with a method of creating plans and tasks, relate those tasks with the data obtained from the penetration testing procedure and finally make those data accessible to the whole team of pen testers.

7.4 Contribution

Our research product was a framework that can improve penetration testing auditability. This framework is a standardized methodology that can increase the quality and quantity of auditability data. Also, it improves the performance of auditability process and enhances communication and collaboration within a pen testing team. The prototype created is an example of how the framework can be implemented and the test of it showed in which extent the auditability process and collaboration can be improved.

Our framework is proposed with freedom of pen testers in mind. As we stated at Literature Survey, Chapter 2, penetration testing is characterized as an art. This leads to the conclusion, that pen testers need to be allowed to use any tool or any method that they want for penetration testing purposes. That is why our prototype allows of uploading and download any type of file. In contradiction with other, already discussed, collaboration tools that use add-ons for specified tools as well as databases that limits the type of information that the users can enrol.

Moreover, the prototype proved that planning and task sharing can be included in penetration testing auditability and increase collaboration through it. The idea of creating tasks that a pen tester can share and relate those tasks with data proved to be a significant feature of the prototype, hence the framework.

7.5 Further Work

The framework might be enhanced after the completion of two types of research that we were not able to perform, due to the limited time that this project had. The first research is whether network traffic capturing can be used for penetration testing auditability purposes. We removed that as an option for our framework as packet capturing can consume a huge amount of data space. In addition, we have the problem of relating the packets with the actions performed and even worst, in case of secure channels the packets are encrypted. However, there is a possibility that packet headers might be helpful on penetration testing auditability when someone find out a method that relates them with a pen tester's actions.

Another research is to find out what type of data can be stored in a database in a unified way. In our project we tried to cover both network and application penetration testing. A valuable further work will be to find out a method that can store both of these pen testing types in a unified way, so that a user can query them.

Our `capture.py` script can capture the command line streams, which include the special characters that a terminal uses. These characters are mainly for design purposes, such as colours. This is both a feature and a

limitation. By using the `cat` command, the user can see exactly how the command line interface looked like. The problem that arises, on the other hand, is that if someone uses a normal editor, these special characters can make the produced log file difficult to read. A further work could be a tool that can filter and remove these characters.

Our prototype's main goal was to prove that our framework improves penetration testing auditability. This leads to the conclusion, that the prototype is not a completed working system. It needs to be implemented with security in mind, something that we did not have the time to do. Access controls lists (ACLs) is a feature that could actually be used to support this perspective. Moreover, the implementation of the database and the code of the web application were not created taking into consideration SQL injection attacks, code injection and fuzzing. A proper implementation of the framework must include security as the data stored in it are of high sensitivity.

Bibliography

[1] Asciinema. How it works. https://asciinema.org/docs/how-it-works. Online Access on 17-01-2016.

[2] Andrew Austin and Laurie Williams. One technique is not enough: A comparison of vulnerability discovery techniques. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 97–106. IEEE, 2011.

[3] Rahmat Budiarto, Sureswaran Ramadass, Azman Samsudin, and Salah Noori. Development of penetration testing model for increasing network security. 2004.

[4] CoreSecurity. Penetration testing overview. http://www.coresecurity.com/penetration-testing-overview. Online Access on 07-01-2016.

[5] Daniel Geer and John Harthorne. Penetration testing: A duet. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 185–195. IEEE, 2002.

[6] gobby.github.io. Gobby: a collaborative text editor. https://gobby.github.io/, 2015. Online Access on 24-01-2016.

[7] Gremwell. Video: Using magictree for analysing data. http://www.gremwell.com/video_using_magictree_for_analysing_data. Online Access on 06-01-2016.

[8] Gremwell. What is magictree? http://www.gremwell.com/what_is_magictree. Online Access on 06-01-2016.

[9] Kurt Grutzmacher. Welcome to kvasir! https://github.com/KvasirSecurity/Kvasir/wiki. Online Access on 06-01-2016.

[10] Kurt Grutzmacher. Introducing kvasir. http://blogs.cisco.com/security/introducing-kvasir, 23-09-2013. Online Access on 06-01-2016.

[11] Liwen He and Nikolai Bode. Network penetration testing. In *EC2ND 2005*, pages 3–12. Springer, 2006.

[12] html-form guide.com. Creating a registration form using php. `http://www.html-form-guide.com/php-form/php-registration-form.html`, 2012. Online Access on 27-01-2016.

[13] kali.org. What is kali linux ? `http://docs.kali.org/introduction/what-is-kali-linux`, 2015. Online Access on 24-01-2016.

[14] Michael Kerrisk. Script(1). `http://man7.org/linux/man-pages/man1/script.1.html`. Online Access on 06-01-2016.

[15] Michael Kerrisk. Scriptreplay(1). `http://man7.org/linux/man-pages/man1/scriptreplay.1.html`. Online Access on 06-01-2016.

[16] Linux Mint. iptux. `http://community.linuxmint.com/software/view/iptux`, 2014. Online Access on 24-01-2016.

[17] Nitin A Naik, Gajanan D Kurundkar, Santosh D Khamitkar, and Namdeo V Kalyankar. Penetration testing: A roadmap to network security. *arXiv preprint arXiv:0912.3970*, 2009.

[18] JAKOB NIELSEN. Thinking aloud: The #1 usability tool. `https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/`, 2012. Online Access on 28-01-2016.

[19] Charles P Pfleeger, Shari Lawrence Pfleeger, and Mary Frances Theofanos. A methodology for penetration testing. *Computers & Security*, 8(7):613–620, 1989.

[20] python.org. 33.8. pty âĂŤ pseudo-terminal utilities. `https://docs.python.org/3.2/library/pty.html`. Online Access on 24-01-2016.

[21] Margaret Rouse. Definition framework. `http://whatis.techtarget.com/definition/framework`, 2015. Online Access on 23-01-2016.

[22] Darrien Rushing, Jason Guidry, and Ihssan Alkadi. Collaborative penetration-testing and analysis toolkit (cpat). In *Aerospace Conference, 2015 IEEE*, pages 1–9. IEEE, 2015.

[23] Karen Scarfone, Murugiah Souppaya, Amanda Cody, and Angela Orebaugh. Technical guide to information security testing and assessment. *NIST Special Publication*, 800:115, 2008.

[24] shutter project.org. About. `http://shutter-project.org/about/`. Online Access on 24-01-2016.

[25] Tom Steele and Dan Kottmann. Def con 21 - tom steele and dan kottmann - collaborative penetration testing with lair. https://www.youtube.com/watch?v=3JwSsfx-H40, 2013. Online Access on 17-01-2016.

[26] Christian Weiske. Tomboy. https://wiki.gnome.org/Apps/Tomboy, 2014. Online Access on 24-01-2016.

[27] Wikipedia. Penetration test. https://en.wikipedia.org/wiki/Penetration_test. Online Access on 06-01-2016.

Appendix A

Code

A.1 capture.py

```
import os, time, sys
import pty
from datetime import datetime

def read(fd):
  data = os.read(fd, 1024)
  write = data
  if data.endswith('\n'):
    write= data + 'ESC[1;32m'+
       datetime.utcnow().strftime("%d-%m-%Y␣␣%H:%M:%S")
       +'ESC[0m'+ '\n'
  file.write(write)
  return data

shell = 'sh'
filename = raw_input('Enter␣the␣name␣of␣your␣log␣file:␣')
filename= filename + '_' +
   datetime.utcnow().strftime("%d-%m-%Y_%H:%M:%S")
mode = 'w'
if os.environ.has_key('SHELL'):
  shell = os.environ['SHELL']

file = open(filename, mode)
sys.stdout.write('Capture␣started,␣file␣is:␣%s\n' %
   filename)
file.write('Capture␣started␣at␣UTC␣time␣%s\n' %
   datetime.utcnow().strftime("%d-%m-%Y␣␣%H:%M:%S"))
pty.spawn(shell, read)
file.write('Capture␣stopped␣at␣UTC␣time␣%s\n' %
   datetime.utcnow().strftime("%d-%m-%Y␣␣%H:%M:%S"))
sys.stdout.write('Capture␣stopped,␣file␣is␣%s\n' %
   filename)
```

A.2 Home page

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="./bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="./scripts/jquery-1.12.0.min.js"></script>
    <script
       src="./bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Welcome to CPTO!</title>
  </head>
  <body>
    <div>
      <div class = "page-header␣text-center">
        <h1>Collaborate Pen Testing
           Organizer<br><small>Welcome!</small></h1>
      </div>
      <div class = "text-center">
        <h1><small>Please, login if you are an already
           registered pentester!</small></h1>
        <a href='./login/login.php' class="btn-primary␣
           btn-lg␣btn-block">Login</a><br>
      </div>
      <div class = "text-center">
        <h1><small>Please, sign up if you are a new
           pentester!</small></h1>
        <a href='./register/register.php'
           class="btn-primary␣btn-lg␣
           btn-block">Register</a>
      </div>
    </div>
  </body>
</html>
```

A.3 register.php

```
<?PHP
/*
  ------------------------------------------------------------------------

  The following code was taken and modified from:
  http://www.html-form-guide.com/files/php-form/RegistrationForm.zip

  ------------------------------------------------------------------------
*/

require_once("../include/membersite_config.php");
```

```
# Check if a user registration was submitted and if it
   was correct
# redirect to thank-you-regd.html
if(isset($_POST['submitted']))
{
   if($fgmembersite->RegisterUser())
   {
        $fgmembersite->RedirectToURL("./thank-you-regd.html");
   }
}

?>

<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Registration Form</title>
    <script type='text/javascript'
       src='../scripts/gen_validatorv31.js'></script>
    <link rel="STYLESHEET" type="text/css"
       href="../style/pwdwidget.css" />
    <script src="../scripts/pwdwidget.js"
       type="text/javascript"></script>
  </head>
  <body>
    <!-- Form Code Start -->
    <div id='fg_membersite'>
    <form id='register' action='<?php echo
       $fgmembersite->GetSelfScript(); ?>'
       method='post' accept-charset='UTF-8'>
      <fieldset>
        <div class = "page-header text-center">
          <h1>Collaborate Pen Testing
             Organizer<br><small>Register</small></h1>
        </div>

        <input type='hidden' name='submitted'
           id='submitted' value='1'/>

        <div class="text-center">* required
           fields</div><br/>
```

```
    <div><span class='error'><?php echo
       $fgmembersite->GetErrorMessage();
       ?></span></div>

    <div class="container text-center">
      <label for='name' >Your Full Name*:
         </label><br/>
      <input type='text' name='name' id='name'
         value='<?php echo
         $fgmembersite->SafeDisplay('name') ?>'
         maxlength="50" /><br/>
      <span id='register_name_errorloc'
         class='error'></span>
      <label for='email' >Email
         Address*:</label><br/>
      <input type='text' name='email' id='email'
         value='<?php echo
         $fgmembersite->SafeDisplay('email') ?>'
         maxlength="50" /><br/>
      <span id='register_email_errorloc'
         class='error'></span>
      <label for='username' >Username*:</label><br/>
      <input type='text' name='username'
         id='username' value='<?php echo
         $fgmembersite->SafeDisplay('username') ?>'
         maxlength="50" /><br/>
      <span id='register_username_errorloc'
         class='error'></span>
      <label for='password' >Password*:</label><br/>
      <div class="pagination centering"
         id='thepwddiv'></div>
      <noscript>
        <input type='password' name='password'
          id='password' maxlength="50" />
      </noscript>
      <div id='register_password_errorloc'
         class='error' style='clear:both'></div>
    </div>

    <div class="container text-center">
      <input type='submit' name='Submit'
         value='Submit' class="btn btn-primary"/>
    </div>
  </fieldset>
</form>
<script type='text/javascript'>
// <![CDATA[
  var pwdwidget = new
     PasswordWidget('thepwddiv','password');
```

```
      pwdwidget.MakePWDWidget();

      var frmvalidator  = new Validator("register");
      frmvalidator.EnableOnPageErrorDisplay();
      frmvalidator.EnableMsgsTogether();

      frmvalidator.addValidation("name","req","Please␣
         provide␣your␣name");

      frmvalidator.addValidation("email","req","Please␣
         provide␣your␣email␣address");

      frmvalidator.addValidation("email","email","Please␣
         provide␣a␣valid␣email␣address");

      frmvalidator.addValidation("username","req","Please␣
         provide␣a␣username");

      frmvalidator.addValidation("password","req","Please␣
         provide␣a␣password");

    // ]]>
    </script>
    <!--
    Form Code End
    -->
  </body>
</html>
```

A.4 thank-you-regd.html

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Thank you!</title>
  </head>
  <body>
    <div>
      <div class = "page-header␣text-center">
        <h1>Collaborate Pen Testing
           Organizer<br><small>Thanks for
           registering!</small></h1>
      </div>
```

```
      <div class = "text-center">
        <h2>Your registration is now complete!</h2>
        <a href='../login/login.php' class="btn-primary␣
            btn-lg␣btn-block">Click here to login</a>
      </div>
    </div>
  </body>
</html>
```

A.5 Plan page

```
<?php
require_once("../include/membersite_config.php");

# Check if a user is logged in
if(!$fgmembersite->CheckLogin())
{
    $fgmembersite->RedirectToURL("login.php");
    exit;
}
?>
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="../Style/dropzone.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <script type="text/javascript"
       src="../scripts/bootstrap-filestyle.min.js">
       </script>
    <title>Plan</title>
  </head>
  <body>
    <?php
      # Dir path of selected job
      $directory=($_POST['job']);
    ?>
    <div class = "page-header">
          <h1>Collaborate Pen Testing Organizer<br>
        <!-- Get username -->
        <small> User: <?= $fgmembersite->UserFullName();
           ?> </small> <a href='../logout/logout.php'
           class="btn␣btn-primary">Logout</a> <br>
        <small>Choose a task from the lists provided to
           work with</small>
```

```
  </h1>
</div>
<div class="container">
  <h2>Penetration Testing Phases</h2>
  <ul class="nav␣nav-pills">
    <li class="active"><a data-toggle="pill"
       href="#home">Introductory Planning and
       Preparation</a></li>
    <li><a data-toggle="pill"
       href="#menu1">Discovery and
       Investigation</a></li>
    <li><a data-toggle="pill"
       href="#menu2">Assessment and
       Strategy</a></li>
    <li><a data-toggle="pill"
       href="#menu3">Exploitation /
       Invasion</a></li>
    <li><a data-toggle="pill"
       href="#menu4">Maintaining Access</a></li>
    <li><a data-toggle="pill"
       href="#menu5">Reporting /
       Documenting</a></li>
  </ul>
  <div class="tab-content">
    <div id="home" class="tab-pane␣fade␣in␣active">
      <div class="panel␣panel-default"
         id="PlanningPreparation">
          <h3>Introductory Planning and
             Preparation</h3>
          <?php
            # Fill the tasks of the Introductory
               phase
            taskfill('Introductory',$directory);
          ?>
      </div>
      <form class="form-inline" role="form"
         method="post">
        <div class="form-group">
          <label for="Filename">Task Name: </label>
          <input type="text" class="form-control"
             name="name">
          <input type="hidden" name="phase"
             value="Introductory">
          <?php
            echo '<input type="hidden"
               name="directory"
               value="'.$directory.'">';
          ?>
        </div>
```

```
    <div class="form-group">
      <button type="submit" class="btn␣
         btn-primary"
         formaction="addtask.php">Add
         Task</button>
    </div>
  </form>
</div>
<div id="menu1" class="tab-pane␣fade">
  <div class="panel␣panel-default"
     id="PlanningPreparation">
    <h3>Discovery and Investigation</h3>
    <?php
        # Fill the tasks of the Discovery phase
        taskfill('Discovery',$directory);
    ?>
  </div>
  <form class="form-inline" role="form"
     method="post">
    <div class="form-group">
      <label for="Filename">Task Name: </label>
      <input type="text" class="form-control"
         name="name">
      <input type="hidden" name="phase"
         value="Discovery">
      <?php
        echo '<input type="hidden"
           name="directory"
           value="'.$directory.'">';
      ?>
    </div>
    <div class="form-group">
      <button type="submit" class="btn␣
         btn-primary"
         formaction="addtask.php">Add
         Task</button>
    </div>
  </form>
</div>
<div id="menu2" class="tab-pane␣fade">
  <div class="panel␣panel-default"
     id="PlanningPreparation">
    <h3>Assessment and Strategy</h3>
    <?php
        # Fill the tasks of the Assessment phase
        taskfill('Assessment',$directory);
    ?>
  </div>
  <form class="form-inline" role="form"
```

```
      method="post">
    <div class="form-group">
      <label for="Filename">Task Name: </label>
      <input type="text" class="form-control"
         name="name">
      <input type="hidden" name="phase"
         value="Assessment">
      <?php
        echo '<input type="hidden"
           name="directory"
           value="'.$directory.'">';
      ?>
    </div>
    <div class="form-group">
      <button type="submit" class="btn␣
         btn-primary"
         formaction="addtask.php">Add
         Task</button>
    </div>
  </form>
</div>
<div id="menu3" class="tab-pane␣fade">
  <div class="panel␣panel-default"
     id="PlanningPreparation">
    <h3>Exploitation / Invasion</h3>
    <?php
        # Fill the tasks of the Exploitation
           phase
        taskfill('Exploitation',$directory);
    ?>
  </div>
  <form class="form-inline" role="form"
     method="post">
    <div class="form-group">
      <label for="Filename">Task Name: </label>
      <input type="text" class="form-control"
         name="name">
      <input type="hidden" name="phase"
         value="Exploitation">
      <?php
        echo '<input type="hidden"
           name="directory"
           value="'.$directory.'">';
      ?>
    </div>
    <div class="form-group">
      <button type="submit" class="btn␣
         btn-primary"
         formaction="addtask.php">Add
```

```
          Task</button>
      </div>
    </form>
  </div>
  <div id="menu4" class="tab-pane␣fade">
    <div class="panel␣panel-default"
       id="PlanningPreparation">
      <h3>Maintaining Access</h3>
      <?php
          # Fill the tasks of the Maintaining phase
          taskfill('Maintaining',$directory);
      ?>
    </div>
    <form class="form-inline" role="form"
       method="post">
      <div class="form-group">
        <label for="Filename">Task Name: </label>
        <input type="text" class="form-control"
           name="name">
        <input type="hidden" name="phase"
           value="Maintaining">
        <?php
          echo '<input type="hidden"
             name="directory"
             value="'.$directory.'">';
        ?>
      </div>
      <div class="form-group">
        <button type="submit" class="btn␣
           btn-primary"
           formaction="addtask.php">Add
           Task</button>

      </div>
    </form>
  </div>
  <div id="menu5" class="tab-pane␣fade">
    <div class="panel␣panel-default"
       id="PlanningPreparation">
      <h3>Reporting / Documenting</h3>
      <?php
          # Fill the tasks of the Reporting phase
          taskfill('Reporting',$directory);
      ?>
    </div>
    <form class="form-inline" role="form"
       method="post">
      <div class="form-group">
        <label for="Filename">Task Name: </label>
```

```
            <input type="text" class="form-control"
               name="name">
            <input type="hidden" name="phase"
               value="Reporting">
            <?php
              echo '<input type="hidden"
                 name="directory"
                 value="'.$directory.'">';
            ?>
          </div>
          <div class="form-group">
            <button type="submit" class="btn␣
               btn-primary"
               formaction="addtask.php"><span
               class="glyphicon␣
               glyphicon-plus-sign"></span> Add
               Task</button>
          </div>
        </form>
      </div>
    </div>
  </div>
</body>
</html>
<?php
  function taskfill($phase,$directory ) {
    # Get array of all files and dirs inside the
       specified path
    $tasks = array_diff(scandir($directory."/".$phase),
       array('..', '.'));
    # Check if array is empty
    if (!empty($tasks)){
      # Loop through tasks
      foreach ($tasks as &$value) {
        # Check if color is red
        if (file_exists
           ($directory."/".$phase."/".$value."/CoLoR/red.color"
           ))
        {
          # Set the background to red
          $color='style="background-color:red"';
        # Check if color is green
        }elseif(file_exists
           ($directory."/".$phase."/".$value."/CoLoR/green.color"
           ))
        {
          # Set the background to green
          $color='style="background-color:lightgreen"';
        }
```

```
else
{
  # Set the background to white
  $color='style="background-color:white"';
}
# Print tasks and add remove button
echo '<div class="panel␣panel-primary">
    <div class="panel-heading␣
       form-inline">'.$value.'    
       <form class="form-group" role="form"
       method="post">
      <input type="hidden" name="value"
         value="'.$value.'">
      <input type="hidden" name="directory"
         value="'.$directory.'">
      <input type="hidden" name="phase"
         value="'.$phase.'">
      <button type="submit" class="btn␣
         btn-default" hspace="100"
         formaction="color.php" '.$color.'
         ></button>
    </form>   <form
       class="form-group" role="form"
       method="post">
      <input type="hidden" name="value"
         value="'.$value.'">
      <input type="hidden" name="directory"
         value="'.$directory.'">
      <input type="hidden" name="phase"
         value="'.$phase.'">
      <button type="submit" class="btn␣
         btn-danger␣btn-xs"
         formaction="deletetask.php"><span
         class="glyphicon␣
         glyphicon-remove-sign"></button></form></div>';
# Get array of all files and dirs inside the
   specified path
$users =
   array_diff(scandir($directory."/".$phase."/".$value),
   array('..', '.'));
# Check if array is greater than 1 (if its 1
   there no users, just the dir CoLoR)
if (sizeof($users) > 1){
  # Loop through the array
  foreach ($users as &$user) {
    # Check if user is CoLoR
    if($user!='CoLoR'){
      # Check if user dir is empty
      if (count ( glob
```

```
            ($directory."/".$phase."/".$value."/".$user."/*")
            ) === 0 )
        {
          # Remove dir
          rmdir($directory."/".$phase."/".$value."/".$user);
        }
        else
        {
          # Print user panel
          echo '<div class="panel panel-info">
              <div
                 class="panel-heading">'.$user.'</div>';
          # Get array of all files and dirs inside
             the specified path
          $files = array_diff( scandir
             ($directory."/".$phase."/".$value."/".$user),
             array('..', '.'));
          # Loop through the array
          foreach ($files as &$file) {
            # Print files and remove button
            echo '<div class="panel-body
               form-inline">
              <a href=" '.$directory.' '.$phase.
                 ' '.$value.' '.$user.' '
                 .$file.'" download>'.$file.'</a>
              <form class="form-group" role="form"
                 method="post">
                <input type="hidden" name="name"
                   value="'.$file.'">
                <input type="hidden"
                   name="directory"
                   value="'.$directory.'">
                <input type="hidden"
                   name="location"
                   value="'.$phase.'/'.$value.'/'.$user.'">
                <button type="submit" class="btn
                   btn-danger btn-xs"
                   formaction="deletefile.php"><span
                   class="glyphicon
                   glyphicon-remove-sign"></span></button>
              </form>
            </div>';
          }
          echo '</div>';
        }
      }
    }
  }
}
else
```

```
        {
          echo '<div class="panel-body">No Files</div>';
        }
        # Print upload form
        echo '<form enctype="multipart/form-data"
            action="upload.php" method="POST">
             <input type="hidden" name="value"
                value="'.$value.'">
             <input type="hidden" name="directory"
                value="'.$directory.'">
             <input type="hidden" name="phase"
                value="'.$phase.'">
             <input type="file" class="filestyle"
                data-buttonName="btn-primary"
                data-iconName="glyphicon␣
                glyphicon-inbox"
                data-buttonBefore="true" name="uploaded">
             <button class="btn␣btn-primary"
                type="submit"><span class="glyphicon␣
                glyphicon-upload"></span> Upload File
                </button></form>
          </form>
          </div>';
      }

    }
    else
    {
      echo '<div class="panel-heading">No Tasks</div>';
    }
  }
?>
```

A.6 addtask.php

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Create Task</title>
  </head>
  <body>
    <?php
      # Remove warning messages
```

```
      error_reporting(E_ALL ^ E_WARNING);
      $directory = $_POST["directory"];
      $filename = $_POST["name"];
      $phase = $_POST["phase"];
      # Check if filename was given
      if(!empty($filename)){
        $filename = $directory."/".$phase."/".$filename;
        # Create dir with name filename and inside
            create the CoLoR dir
        if (mkdir($filename,0700)){
          mkdir($filename."/CoLoR",0700);
          echo "<label><h1>You␣have␣succesfully␣created␣
              a␣new␣task.</h1></label>";
        }else{
          echo "<label><h1>An␣error␣occured.␣Propably␣
              the␣name␣given␣does␣already␣
              exist.</h1></label>";
        }
      }
      else{
        echo "<label><h1>Please␣provide␣a␣name␣for␣the␣
            new␣task.</h1></label>";
      }
    ?>
    <form action="./index.html"  method="post">
      <?php
        echo '<input type="hidden" name="job"
            value="'.$directory.'">';
      ?>
      <button class="btn␣btn-primary"
          type="submit"><span class="glyphicon␣
          glyphicon-circle-arrow-left"></span> Back
          </button>
    </form>
  </body>
</html>
```

A.7 color.php

```
<?php
  $directory = $_POST['directory'];
  $value= $_POST['value'];
  $phase = $_POST['phase'];

  # Check if file red.color exists
  if (file_exists
     ($directory."/".$phase."/".$value."/CoLoR/red.color"
     ))
  {
```

```
    # Delete red.color
    unlink
       ($directory."/".$phase."/".$value."/CoLoR/red.color"
       );
    # Create green.color
    touch($directory."/".$phase."/".$value."/CoLoR/green.color"
       );
  }
  # Check if file green.color exists
  elseif(file_exists
     ($directory."/".$phase."/".$value."/CoLoR/green.color"
     ))
  {
    # Delete green.color
    unlink
       ($directory."/".$phase."/".$value."/CoLoR/green.color"
       );
  }
  else
  {
    # Create red.color
    touch($directory."/".$phase."/".$value."/CoLoR/red.color");
  }
?>
<!DOCTYPE html>
<html>
  <form action="./index.html"  method="post"
     id='redirect'>
    <?php
      echo '<input type="hidden" name="job"
         value="'.$directory.'">';
    ?>
  </form>
</html>

<script type="text/javascript">
  // Call form action
  document.getElementById("redirect").submit();
</script>
```

A.8 deletefile.php

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
```

```
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Delete File</title>
  </head>
  <body>
    <?php
      # Remove warning messages
      error_reporting(E_ALL ^ E_WARNING);
      $directory = $_POST["directory"];
      $filename = $_POST["name"];
      $location = $_POST["location"];
      $path= $directory."/".$location;
      # Check if path is empty
      if(!empty($path)){
        $path = $path."/".$filename;
        # Check if file exists
        if(file_exists($path)){
          # Delete file
          unlink($path);
          echo "<label><h1>You␣have␣succesfully␣deleted␣
             a␣file.</h1></label>";
        }else{
          echo "<label><h1>The␣file␣provided␣does␣not␣
             exist.</h1></label>";
        }
      }else{
        echo "<label><h1>Please␣provide␣a␣name␣for␣the␣
           file␣to␣be␣deleted.</h1></label>";
      }
    ?>
    <form action="./index.html"  method="post">
      <?php
        echo '<input type="hidden" name="job"
           value="'.$directory.'">';
      ?>
      <button class="btn␣btn-primary"
         type="submit"><span class="glyphicon␣
         glyphicon-circle-arrow-left"></span> Back
         </button>
    </form>
  </body>
</html>
```

A.9 deletetask.php

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
```

```
      href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
      src="../scripts/jquery-1.12.0.min.js"></script>
    <script
      src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Delete Task</title>
  </head>
  <body>
    <?php
      # Remove warning messages
      error_reporting(E_ALL ^ E_WARNING);
      $directory = $_POST["directory"];
      $filename = $_POST["value"];
      $phase = $_POST["phase"];
      # Check if filename is empty
      if(!empty($filename)){
        $location = $directory."/".$phase."/".$filename;
        # Check if dir exists
        if(file_exists($location)){
          Delete($location);
          echo "<label><h1>You␣have␣succesfully␣deleted␣
            a␣task.</h1></label>";
        }else{
          echo "<label><h1>The␣task␣provided␣does␣not␣
            exist.</h1></label>";
        }
      }else{
        echo "<label><h1>Please␣provide␣a␣name␣for␣the␣
          task␣to␣be␣deleted.</h1></label>";
      }
    ?>
    <form action="./index.html"  method="post">
      <?php
        echo '<input type="hidden" name="job"
          value="'.$directory.'">';
      ?>
      <button class="btn␣btn-primary"
        type="submit"><span class="glyphicon␣
        glyphicon-circle-arrow-left"></span> Back
        </button>
    </form>
  </body>
</html>

<?php
  # Recursive delete of the files and dirs inside the
    specified path
  function Delete($path)
  {
```

```
    if (is_dir($path) === true)
    {
      $files = array_diff(scandir($path), array('.',
          '..'));
      foreach ($files as $file)
      {
        Delete(realpath($path) . '/' . $file);
      }
      return rmdir($path);
    }
    else if (is_file($path) === true)
    {
      return unlink($path);
    }
    return false;
  }
?>
```

A.10 upload.php

```
<?php
require_once("../include/membersite_config.php");

# Check if user is logged in
if(!$fgmembersite->CheckLogin())
{
    $fgmembersite->RedirectToURL("login.php");
    exit;
}
?>

<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Upload File</title>
  </head>
  <body>
    <?php
      $directory = $_POST['directory'];
      $task= $_POST['value'];
      $phase = $_POST['phase'];
      $ds = DIRECTORY_SEPARATOR;
      $user= $fgmembersite->UserFullName();
```

```
      $target =
         $directory.$ds.$phase.$ds.$task.$ds.$user.$ds;
      $target = $target . basename(
         $_FILES['uploaded']['name']) ;
      $ok=1;
      # Check if file exists
      if
         (!file_exists($directory.$ds.$phase.$ds.$task.$ds.$ds.$user))
      {
        # Create user dir
        mkdir($directory.$ds.$phase.$ds.$task.$ds.$ds.$user,0700);
      }
      # Move the uploaded file to the specified location
      if(move_uploaded_file($_FILES['uploaded']['tmp_name'],
         $target))
      {
        echo "<label><h1>The␣file␣".basename(
           $_FILES['uploaded']['name'])."␣␣has␣been␣
           uploaded.</h1></label>";
      }
      else
      {
        echo "<label><h1>Sorry,␣there␣was␣a␣problem␣
           uploading␣your␣file.</h1></label>";
      }
    ?>
    <form action="./index.html"  method="post">
      <?php
        echo '<input type="hidden" name="job"
           value="'.$directory.'">';
      ?>
      <button class="btn␣btn-primary"
         type="submit"><span class="glyphicon␣
         glyphicon-circle-arrow-left"></span> Back
         </button>
    </form>
  </body>
</html>
```

A.11 logout.php

```
<?PHP
/*
  ------------------------------------------------------------------------

  The following code was taken and modified from:
  http://www.html-form-guide.com/files/php-form/RegistrationForm.zip

  ------------------------------------------------------------------------
```

```
*/

require_once("../include/membersite_config.php");

$fgmembersite->LogOut();

?>

<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Logout</title>
    <script type='text/javascript'
       src='../scripts/gen_validatorv31.js'></script>
  </head>
  <body>
    <div>
      <div class = "page-header text-center">
        <h1>Collaborate Pen Testing
           Organizer<br><small>You have logged
           out</small></h1>
      </div>
      <div class = "text-center">
        <a href='../index.html' class="btn-primary
           btn-lg btn-block">Home</a><br>
        <a href='../login/login.php' class="btn-primary
           btn-lg btn-block">Login</a><br>
      </div>
    </div>
  </body>
</html>
```

A.12 login.php

```
<?PHP
/*
  ------------------------------------------------------------------------

  The following code was taken and modified from:
  http://www.html-form-guide.com/files/php-form/RegistrationForm.zip

  ------------------------------------------------------------------------
*/
```

```
require_once("../include/membersite_config.php");

# Check if user credentials are correct and if yes logs
   him/her in
# and redirects to login-home.php
if(isset($_POST['submitted']))
{
   if($fgmembersite->Login())
   {
        $fgmembersite->RedirectToURL("login-home.php");
   }
}

?>

<!DOCTYPE html PUBLIC>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Login</title>
    <script type='text/javascript'
       src='../scripts/gen_validatorv31.js'></script>
    <link rel="STYLESHEET" type="text/css"
       href="../style/pwdwidget.css" />
    <script src="../scripts/pwdwidget.js"
       type="text/javascript"></script>
  </head>
  <body>
    <!-- Form Code Start -->
    <div id='fg_membersite'>
      <form id='login' action='<?php echo
         $fgmembersite->GetSelfScript(); ?>'
         method='post' accept-charset='UTF-8'>
        <fieldset>
          <div class = "page-header text-center">
            <h1>Collaborate Pen Testing
               Organizer<br><small>Login</small></h1>
          </div>

          <input type='hidden' name='submitted'
             id='submitted' value='1'/>

          <div class="text-center">* required
```

```
            fields</div><br/>

        <div><span class='error'><?php echo
            $fgmembersite->GetErrorMessage();
            ?></span></div>
        <div class="container text-center">
          <label for='username'
            >Username*:</label><br/>
          <input type='text' name='username'
            id='username' value='<?php echo
            $fgmembersite->SafeDisplay('username')
            ?>' maxlength="50" /><br/>
          <span id='login_username_errorloc'
            class='error'></span>
        </div>

        <div class="container text-center">
          <label for='password'
            >Password*:</label><br/>
          <input type='password' name='password'
            id='password' maxlength="50" /><br/><br/>
          <span id='login_password_errorloc'
            class='error'></span>
        </div>

        <div class="container text-center">
            <input type='submit' name='Submit'
              value='Submit' class="btn
              btn-primary"/>
        </div>
      </fieldset>
    </form>
    <script type='text/javascript'>
    // <![CDATA[

      var frmvalidator  = new Validator("login");
      frmvalidator.EnableOnPageErrorDisplay();
      frmvalidator.EnableMsgsTogether();

      frmvalidator.addValidation("username","req","Please
         provide your username");

      frmvalidator.addValidation("password","req","Please
         provide the password");

    // ]]>
    </script>
</div>
<!--
```

```
    Form Code End
    -->
  </body>
</html>
```

A.13 login-home.php

```
<?PHP
/*
  ----------------------------------------------------------------------

  The following code was taken and modified from:
  http://www.html-form-guide.com/files/php-form/RegistrationForm.zip

  ----------------------------------------------------------------------
*/

require_once("../include/membersite_config.php");

# Check if a user is logged in
if(!$fgmembersite->CheckLogin())
{
    $fgmembersite->RedirectToURL("login.php");
    exit;
}

?>

<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Home page</title>
  </head>
  <body>
    <div>
      <div class = "page-header␣text-center">
        <h1>Collaborate Pen Testing
           Organizer<br><small>Home Page</small></h1>
      </div>
      <div class = "text-center">
        <h2>Welcome back <?=
           $fgmembersite->UserFullName(); ?>!</h2>
        <a href='../jobs/index.html' class="btn-primary␣
```

```
            btn-lg␣btn-block">Go to Projects</a></br>
          <a href='../changepassword/change-pwd.php'
            class="btn-primary␣btn-lg␣btn-block">Change
            password</a><br>
          <a href='../logout/logout.php'
            class="btn-primary␣btn-lg␣
            btn-block">Logout</a>
        </div>
      </div>
    </body>
</html>
```

A.14 Jobs page

```
<?php
require_once("../include/membersite_config.php");

# Check if a user is logged in
if(!$fgmembersite->CheckLogin())
{
    $fgmembersite->RedirectToURL("login.php");
    exit;
}
?>

<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Projects</title>
  </head>
  <body>
    <div class = "page-header">
      <h1>Collaborate Pen Testing Organizer<br>
        <!-- Get the username -->
        <small> User: <?= $fgmembersite->UserFullName();
           ?> </small> <a href='../logout/logout.php'
           class="btn␣btn-primary">Logout</a> <br>
        <small>Choose a project from the provided
           lists</small>
      </h1>
    </div>
    <div>
          <div class="list-group">
```

```
  <h2>Current Jobs</h2>
  <?php
    $directory = "../Projects/Current";
    # Get array of all files and dirs inside the
       specified path
    $jobs = array_diff(scandir($directory),
       array('..', '.'));
    # Check if array is empty
    if (!empty($jobs)){
      # Loop through the array and print current
         jobs
      foreach ($jobs as &$value) {
        # Print form that creates clickable links
           for the jobs and the achive button
        echo '<form class="form-inline"
           role="form" method="post">
              <input type="hidden" name="job"
                 value="'.$directory.'/'.$value.'">
              <input type="hidden" name="name"
                 value="'.$value.'">
              <button type="submit" class="btn␣
                 btn-default"
                 formaction="../plan/index.html"
                 >'.$value.'</button>
              <button type="submit" class="btn␣
                 btn-danger"
                 formaction="moveArchived.php"><span
                 class="glyphicon␣
                 glyphicon-floppy-disk"></button>
           </form>';
        }
    }
    else
    {
      echo '<div>No jobs</div>';
    }
  ?>
    </div>
<form class="form-inline" role="form"
   method="post">
  <div class="form-group">
    <label for="Filename">Job Name: </label>
    <input type="text" class="form-control"
       name="name">
  </div>
  <div class="form-group">
    <button type="submit" class="btn␣btn-primary"
       formaction="addcurent.php"><span
       class="glyphicon␣
```

```
            glyphicon-plus-sign">Create</button>
        </div>
      </form>
          <div class="list-group">
        <h2>Archived Jobs</h2>
        <?php
          $directory = "../Projects/Archived";
          # Get array of all files and dirs inside the
             specified path
          $jobs = array_diff(scandir($directory),
             array('..', '.'));
          # Check if array is empty
          if (!empty($jobs)){
            # Loop through the array and print archived
               jobs
            foreach ($jobs as &$value) {
              # Print form that creates clickable links
                 for the jobs and the delete button
              echo '<form class="form-inline"
                 role="form" method="post"
                 action="../plan/index.html">
                    <input type="hidden" name="job"
                       value="'.$directory.'/'.$value.'">
                    <input type="hidden" name="name"
                       value="'.$value.'">
                    <button type="submit" class="btn␣
                       btn-default"
                       formaction="../plan/index.html"
                       >'.$value.'</button>
                    <button type="submit" class="btn␣
                       btn-danger"
                       formaction="deleteFolder.php"><span
                       class="glyphicon␣
                       glyphicon-remove-sign"></button></button>
                 </form>';
            }
          }
          else
          {
            echo '<div>No jobs</div>';
          }
        ?>

          </div>
    </div>
  </body>
</html>
```

A.15 moveArchived.php

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Archive Project</title>
  </head>
  <body>
    <?php
      # Remove warning messages
      error_reporting(E_ALL ^ E_WARNING);
      $filename = $_POST["name"];
      # Check if filename was given
      if(!empty($filename)){
        $currentlocation =
           "../Projects/Current/".$filename;
        $newlocation = "../Projects/Archived/".$filename;
        # Move dir or file from old to new location
        if (rename($currentlocation,$newlocation)){
          echo "<label><h1>You␣have␣succesfully␣archived␣
             a␣project.</h1></label>";
        }else {
          echo "<label><h1>An␣error␣occured.␣Propably␣
             the␣name␣given␣does␣not␣
             exist.</h1></label>";
        }
      }else{
        echo "<label><h1>Please␣provide␣a␣name␣for␣the␣
           project␣to␣be␣Archived.</h1></label>";
      }
    ?>
    <form action="./index.html">
      <button class="btn␣btn-primary"
         type="submit"><span class="glyphicon␣
         glyphicon-circle-arrow-left"></span> Back
         </button>
    </form>
  </body>
</html>
```

A.16 deleteFolder.php

```
<!DOCTYPE html>
```

```
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Delete Project</title>
  </head>
  <body>
    <?php
      # Remove warning messages
      error_reporting(E_ALL ^ E_WARNING);
      $filename = $_POST["name"];
      # Check if filename is empty
      if(!empty($filename)){
        $location = "../Projects/Archived/".$filename;
        # If file exists, delete it
        if(file_exists($location)){
          Delete($location);
          echo "<label><h1>You␣have␣succesfully␣deleted␣
             a␣project.</h1></label>";
        }else{
          echo "<label><h1>The␣project␣provided␣does␣not␣
             exist.</h1></label>";
        }
      }else{
        echo "<label><h1>Please␣provide␣a␣name␣for␣the␣
           project␣to␣delete.</h1></label>";
      }
    ?>
    <form action="./index.html">
      <button class="btn␣btn-primary"
         type="submit"><span class="glyphicon␣
         glyphicon-circle-arrow-left"></span> Back
         </button>
    </form>
  </body>
</html>

<?php
  # Recursive delete of the files and dirs inside the
     specified path
  function Delete($path)
  {
    if (is_dir($path) === true)
    {
      $files = array_diff(scandir($path), array('.',
```

```
          '..'));
      foreach ($files as $file)
      {
          Delete(realpath($path) . '/' . $file);
      }
      return rmdir($path);
    }
    else if (is_file($path) === true)
    {
      return unlink($path);
    }
    return false;
  }
?>
```

A.17 addcurent.php

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Create Project</title>
  </head>
  <body>
    <?php
      # Remove warning messages
      error_reporting(E_ALL ^ E_WARNING);
      $filename = $_POST["name"];
      # Check if filename was given
      if(!empty($filename)){
        # Check if file exists
        if (file_exists (
           "../Projects/Archived/".$filename))
        {
          echo "<label><h1>The␣Project␣name␣exist␣in␣the␣
             Archived␣directory.</h1></label>";
        }else
        {
          # Create dir with name filename and inside
             create the needed dirs
          $filename = "../Projects/Current/".$filename;
          if (mkdir($filename,0700)){
            mkdir($filename."/Assessment",0700);
            mkdir($filename."/Discovery",0700);
```

```
            mkdir($filename."/Exploitation",0700);
            mkdir($filename."/Introductory",0700);
            mkdir($filename."/Maintaining",0700);
            mkdir($filename."/Reporting",0700);
            echo "<label><h1>You␣have␣succesfully␣
               created␣a␣new␣project.</h1></label>";
          }else{
            echo "<label><h1>An␣error␣occured.␣Propably␣
               the␣name␣given␣does␣already␣
               exist.</h1></label>";
          }
        }
      }
      else{
        echo "<label><h1>Please␣provide␣a␣name␣for␣the␣
           new␣project.</h1></label>";
      }
    ?>
    <form action="./index.html">
      <button class="btn␣btn-primary"
         type="submit"><span class="glyphicon␣
         glyphicon-circle-arrow-left"></span> Back
         </button>
    </form>
  </body>
</html>
```

A.18 fg_membersite.php

```
<?PHP
/*
  ----------------------------------------------------------------------
      Registration/Login script from HTML Form Guide
      Version 1.0

      This program is free software published under the
      terms of the GNU Lesser General Public License.
      http://www.gnu.org/copyleft/lesser.html

  The following code was taken and modified from:

  http://www.html-form-guide.com/files/php-form/RegistrationForm.zip
  http://www.html-form-guide.com/php-form/php-registration-form.html
  http://www.html-form-guide.com/php-form/php-login-form.html
  ----------------------------------------------------------------------
*/

require_once("formvalidator.php");
```

```
class FGMembersite
{
    var $admin_email;
    var $from_address;
    var $username;
    var $pwd;
    var $database;
    var $tablename;
    var $connection;
    var $rand_key;
    var $error_message;

    //-----Initialization -------
    function FGMembersite()
    {
        $this->sitename = 'os3.nl';
        $this->rand_key = '0iQx5oBk66oVZep';
    }

    function
        InitDB($host,$uname,$pwd,$database,$tablename)
    {
        $this->db_host  = $host;
        $this->username = $uname;
        $this->pwd  = $pwd;
        $this->database  = $database;
        $this->tablename = $tablename;

    }

    function SetAdminEmail($email)
    {
        $this->admin_email = $email;
    }

    function SetWebsiteName($sitename)
    {
        $this->sitename = $sitename;
    }

    function SetRandomKey($key)
    {
        $this->rand_key = $key;
    }

    //-------Main Operations ----------------------
    function RegisterUser()
    {
        if(!isset($_POST['submitted']))
```

```
    {
        return false;
    }

    $formvars = array();

    if(!$this->ValidateRegistrationSubmission())
    {
        return false;
    }

    $this->CollectRegistrationSubmission($formvars);

    if(!$this->SaveToDatabase($formvars))
    {
        return false;
    }
    return true;
}

function Login()
{
    if(empty($_POST['username']))
    {
        $this->HandleError("UserName is empty!");
        return false;
    }

    if(empty($_POST['password']))
    {
        $this->HandleError("Password is empty!");
        return false;
    }

    $username = trim($_POST['username']);
    $password = trim($_POST['password']);

    if(!isset($_SESSION)){ session_start(); }
    if(!$this->CheckLoginInDB($username,$password))
    {
        return false;
    }

    $_SESSION[$this->GetLoginSessionVar()] =
        $username;

    return true;
}
```

```
function CheckLogin()
{
      if(!isset($_SESSION)){ session_start(); }

      $sessionvar = $this->GetLoginSessionVar();

      if(empty($_SESSION[$sessionvar]))
      {
          return false;
      }
      return true;
}

function UserFullName()
{
     return
         isset($_SESSION['name_of_user'])?$_SESSION['name_of_user']:'';
}

function UserEmail()
{
     return
         isset($_SESSION['email_of_user'])?$_SESSION['email_of_user']:'';
}

function LogOut()
{
     session_start();

     $sessionvar = $this->GetLoginSessionVar();

     $_SESSION[$sessionvar]=NULL;

     unset($_SESSION[$sessionvar]);
}

function ChangePassword()
{
     if(!$this->CheckLogin())
     {
          $this->HandleError("Not logged in!");
          return false;
     }

     if(empty($_POST['oldpwd']))
     {
          $this->HandleError("Old password is empty!");
          return false;
     }
```

```
    if(empty($_POST['newpwd']))
    {
        $this->HandleError("New password is empty!");
        return false;
    }

    $user_rec = array();
    if(!$this->GetUserFromEmail($this->UserEmail(),$user_rec))
    {
        return false;
    }

    $pwd = trim($_POST['oldpwd']);

    if($user_rec['password'] != md5($pwd))
    {
        $this->HandleError("The old password does
           not match!");
        return false;
    }
    $newpwd = trim($_POST['newpwd']);

    if(!$this->ChangePasswordInDB($user_rec,
       $newpwd))
    {
        return false;
    }
    return true;
}

//-------Public Helper functions -------------
function GetSelfScript()
{
    return htmlentities($_SERVER['PHP_SELF']);
}

function SafeDisplay($value_name)
{
    if(empty($_POST[$value_name]))
    {
        return'';
    }
    return htmlentities($_POST[$value_name]);
}

function RedirectToURL($url)
{
    header("Location: $url");
    exit;
```

```
}

function GetSpamTrapInputName()
{
    return 'sp'.md5('KHGdnbvsgst'.$this->rand_key);
}

function GetErrorMessage()
{
    if(empty($this->error_message))
    {
        return '';
    }
    $errormsg =
       nl2br(htmlentities($this->error_message));
    return $errormsg;
}

//-------Private Helper functions-----------
function HandleError($err)
{
    $this->error_message .= $err."\r\n";
}

function HandleDBError($err)
{
    $this->HandleError($err."\r\n␣
       mysqlerror:".mysql_error());
}

function GetFromAddress()
{
    if(!empty($this->from_address))
    {
        return $this->from_address;
    }

    $host = $_SERVER['SERVER_NAME'];

    $from ="nobody@$host";
    return $from;
}

function GetLoginSessionVar()
{
    $retvar = md5($this->rand_key);
    $retvar = 'usr_'.substr($retvar,0,10);
    return $retvar;
}
```

```
function CheckLoginInDB($username,$password)
{
    if(!$this->DBLogin())
    {
        $this->HandleError("Database␣login␣failed!");
        return false;
    }
    $username = $this->SanitizeForSQL($username);
    $pwdmd5 = md5($password);
    $qry = "Select␣name,␣email␣from␣$this->tablename␣
       where␣username='$username'␣and␣
       password='$pwdmd5'";

    $result = mysql_query($qry,$this->connection);

    if(!$result || mysql_num_rows($result) <= 0)
    {
        $this->HandleError("Error␣logging␣in.␣The␣
           username␣or␣password␣does␣not␣match");
        return false;
    }

    $row = mysql_fetch_assoc($result);

    $_SESSION['name_of_user']  = $row['name'];
    $_SESSION['email_of_user'] = $row['email'];

    return true;
}

function ResetUserPasswordInDB($user_rec)
{
    $new_password = substr(md5(uniqid()),0,10);

    if(false ==
       $this->ChangePasswordInDB($user_rec,$new_password))
    {
        return false;
    }
    return $new_password;
}

function ChangePasswordInDB($user_rec, $newpwd)
{
    $newpwd = $this->SanitizeForSQL($newpwd);

    $qry = "Update␣$this->tablename␣Set␣
       password='".md5($newpwd)."'␣Where␣␣
```

```
        id_user=".$user_rec['id_user']."";

    if(!mysql_query( $qry ,$this->connection))
    {
        $this->HandleDBError("Error␣updating␣the␣
            password␣\nquery:$qry");
        return false;
    }
    return true;
}

function GetUserFromEmail($email,&$user_rec)
{
    if(!$this->DBLogin())
    {
        $this->HandleError("Database␣login␣failed!");
        return false;
    }
    $email = $this->SanitizeForSQL($email);

    $result = mysql_query("Select␣*␣from␣
        $this->tablename␣where␣
        email='$email'",$this->connection);

    if(!$result || mysql_num_rows($result) <= 0)
    {
        $this->HandleError("There␣is␣no␣user␣with␣
            email:␣$email");
        return false;
    }
    $user_rec = mysql_fetch_assoc($result);

    return true;
}

function GetResetPasswordCode($email)
{
   return
      substr(md5($email.$this->sitename.$this->rand_key),0,10);
}

function ValidateRegistrationSubmission()
{
    //This is a hidden input field. Humans won't
        fill this field.
    if(!empty($_POST[$this->GetSpamTrapInputName()])
        )
    {
        //The proper error is not given intentionally
```

```
            $this->HandleError("Automated␣submission␣
               prevention:␣case␣2␣failed");
            return false;
        }

        $validator = new FormValidator();
        $validator->addValidation("name","req","Please␣
           fill␣in␣Name");
        $validator->addValidation("email","email","The␣
           input␣for␣Email␣should␣be␣a␣valid␣email␣
           value");
        $validator->addValidation("email","req","Please␣
           fill␣in␣Email");
        $validator->addValidation("username","req","Please␣
           fill␣in␣UserName");
        $validator->addValidation("password","req","Please␣
           fill␣in␣Password");

        if(!$validator->ValidateForm())
        {
            $error='';
            $error_hash = $validator->GetErrors();
            foreach($error_hash as $inpname => $inp_err)
            {
                $error .= $inpname.':'.$inp_err."\n";
            }
            $this->HandleError($error);
            return false;
        }
        return true;
    }

    function CollectRegistrationSubmission(&$formvars)
    {
        $formvars['name'] =
           $this->Sanitize($_POST['name']);
        $formvars['email'] =
           $this->Sanitize($_POST['email']);
        $formvars['username'] =
           $this->Sanitize($_POST['username']);
        $formvars['password'] =
           $this->Sanitize($_POST['password']);
    }

    function SaveToDatabase(&$formvars)
    {
        if(!$this->DBLogin())
        {
            $this->HandleError("Database␣login␣failed!");
```

```
            return false;
        }
        if(!$this->Ensuretable())
        {
            return false;
        }
        if(!$this->IsFieldUnique($formvars,'email'))
        {
            $this->HandleError("This email is already
               registered");
            return false;
        }

        if(!$this->IsFieldUnique($formvars,'username'))
        {
            $this->HandleError("This UserName is already
               used. Please try another username");
            return false;
        }
        if(!$this->InsertIntoDB($formvars))
        {
            $this->HandleError("Inserting to Database
               failed!");
            return false;
        }
        return true;
    }

    function IsFieldUnique($formvars,$fieldname)
    {
        $field_val =
           $this->SanitizeForSQL($formvars[$fieldname]);
        $qry = "select username from $this->tablename
           where $fieldname='".$field_val."'";
        $result = mysql_query($qry,$this->connection);
        if($result && mysql_num_rows($result) > 0)
        {
            return false;
        }
        return true;
    }

    function DBLogin()
    {

        $this->connection =
           mysql_connect($this->db_host,$this->username,$this->pwd);

        if(!$this->connection)
```

```
    {
        $this->HandleDBError("Database␣Login␣failed!␣
            Please␣make␣sure␣that␣the␣DB␣login␣
            credentials␣provided␣are␣correct");
        return false;
    }
    if(!mysql_select_db($this->database,
        $this->connection))
    {
        $this->HandleDBError('Failed to select
            database: '.$this->database.' Please
            make sure that the database name
            provided is correct');
        return false;
    }
    if(!mysql_query("SET␣NAMES␣
        'UTF8'",$this->connection))
    {
        $this->HandleDBError('Error setting utf8
            encoding');
        return false;
    }
    return true;
}

function Ensuretable()
{
    $result = mysql_query("SHOW␣COLUMNS␣FROM␣
        $this->tablename");
    if(!$result || mysql_num_rows($result) <= 0)
    {
        return $this->CreateTable();
    }
    return true;
}

function CreateTable()
{
    $qry = "Create␣Table␣$this->tablename␣(".
            "id_user␣INT␣NOT␣NULL␣AUTO_INCREMENT␣,".
            "name␣VARCHAR(␣128␣)␣NOT␣NULL␣,".
            "email␣VARCHAR(␣64␣)␣NOT␣NULL␣,".
            "username␣VARCHAR(␣16␣)␣NOT␣NULL␣,".
            "password␣VARCHAR(␣32␣)␣NOT␣NULL␣,".
            "PRIMARY␣KEY␣(␣id_user␣)".
            ")";

    if(!mysql_query($qry,$this->connection))
    {
```

```
        $this->HandleDBError("Error␣creating␣the␣
            table␣\nquery␣was\n␣$qry");
        return false;
    }
    return true;
}

function InsertIntoDB(&$formvars)
{
    $insert_query = 'insert into
        '.$this->tablename.'(
            name,
            email,
            username,
            password
            )
            values
            (
            '',␣.␣
                $this->SanitizeForSQL($formvars['name'])␣
                .␣'',
            '',␣.␣
                $this->SanitizeForSQL($formvars['email'])␣
                .␣'',
            '',␣.␣
                $this->SanitizeForSQL($formvars['username'])␣
                .␣'',
            '',␣.␣md5($formvars['password'])␣.␣'"
            )';
    if(!mysql_query( $insert_query
        ,$this->connection))
    {
        $this->HandleDBError("Error␣inserting␣data␣
            to␣the␣table\nquery:$insert_query");
        return false;
    }
    return true;
}

function SanitizeForSQL($str)
{
    if( function_exists( "mysql_real_escape_string"
        ) )
    {
          $ret_str = mysql_real_escape_string( $str
              );
    }
    else
    {
```

```
            $ret_str = addslashes( $str );
        }
        return $ret_str;
    }

    /*
    Sanitize() function removes any potential threat
       from the
    data submitted. Prevents email injections or any
       other hacker attempts.
    if $remove_nl is true, newline characters are
       removed from the input.
    */
    function Sanitize($str,$remove_nl=true)
    {
        $str = $this->StripSlashes($str);

        if($remove_nl)
        {
            $injections = array('/(\n+)/i',
                '/(\r+)/i',
                '/(\t+)/i',
                '/(%0A+)/i',
                '/(%0D+)/i',
                '/(%08+)/i',
                '/(%09+)/i'
                );
            $str = preg_replace($injections,'',$str);
        }

        return $str;
    }

    function StripSlashes($str)
    {
        if(get_magic_quotes_gpc())
        {
            $str = stripslashes($str);
        }
        return $str;
    }
}

?>
```

A.19 formvalidator.php

```
<?PHP
/*
```

```
  ---------------------------------------------------------------------
        PHP Form Validator (formvalidator.php)
      Version 1.1

  This program is free software published under the
  terms of the GNU Lesser General Public License.
  http://www.gnu.org/copyleft/lesser.html

  The following code was taken without any change from:

  http://www.html-form-guide.com/files/php-form/RegistrationForm.zip
  http://www.html-form-guide.com/php-form/php-form-validation.html
  ---------------------------------------------------------------------
*/

/**
* Carries information about each of the form validations
*/
class ValidatorObj
{
  var $variable_name;
  var $validator_string;
  var $error_string;
}

/**
* Base class for custom validation objects
**/
class CustomValidator
{
  function DoValidate(&$formars,&$error_hash)
  {
    return true;
  }
}

/** Default error messages **/
define("E_VAL_REQUIRED_VALUE","Please␣enter␣the␣value␣
   for␣%s");
define("E_VAL_MAXLEN_EXCEEDED","Maximum␣length␣exceeded␣
   for␣%s.");
define("E_VAL_MINLEN_CHECK_FAILED","Please␣enter␣input␣
   with␣length␣more␣than␣%d␣for␣%s");
define("E_VAL_ALNUM_CHECK_FAILED","Please␣provide␣an␣
   alpha-numeric␣input␣for␣%s");
define("E_VAL_ALNUM_S_CHECK_FAILED","Please␣provide␣an␣
   alpha-numeric␣input␣for␣%s");
define("E_VAL_NUM_CHECK_FAILED","Please␣provide␣numeric␣
   input␣for␣%s");
```

```
define("E_VAL_ALPHA_CHECK_FAILED","Please␣provide␣
    alphabetic␣input␣for␣%s");
define("E_VAL_ALPHA_S_CHECK_FAILED","Please␣provide␣
    alphabetic␣input␣for␣%s");
define("E_VAL_EMAIL_CHECK_FAILED","Please␣provide␣a␣
    valida␣email␣address");
define("E_VAL_LESSTHAN_CHECK_FAILED","Enter␣a␣value␣less␣
    than␣%f␣for␣%s");
define("E_VAL_GREATERTHAN_CHECK_FAILED","Enter␣a␣value␣
    greater␣than␣%f␣for␣%s");
define("E_VAL_REGEXP_CHECK_FAILED","Please␣provide␣a␣
    valid␣input␣for␣%s");
define("E_VAL_DONTSEL_CHECK_FAILED","Wrong␣option␣
    selected␣for␣%s");
define("E_VAL_SELMIN_CHECK_FAILED","Please␣select␣
    minimum␣%d␣options␣for␣%s");
define("E_VAL_SELONE_CHECK_FAILED","Please␣select␣an␣
    option␣for␣%s");
define("E_VAL_EQELMNT_CHECK_FAILED","Value␣of␣%s␣should␣
    be␣same␣as␣that␣of␣%s");
define("E_VAL_NEELMNT_CHECK_FAILED","Value␣of␣%s␣should␣
    not␣be␣same␣as␣that␣of␣%s");

/**
* FormValidator: The main class that does all the form
    validations
**/
class FormValidator
{
  var $validator_array;
    var $error_hash;
  var $custom_validators;

  function FormValidator()
  {
    $this->validator_array = array();
        $this->error_hash = array();
    $this->custom_validators=array();
  }

  function AddCustomValidator(&$customv)
  {
    array_push($this->custom_validators,$customv);
  }

  function addValidation($variable,$validator,$error)
  {
    $validator_obj = new ValidatorObj();
    $validator_obj->variable_name = $variable;
```

```
  $validator_obj->validator_string = $validator;
  $validator_obj->error_string = $error;
  array_push($this->validator_array,$validator_obj);
}

  function GetErrors()
  {
      return $this->error_hash;
  }

function ValidateForm()
{
  $bret = true;

  $error_string="";
  $error_to_display = "";

  if(strcmp($_SERVER['REQUEST_METHOD'],'POST')==0)
  {
    $form_variables = $_POST;
  }
  else
  {
    $form_variables = $_GET;
  }

      $vcount = count($this->validator_array);

  foreach($this->validator_array as $val_obj)
  {
    if(!$this->ValidateObject($val_obj,$form_variables,$error_string))
    {
      $bret = false;
              $this->error_hash[$val_obj->variable_name]
                 = $error_string;
    }
  }

  if(true == $bret && count($this->custom_validators)
     > 0)
  {
          foreach( $this->custom_validators as
             $custom_val)
    {
      if(false ==
         $custom_val->DoValidate($form_variables,$this->error_hash))
      {
        $bret = false;
      }
```

```
    }
  }
  return $bret;
}

function
   ValidateObject($validatorobj,$formvariables,&$error_string)
{
  $bret = true;

  $splitted =
     explode("=",$validatorobj->validator_string);
  $command = $splitted[0];
  $command_value = '';

  if(isset($splitted[1]) && strlen($splitted[1])>0)
  {
    $command_value = $splitted[1];
  }

  $default_error_message="";

  $input_value ="";

  if(isset($formvariables[$validatorobj->variable_name]))
  {
   $input_value =
      $formvariables[$validatorobj->variable_name];
  }

  $bret =
     $this->ValidateCommand($command,$command_value,$input_value,
                $default_error_message,
                $validatorobj->variable_name,
                $formvariables);

  if(false == $bret)
  {
    if(isset($validatorobj->error_string) &&
      strlen($validatorobj->error_string)>0)
    {
      $error_string = $validatorobj->error_string;
    }
    else
    {
      $error_string = $default_error_message;
    }

  }
```

```
  return $bret;
}

function validate_req($input_value,
   &$default_error_message, $variable_name)
{
  $bret = true;
      if(!isset($input_value) ||
    strlen($input_value) <=0)
  {
    $bret=false;
    $default_error_message =
        sprintf(E_VAL_REQUIRED_VALUE,$variable_name);
  }
  return $bret;
}

function validate_maxlen($input_value,$max_len,
   $variable_name,&$default_error_message)
{
  $bret = true;
  if(isset($input_value) )
  {
    $input_length = strlen($input_value);
    if($input_length > $max_len)
    {
      $bret=false;
      $default_error_message =
          sprintf(E_VAL_MAXLEN_EXCEEDED,$variable_name);
    }
  }
  return $bret;
}

function
   validate_minlen($input_value,$min_len,$variable_name,
   &$default_error_message)
{
  $bret = true;
  if(isset($input_value) )
  {
    $input_length = strlen($input_value);
    if($input_length < $min_len)
    {
      $bret=false;
      $default_error_message =
          sprintf(E_VAL_MINLEN_CHECK_FAILED,$min_len,$variable_name);
    }
  }
```

```
  return $bret;
}

function test_datatype($input_value,$reg_exp)
{
  if(ereg($reg_exp,$input_value))
  {
    return false;
  }
  return true;
}

function validate_email($email)
{
  return preg_match(
     "/^[_\.0-9a-zA-Z-]+@([0-9a-zA-Z][0-9a-zA-Z-]+\.)+[a-zA-Z]{2,6}$/i",
     $email);
}

function
   validate_for_numeric_input($input_value,&$validation_success)
{

  $more_validations=true;
  $validation_success = true;
  if(strlen($input_value)>0)
  {

    if(false == is_numeric($input_value))
    {
      $validation_success = false;
      $more_validations=false;
    }
  }
  else
  {
    $more_validations=false;
  }
  return $more_validations;
}

function validate_lessthan($command_value,$input_value,
             $variable_name,&$default_error_message)
{
  $bret = true;
  if(false ==
     $this->validate_for_numeric_input($input_value,
                                       $bret))
  {
```

```
    return $bret;
  }
  if($bret)
  {
    $lessthan = doubleval($command_value);
    $float_inputval = doubleval($input_value);
    if($float_inputval >= $lessthan)
    {
      $default_error_message =
         sprintf(E_VAL_LESSTHAN_CHECK_FAILED,
                  $lessthan,
                  $variable_name);
      $bret = false;
    }
  }
  return $bret ;
}

function
   validate_greaterthan($command_value,$input_value,
   $variable_name,&$default_error_message)
{
  $bret = true;
  if(false ==
     $this->validate_for_numeric_input($input_value,$bret))
  {
    return $bret;
  }
  if($bret)
  {
    $greaterthan = doubleval($command_value);
    $float_inputval = doubleval($input_value);
    if($float_inputval <= $greaterthan)
    {
      $default_error_message =
         sprintf(E_VAL_GREATERTHAN_CHECK_FAILED,
                  $greaterthan,
                  $variable_name);
      $bret = false;
    }
  }
  return $bret ;
}

  function
     validate_select($input_value,$command_value,
     &$default_error_message,$variable_name)
  {
    $bret=false;
```

```
  if(is_array($input_value))
  {
    foreach($input_value as $value)
    {
      if($value == $command_value)
      {
        $bret=true;
        break;
      }
    }
  }
  else
  {
    if($command_value == $input_value)
    {
      $bret=true;
    }
  }
      if(false == $bret)
      {
          $default_error_message =
             sprintf(E_VAL_SHOULD_SEL_CHECK_FAILED,
                                          $command_value,$variable_name);
      }
    return $bret;
  }

function
    validate_dontselect($input_value,$command_value,
    &$default_error_message,$variable_name)
{
    $bret=true;
  if(is_array($input_value))
  {
    foreach($input_value as $value)
    {
      if($value == $command_value)
      {
        $bret=false;
        $default_error_message =
           sprintf(E_VAL_DONTSEL_CHECK_FAILED,$variable_name);
        break;
      }
    }
  }
  else
  {
    if($command_value == $input_value)
    {
```

```
      $bret=false;
      $default_error_message =
         sprintf(E_VAL_DONTSEL_CHECK_FAILED,$variable_name);
    }
  }
  return $bret;
}

function
   ValidateCommand($command,$command_value,$input_value,
   &$default_error_message,$variable_name,$formvariables)
{
  $bret=true;
  switch($command)
  {
    case 'req':
          {
            $bret = $this->validate_req($input_value,
               $default_error_message,$variable_name);
            break;
          }

    case 'maxlen':
          {
            $max_len = intval($command_value);
            $bret =
               $this->validate_maxlen($input_value,$max_len,
               $variable_name,
                      $default_error_message);
            break;
          }

    case 'minlen':
          {
            $min_len = intval($command_value);
            $bret =
               $this->validate_minlen($input_value,$min_len,
               $variable_name,
                    $default_error_message);
            break;
          }

    case 'alnum':
          {
            $bret=
               $this->test_datatype($input_value,"[^A-Za-z0-9]");
            if(false == $bret)
            {
              $default_error_message =
```

```
                sprintf(E_VAL_ALNUM_CHECK_FAILED,$variable_name);
            }
            break;
        }

    case 'alnum_s':
        {
            $bret=
                $this->test_datatype($input_value,"[^A-Za-z0-9]");
            if(false == $bret)
            {
              $default_error_message =
                sprintf(E_VAL_ALNUM_S_CHECK_FAILED,$variable_name);
            }
            break;
        }

    case 'num':
        case 'numeric':
        {
            $bret=
                $this->test_datatype($input_value,"[^0-9]");
            if(false == $bret)
            {
              $default_error_message =
                sprintf(E_VAL_NUM_CHECK_FAILED,$variable_name);
            }
            break;
        }

    case 'alpha':
        {
            $bret=
                $this->test_datatype($input_value,"[^A-Za-z]");
            if(false == $bret)
            {
              $default_error_message =
                sprintf(E_VAL_ALPHA_CHECK_FAILED,$variable_name);
            }
            break;
        }
    case 'alpha_s':
        {
            $bret=
                $this->test_datatype($input_value,"[^A-Za-z␣
                ]");
            if(false == $bret)
            {
              $default_error_message =
```

```
                sprintf(E_VAL_ALPHA_S_CHECK_FAILED,$variable_name);
          }
          break;
        }
case 'email':
        {
          if(isset($input_value) &&
             strlen($input_value)>0)
          {
            $bret=
               $this->validate_email($input_value);
            if(false == $bret)
            {
              $default_error_message =
                 E_VAL_EMAIL_CHECK_FAILED;
            }
          }
          break;
        }
case "lt":
case "lessthan":
        {
          $bret =
             $this->validate_lessthan($command_value,
                      $input_value,
                      $variable_name,
                      $default_error_message);
          break;
        }
case "gt":
case "greaterthan":
        {
          $bret =
             $this->validate_greaterthan($command_value,
                      $input_value,
                      $variable_name,
                      $default_error_message);
          break;
        }

case "regexp":
        {
          if(isset($input_value) &&
             strlen($input_value)>0)
          {
            if(!preg_match("$command_value",$input_value))
            {
              $bret=false;
              $default_error_message =
```

```
                sprintf(E_VAL_REGEXP_CHECK_FAILED,$variable_name);
            }
          }
          break;
        }
case "dontselect":
case "dontselectchk":
    case "dontselectradio":
      {
        $bret =
           $this->validate_dontselect($input_value,
                           $command_value,
                           $default_error_message,
                          $variable_name);
         break;
      }

    case "shouldselchk":
    case "selectradio":
                {
                       $bret =
                          $this->validate_select($input_value,
                $command_value,
                $default_error_message,
               $variable_name);
                       break;
                }
case "selmin":
      {
        $min_count = intval($command_value);

        if(isset($input_value))
                       {
            if($min_count > 1)
            {
                $bret = (count($input_value) >=
                   $min_count )?true:false;
            }
                           else
                           {
                             $bret = true;
                           }
                       }
        else
        {
          $bret= false;
          $default_error_message =
             sprintf(E_VAL_SELMIN_CHECK_FAILED,
             $min_count,$variable_name);
```

```
          }

          break;
        }
case "selone":
        {
          if(false == isset($input_value)||
            strlen($input_value)<=0)
          {
            $bret= false;
            $default_error_message =
               sprintf(E_VAL_SELONE_CHECK_FAILED,$variable_name);
          }
          break;
        }
case "eqelmnt":
        {

          if(isset($formvariables[$command_value]) &&
             strcmp($input_value,$formvariables[$command_value])==0
                )
          {
            $bret=true;
          }
          else
          {
            $bret= false;
            $default_error_message =
               sprintf(E_VAL_EQELMNT_CHECK_FAILED,
               $variable_name,$command_value);
          }
        break;
        }
 case "neelmnt":
        {
          if(isset($formvariables[$command_value]) &&
             strcmp($input_value,$formvariables[$command_value])
                !=0 )
          {
            $bret=true;
          }
          else
          {
            $bret= false;
            $default_error_message =
               sprintf(E_VAL_NEELMNT_CHECK_FAILED,
               $variable_name,$command_value);
          }
          break;
```

```
            }

    }
    return $bret;
  }
}

?>
```

A.20 membersite_config.php

```
<?PHP
/*
  ----------------------------------------------------------------------
  This program is free software published under the
      terms of the GNU Lesser General Public License.
  http://www.gnu.org/copyleft/lesser.html

  The following code was taken and modified from:

  http://www.html-form-guide.com/files/php-form/RegistrationForm.zip
  ----------------------------------------------------------------------
*/

require_once("../include/fg_membersite.php");

$fgmembersite = new FGMembersite();

//Provide your site name here
$fgmembersite->SetWebsiteName('test.nl');

//Provide the email address where you want to get
   notifications
$fgmembersite->SetAdminEmail('smaritsas@os3.nl');

//Provide your database login details here:
//hostname, user name, password, database name and table
   name
//note that the script will create the pentesters table
//by itself on submitting register.php for the first time
$fgmembersite->InitDB(/*hostname*/'localhost',
                      /*username*/'root',
                      /*password*/'admin',
                      /*database name*/'cpto',
                      /*table name*/'pentesters');

//For better security. Get a random string from this
   link: http://tinyurl.com/randstr
//and put it here
```

```
$fgmembersite->SetRandomKey('qSRcVS6DrTzrPvr');

?>
```

A.21 change-pwd.php

```
<?PHP
/*
  ------------------------------------------------------------------------

  The following code was taken and modified from:
  http://www.html-form-guide.com/files/php-form/RegistrationForm.zip

  ------------------------------------------------------------------------
*/

require_once("../include/membersite_config.php");

# Check if a user is logged in
if(!$fgmembersite->CheckLogin())
{
  $fgmembersite->RedirectToURL("login.php");
  exit;
}

# Change password, log out user, and then redirect to
   changed-pwd.html page
if(isset($_POST['submitted']))
{
  if($fgmembersite->ChangePassword())
  {
    $fgmembersite->LogOut();
    $fgmembersite->RedirectToURL("../changepassword/changed-pwd.html");
  }
}

?>

<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Change password</title>
    <script type='text/javascript'
```

```
    src='../scripts/gen_validatorv31.js'></script>
  <link rel="STYLESHEET" type="text/css"
    href="../style/pwdwidget.css" />
  <script src="../scripts/pwdwidget.js"
    type="text/javascript"></script>
</head>
<body>
  <!-- Form Code Start -->
  <div id='fg_membersite'>
    <form id='changepwd' action='<?php echo
      $fgmembersite->GetSelfScript(); ?>'
      method='post' accept-charset='UTF-8'>
     <fieldset>
      <div class = "page-header text-center">
        <h1>Collaborate Pen Testing
          Organizer<br><small>Password
          Change</small></h1>
      </div>

      <input type='hidden' name='submitted'
        id='submitted' value='1'/>

      <div class="text-center">* required
        fields</div></br>

      <!-- Put the error message from
        fg_membersite.php into a div -->
      <div><span class='error'><?php echo
        $fgmembersite->GetErrorMessage();
        ?></span></div>

      <div class="container text-center">
        <label for='oldpwd' >Old
          Password*:</label><br/>
        <div class="pagination centering"
          id='oldpwddiv' ></div><br/>
        <noscript>
          <input type='password' name='oldpwd'
            id='oldpwd' maxlength="50" />
        </noscript>
        <span id='changepwd_oldpwd_errorloc'
          class='error'></span>
      </div>

      <div class="container text-center">
        <label for='newpwd' >New
          Password*:</label><br/>
        <div class="pagination centering"
          id='newpwddiv' ></div>
```

```
            <noscript>
              <input type='password' name='newpwd'
                 id='newpwd' maxlength="50" />
            </noscript>
            <span id='changepwd_newpwd_errorloc'
               class='error'></span>
          </div>
          <div class="container␣text-center">
            <input type='submit' name='Submit'
               value='Submit' class="btn␣btn-primary"/>
          </div>
        </fieldset>
      </form>
      <script type='text/javascript'>
      // <![CDATA[
        var pwdwidget = new
           PasswordWidget('oldpwddiv','oldpwd');
        pwdwidget.enableGenerate = false;
        pwdwidget.enableShowStrength=false;
        pwdwidget.enableShowStrengthStr =false;
        pwdwidget.MakePWDWidget();

        var pwdwidget = new
           PasswordWidget('newpwddiv','newpwd');
        pwdwidget.MakePWDWidget();


        var frmvalidator  = new Validator("changepwd");
        frmvalidator.EnableOnPageErrorDisplay();
        frmvalidator.EnableMsgsTogether();

        frmvalidator.addValidation("oldpwd","req","Please␣
           provide␣your␣old␣password");

        frmvalidator.addValidation("newpwd","req","Please␣
           provide␣your␣new␣password");
      // ]]>
        </script>
    </div>
    <!--
    Form Code End
    -->
  </body>
</html>
```

A.22 changed-pwd.html

```
<!DOCTYPE html>
<html>
```

```
  <head>
    <link rel="stylesheet"
       href="../bootstrap-master/dist/css/bootstrap.min.css">
    <script
       src="../scripts/jquery-1.12.0.min.js"></script>
    <script
       src="../bootstrap-master/dist/js/bootstrap.min.js"></script>
    <title>Change password</title>
  </head>
  <body>
    <div>
      <div class = "page-header␣text-center">
        <h1>Collaborate Pen Testing
           Organizer<br><small>You have successfully
           changed your password!</small></h1>
      </div>
      <div class = "text-center">
        <a href='../index.html' class="btn-primary␣
           btn-lg␣btn-block">Home</a><br>
        <a href='../login/login.php' class="btn-primary␣
           btn-lg␣btn-block">Login</a><br>
      </div>
      </div>
    </div>
  </body>
</html>
```

Appendix B

Screen shots

B.1 Home page

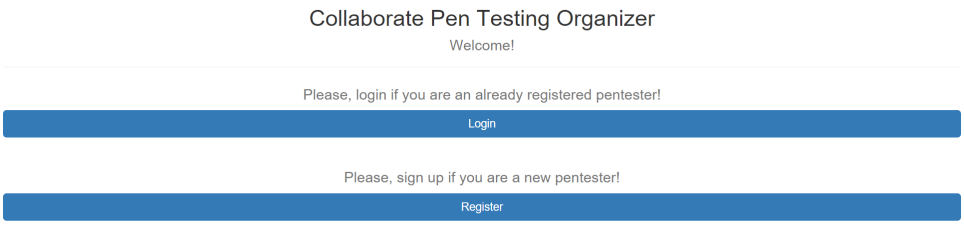

Figure B.1: Home page.

B.2 Register page

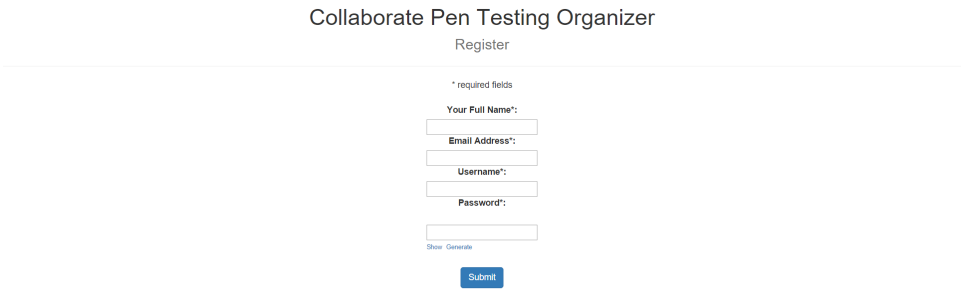

Figure B.2: Register page.

B.3 Login page

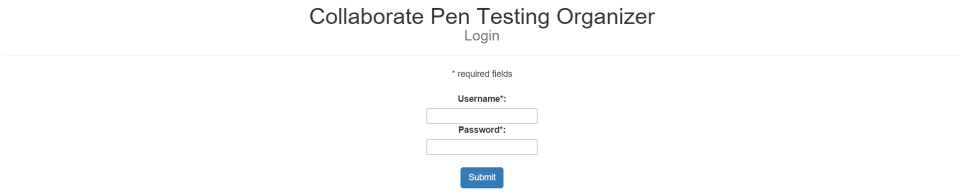

Figure B.3: Login page.

B.4 Profile page

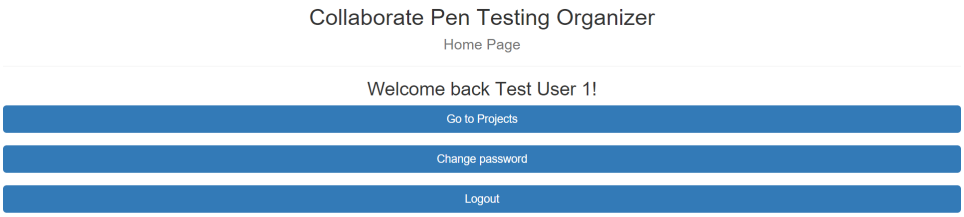

Figure B.4: Profile page.

B.5 Change password page

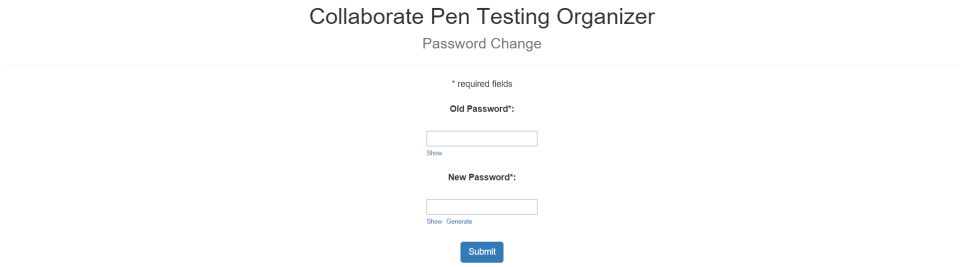

Figure B.5: Change password page.

B.6 Jobs page

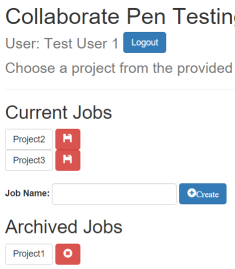

Figure B.6: Jobs page.

B.7 Plan page

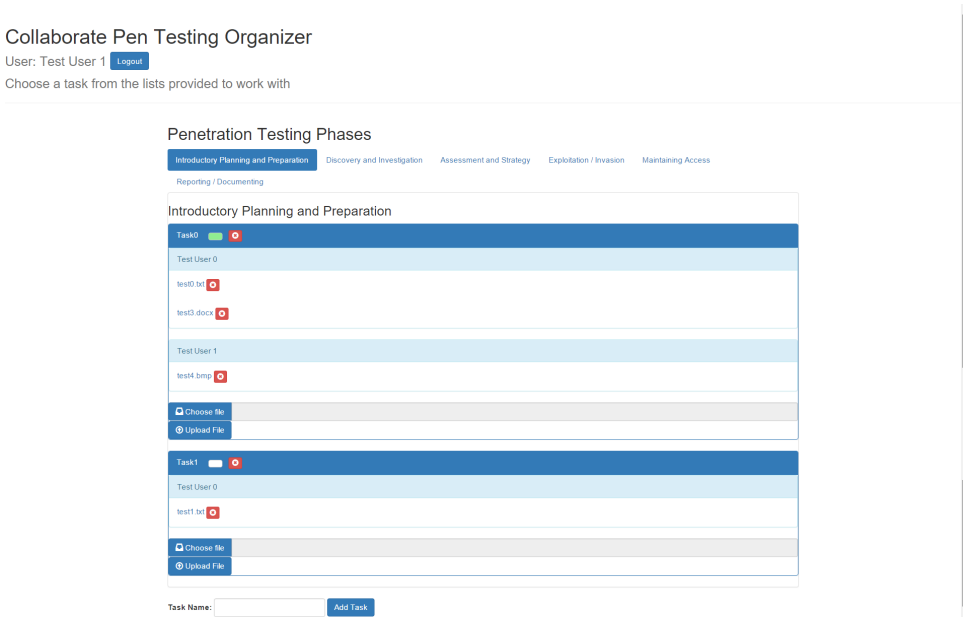

Figure B.7: Plan page.

B.8 Using capture.py

```
root@kali-sne-client:~/workspace# python capture.py
Enter the name of your log file: test
Capture started, file is: test_06-02-2016_16:52:32
root@kali-sne-client:~/workspace# ping -c 3 www.os3.nl
PING www.os3.nl (145.100.96.70) 56(84) bytes of data.
64 bytes from www.os3.nl (145.100.96.70): icmp_seq=1 ttl=57 time=7.66 ms
64 bytes from www.os3.nl (145.100.96.70): icmp_seq=2 ttl=57 time=7.39 ms
64 bytes from www.os3.nl (145.100.96.70): icmp_seq=3 ttl=57 time=7.36 ms

--- www.os3.nl ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 7.365/7.475/7.665/0.152 ms
root@kali-sne-client:~/workspace# systemctl list-dependencies
default.target
● ├─accounts-daemon.service
● ├─arpwatch.service
● ├─gdm.service
● ├─gdomap.service
● ├─irqbalance.service
● ├─saned.service
● ├─speech-dispatcher.service
● ├─stunnel4.service
● ├─systemd-update-utmp-runlevel.service
● ├─thin.service
● └─multi-user.target
●   ├─anacron.service
●   ├─arpwatch.service
●   ├─atd.service
●   ├─binfmt-support.service
●   ├─cron.service
●   ├─dbus.service
●   ├─gdomap.service
●   ├─greenbone-security-assistant.service
●   ├─inetd.service
●   ├─irqbalance.service
●   ├─ModemManager.service

root@kali-sne-client:~/workspace# exit
exit
Capture stopped, file is test_06-02-2016_16:52:32
```

Figure B.8: A simple example of capture.py script.

B.9 Output of capture.py

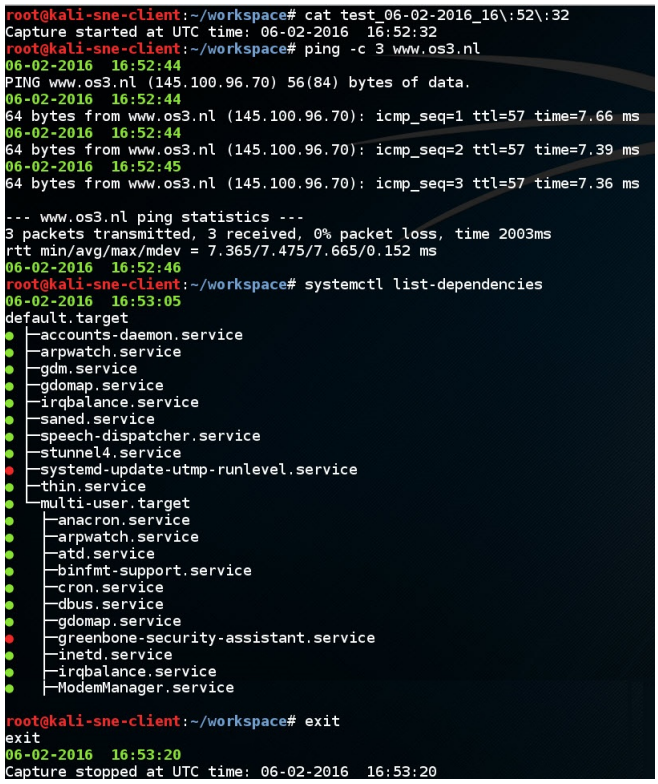

Figure B.9: Output of capture.py script.