School of Computing, Maths and Digital Technologies

Mark Bellingham

BSc Computing

14032098

Website Security

Detecting and Preventing Website Defacement

Supervisor: Yanlong Zhang

2017

Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Name:	Mark Bellingham
Date:	24/04/2017

Abstract

This project demonstrates a method preventing website defacement, by downloading and hashing the source code and then monitoring the website for changes to the hashes. If a change is found it responds by locating the IP addresses in the web server's log that correspond to the time of the attack. The program next sends an email to the website's administrator containing details of the page that was modified and the IP addresses that were accessing at that time. It then restores the website from the backup and continues monitoring. If the website is found to have changed again and the IP address is one that is known to the system, the program blocks that IP address from accessing the website and informs the administrator that it has done so.

Contents

List of Illustrations

1 Introduction

This project aims to develop a way of protecting against a hacker who may be targeting a particular website, trying to hack it again and again as a way of proving something to themselves. The project aims to show a way of identifying an individual attacker who is making repeated unauthorised changes to to a web-page, so that they can be blocked from accessing the site.

The first section deals with the literature review, where similar projects are identified and analysed to see if any methods or research would be useful to include in this project. It also serves to help the author avoid repeating work that has been done before.

The following section looks at the design of the program, detailing the features and methods that should be included. This section is divided into three parts: requirement analysis, structural design, and detailed design. The requirement analysis identifies the features that the program needs to have, so that the program may be designed in a logical and organised fashion. The structural design shows how the different elements of the project and program are related to each other, and the sequence of events as the program operates. The detailed design outlines each individual function or method, explaining how they are to be achieved.

The next section presents the implementation of the program and its output, by use of screenshots and details of the various messages that are imparted to the user via email and through the console.

In the evaluation, the author will be reflecting on their achievements at all stages of this project, focussing primarily on the areas of Product, Research, System Design and Time Management. This section also includes ideas for how the program could be improved.

The report finishes with a conclusion that draws together the arguments and execution of the project. It shows where the project has been successful and where it could have been done differently.

At the end of the report are the references, bibliography and appendices, which include the Terms of Reference, Project Plan and Ethics Form.

2 Literature Review

2.1 Introduction

The problem of website hacking is not one that is going to go away, no matter how much security is put in place. As the online world becomes more interactive collecting information from its users, as convenience dictates that the administration portal be accessible from external IP addresses, there will always be a hole or chink in the armour that can be exploited. There can be no one single solution to website or network security and instead security should comprise of many different layers, each one *“a different component in the network with a specific task in the protection scheme”* (Young, 2015). Although a particular layer may have holes or weaknesses, these weaknesses would be protected on another layer and that when all layers are combined, all the holes would be plugged. Even then, no system can be fully protected. The only completely safe system is one which is unplugged from the network, which is impossible for a website. Hacking prevention will always fail at some point. Therefore as part of the security toolkit, a monitoring system, while not preventing intrusions, can serve to frustrate an attacker. Intruders often don’t complete their objective in a few minutes or hours. Instead they will return again and again for months or even years (Bejtlich, 2013).

The approach that will be the focus of this paper, is to automatically detect when a page has been illegally modified and by whom, the aim being to block someone who is making repeated visits to a website in order to deface it.

2.2 Detection Methods

To begin with we will look at three systems that monitor the server for abnormal traffic. How does the system know which traffic is legitimate and which is abnormal? Many hackers will disguise themselves as legitimate traffic by using common port numbers such as port 80 (http/https) or port 443 (TLS/SSL). They may also be using https or SSL to encrypt their traffic, knowing that traffic on these ports using these protocols will likely be allowed through even if the system is unsure of its purpose. The solution to this problem proposed in the paper “Two-Phased Method for Detecting Evasive Network Attack Channels” (Cao et al., 2014) is one that checks against two core elements of the SSL security system. First inspect the SSL handshake, making sure that it conforms to the standard SSL session structure. Second, inspect the trustworthiness of the certificate using a knowledge-based approach and determine whether the certificate passes a few key rules. The paper “Design and Implementation of Automatic Defensive Websites Tamper-Resistant

System" (Huo et al., 2012) instead suggests monitoring the system log for file changes. They surmise that the complexity and diversity of operating systems combined with third-party security software and web applications creates many vulnerabilities which can be exploited. Their system, if it detected an unauthorised change, would immediately retrieve the backup to replace the hacked files. It would also send a message via email or SMS to the website administrator and, if necessary, shut off web access to the web server, which seems a bit extreme and could potentially be expensive for an organisation which relies on the durability of its website. This system would have a number of different devices which each perform a specific function - monitoring, retrieving and sending the backup, and administrator control. This in itself introduces complexity and potential points of failure. Finally there Is the idea from the paper *"Proposing a real time internal intrusion detection system towards a secured development of e-government web site"* (Al- Khanjari et al., 2013) who have devised a program in Java which monitors the HTML classes. This method would operate separate from the site itself and would hopefully detect both intruders and people who are "abusing their privileges", the so-called insider threat.

The problem with each of these methods is that they only monitor a very specific set of circumstances, and those circumstances are quite easy to bypass. For example there is no guarantee that the hacker will use https or SSL, they could even be attacking from inside the network. Likewise it is not particularly difficult for any serious hacker who knows what he is doing to modify the contents and timestamp of the logfile to hide his tracks. For example, using Linux the command

```
"touch -a -m -t 203801181205.09 file.log"
```

would change the access and modified time of file.log to 18th Jan 2038 at 12hrs:05min:09sec. In Windows a similar result can be achieved using Powershell and the command

```
"$(Get-Item file.log).lastwritetime
  =$(Get-Date "18/01/2038 12:05")
```

so unless you are completely sure that you have full control over the log files, and as mentioned in the introduction you can never be completely sure of anything security related, then this information is useful but not 100% reliable. The method that tracks changes to the HTML classes would not work either because a hacker may decide to keep the classes in order to make use of the

CSS file. Also this method seems quite complicated keeping track of changes and updates as the page evolves over time.

2.3 Methods That Use Hashing

The paper “Web Defacement: Threat Analysis and Modelling” (Mishra and Singh, 2015) compares 5 different types of hashing algorithms to see which would be the most useful in identifying whether a webpage has changed. The five algorithms are CRC32, MD5, SHA512, PSNR, and SSIM. A number of different types of threats are identified and each of them subjected to each of the different hashes to see if the change can be detected. The results are compiled into a table in order to show which is the best. It concludes that CRC32 cannot provide authentication, so cannot fully determine that the data was intentionally altered. MD5 suffers from collision because it gives the same hash value for two web pages of a certain complexity. PSNR (Peak Signal to Noise Ratio) and SSIM (Structural Similarity) do not support text integrity. Therefore the paper states that SHA512 gives the best results and this conclusion is also borne out by their test results, with SHA512 having the highest level of accuracy.

“An Approach to Reveal Website Defacement” (Gurjwar et al., 2013) offers a solution that is, in many ways, similar to the one that will be explored in this project. It also describes the idea of continuous monitoring, hashing pages and checking the hashes with those stored in a database. They have devised a fairly complicated method of extracting the data from a particular page, by splitting up the text and images and using a different method on each. They have also decided to strip out all JavaScript or other scripts and other large multimedia elements, saying that it is unnecessary to check these. This paper also compares the hashing methods CRC32, MD5, PSNR and SSiM. It discounts CRC32, PSNR and SsiM for the same reasons as the previous paper. The paper then goes on to compare MD5 with a number of iterations of the SHA hash, eventually coming to the same conclusion as Mishra and Singh, that SHA512 gives the highest level of accuracy.

This solution would be written in C# because the author wants a business-like application. While this is understandable, there is plenty of merit in creating a lightweight application that outputs in simple HTML or text, which can be read almost universally and easily incorporated into a larger suite of products.

These studies both have an excellent amount of clear and well defined data and will be extremely useful in helping to decide which hashing algorithm to use in my project.

2.4 Getting Data From The Logs

Logging access and modified times and analysing the data plays a key part in identifying an attacker. Yet as noted earlier, how can we be sure that this data is reliable and what is the best way to manage what is likely to be a large amount of data. “Identifying threats in real time” (Macrae, 2013) says that logging generates huge amounts of data which is usually managed in an inefficient and disparate manner. Better use of this logging data would help IT teams to analyse it. They say that any system which seeks to maximise the value of its log data should not only *“have log management capabilities, but should also incorporate real-time analysis, advanced correlation, reporting technology and immediate remediation capabilities.”* In this way, any abnormal activity is identified as it happens, and organisations are able to automatically tie together seemingly unrelated incidents, with potential danger being flagged as it occurs. Not only does this level of monitoring ensure that intrusions are more likely to be identified, it also becomes much more difficult for hackers to get around such a system, because they would have to modify the logs in both the network control system and the log management system at the same time.

(Thakore et al., 2016) in “A Quantitative Methodology for Security Monitor Deployment” say that *“Detection of an event may require the correlation of information from multiple monitors.”* They suggest using the Apache server access log, the Linux syslog or even installing Snort (www.snort.org), which is an *“open source intrusion prevention system capable of real-time traffic analysis and packet logging”*. This helps to mitigate the problem noted earlier that it is possible that the hacker may be trying to hide their tracks by altering the content or timestamps of the log files. Comparing the data in each gives a greater chance that this data is accurate. The paper “Anomaly-Based Intrusion Detection and Prevention System on Website Usage using Rule-Growth Sequential Pattern Analysis” (Pramono and Suhardi, 2014) builds on this idea by trying to identify malicious user behaviour by analysing the logs to determine a sequence of events that always happen in a particular order (say a,b,c), which are then always followed by another set sequence (d,e,f,g). If a different sequence happens or something happens out of order, this will be classed as a malicious event. It filters the access log based on activity, sorts them by distinct IP address and by time. In this way, reducing the size of the log to something more manageable. One problem with this system is noise, where the server log and the user behaviour do not match.

It becomes clear then, that log monitoring must be approached from several angles. Monitoring both the system and the server log is a great idea and quite easily achievable. Behaviour analysis, while interesting, is probably beyond the scope of this project.

2.5 Hackers Who Hide Their Identity

Of course, even the best logging strategies fall down against a hacker who is hiding their identity through proxies or TOR. This problem is discussed in “Hackers Topology Matter Geography” (Rechavi et al., 2015). On the internet, geographical boundaries are almost non-existent. Vague privacy and comprehensive anonymity make it easy for hackers to hide on the online. For their project they created two honeypot computers, which would gather information about trespassing events. Although their research only monitored attacks on SSH servers, making it of limited value to this project, it does provide some insight into behaviours, most notably concluding that while the IP address of the hacker is most probably not their real IP, it does identify the most available IP address that they can use.

If the attacker is using the TOR network, there is little you can do without government or ISP level resources. TOR very effectively hides an internet user from the service that they are trying to access. It should not be vilified since it has both good and bad uses, enabling people who live under oppressive governments to access information freely as well as hiding criminals and terrorists. (Park et al., 2015) in “New Detection Method and Countermeasure of Cyber Attacks In Mix Networks” say that criminals choose countries with high available bandwidth but by controlling a large number of exit nodes in a particular country and by collaborating with other countries, it is possible to know which country an attacker is coming from, even if you cannot identify the attacker himself. By monitoring the exit nodes and having real-time access to their Access Logs you can detect access frequency for a particular IP and block them.

2.6 Conclusion

From this research several conclusions can be made. Simple monitoring of any one single element of the system will not be enough. SHA512 hashing algorithm looks to be a strong candidate for the hash that will be used in this project, although further tests can be easily and quickly set up to confirm this. Tests will also have to be carried out to see if it is worthwhile dividing the pages into their different elements and hashing them separately, although the option being considered at the moment is to hash the page as it is delivered to the user, with all elements combined in a single hash. Monitoring both the server and system log is possible and will certainly be considered. It is heartening to see that this system may be of limited value against people using proxies, though for obvious reasons will not stop those using TOR.

3 Design

3.1 Requirement Analysis

3.1.1 Database

Store hashes of pages and IP addresses in a MySQL database

Make the database password restricted so that the data contained within cannot be modified by an intruder

3.1.2 Program

Get pages from the website at regular intervals

Since webpages are made up of several elements, the program will need to collect all of it using and zip as a single file before hashing.

Get log(s) from the server(s) when triggered

Retrieve logs from the server and extract IP addresses for a particular timestamp. The best approach would be to extract information from both the web server log and the system log as advised by (Thakore et al., 2016) in "A Quantitative Methodology for Security Monitor Deployment"

Investigate locking the logfiles so that they cannot be modified by the intruder, but still be usable by the program and administrator

Hash pages using SHA512

The SHA512 hashing algorithm was found to be the most accurate in detecting changes to webpages in both of the papers that studied hashing (Mishra and Singh, 2015), (Gurjwar et al., 2013)

Add hashes and IP addresses to the database

Retrieve hashes and IP addresses from the database

Compare hashes and IP addresses

Hashes change substantially for minor changes at the source so it will be sufficient to check for exact matches

Start and stop monitoring

When the website is being updated, the monitoring function will need to be paused so that the alerts are not triggered, then started again using the Initialise function

Used at the start of the analysis and whenever a page has been updated it will first generate hashes of the required pages, store them in the database and then start the monitoring process

Alert when triggered via email and/or text

Restore website to state at the start of the program

Block specific users from accessing the website

3.1.3 Website

3 or 4 page website with basic interactivity

Very few websites are completely static so the website in this project should include some JavaScript to show that it works with these types of pages. However, a fully comprehensive website is not needed to show proof of concept

3.2 Structural Design

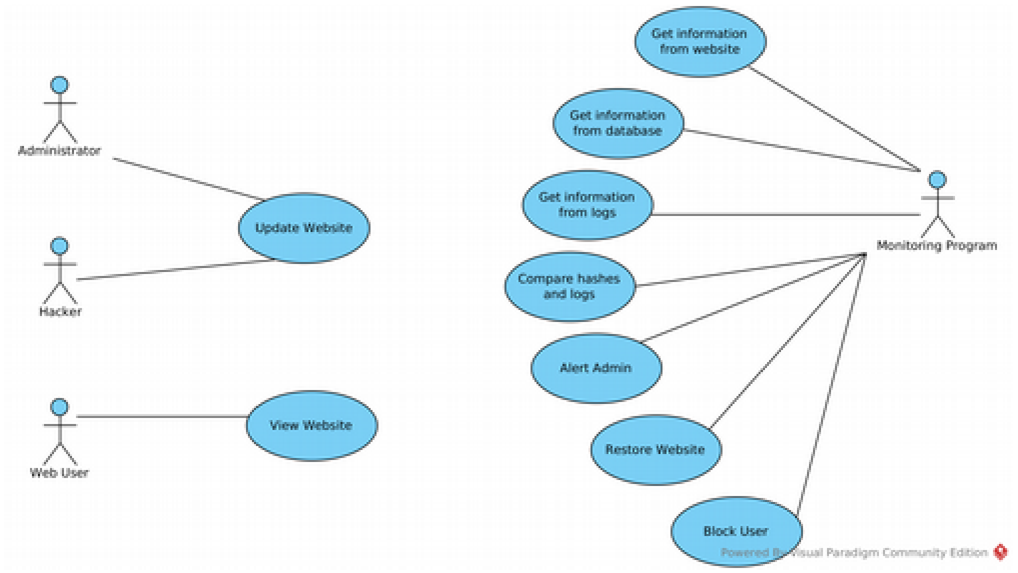


Illustration 3.1: Use case diagram

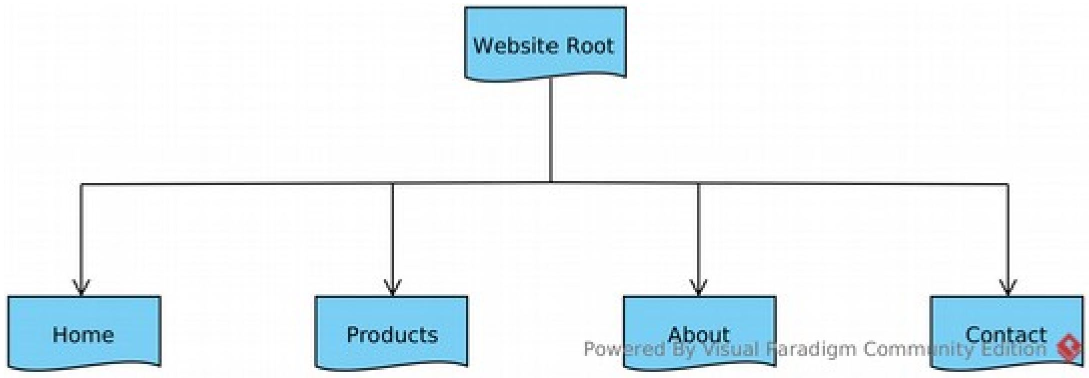


Illustration 3.2: Website Overview

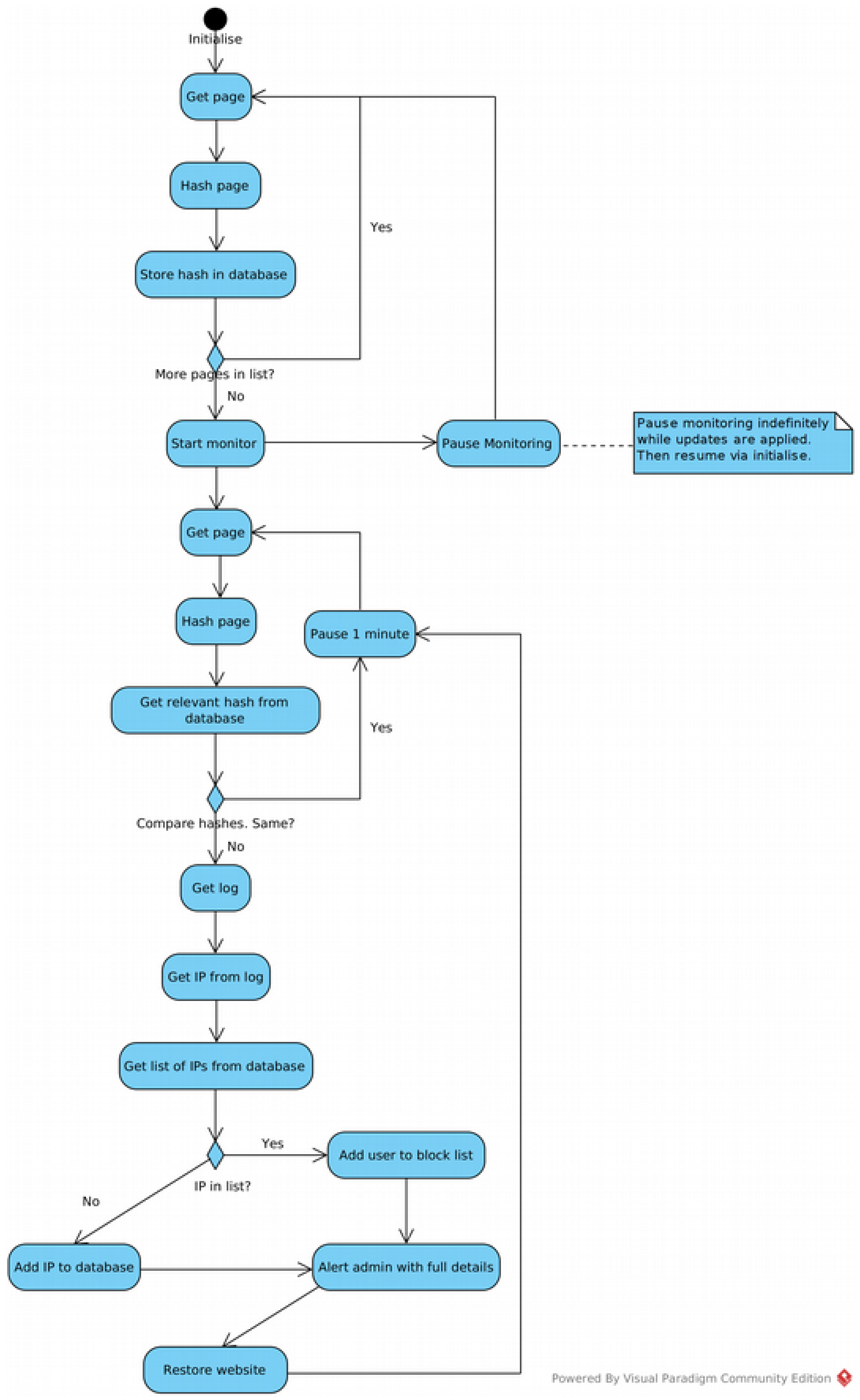


Illustration 3.3: Program flow

3.3 Detailed Design

3.3.1 Overall Program Design

The program will incorporate a controller that performs the business logic with functions such as hashing the pages and reading the access log being in their own classes and called upon when necessary.

3.3.2 Access To The Database

Access to the database will be established as a DAO (Data Accessor Object) model, which is a separate class that handles all database interactions. Using a DAO model means that the database can be replaced in the future by (for example) a cloud data store or other type of database without needing to rewrite the whole program.

The DAO contains methods to connect to and disconnect from the database, which will be used by the CRUD methods. These methods should not be available outside this class.

Method to open a database connection

```
Connect to the database
Define username and password
Define url to the database
Create connection object
Return connection object
```

It also contains methods to Create, Retrieve, Update and Delete entries in the database. These methods should be available to the calling program, which passes arguments. Where relevant, they return an object to be interpreted by the program.

Method that returns all records from a database table

```
Open database connection
Create array with key value pairs
Create select all query
Execute query
While result has more records:
    Add the name and hash to the array
Close database connection
Return array
```

Method to delete all records from the pages table

```
     Function takes page name as argument
     Open database connection
     Create basic delete query
     Add parameter to query
     Execute query
     Close database connection
```

Method to insert an IP address into the database

```
     Function takes two strings, IP address and date
     Open database connection
     Create basic database insert query
     Add parameters to query
     Execute query
     Close database connection
```

3.3.3 Database Tables

The database will be created in MySQL. It should contain two tables.

One table will store the hashes of the pages and the corresponding page names. This table will store informations about HTML, JavaScript and CSS type pages

Page name	SHA512 hash

The second table will store the IP address of people who accessed the website in a time-frame identified by the program, along with their access timestamp.

IP address	Date

3.3.4 The Website

The website needs to only be a simple one since the program only needs to prove the concept of a monitoring system. It is proposed to create a four page website with some basic media and interactivity. These would incorporate HTML, pictures, links, and CSS.

3.3.5 Method For Retrieving The Page From The Web Server

The controller holds a list of the pages that are to be monitored. It requests each one in turn and saves the data stream so that it can be passed to the hashing function.

Method to retrieve the page from the web server

```
    Function takes page address as argument
    Connect to the page
    While the data is streaming:
        Read the data line by line
        Concatenate the lines to a single string
    Send the final string to the hash function
    Return the hash value
```

3.3.6 Method For Hashing The Pages

The hashing algorithm will be SHA512 as recommended by (Mishra and Singh, 2015), and (Gurjwar et al., 2013). The page code should be passed to the hashing method as a String argument. The function then hashes the code and returns the hash value.

Method for hashing the page code

```
    Function takes String as argument
    Hash the string using SHA512
    Return the hash value
```

3.3.7 Restore The Website From Backup

The program should have the facility to restore the website from backup files immediately so that normal functionality can resume without delay. It will do this simply by copying all of the files from the backup to the live location.

Method for restoring the website

```
    Define backup page locations
    Add them to an array
    Define live website locations
    Add them to an array
    For Loop same length as array
        Get page from backup location
        Copy page to corresponding live location
```

3.3.8 Method For Extracting Information From The Log File

The log file is to be parsed line by line until the program finds an entry with a timestamp that matches the one being searched for. When such an entry is found, that line is split into its component parts and the IP address is stored in a variable array while the rest of the log is parsed. Once parsing is complete, the IP addresses in the variable array can be used in the email that is sent to the Administrator.

Method for reading the log file

```
Define the log file location
Create array to hold the results
Read the log file into a buffer
While the buffer has more lines:
     Split the line into components
     If the data component matches date passed:
          Save IP address in the array
Return array
```

3.3.9 Method For Sending An Email To The Administrator

The program needs to have a defined email address that will be used for the purposes of this project. It will take the body of the email as an argument, which will have been formed by the controlling program and will contain information that will be useful to the receiver of the email, such as the page hashes that have triggered this event, the IP address responsible and the page affected. The program needs to define all the properties that go towards making up an email, such as host, port number, authentication type, content type, date and sender’s address. Once the email have been formulated and sent successfully, the program will post notification to the console indicating this. This method is as described on the website JournalDev JavaMail Example – Send Mail in Java using SMTP (Pankaj, 2016)

Method for sending an email

```
Message body passed as argument
Define the properties that make up an email
Confirm authentication with the server
Include the compiled information in the body
Send the email
Create notification of success
```

3.3.10 Program Controller

The controller is the core of the program. It manages the business logic and directs data and traffic to other functions as and when required. To begin with it removes all the page hashes from the database in order to start afresh. This is because the program will be restarted after the website has been updated and it should use only the most up to date information. Once this has been done the program can define which pages will be monitored, get the code, hash it and insert the hashes into the database.

Method for initial page hashing

```
     Delete old page hashes from database
     Get webpage names from db and add them to an array
     For each address in the array:
          Get the page hash
          Insert the address and hash into the database
```

Once the initial set up is complete, the program can start the monitoring process. It begins by getting all the pages and hashes into an array with key value pairs so that they can be linked. The program then runs a continuous loop that gets the hash for each page and checks it against the first hash that was created initially. If the hashes for all of the pages match, the program pauses for 60 seconds and then starts again. The first part of this process is detailed in the pseudocode below.

Method for monitoring the pages

```
     While true:
          For each page to be checked:
               Get the hash for the page code
               Compare it against the first hash
               If hashes match:
                    Continue
               Else:
                    .. .. ..
          Once all pages have been checked, loop
          through them again
```

If the hashes do not match, the program commences a chain of events that conclude with notifying the administrator by email and restoring the website back to its original state. To begin with it gets the current time and passes that to the Log Reader, which returns an array of IP addresses that were accessing the website at that time. The program then retrieves all of the IP addresses that have previously been logged so that they can be compared with the new set. The program then loops through all the IP addresses from the Log Reader, inserts each one with the time into the database and compares it with the ones previously extracted from the database. In this way it can inform the administrator about whether this IP address is known to the system or if it is a new one. The administrator can use this information to decide whether to block this IP address from accessing the website in the future. Finally the program copies the website files from the backup location to the live location and restores it to its original state. The pseudocode below continues from where the previous block left off.

Method for when the hashes do not match

```
     Else:
          Get current time
          Find entry in access_log that matches time
          Extract IP address(es)
          Compare IP address(es) against database
          If IP address is known:
               .. .. ..
          If IP address is not known:
               Send email with relevant details
          Add IP(s) with log time to database
          Restore website from backup
          Resume monitoring
```

3.3.11 Method For Blocking An IP Address

If, when analysing the IP addresses from the web server log, the program encounters an IP address that it already knows about, it must do all of the above plus block that IP address from accessing the website in the future. It does this by adding the IP address to the .htaccess file located in the website’s root directory.

Method for blocking an IP address

```
     If the IP address is known:
          Get the .htaccess file
          Amend it with the IP address
          Close the file
          Include this information in the email body
```

3.3.12 Interface

The interface for the program is a simple command line one. The program is designed to be set and then left running without further oversight unless there have been changes made to the website. Therefore it is designed to have lower overheads and a graphical interface is not necessary.

3.4 Design Evaluation

This design meets all of the objectives and deliverables in the Terms Of Reference (Appendix A). In addition it actually goes further, in implementing a simple block on an attacker. It is relatively straightforward for someone with a moderate amount of programming knowledge to implement and can be created using free and open-source tools. The resources needed to run the program as designed should not be a problem for any modern computer that is capable of running a web server, making the whole project extremely cost-effective.

Function wise, once it has been set up it couldn’t be simpler to run since it is designed to be set and left, with changes to website pages requiring only a program restart. Adding new pages to the database is the only area that might trip up a complete novice but the SQL language is very easy to comprehend, even for a non-programmer.

4 Implementation

4.1 Program Code

The full source code is available in the accompanying program. In this chapter, some snippets of code that explain how the key features of the program work will be examined in detail. All code snippets in this chapter are taken from the Eclipse IDE and use Eclipse's colour scheme.

4.1.1 Data Access Object

The program uses a Data Access Object (DAO) for accessing the database. This object is a stand-alone class which contains all the methods for connecting to the database, performing some kind of function on the database and returning

Method for connecting to the database

```
// Method for connecting to the database, returns an open connection
      private Connection openConnection() {
            String userid = "mark";
            String userpass = "Excite10";
            String url =
                  "jdbc:mysql://localhost:3306/web_sec_project";
            Connection conn = null;

            try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
                  conn = DriverManager.getConnection(url, userid,
                              userpass);
                  conn.createStatement();
            } catch (InstantiationException | IllegalAccessException
                  | ClassNotFoundException | SQLException e) {
                  System.err.println(e);
            }
            return conn;
      }

      // Method for closing the connection to the database
      private void closeConnection(Connection conn) {
            try {
                  conn.close();
            } catch (SQLException e) {
                  e.printStackTrace();
            }
      }
```

the results to the program, either as an object or whatever datatype is most suitable. The advantage of this approach is that all the methods for accessing the database are separate from the main program. In this way the database could be substituted for a different type, for example a cloud store, without affecting any of the program logic code. (Parsons, 2008)

The DAO begins with two methods for connecting to the database. These methods are private so that they can only be accessed within this class. The first method contains the authentication details and returns an open connection object. The second takes an open connection as argument and closes it.

An example of a method for extracting data from the database is shown next. This particular example is for saving an IP address with the data accessed into the database but the other methods are very similar.

To begin with it takes the IP address and the date accessed as arguments. The basic SQL string is created, with question marks for the parameters and an open connection object is created. All the methods in the DAO use the Prepared Statement interface for executing the statement because this is a secure standard means of interacting with the database. It allows the program use and reuse dynamic queries and it protects the database from SQL injection attacks (Long et al., 2012). The Prepared Statement insert the parameters and executes the statement. The instruction to close the connection is in the 'finally' block so that even if the statement fails for any reason it would always carry out this part.

Method for inserting an IP address into the database

```
	// Method for inserting an IP address into the database
	public void insertIP(String ipAddress, String date) {
		String sql = "INSERT INTO ip_addresses (ip_address,
		                   date_accessed) VALUES (?, ?)";
		Connection conn = openConnection();

		try {
			PreparedStatement pstmt =
				conn.prepareStatement(sql);
			pstmt.setString(1, ipAddress);
			pstmt.setString(2, date);
			pstmt.executeUpdate();
		} catch (SQLException e) {
			e.printStackTrace();
		} finally {
			closeConnection(conn);
		}
	}
```

4.1.2 Hashing the pages

The method that hashes the pages begins by using an input streamer to read the data stream that is being downloaded over the network, the address provided by argument to the function. This input is fed to a buffered reader and each line from the buffered reader is concatenated into a String, which is then passed to the hashing function.

Method for hashing the web page

```
      public static String hash(String address) throws
                  IOException, NoSuchAlgorithmException {

            // Create a URL from the String and print it
                  to the console for troubleshooting
            URL url = new URL(address);
            // System.out.println(url);

            // Use BufferedReader to read the stream of
                  data from the network
            InputStreamReader isr = new
                  InputStreamReader(url.openStream());
            BufferedReader br = new BufferedReader(isr);

            // Convert the data stream in the
                  BufferedReader to a String
            String input, pageSource = "";
            while ((input = br.readLine()) != null) {
                  pageSource += input;
            }
            br.close();

            // Get a hash value form the page source
            String sha512 = HashFunction.hash(pageSource);

            // Print the hash value for troubleshooting
            // System.out.println(sha512);

            return sha512;
}
```

The Message Digest is a Java Class that is capable of converting text into many different types of hashes. This program uses it to convert the page code into a SHA512 hash as explained in Chapter 3.3.6. Once the hashing function receives the string of page code it converts it into bytes as required by the Message Digest. The type of hash being used is defined and then the code is converted into a hash. Using a buffer, the hash is converted to hex format and then back into a string, which can be passed back to the calling program.

Hashing Function

```
	public static String hash (String pageSource) throws
		NoSuchAlgorithmException {

		// Convert the String to bytes as required by
			MessageDigest
		byte [] input = pageSource.getBytes();

		// Get a Message Digest object that uses the SHA512
			algorithm
		MessageDigest SHA512 = MessageDigest.getInstance("SHA-
			512");

		// Compute digest (hash value) of the page
		SHA512.update(input);
		byte [] digest = SHA512.digest();

		// Convert byte digest to hex format
		StringBuffer hexDigest = new StringBuffer();
		for (int i = 0; i < digest.length; i++) {
			hexDigest.append(Integer.toString((digest[i]&0xff)
				+ 0x100,16).substring(1));
		}
		return hexDigest.toString();
	}
```

4.1.3 Reading the logfile

Initially the program must define the source of the log file, read it in using a File Input Stream, buffer using the Buffered Input Stream and then read the data using the Data Input Stream. The program code on the next page starts at this point, where it is reading the file data line by line.

The program splits each line of the data into their component parts and puts them into an array. It can then isolate and extract the date and time. The minute value is also isolated from the log date and the error date passed in as argument. These two values are compared to see if they are identical or if the log value is one less than the error value. Although this means that the comparison window is not exactly 1 minute, this is a simpler and more straightforward solution, and does not in any way affect the reliability of the data. In order to compare exactly one minute, both date values, currently held as Strings, would need to be converted to that of type Calendar, which means importing an extra module, increasing the size and complexity of the program for no real benefit. The practical effect is that the comparison window is extended to up to 2 minutes, but is more likely to average 1.5 minutes.

If there is a match, the program extracts the IP address that corresponds to that time and if the IP address is not one that belongs to the host computer, and it is not one that has already been identified, it is added to an Array List.

Method for reading the log file

```
     while (dis.available() != 0) {
            String logEntry = dis.~~readLine()~~;

            // Split the log entry into its constituent parts and
                  extract the date
            String[] logEntryParts = logEntry.split(" ");
            logDate = logEntryParts[3];
            logDate = logDate.substring(1);

            // Get the minute value from the date and convert to
                  integer
            int logDate_minuteVal =
                  Integer.parseInt(logDate.substring(15, 17));
            int errorTime_minuteVal =
                  Integer.parseInt(errorTime.substring(15,17));

            // Compare the minute value from the log with that from
                  the program
            if (logDate_minuteVal == errorTime_minuteVal ||
                  logDate_minuteVal == errorTime_minuteVal - 1) {

            //System.out.println("errorTime: " + errorTime);
            //System.out.println("logDate:   " + logDate);

            // If there is a match, the IP address for that
            // entry is the one we are looking for
            ipAddress = logEntryParts[0];
            //System.out.println("Log IP Address: " + ipAddress);
            }

            // Create an ArrayList of all found IP addresses,
                  excluding the localhost ones
            // Some will be duplicate because the web server logs an
                  IP address for each component of the page that is
                  downloaded. The ArrayList should only record this
                  once
            if (!ipAddresses.contains(ipAddress) && !
      ipAddress.equals("127.0.0.1") && !ipAddress.equals("::1") && !
      ipAddress.equals("192.168.1.189") && !ipAddress.equals("")) {

                  ipAddresses.add(ipAddress);
            }
      }
```

Once the whole access log has been examined, the resulting array of IP addresses is passed back to the Controller.

4.1.4 Restoring the website

The program needs to automatically restore the website to how it was originally and it does this by copying the files from a backup location and overwriting the files in the live location. The program replaces the entire website in this way.

Although this could be considered to be a waste of resources, the website being tested here is very small and copying is almost instant. If it is known that a hacker has gained access to the website files, it is perhaps safer to copy everything so that you can be sure it is all correct. Hopefully this is a procedure that would not need to happen very often, if at all so the trade off would be worth it. For a very large site that would take a long time too copy, the program could easily be altered to only replace the file(s) that have been modified.

Method for restoring the website

```
public static void restore() {

	DAO dao = new DAO();

	// Create ArrayLists of type Path to store the
		locations of the live and backup files
	ArrayList<Path> backup_pages = new ArrayList<Path>();
	ArrayList<Path> live_pages = new ArrayList<Path>();

	// Get the page URLs from the database
	ArrayList<String> pageNames = dao.getPageNames();

	// For each URL returned, extract the page name and
		define the path for the live and backup locations
	for (String page : pageNames) {
		page = page.substring(31);
		Path path =
		Paths.get("/home/mark/Sites/ProjectWebSec/" + page);
			live_pages.add(path);
	}

	for (String page : pageNames) {
		page = page.substring(31);
		Path path =
	Paths.get("/home/mark/Sites/ProjectWebSec_backup/" + page);
		backup_pages.add(path);
	}


	// Replace the live pages with those from the backup
	for (int i = 0; i < backup_pages.size(); i++) {
		Path from = backup_pages.get(i);
		Path to = live_pages.get(i);
		try {
			Files.copy(from, to,
				StandardCopyOption.REPLACE_EXISTING);
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}
```

The function begins by getting a list of all pages that are being monitored from the database. It creates separate Array Lists for the live and backup locations and then creates a Path object for each page. It then quite simply loops though the array of backup pages and for each one, replaces the corresponding live page.

4.1.5 Blocking an IP address from accessing the website

The program needs to be able to block the IP address of a persistent hacker. There are several ways of achieving this, with varying results. The method chosen for this project is a fairly simple and basic one that will only deter the type of hacker that is doing it for fun. It is hoped that someone who is doing this for kicks will soon get bored of seeing their efforts replaced so quickly, and of having to use a different proxy to be able to view the results of each attack, that they will move on to an easier target.

Blocking a more determined hacker would mean doing things like changing the rules in the IP tables of the firewall, which are beyond the scope of this program. However, the system administrator should have all the information they need in order to start probing the system logs from the email notifications and the database of IP addresses with their access times.

As noted in Chapter 2.1, good network security is the product of many elements that work together rather than a single one size fits all approach.

Method for blocking a specific IP address

```java
public static void appendIP(String ipAddress) {

      // Define the string to append to the htaccess file
      String textToInsert = "deny from " + ipAddress;

      // Define the location of the htaccess file
      String htaccessFile =
            "/home/mark/Sites/ProjectWebSec/.htaccess";

      try {
            // Get a new BufferedWriter
            BufferedWriter writer = new BufferedWriter(new
                  FileWriter(htaccessFile, true));

            // Append the contents
            writer.write(textToInsert);
            writer.newLine();

            // Close the file
            writer.close();

      } catch (IOException e) {
            e.printStackTrace();
      }
}
```

As stated earlier, the method is a very simple one. It involves appending to the .htaccess file that is located in the root directory of the website. The program takes the IP address as an argument, defines the String that will be added to the file, locates the .htaccess file and writes to it using a Buffered Writer and a File Writer. It then adds a new line to the document ready for the next entry and closes the file.

4.2 Program Execution, Output and Initial Testing

The first four images show the website that is to be monitored.

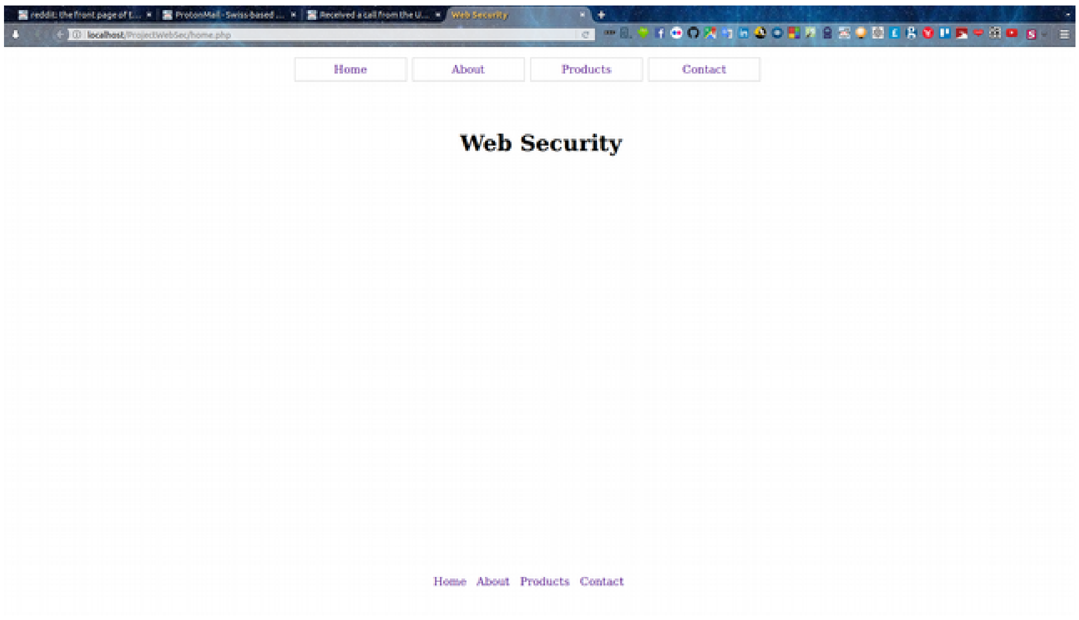
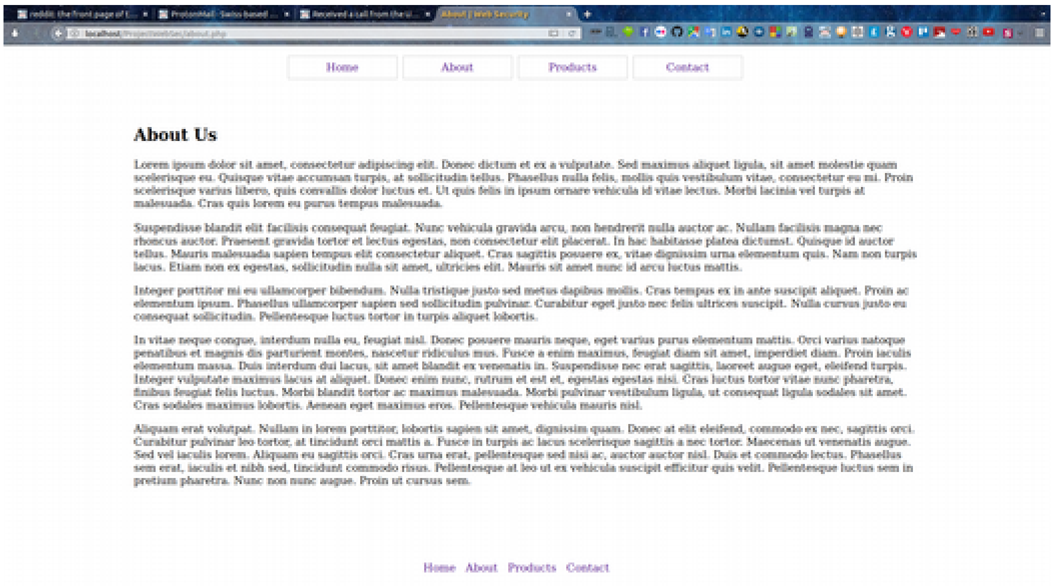


Illustration 4.1: Website Home Page


localhost/ProjectWebSec/about.php
Home
About
Products
Contact
Home About Products Contact

About Us

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec dictum et ex a vulputate. Sed maximus aliquet ligula, sit amet molestie quam scelerisque eu. Quisque vitae accumsan turpis, at sollicitudin tellus. Phasellus nulla felis, mollis quis vestibulum vitae, consectetur eu mi. Proin scelerisque varius libero, quis convallis dolor luctus et. Ut quis felis in ipsum ornare vehicula id vitae lectus. Morbi lacinia vel turpis at malesuada. Cras quis lorem eu purus tempus malesuada.

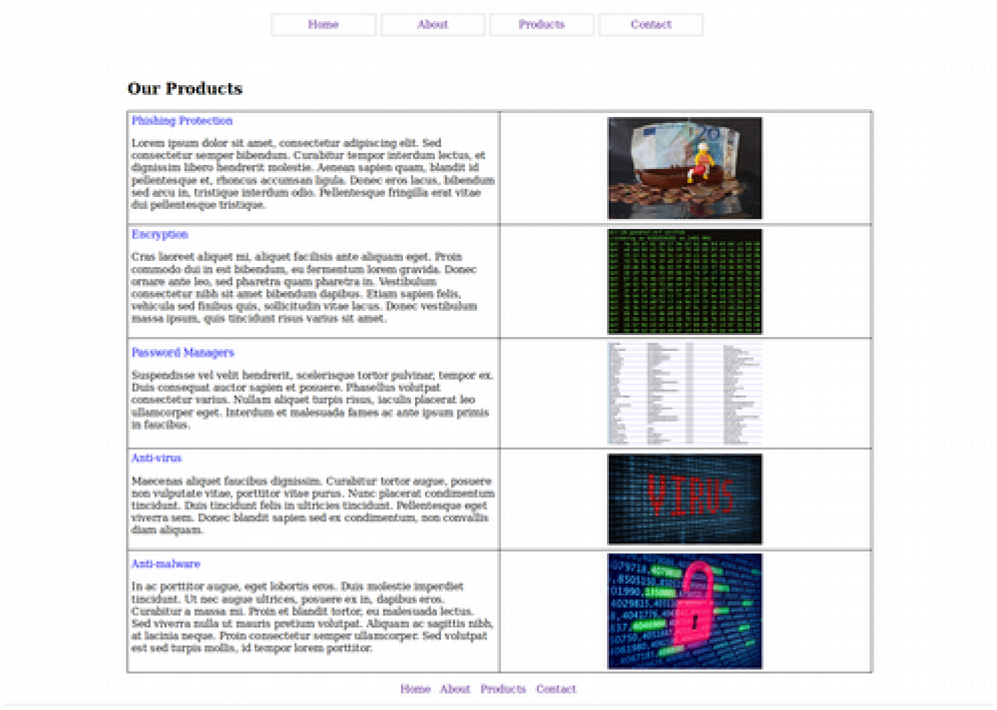
Suspendisse blandit elit facilisis consequat feugiat. Nunc vehicula gravida arcu, non hendrerit nulla auctor ac. Nullam facilisis magna nec rhoncus auctor. Praesent gravida tortor et lectus egestas, non consectetur elit placerat. In hac habitasse platea dictumst. Quisque id auctor tellus. Mauris malesuada sapien tempus elit consectetur aliquet. Cras sagittis posuere ex, vitae dignissim urna elementum quis. Nam non turpis lacus. Etiam non ex egestas, sollicitudin nulla sit amet, ultricies elit. Mauris sit amet nunc id arcu luctus mattis.

Integer porttitor mi eu ullamcorper bibendum. Nulla tristique justo sed metus dapibus mollis. Cras tempus ex in ante suscipit aliquet. Proin ac elementum ipsum. Phasellus ullamcorper sapien sed sollicitudin pulvinar. Curabitur eget justo nec felis ultrices suscipit. Nulla cursus justo eu consequat sollicitudin. Pellentesque luctus tortor in turpis aliquet lobortis.

In vitae neque congue, interdum nulla eu, feugiat nisl. Donec posuere mauris neque, eget varius purus elementum mattis. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Fusce a enim maximus, feugiat diam sit amet, imperdiet diam. Proin iaculis elementum massa. Duis interdum dui lacus, sit amet blandit ex venenatis in. Suspendisse nec erat sagittis, laoreet augue eget, eleifend turpis. Integer vulputate maximus lacus at aliquet. Donec enim nunc, rutrum et est et, egestas egestas nisi. Cras luctus tortor vitae nunc pharetra, finibus feugiat felis luctus. Morbi blandit tortor ac maximus malesuada. Morbi pulvinar vestibulum ligula, ut consequat ligula sodales sit amet. Cras sodales maximus lobortis. Aenean eget maximus eros. Pellentesque vehicula mauris nisl.

Aliquam erat volutpat. Nullam in lorem porttitor, lobortis sapien sit amet, dignissim quam. Donec at elit eleifend, commodo ex nec, sagittis orci. Curabitur pulvinar leo tortor, at tincidunt orci mattis a. Fusce in turpis ac lacus scelerisque sagittis a nec tortor. Maecenas ut venenatis augue. Sed vel iaculis lorem. Aliquam eu sagittis orci. Cras urna erat, pellentesque sed nisi ac, auctor auctor nisl. Duis et commodo lectus. Phasellus sem erat, iaculis et nibh sed, tincidunt commodo risus. Pellentesque at leo ut ex vehicula suscipit efficitur quis velit. Pellentesque luctus sem in pretium pharetra. Nunc non nunc augue. Proin ut cursus sem.

Illustration 4.2: Website About Us



Our Products

Phishing Protection Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed consectetur semper bibendum. Curabitur tempor interdum lectus, et dignissim libero hendrerit molestie. Aenean sapien quam, blandit id pellentesque et, rhoncus accumsan ligula. Donec eros lacus, bibendum sed arcu in, tristique interdum odio. Pellentesque fringilla erat vitae dui pellentesque tristique.	
Encryption Cras laoreet aliquet mi, aliquet facilisis ante aliquam eget. Proin commodo dui in est bibendum, eu fermentum lorem gravida. Donec ornare ante leo, sed pharetra quam pharetra in. Vestibulum consectetur nibh sit amet bibendum dapibus. Etiam sapien felis, vehicula sed finibus quis, sollicitudin vitae lacus. Donec vestibulum massa ipsum, quis tincidunt risus varius sit amet.	
Password Managers Suspendisse vel velit hendrerit, scelerisque tortor pulvinar, tempor ex. Duis consequat auctor sapien et posuere. Phasellus volutpat consectetur varius. Nullam aliquet turpis risus, iaculis placerat leo ullamcorper eget. Interdum et malesuada fames ac ante ipsum primis in faucibus.	
Anti-virus Maecenas aliquet faucibus dignissim. Curabitur tortor augue, posuere non vulputate vitae, porttitor vitae purus. Nunc placerat condimentum tincidunt. Duis tincidunt felis in ultricies tincidunt. Pellentesque eget viverra sem. Donec blandit sapien sed ex condimentum, non convallis diam aliquam.	
Anti-malware In ac porttitor augue, eget lobortis eros. Duis molestie imperdiet tincidunt. Ut nec augue ultrices, posuere ex in, dapibus eros. Curabitur a massa mi. Proin et blandit tortor, eu malesuada lectus. Sed viverra nulla ut mauris pretium volutpat. Aliquam ac sagittis nibh, at lacinia neque. Proin consectetur semper ullamcorper. Sed volutpat est sed turpis mollis, id tempor lorem porttitor.	

Home About Products Contact

Illustration 4.3: Website Products page

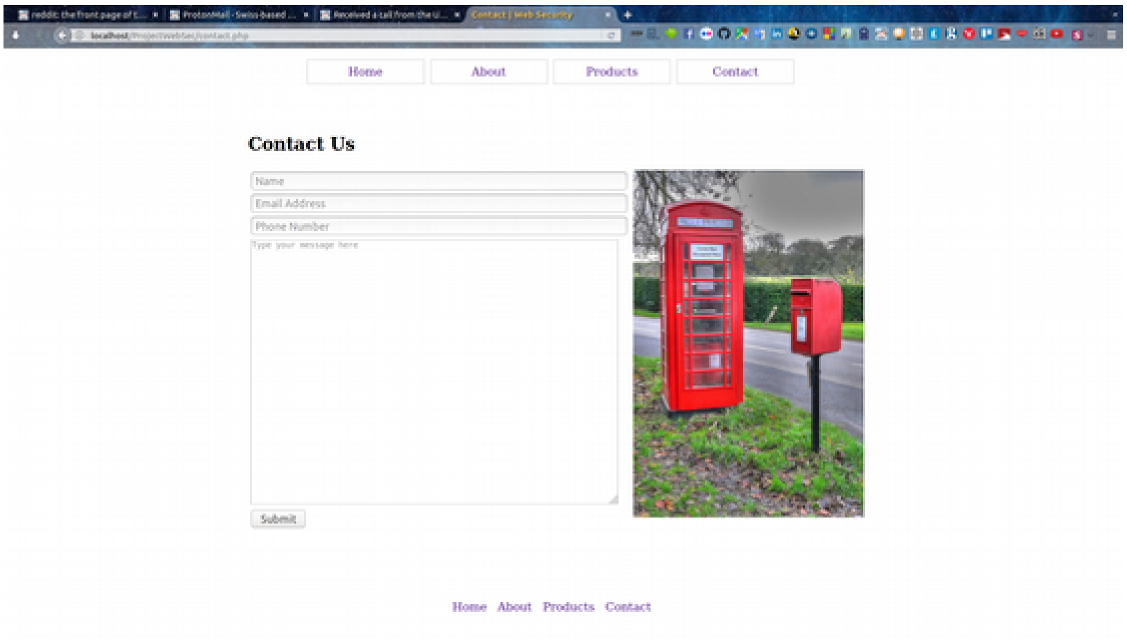
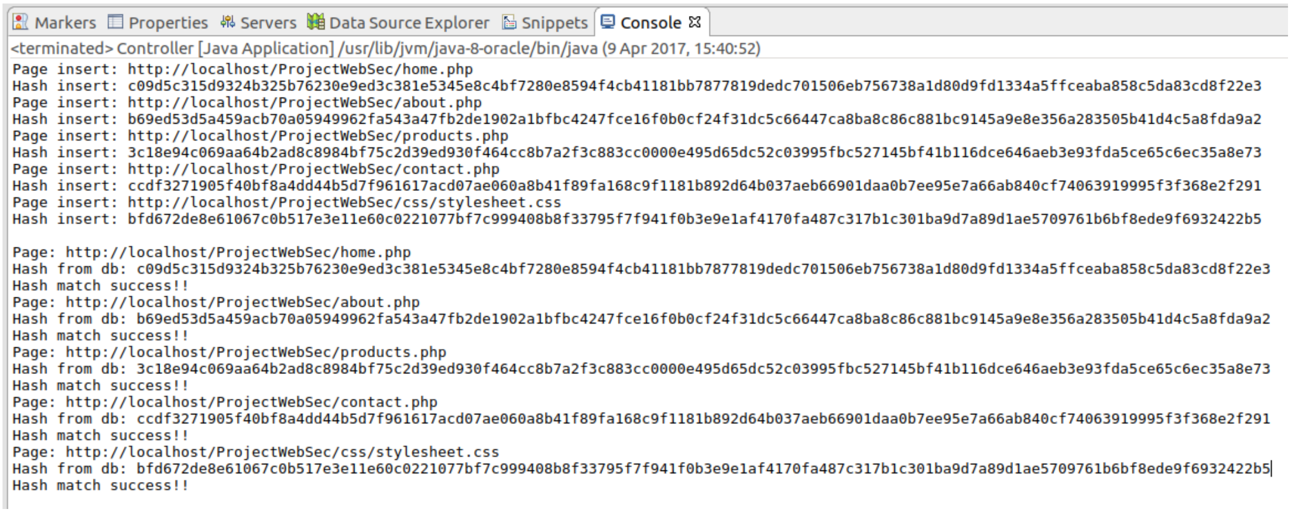


Illustration 4.4: Website Contact Page

It is a simple 4 page website with text, images and some basic formatting.

Illustration 4.5 shows the output to the console when the program has just started and everything is running normally. The first section confirms that the page hashes have been inserted into the database, and following is a list of the pages that have been checked, their hash value and a message indicating that the checked hash value matches the original.

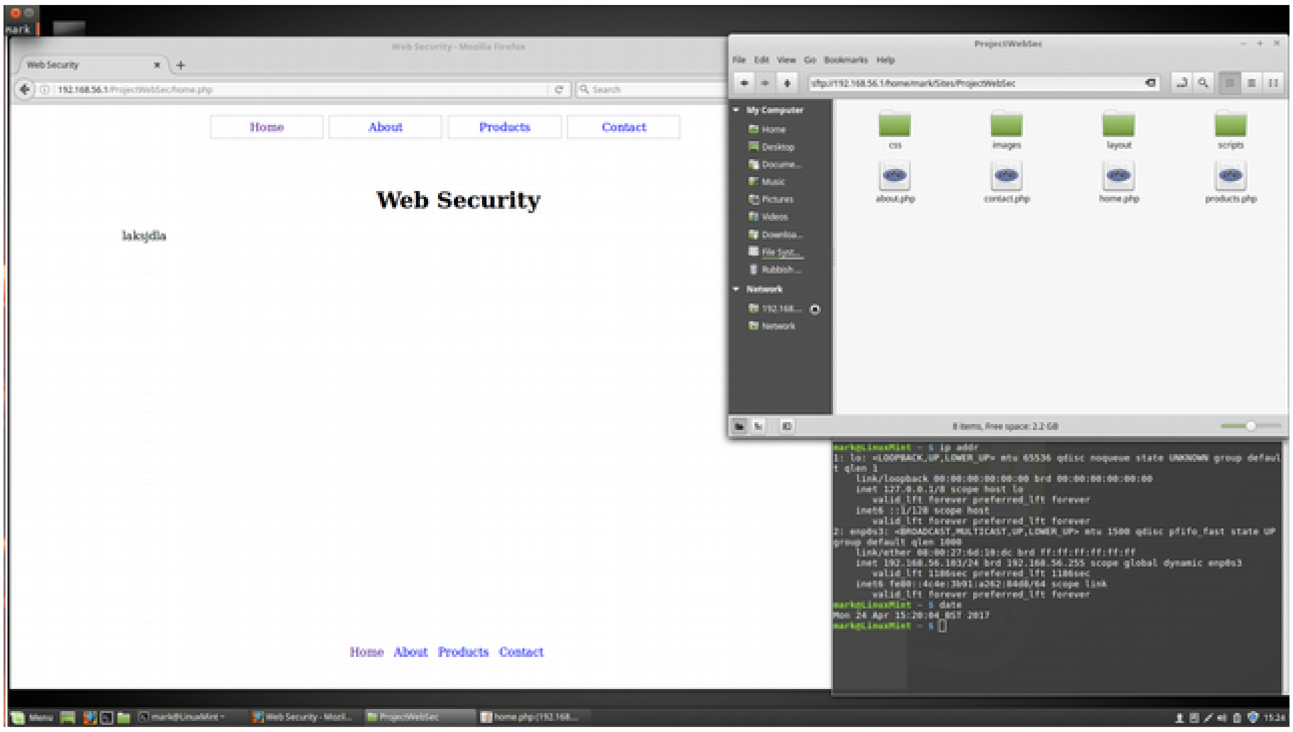


```
<terminated> Controller [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (9 Apr 2017, 15:40:52)
Page insert: http://localhost/ProjectWebSec/home.php
Hash insert: c09d5c315d9324b325b76230e9ed3c381e5345e8c4bf7280e8594f4cb41181bb7877819dedc701506eb756738a1d80d9fd1334a5ffceaba858c5da83cd8f22e3
Page insert: http://localhost/ProjectWebSec/about.php
Hash insert: b69ed53d5a459acb70a05949962fa543a47fb2de1902a1bfbc4247fce16f0b0cf24f31dc5c66447ca8ba8c86c881bc9145a9e8e356a283505b41d4c5a8fda9a2
Page insert: http://localhost/ProjectWebSec/products.php
Hash insert: 3c18e94c069aa64b2ad8c8984bf75c2d39ed930f464cc8b7a2f3c883cc0000e495d65dc52c03995fbc527145bf41b116dce646aeb3e93fda5ce65c6ec35a8e73
Page insert: http://localhost/ProjectWebSec/contact.php
Hash insert: ccdf3271905f40bf8a4dd44b5d7f961617acd07ae060a8b41f89fa168c9f1181b892d64b037aeb66901daa0b7ee95e7a66ab840cf74063919995f3f368e2f291
Page insert: http://localhost/ProjectWebSec/css/stylesheet.css
Hash insert: bfd672de8e61067c0b517e3e11e60c0221077bf7c999408b8f33795f7f941f0b3e9e1af4170fa487c317b1c301ba9d7a89d1ae5709761b6bf8ede9f6932422b5

Page: http://localhost/ProjectWebSec/home.php
Hash from db: c09d5c315d9324b325b76230e9ed3c381e5345e8c4bf7280e8594f4cb41181bb7877819dedc701506eb756738a1d80d9fd1334a5ffceaba858c5da83cd8f22e3
Hash match success!!
Page: http://localhost/ProjectWebSec/about.php
Hash from db: b69ed53d5a459acb70a05949962fa543a47fb2de1902a1bfbc4247fce16f0b0cf24f31dc5c66447ca8ba8c86c881bc9145a9e8e356a283505b41d4c5a8fda9a2
Hash match success!!
Page: http://localhost/ProjectWebSec/products.php
Hash from db: 3c18e94c069aa64b2ad8c8984bf75c2d39ed930f464cc8b7a2f3c883cc0000e495d65dc52c03995fbc527145bf41b116dce646aeb3e93fda5ce65c6ec35a8e73
Hash match success!!
Page: http://localhost/ProjectWebSec/contact.php
Hash from db: ccdf3271905f40bf8a4dd44b5d7f961617acd07ae060a8b41f89fa168c9f1181b892d64b037aeb66901daa0b7ee95e7a66ab840cf74063919995f3f368e2f291
Hash match success!!
Page: http://localhost/ProjectWebSec/css/stylesheet.css
Hash from db: bfd672de8e61067c0b517e3e11e60c0221077bf7c999408b8f33795f7f941f0b3e9e1af4170fa487c317b1c301ba9d7a89d1ae5709761b6bf8ede9f6932422b5
Hash match success!!
```

Illustration 4.5: Program Output at Start

The program will continue to output the second part of this screenshot at intervals of 1 minute unless it finds a mismatch in the hashes. Illustration 4.6 is



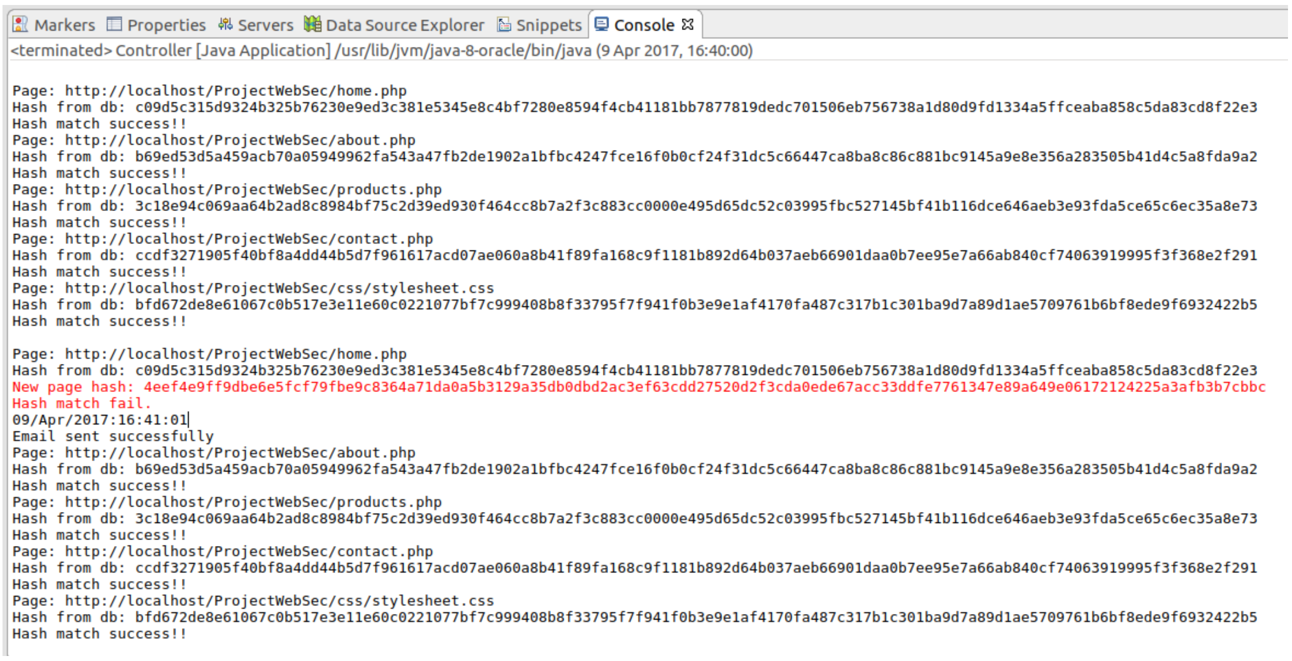
```
mark@LinuxMint ~ $ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:6d:10:dc brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.103/24 brd 192.168.56.255 scope global dynamic enp0s3
       valid_lft 1186sec preferred_lft 1186sec
    inet6 fe80::4c4e:3b91:a262:8448/64 scope link
       valid_lft forever preferred_lft forever
mark@LinuxMint ~ $ date
Mon 24 Apr 15:20:04 BST 2017
mark@LinuxMint ~ $
```

Illustration 4.6: Hacker Computer

of the hacker's computer and it shows access to the website files, modifying the content of one of the files and the output as displayed by the web server.

The terminal window shows the IP address of the computer and the time of attack, which will be confirmed in the email to the administrator.

Now the program will display output informing the user that the page has been modified and that an email with details has been sent to the administrator. This output is shown in illustration 4.7



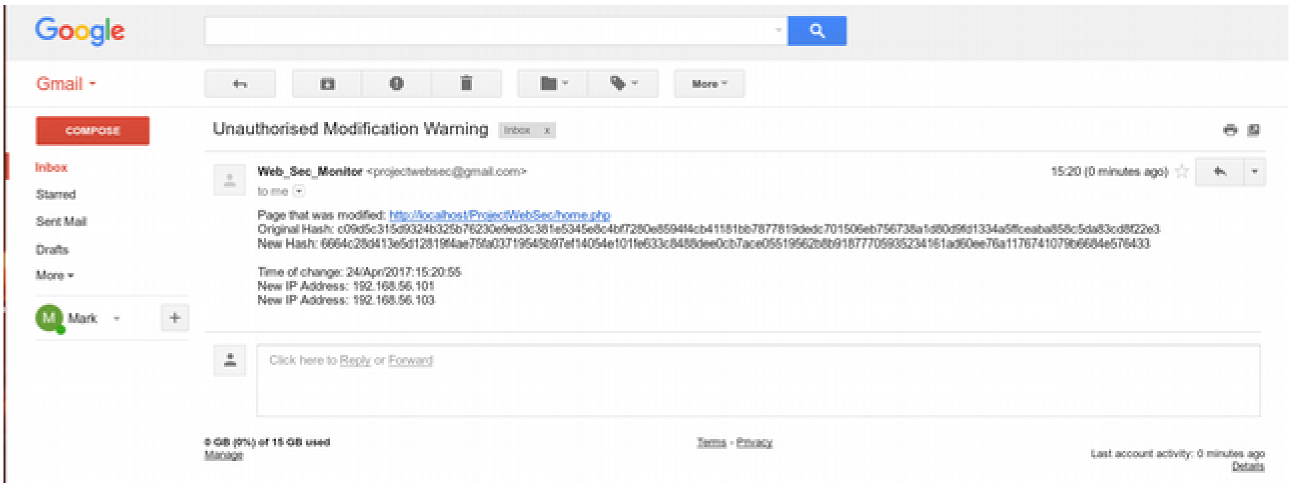
```
<terminated> Controller [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (9 Apr 2017, 16:40:00)

Page: http://localhost/ProjectWebSec/home.php
Hash from db: c09d5c315d9324b325b76230e9ed3c381e5345e8c4bf7280e8594f4cb41181bb7877819dedc701506eb756738a1d80d9fd1334a5ffceaba858c5da83cd8f22e3
Hash match success!!
Page: http://localhost/ProjectWebSec/about.php
Hash from db: b69ed53d5a459acb70a05949962fa543a47fb2de1902a1bfbc4247fce16f0b0cf24f31dc5c66447ca8ba8c86c881bc9145a9e8e356a283505b41d4c5a8fda9a2
Hash match success!!
Page: http://localhost/ProjectWebSec/products.php
Hash from db: 3c18e94c069aa64b2ad8c8984bf75c2d39ed930f464cc8b7a2f3c883cc0000e495d65dc52c03995fbc527145bf41b116dce646aeb3e93fda5ce65c6ec35a8e73
Hash match success!!
Page: http://localhost/ProjectWebSec/contact.php
Hash from db: ccdf3271905f40bf8a4dd44b5d7f961617acd07ae060a8b41f89fa168c9f1181b892d64b037aeb66901daa0b7ee95e7a66ab840cf74063919995f3f368e2f291
Hash match success!!
Page: http://localhost/ProjectWebSec/css/stylesheet.css
Hash from db: bfd672de8e61067c0b517e3e11e60c0221077bf7c999408b8f33795f7f941f0b3e9e1af4170fa487c317b1c301ba9d7a89d1ae5709761b6bf8ede9f6932422b5
Hash match success!!

Page: http://localhost/ProjectWebSec/home.php
Hash from db: c09d5c315d9324b325b76230e9ed3c381e5345e8c4bf7280e8594f4cb41181bb7877819dedc701506eb756738a1d80d9fd1334a5ffceaba858c5da83cd8f22e3
New page hash: 4eef4e9ff9dbe6e5fcf79fbe9c8364a71da0a5b3129a35db0dbd2ac3ef63cdd27520d2f3cda0ede67acc33ddfe7761347e89a649e06172124225a3afb3b7cbbc
Hash match fail.
09/Apr/2017:16:41:01
Email sent successfully
Page: http://localhost/ProjectWebSec/about.php
Hash from db: b69ed53d5a459acb70a05949962fa543a47fb2de1902a1bfbc4247fce16f0b0cf24f31dc5c66447ca8ba8c86c881bc9145a9e8e356a283505b41d4c5a8fda9a2
Hash match success!!
Page: http://localhost/ProjectWebSec/products.php
Hash from db: 3c18e94c069aa64b2ad8c8984bf75c2d39ed930f464cc8b7a2f3c883cc0000e495d65dc52c03995fbc527145bf41b116dce646aeb3e93fda5ce65c6ec35a8e73
Hash match success!!
Page: http://localhost/ProjectWebSec/contact.php
Hash from db: ccdf3271905f40bf8a4dd44b5d7f961617acd07ae060a8b41f89fa168c9f1181b892d64b037aeb66901daa0b7ee95e7a66ab840cf74063919995f3f368e2f291
Hash match success!!
Page: http://localhost/ProjectWebSec/css/stylesheet.css
Hash from db: bfd672de8e61067c0b517e3e11e60c0221077bf7c999408b8f33795f7f941f0b3e9e1af4170fa487c317b1c301ba9d7a89d1ae5709761b6bf8ede9f6932422b5
Hash match success!!
```

Illustration 4.7: Program Output With Hash Mismatch

Illustration 4.8 shows the email that was sent to the administrator's Gmail account. It has a warning message in the subject line and the email body gives details of which page was modified, the original hash and modified hash, confirming page status, the IP address(es) that were accessing the page at the time of modification and whether this IP address is known to the system.



Unauthorised Modification Warning Inbox x

Web_Sec_Monitor <projectwebsec@gmail.com>
to me
15:20 (0 minutes ago)

Page that was modified: http://localhost/ProjectWebSec/home.php
Original Hash: c09d5c315d9324b325b76230e9ed3c381e5345e8c4bf7280e8594f4cb41181bb7877819dedc701506eb756738a1d80d9fd1334a5ffceaba858c5da83cd8f22e3
New Hash: 6664c28d413e5d12819f4ae79fa03719545b97ef14054e101fe633c8488dee0cb7ace05519562b8b91877705935234161ad60ee76a1176741079b6684e576433

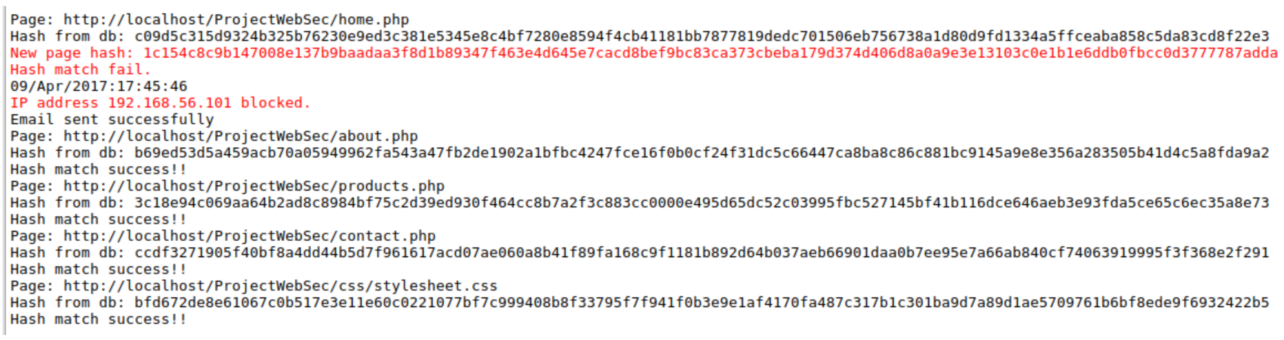
Time of change: 24/Apr/2017:15:20:55
New IP Address: 192.168.56.101
New IP Address: 192.168.56.103

Click here to Reply or Forward

0 GB (0%) of 15 GB used
Manage
Terms - Privacy
Last account activity: 0 minutes ago
Details

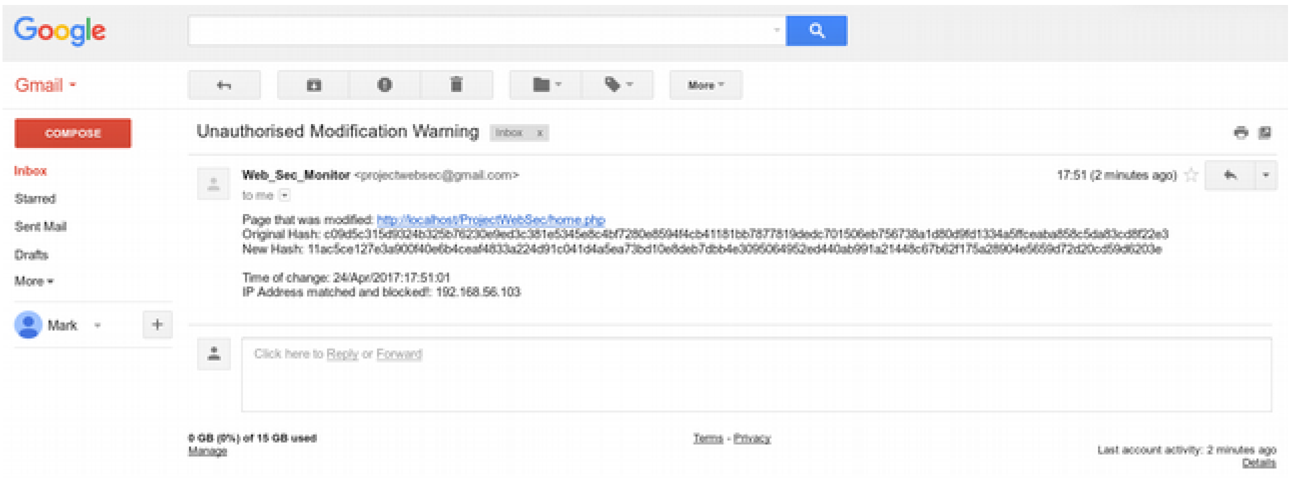
Illustration 4.8: Email Output

The program then restores the website from the backup and continues to monitor it. If it finds another hash mismatch and the IP address accessing the website is one that is known to the system, the email message informs the administrator (illustration 4.9 and 4.10) and that IP address is blocked from accessing the website. This is done by amending to the .htaccess file located in the website root with the word deny and the IP address of the hacker.



```
Page: http://localhost/ProjectWebSec/home.php
Hash from db: c09d5c315d9324b325b76230e9ed3c381e5345e8c4bf7280e8594f4cb41181bb7877819dedc701506eb756738a1d80d9fd1334a5ffceaba858c5da83cd8f22e3
New page hash: 1c154c8c9b147008e137b9baadaa3f8d1b89347f463e4d645e7cacd8bef9bc83ca373cbeba179d374d406d8a0a9e3e13103c0e1b1e6ddb0fbcc0d3777787adda
Hash match fail.
09/Apr/2017:17:45:46
IP address 192.168.56.101 blocked.
Email sent successfully
Page: http://localhost/ProjectWebSec/about.php
Hash from db: b69ed53d5a459acb70a05949962fa543a47fb2de1902a1bfbc4247fce16f0b0cf24f31dc5c66447ca8ba8c86c881bc9145a9e8e356a283505b41d4c5a8fda9a2
Hash match success!!
Page: http://localhost/ProjectWebSec/products.php
Hash from db: 3c18e94c069aa64b2ad8c8984bf75c2d39ed930f464cc8b7a2f3c883cc0000e495d65dc52c03995fbc527145bf41b116dce646aeb3e93fda5ce65c6ec35a8e73
Hash match success!!
Page: http://localhost/ProjectWebSec/contact.php
Hash from db: ccdf3271905f40bf8a4dd44b5d7f961617acd07ae060a8b41f89fa168c9f1181b892d64b037aeb66901daa0b7ee95e7a66ab840cf74063919995f3f368e2f291
Hash match success!!
Page: http://localhost/ProjectWebSec/css/stylesheet.css
Hash from db: bfd672de8e61067c0b517e3e11e60c0221077bf7c999408b8f33795f7f941f0b3e9e1af4170fa487c317b1c301ba9d7a89d1ae5709761b6bf8ede9f6932422b5
Hash match success!!
```

Illustration 4.9: Program Output With IP Address Blocked



Unauthorised Modification Warning

Web_Sec_Monitor <projectwebsec@gmail.com>
to me
17:51 (2 minutes ago)

Page that was modified: http://localhost/ProjectWebSec/home.php
Original Hash: c09d5c315d9324b325b76230e9ed3c381e5345e8c4bf7280e8594f4cb41181bb7877819dedc701506eb756738a1d80d9fd1334a5ffceaba858c5da83cd8f22e3
New Hash: 11ac5ce127e3a900f40e6b4ceaf4833a224d91c041d4a5ea73bd10e8deb7dbb4e3095064952ed440ab991a21448c67b62f175a28904e5659d72d20cd59d6203e

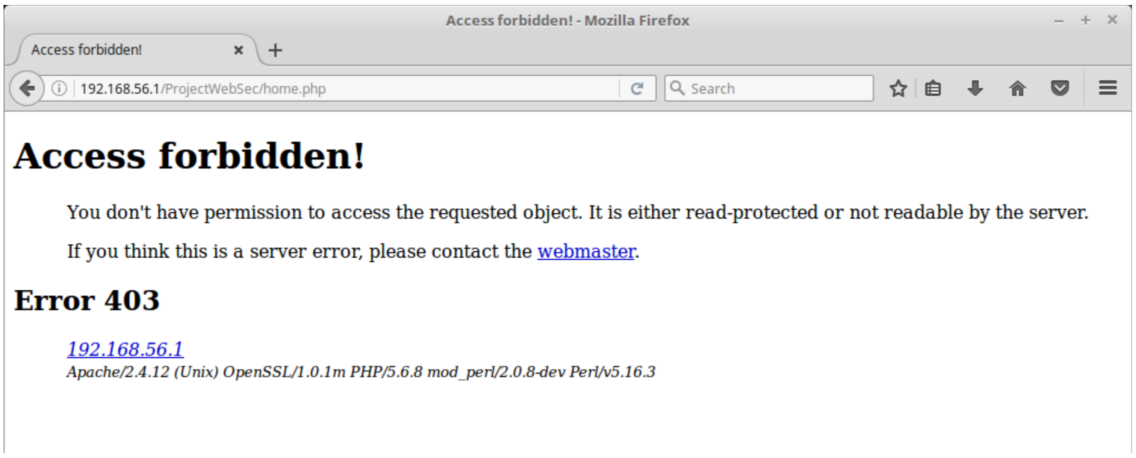
Time of change: 24/Apr/2017:17:51:01
IP Address matched and blocked!: 192.168.56.103

Click here to Reply or Forward

0 GB (0%) of 15 GB used
Manage
Terms - Privacy
Last account activity: 2 minutes ago
Details

Illustration 4.10: Email showing IP address blocked

Finally Illustration 4.11 confirms that the website is indeed blocked for the hacker.


Access forbidden! - Mozilla Firefox

Access forbidden!

192.168.56.1/ProjectWebSec/home.php

Access forbidden!

You don't have permission to access the requested object. It is either read-protected or not readable by the server.

If you think this is a server error, please contact the webmaster.

Error 403

192.168.56.1
Apache/2.4.12 (Unix) OpenSSL/1.0.1m PHP/5.6.8 mod_perl/2.0.8-dev Perl/v5.16.3

Illustration 4.11: Website Blocked

4.3 Program Testing

The previous section demonstrates the program operating under ideal conditions, but the program needs to show that it performs accurately in a variety of situations. In this section a series of tests will show how it behaves under different conditions, confirm if it is working correctly and determine whether any modifications need to be carried out.

Test #	Test Activity	Expected Result	Actual Result
1	Hacker changes file then restores it. File modified date will change but hash will remain the same	Program does not register any change	Program does not register any change
2	One hacker is blocked then a different hacker makes changes	Program should detect the change but not block the user	Program correctly does not block the second hacker until they have made multiple changes.
3	Hacker 1 modifies website. Then Hacker 2 modifies website. Then Hacker 1 modifies website again.	Program should correctly identify which hacker made repeated attacks.	This works but only if enough time has elapsed between hacks. The method of comparing dates means that there could be a small amount of overlap between one checking period and the next.

4	Hacker deletes file	Program will detect file is missing.	Program detected file was missing. This caused a hash mismatch, which triggered the alert and restore section of the program.
5	Hacker deletes entire website	Program will detect missing files and restore	Program detected missing files, which caused a hash mismatch and triggered the alert and restore part of the program. Program successfully restored the pages in the root of the website but was not able to recreate the folder structure for the other files. Program then hung.

5 Evaluation

5.1 Achievements

5.1.1 Product

The program successfully achieves all of its stated aims. It monitors the website, determines when a page has been modified, identifies the IP addresses of those using the site at the time of the modification, sends a message to the administrator informing them of what has happened.

The program then goes beyond that which was first proposed, by automatically restoring the website to its original state, and blocking the IP address of a user that has made repeated changes to the website.

5.1.2 Research

The research proved useful in that it identified which of the hashing algorithms would be most useful when creating hashes of the pages. None of the methods for identifying website hackers were exactly the same as the product in this report, showing that it is an original piece of work. Other useful elements of the research included learning which log files would hold the most relevant information. It was reassuring to find out that the program could never be a full and total solution to the problem, only a small part of the whole.

5.1.3 System Design

The program has been designed in a clear, object-orientated manner, with different classes for each of the core functions. The code includes such design patterns as Data Access Object and Prepared Statements which ensure that database access is clean and such access cannot corrupt the database. The program provides useful information to the user at all stages of its operation in a meaningful yet simple way, using console messages for ongoing proceedings and emails for key events.

The program is well commented throughout and uses descriptive names for variables, classes and functions. In this way it would be easy for another programmer to understand it and modify it without needing to refer back to the original author.

5.1.4 Time Management

The program, accompanying report and presentation were all successfully completed in advance of the final deadline. Time was set aside each week and although not all of the interim deadlines were met, due to pressures of other assignments and confusion over exactly how to implement the project, there was never any danger of the project not being completed by the final deadline.

5.2 Improvements

5.2.1 Restore only the page that was modified

The current operation of the program is to restore the entire website once an attack has been discovered. This is to ensure that the website is fully operational in the way it was intended. While this is an acceptable method for a fairly small site such as the one in this paper, it might suffer a performance hit on a very large site. To counter this, the program could be modified to only restore the page that was affected.

6 Conclusion

6.1 Concluding Remarks

This project has shown that a method for monitoring a website and providing basic protection against hackers is both viable and cost effective. The program in this project has been shown to be effective in all areas of its operation. Any website administrator could use a program such as the one demonstrated here to help keep their site free from unwanted modifications, while only needing a small amount of extra resources.

6.2 Future Work

6.2.1 Multimedia Analysis

The program could be further improved by analysing the images and other multimedia content for changes. With the system as it stands, a potential attacker could upload an image that has the same name as one that is already in place, but with a different message. A simple way of doing this would be to monitor the dimensions and file size of the images, since it would be difficult to generate a compressed image with a different message that still matches these attributes.

6.2.2 Interface

The program does not have an interface and is controlled via the command line or terminal. While the reason for this has been justified in Chapter 3.3.12, a simple interface that allows the input of page names to the database and for starting and stopping the program. It is expected that potential users of this program will be experienced and knowledgable technical operators, although may not necessarily be the case.

References

Al- Khanjari, Z., Alanee, A., Kraiem, N., Jamoussi, Y., 2013. Proposing a real time internal intrusion detection system towards a secured development of e-government web site. Eur. Sci. J. 3 SE, 27.

Bejtlich, R., 2013. The Practice of Network Security Monitoring: Understanding Incident Detection and Response. No Starch Press.

Cao, Z., Xiong, G., Zhao, Y., Guo, L., Fang, B., 2014. Two-phased method for detecting evasive network attack channels. China Commun. 11, 47–58. doi:10.1109/CC.2014.6911087

Gurjwar, R.K., Sahu, D.R., Tomar, D.S., 2013. An Approach to Reveal Website Defacement. Int. J. Comput. Sci. Inf. Secur. 11, 73–79.

Huo, J., Qu, H., Liu, L., 2012. Design and Implementation of Automatic Defensive Websites Tamper-Resistant System. J. Softw. 7. doi:10.4304/jsw.7.10.2379-2386

Long, F., Mohindra, D., Seacord, R.C., Sutherland, D.F., Svoboda, D., 2012. The CERT Oracle Secure Coding Standard for Java, 1st ed. Pearson Education Inc.

Macrae, A., 2013. Identifying threats in real time. Netw. Secur. 2013, 5–8. doi:10.1016/S1353-4858(13)70119-3

Mishra, C.M., Singh, R.D., 2015. Web Defacement: Threat Analysis and Modelling. Int. J. Comput. Secur. Source Code Anal. 1, 26–30.

Pankaj, 2016. JavaMail Example - Send Mail in Java using SMTP - JournalDev [WWW Document]. JournalDev. URL http://www.journaldev.com/2532/javamail-example-send-mail-in-java-smtp (accessed 1.30.17).

Park, K.C., Shin, H., Park, W.H., Lim, J.I., 2015. New detection method and countermeasure of cyber attacks in mix networks. Multimed. Tools Appl. 74, 6509–6518. doi:10.1007/s11042-014-2127-7

Parsons, D., 2008. Dynamic Web Application Development using XML and Java. Cengage Learning.

Pramono, Y.W.T., Suhardi, 2014. Anomaly-based intrusion detection and prevention system on website usage using rule-growth sequential pattern analysis: Case study: Statistics of Indonesia (BPS) website, in: 2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA). Presented at the 2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA), pp. 203–208. doi:10.1109/ICAICTA.2014.7005941

Rechavi, A., Berenblum, T., Maimon, D., Sevilla, I.S., 2015. Hackers topology matter geography: Mapping the dynamics of repeated system trespassing events networks, in: 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). Presented at the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 795–804. doi:10.1145/2808797.2808873

Thakore, U., Weaver, G.A., Sanders, W.H., 2016. A Quantitative Methodology for Security Monitor Deployment, in: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Presented at the 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 1–12. doi:10.1109/DSN.2016.10

Young, D., 2015. Network Security “Security of Today and Tomorrow.” IT Showc. Proc. 120.

7 Bibliography

Borgolte, K., Kruegel, C. and Vigna, G. (2014). Relevant change detection. *Proceedings of the 23rd International Conference on World Wide Web - WWW '14 Companion*, [online] pp.595 - 598. Available at: https://dl.acm.org/citation.cfm?doid=2567948.2578039 [Accessed 17 Apr. 2017].

Burgett, S. (2003). *How to Programmatically add IP Addresses to IIS's Deny Access List using C# and WMI - CodeProject*. [online] Codeproject.com. Available at: https://www.codeproject.com/articles/4671/how-to-programmatically-add-ip-addresses-to-iis-s [Accessed 17 Apr. 2017].

CompileTimeError. (2017). *Java: Download/Extract all images from website URL in java*. [online] Available at: http://www.compiletimerror.com/2013/08/java-downloadextract-all-images-from.html [Accessed 17 Apr. 2017].

Gupta, P. (2015). *Cryptography and Network Security*. 1st ed. Delhi: PHI Learning.

Hortonworks. (2017). *Hadoop tutorial: How to Refine and Visualize Server Log Data*. [online] Available at: https://hortonworks.com/hadoop-tutorial/how-to-refine-and-visualize-server-log-data/ [Accessed 17 Apr. 2017].

Htaccess-guide.com. (2017). *What is .htaccess? - Apache .htaccess Guide, Tutorials & Examples*. [online] Available at: http://www.htaccess-guide.com/ [Accessed 17 Apr. 2017].

IGN, M. (2015). Indonesia Web Defacement Attacks Analysis for Anti Web Defacement. *Ticom*, [online] 3(3). Available at: http://ejurnal.net/portal/index.php/ticom/article/view/309 [Accessed 17 Mar. 2017].

Java.net. (2017). *JavaMail API Reference Implementation: Wiki: Home — Project Kenai*. [online] Available at: https://java.net/projects/javamail/pages/Home [Accessed 17 Apr. 2017].

Lyne, J. (2017). *Forbes Welcome*. [online] Forbes.com. Available at: https://www.forbes.com/sites/jameslyne/2013/09/06/30000-web-sites-hacked-a-day-how-do-you-host-yours/#152be5ee1738 [Accessed 17 Apr. 2017].

Niemela, J. and Kesti, V. (2014). *Detecting Unauthorised Changes to Website Content*. US20140317754 A1.

Snort.org. (2017). *Snort - Network Intrusion Detection & Prevention System*. [online] Available at: https://www.snort.org/ [Accessed 17 Apr. 2017].

The HTML Editor Kit. (2017). 1st ed. [ebook] O'Reilly. Available at: http://examples.oreilly.com/jswing2/code/goodies/HtmlEdKit.pdf [Accessed 17 Apr. 2017].

Appendices

Appendix A

Terms of Reference

Course Learning Outcomes

• To equip students with the practical skills and the theoretical underpinning necessary for the analysis, design and construction of information systems using a variety of methods;

• To enable students to gain an advanced understanding of database management systems, and to develop advanced skills in their exploitation;

• To develop the ability to design, specify, construct and maintain software; and

• To equip students with skills and attributes that will enhance employment opportunities in general computing and IT.

Project Background

Current estimates suggest that as many as 30,000 websites are hacked each day. While there are plenty of tutorials showing how to secure your website, not everyone follows best practices, perhaps for reasons of convenience and/or ignorance and even the best security is not necessarily perfect as technology advances. In addition, a particular website may find itself the target of a sustained and repeated attack from a hacker who wants to prove something to themselves.

A defaced website can not only be an embarrassment, causing harm to your reputation but can have much more serious consequences. A hacker could change the words or pictures or worse they could insert a phishing link, which could ultimately cause the website to be removed from Google Search results and the hosting company will likely close down and blacklist the site.

Aim

This project aims to develop a way of protecting against a hacker who may be targeting a particular website, trying to hack it again and again as a way of proving something to themselves. The project aims to show a way of identifying an individual attacker who is making repeated unauthorised changes to to a web-page, so that they can be blocked from accessing the site

Related Work

Example 1: Detecting Unauthorised Changes to Website Content

This is a US patent for a system which monitors websites and aims to prevent unauthorised changes. It does this by collecting and storing authorised content policy sets and tries to identify websites which do not conform to these policies, whereby it will alert the website administrator. This is different to the solution proposed in this paper which will collect hashes for key pages and monitor those for changes.

Example 2: Monitor Hacked Files Tool

This is a php script which is designed to run, by default every 60 minutes, and checks the time-stamp of the files on the server to see if they have changed. This solution has two key differences which are addressed in the solution in this project. Firstly it only looks at the time-stamps not the files themselves. With the right access it is possible for a hacker to cover their tracks by manipulating the time-stamps. Also the default setting is to only check every hour, compared to the solution in this project which checks every minute. Although the code is open-source and can be modified with the right knowledge, many people won't know how to do this. A lot of damage can be done in an hour! By checking more frequently the system is more likely to catch an attacker in the act and block them.

Proposed Solution

The solution in this project proposes to create a database of hashes of certain key pages. These pages will then be monitored every 60 seconds and the hashes of the live pages will be checked against the hashes in the database. If the hashes are different, this means someone has hacked the website and a list of IP addresses that accessed the page in this time-period will be obtained from the log. The website will be restored from the backup and monitoring continued. If changed hashes show that the website has been hacked again, the IP addresses from the log will be compared with those from previous attacks to see if there is a match. In this way, a repeat attacker can be blocked from accessing the website.

Objectives

- Complete a comprehensive literature review of at least 20 articles or journals on the subject of website security
- Create a basic website and host it on a local server, create hashes of the key pages and store them in a database
- Design, create and test a program using Java which will check the selected pages at specified intervals and compare them with the ones stored in the database. Also
    - When a hash mismatch is found, get IP addresses from the log, store them in the database and compare them with any IP addresses already in the database

- Implement the program and evaluate whether it achieves the aims of the project
- Create a presentation that demonstrates the project
- Write a report containing full details of all the above objectives

Problems

- Actually getting the website hacked – since the project will be hosted on a local web-server, this action will need to be simulated. I have 4 devices on my local network (a HTPC, a laptop, a Raspberry Pi and a mobile phone). One will host, two will act as normal traffic and one to be the hacker machine.
- Will there be enough traffic to show both normal and hacker traffic – again this will need to be simulated
- How to tell whether a repeated IP address is the hacker and not just a return visitor or someone refreshing the page? Two possible solutions:
    - Create a whitelist of IP addresses that are allowed to make changes – would then need to automatically add the new hashes to the database
    - Have an option in the program to pause checking while changes are made

Deliverables

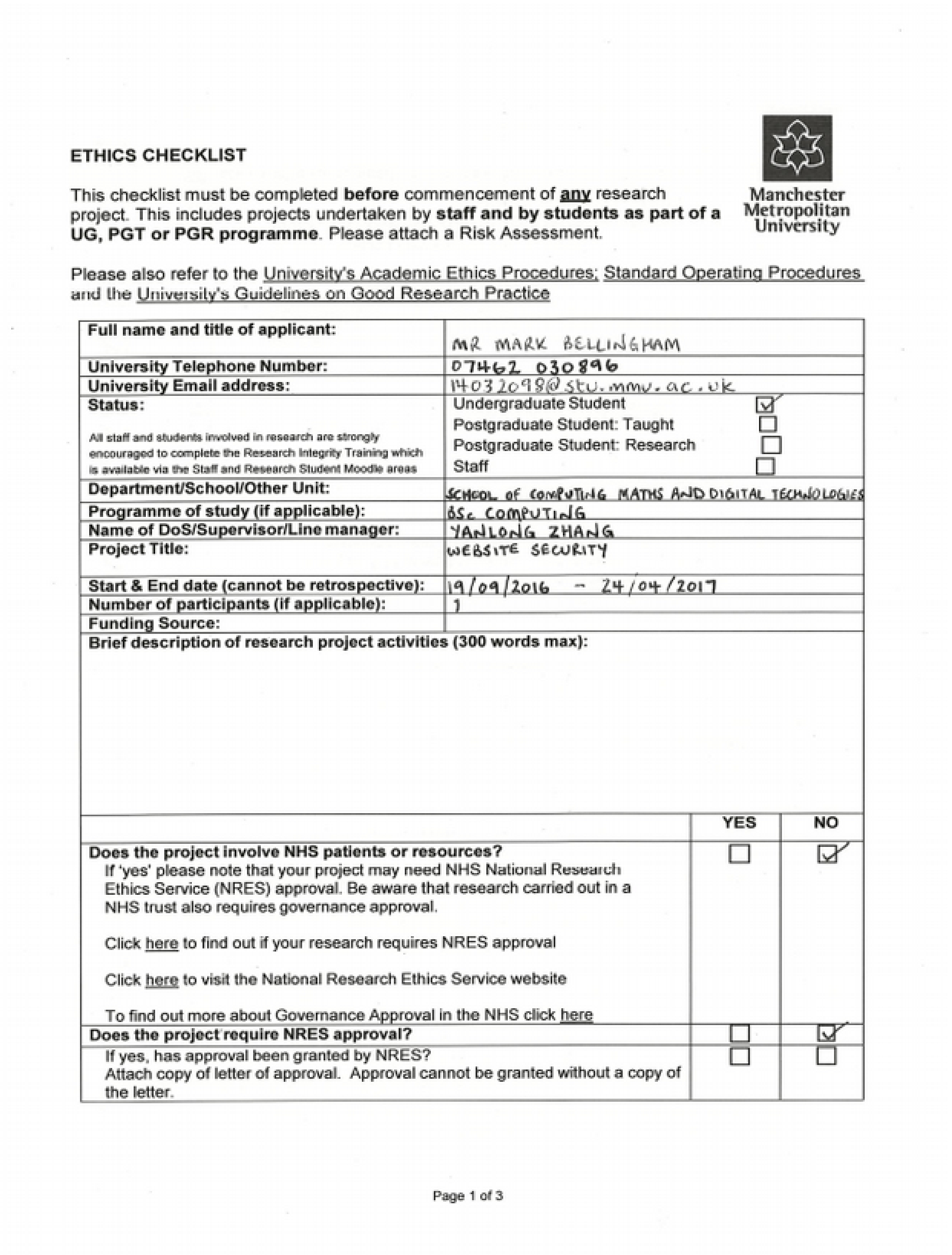
- A website
- Program in Java that does the following:
    - check the website every 60 seconds and returns the hashes of key pages
    - check hashes against those in a database
    - extract the IP addresses from the log for a specified time period and stores them in a database
    - check IP addresses against those in a database
    - alert the user of any hacks and any IP address matches
- Progress reports delivered within set deadlines
- Final report

Timetable All dates are week commencing	
17/10/2016	Terms of Reference
24/10/2016	Ethics Form
31/10/2016	Literature Review
07/11/2016	Literature Review
14/11/2016	Literature Review
21/11/2016	Literature Review
28/11/2016	Product Design
05/12/2016	Product Design
12/12/2016	Product Design
19/12/2016	Product Design and Implementation
26/12/2016	Product Design and Implementation
02/01/2017	Product Design and Implementation
09/01/2017	Implementation and Evaluation
16/01/2017	Implementation and Evaluation
23/01/2017	Implementation and Evaluation
30/01/2017	Evaluation
06/02/2017	Report Writing
20/02/2017	Report Outline
27/02/2017	Presentation Preparation
06/03/2017	Draft Slides
13/03/2017	Practice Presentation
20/03/2017	Report writing
24/04/2017	Report and Product Presentation Final Presentation Slides

Required Resources

- Web server for hosting - This will be done using the LAMP (Linux, Apache, MySQL, PHP) stack on my private network. The access addresses to be monitored will all be in the 192.168.x.x range.
- Backup facilities – The project will use GitHub for version control and Dropbox for automated backups, as well as a local copy that the program will use.
- Text editor to create the website – there are many free options available such as Notepad++ and Komodo.
- Database to hold hashes and IP logs – will be MySQL
- IDE for creating the Java program – this will be Eclipse.

Ethics Form



ETHICS CHECKLIST

Manchester Metropolitan University

This checklist must be completed **before** commencement of <u>**any**</u> research project. This includes projects undertaken by **staff and by students as part of a UG, PGT or PGR programme**. Please attach a Risk Assessment.

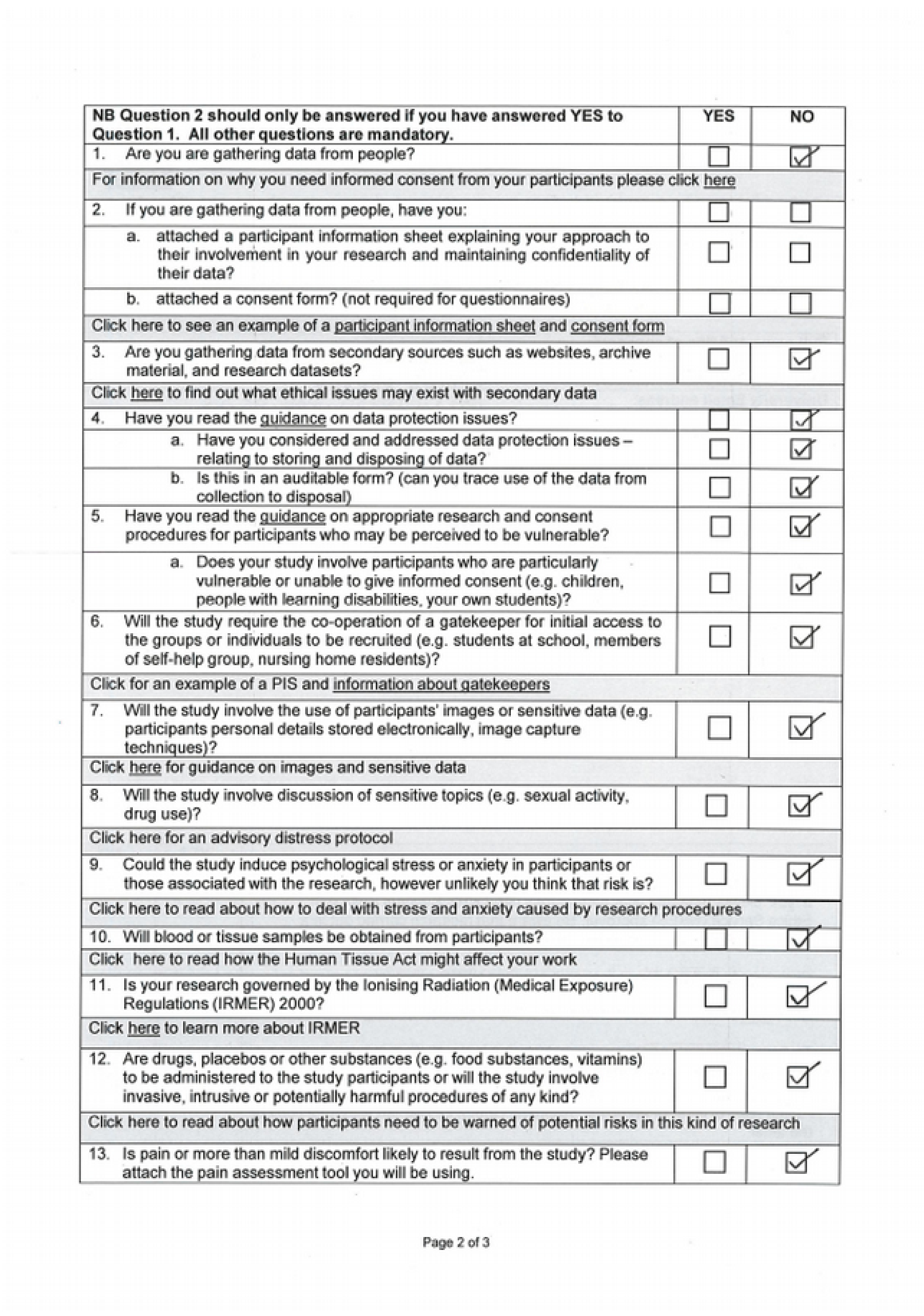
Please also refer to the University's Academic Ethics Procedures; Standard Operating Procedures and the University's Guidelines on Good Research Practice

Full name and title of applicant:	MR MARK BELLINGHAM
University Telephone Number:	07462 030896
University Email address:	14032098@stu.mmu.ac.uk
Status: All staff and students involved in research are strongly encouraged to complete the Research Integrity Training which is available via the Staff and Research Student Moodle areas	Undergraduate Student ☑ Postgraduate Student: Taught ☐ Postgraduate Student: Research ☐ Staff ☐
Department/School/Other Unit:	SCHOOL OF COMPUTING MATHS AND DIGITAL TECHNOLOGIES
Programme of study (if applicable):	BSc COMPUTING
Name of DoS/Supervisor/Line manager:	YANLONG ZHANG
Project Title:	WEBSITE SECURITY
Start & End date (cannot be retrospective):	19/09/2016 - 24/04/2017
Number of participants (if applicable):	1
Funding Source:	

Brief description of research project activities (300 words max):

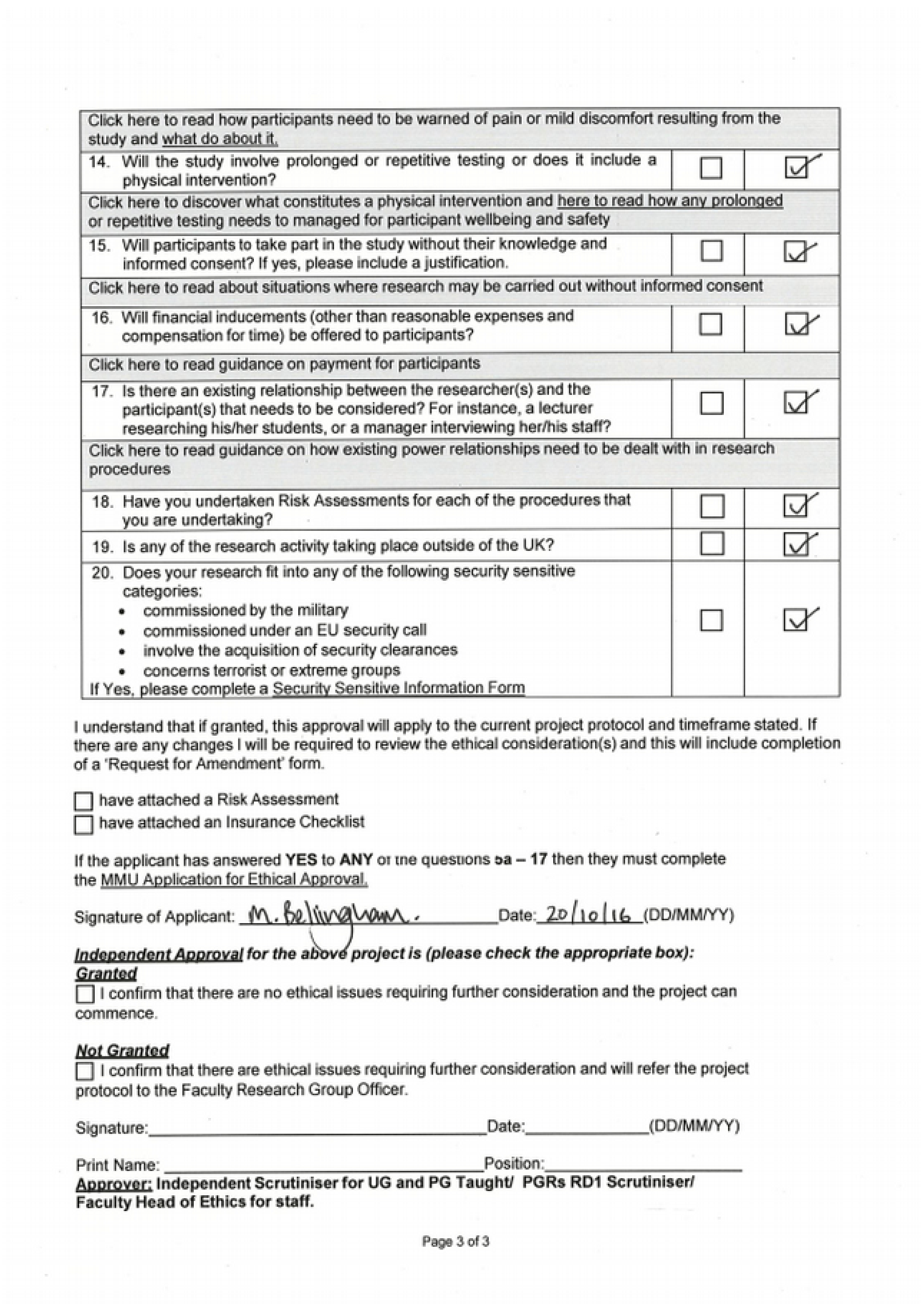
	YES	NO
Does the project involve NHS patients or resources? If 'yes' please note that your project may need NHS National Research Ethics Service (NRES) approval. Be aware that research carried out in a NHS trust also requires governance approval. Click here to find out if your research requires NRES approval Click here to visit the National Research Ethics Service website To find out more about Governance Approval in the NHS click here	☐	☑
Does the project require NRES approval?	☐	☑
If yes, has approval been granted by NRES? Attach copy of letter of approval. Approval cannot be granted without a copy of the letter.	☐	☐

Page 1 of 3



NB Question 2 should only be answered if you have answered YES to Question 1. All other questions are mandatory.	YES	NO
1. Are you are gathering data from people?	☐	☑
For information on why you need informed consent from your participants please click here		
2. If you are gathering data from people, have you:	☐	☐
a. attached a participant information sheet explaining your approach to their involvement in your research and maintaining confidentiality of their data?	☐	☐
b. attached a consent form? (not required for questionnaires)	☐	☐
Click here to see an example of a participant information sheet and consent form		
3. Are you gathering data from secondary sources such as websites, archive material, and research datasets?	☐	☑
Click here to find out what ethical issues may exist with secondary data		
4. Have you read the guidance on data protection issues?	☐	☑
a. Have you considered and addressed data protection issues – relating to storing and disposing of data?	☐	☑
b. Is this in an auditable form? (can you trace use of the data from collection to disposal)	☐	☑
5. Have you read the guidance on appropriate research and consent procedures for participants who may be perceived to be vulnerable?	☐	☑
a. Does your study involve participants who are particularly vulnerable or unable to give informed consent (e.g. children, people with learning disabilities, your own students)?	☐	☑
6. Will the study require the co-operation of a gatekeeper for initial access to the groups or individuals to be recruited (e.g. students at school, members of self-help group, nursing home residents)?	☐	☑
Click for an example of a PIS and information about gatekeepers		
7. Will the study involve the use of participants' images or sensitive data (e.g. participants personal details stored electronically, image capture techniques)?	☐	☑
Click here for guidance on images and sensitive data		
8. Will the study involve discussion of sensitive topics (e.g. sexual activity, drug use)?	☐	☑
Click here for an advisory distress protocol		
9. Could the study induce psychological stress or anxiety in participants or those associated with the research, however unlikely you think that risk is?	☐	☑
Click here to read about how to deal with stress and anxiety caused by research procedures		
10. Will blood or tissue samples be obtained from participants?	☐	☑
Click here to read how the Human Tissue Act might affect your work		
11. Is your research governed by the Ionising Radiation (Medical Exposure) Regulations (IRMER) 2000?	☐	☑
Click here to learn more about IRMER		
12. Are drugs, placebos or other substances (e.g. food substances, vitamins) to be administered to the study participants or will the study involve invasive, intrusive or potentially harmful procedures of any kind?	☐	☑
Click here to read about how participants need to be warned of potential risks in this kind of research		
13. Is pain or more than mild discomfort likely to result from the study? Please attach the pain assessment tool you will be using.	☐	☑

Page 2 of 3



Click here to read how participants need to be warned of pain or mild discomfort resulting from the study and what do about it.		
14. Will the study involve prolonged or repetitive testing or does it include a physical intervention?	☐	☑
Click here to discover what constitutes a physical intervention and here to read how any prolonged or repetitive testing needs to managed for participant wellbeing and safety		
15. Will participants to take part in the study without their knowledge and informed consent? If yes, please include a justification.	☐	☑
Click here to read about situations where research may be carried out without informed consent		
16. Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?	☐	☑
Click here to read guidance on payment for participants		
17. Is there an existing relationship between the researcher(s) and the participant(s) that needs to be considered? For instance, a lecturer researching his/her students, or a manager interviewing her/his staff?	☐	☑
Click here to read guidance on how existing power relationships need to be dealt with in research procedures		
18. Have you undertaken Risk Assessments for each of the procedures that you are undertaking?	☐	☑
19. Is any of the research activity taking place outside of the UK?	☐	☑
20. Does your research fit into any of the following security sensitive categories: • commissioned by the military • commissioned under an EU security call • involve the acquisition of security clearances • concerns terrorist or extreme groups If Yes, please complete a Security Sensitive Information Form	☐	☑

I understand that if granted, this approval will apply to the current project protocol and timeframe stated. If there are any changes I will be required to review the ethical consideration(s) and this will include completion of a 'Request for Amendment' form.

☐ have attached a Risk Assessment
☐ have attached an Insurance Checklist

If the applicant has answered **YES** to **ANY** of the questions **5a – 17** then they must complete the MMU Application for Ethical Approval.

Signature of Applicant: M. Bellingham. Date: 20/10/16 (DD/MM/YY)

***Independent Approval** for the above project is (please check the appropriate box):*
Granted
☐ I confirm that there are no ethical issues requiring further consideration and the project can commence.

Not Granted
☐ I confirm that there are ethical issues requiring further consideration and will refer the project protocol to the Faculty Research Group Officer.

Signature:______________________Date:__________(DD/MM/YY)

Print Name: ______________________Position:__________________
Approver: Independent Scrutiniser for UG and PG Taught/ PGRs RD1 Scrutiniser/ Faculty Head of Ethics for staff.

Page 3 of 3