UNIVERSITY OF BIRMINGHAM

Implementation of a hidden functionality in a web page

Project report, B.Sc. Computer Science

School of Computer Science, University of Birmingham

10th March 2017

AUTHOR: *Sebastian Lewis(1446296)*
SUPERVISOR: *Dr. Erik Tews*

Abstract

This project looks at how people with any sensitive problem can access information on the matter without any fear of others finding out. The aim with this project is to produce a website that will display sensitive information required by a user only to any users who know of some embedded trigger. People often have a need to view information on private or sensitive matters online without raising the suspicions of anyone who has access to their computer. To allow users to do this my aim was to create a web page that will display a second web page of my creation with information on any sensitive subjects they wish to view. The second page will only be shown once the user visits certain pages a certain number of times, this should not be obvious to anyone who doesn't know the pages to visit, therefore allowing users to visit the second page without any worries about their privacy. The solution currently uses a server that once a trigger is activated will serve a web page with a javascript trigger that can be activated and will cause the page to display a second web page hosted on a different server of my creation which contains sensitive information. The issues with current solutions for this problem such as private browsing in most modern browsers is that they can be simply disabled also the user will leave big gaps in their history from the times they browse using any private browsing feature. With my solution the first page currently gets saved to history but not into cache meaning anyone looking for what the user has been browsing will be greeted by a non suspicious generic page (instead of a gap in the history), This page has no indication that there is a hidden trigger and anyone accessing it without knowing the trigger can't get to the sensitive page. The second page does not end up in browser cache or history therefore allowing the user to view information whilst knowing that no evidence will be left behind on their computer.

Location of software

All software for this project can be found at:
https://git-teaching.cs.bham.ac.uk/mod-40cr-proj-2016/sxl496

Acknowledgements

Thanks to my family and friends for support and to Erik Tews for support and guidance.

1. Introduction

The motivation for this project was to allow anyone in need to view information on some sensitive subject which they weren't conformable telling anyone with access to their computer about. This problem could be in a home with a shared computer or for example in a workplace where the user has some issue with a colleague or their employer knowing the subject they are viewing.

The aim of this project was to solve this problem by providing a web page that allowed any user interested in any sensitive subject to view information on it without having to worry about anyone with access to their computer knowing they have viewed the information. To do this would require a web server to be created (running off my home machine) which would display a generic non suspicious web page. Within the web server there would be a trigger to send the user some secret web page with information on some sensitive subject. This report will describe the design and implementation of this solution.

People are often embarrassed to discuss sensitive subjects or may feel ashamed about something they need help with. This solution will allow the secret web page to be simply swapped out with pages that have information on sensitive subjects such as STDs, drug abuse, or whistle blowing along with anything else someone may require information on. Having this server in place will allow users to access it via any browser and with knowledge of the trigger gain access to the information they require while anyone else who accesses the site with no knowledge of the trigger will not be aware of any secret information or hidden functionality. In practice the trigger would be given along with the url to access the page with both changing whenever necessary (e.g. once a month). If a user wanted to get further information on some medical concern a doctor could give them a card with the url and how to reach the secret information/activate the trigger for example.

Throughout the project several different ways to implement both the trigger and secret page were tried which weren't appropriate or possible, these will be discussed in the report. I am happy with the final implementation of the solution but there are many improvements that could be made on top of what I implemented.

In this report I will cover the background material I looked at before implementation including current solutions in place, the analysis and specification I came up with based on analysis of the problem and the background material, the design of the project solution covering ideas I had throughout and what I finally went with, implementing my solution with changes the solution went through and the testing I used to check my solution worked in a way that solved the problem, then how I managed the project, and finally results and evaluation of the project. After this I will discuss how I felt the project went, what went wrong, what went right, what could be improved on and after I will finally conclude by giving a statement on how the project solution addresses the problem given above.

2. Background

The background needed to understand this project is quite simple, normal popular browsers such as Chrome, Firefox, Opera, and Microsoft Edge all save any page a user visits into history. Based on the HTTP header the web server with the page gives the browsers will also save the page and any items within the page, such as HTML, CSS, Javascript, image files, etc. into browser cache. This is then used when reloading the page to reduce the load time.

The majority of modern browsers try to solve the problem of private browsing by having some private browsing mode available such as Chrome's incognito mode, Firefox's private browsing, Opera's private browsing, and Microsoft Edge's InPrivate browsing. All of these versions of private browsing don't save visited web pages to regular history or cache most have their own cache which is deleted once the private browsing window is closed.

Another alternative worth mentioning is Epic Privacy Browser which is a browser designed to provide a user with private browsing. This works similarly to the other forms of private browsing but also routes some requests through proxy servers and blocks trackers from websites. Like the other types of private browsing Epic Privacy Browser has it's own cache that is deleted when the application is closed. There are other browsers designed for users privacy such as Tor Browser. With these browsers and the privacy browsing modes of other browsers users would be able to visit any website on the internet without having that website saved to any history or cache. User's may have any number of sensitive issues that they wish to view information on. Using these browsers would mean that the user could view information on any sensitive subject they may have a need to view.

The other way to prevent a web page from being cached is to alter the HTTP header to specify to the browser that the contents of the HTTP response shouldn't be cached. When I started the project and needed to know how to do this I found a discussion on stackoverflow explaining the different set of headers needed to do this across all browsers. I used these headers on the web server I made which prevented anything from being cached. The solution I set out to implement would use the HTTP headers to prevent the pages from being cached. The HTTP headers could be created and altered by the web server I implemented. I could then use this to ensure that the second page isn't cached and that it wouldn't appear in history therefore not seeming as if the user had visited it.

Evidence of users visiting web pages being on a user's computer is only a part of keeping their privacy however. Even using any of the private browsing modes and having nothing in cache, history, or stored on their computer there would still be network activity viewable by ISPs or an employer. Epic Privacy Browser and similar browsers such as Tor use a Virtual private network to hide the identity of the server you are connecting to while also encrypting traffic so an ISP or employer wouldn't be able to see any of the user's HTTP requests or who the user is connecting to. Opera also has it's own free VPN to hide the user's browsing history from anyone they're sharing a network with.

3. Analysis and Specification

Looking at the problem of users accessing information on sensitive subjects without anyone else having knowledge of what they were looking at, there would have to be some way to give users access to a web page with some sensitive information whilst ensuring it wasn't viewable by other people who use the user's computer. It was that realised a good way to go about doing this would be to implement a website which would seem generic and not suspicious in any way but would display some sensitive information only once a trigger was activated therefore only someone who knew the trigger would be able to access the second page.

An issue that arises with the solution of creating a website is that unlike with a client side solution protecting the user's network traffic and ensuring it isn't viewable becomes impossible. Therefore this could not be included as a requirement for my solution which would only be preventing evidence from being left on the computer but anyone who could monitor network traffic would be able to see the sensitive information page had been visited.

After looking at some current solutions I got a more accurate idea of what I would have to implement to successfully solve the problem. One thing that became obvious whilst looking at the privacy browsing modes in browsers was that they were all client side and can be disabled by an admin so aren't appropriate in some situations; for example at work where an employer can very simply disable the feature, therefore not allowing the user access to any information privately. The other problem with these privacy browsing modes I intended to avoid when coming up with my solution was that there is no history on anything the user visits whilst using them. Leaving no history is good for any user who wants to browse privately but also will leave a big gap in the user's history which may seem suspicious.

While Epic Privacy Browser is effective it has the same problem as the other client side private browsing methods in that it can be disabled as it requires the user to install the browser on their computer which in a work environment might not be possible and would look extremely suspicious if it was possible to do. Leaving no history was also similarly to the privacy browsing modes an issue with this browser.

As HTTP headers were the method I could use to control what is and what isn't cached by the browser this meant that using other pages as the secret page and having the first page connect and display the sensitive information from a URL would be difficult as I couldn't alter the HTTP header of an external page. This meant while private browsing modes could allow the user to view external information on an issue such as allowing them to directly visit the NHS website to view information on any medical concern my solution would likely have to display it's own sensitive information page hosted by me.

To improve upon these client side private browsing methods my solution would have to save the first page into history but not into cache and ensure once the trigger was activated the second page wouldn't appear in history or cache and so would only leave gaps in the user's history from the secret information page. Having the solution work from a website I could host also meant that unlike private browsing unless an admin on the computer knew the site was suspicious it wouldn't be blocked. This left me with the an initial idea for server side solution of having a website with a hidden trigger which once activated would display some hidden page (which I would fill with information on sensitive subjects) which wouldn't appear in the user's history or browser cache.

The requirements for the solution after analysing the problem and previous solutions were to:

- Create a generic non suspicious page
- Embed a trigger into this page
- Display sensitive information once the trigger is activated
- Give no way of a user who doesn't know there is a trigger to find it
- Give no way of a user without the trigger to view the secret page
- Allow the secret page to swapped out for pages with different information
- Have the first page appear in history but not cache
- Ensure the second page does not appear in browser history or cache

4. Design

The design phase for my project was not particularly long as the solution itself did not need much design, it only required some thought on how to approach the implementation. The way the solution I intended to implement was laid out it was easiest to just implement the web pages and then the server. This meant the design phase mainly consisted of initially just coming up with ideas for displaying the secret page and how to implement the trigger then these ideas could be tested on the implemented website during the implementation phase.

The design of exactly how the solution would work was essentially set out during the analysis and specification phase. The way the solution would work was to create a web page which would be hosted on a server I would create then inside the web page there would be a trigger that could be activated to show a secret page. This meant the solution itself would essentially be split into three parts.

One part of the implementation would be the web pages, which would be the easier part. The web pages would just require me to implement a website with HTML and likely javascript, having done this before it would be fine. The second part would be implementing the server to host the website. This would be done in Java with Java Socket programming, the server would also likely host the trigger to send the secret page. The final part of the solution was the trigger itself, I had to spend some time thinking about how I would implement the trigger.

Some of the initial ideas I had during this early design phase would be used in my final solution. The primary idea I had which would be used throughout the rest of the project was to have the secret page within the first page displayed in an iFrame once the trigger was activated.

Image 1: A screenshot of the simple iFrame that was tested

The above image was one of the first ideas I tried to quickly implement using Javascript. I made a HTML page with a single button that when pressed would activate a Javascript function that would append an iFrame displaying the NHS home page into one of the divs in the html page.

I began to create the first web page during the design phase as it was easiest to code it in HTML and see what it would look like. My initial ideas for the web page were quite simple and were just to create a basic web page with some text and images but would also have to include a navigation bar and search field. With the navigation bar and search field being my initial ideas for the trigger.

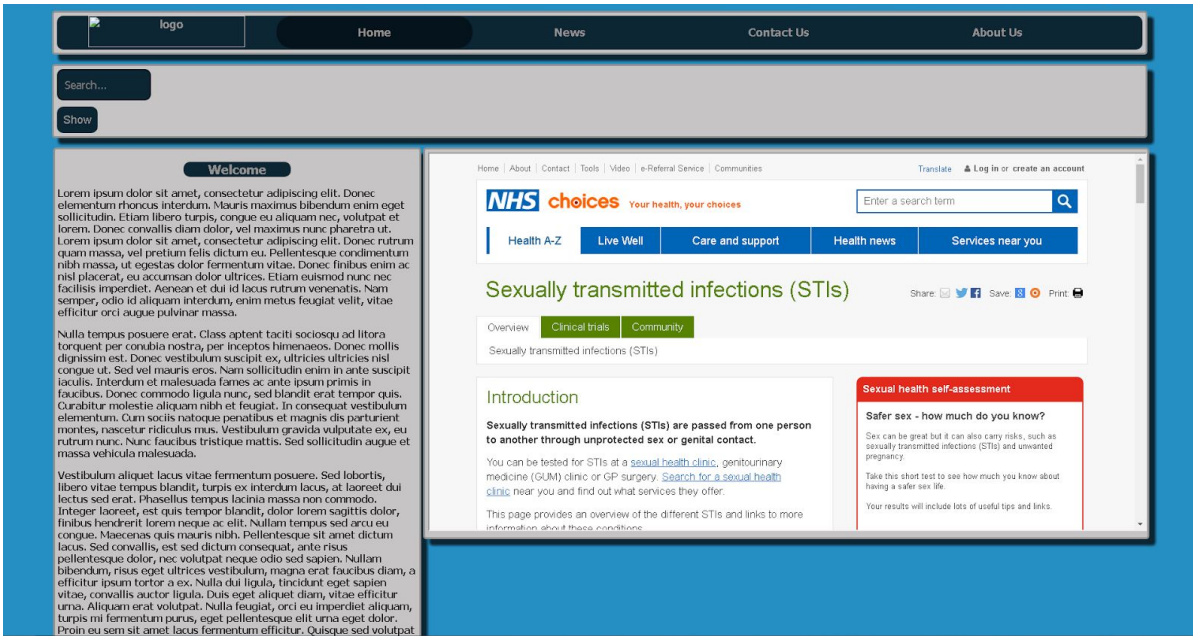

Image 2: A screenshot of the web page design implemented in HTML with CSS

The above image shows what the initial web page I implemented looked like. The website would stay looking essentially the same for the rest of the project, once I began to implement the final trigger on the server I added additional pages which looked similar but with the addition of dummy text and images. Other ideas I had early on were to do with the trigger my initial ideas were to have either a search term where the user would search for a certain term to activate the trigger or have them click on the navigation buttons in a certain order. Having discussed with Erik my supervisor he had also suggested clicking on certain parts of some image in the page as a trigger. The trigger itself would have to be done inside the server most likely as there is no way to hide the Javascript from anyone who connects to the server. If the trigger was done in Javascript then it would be quite simple for anyone to just inspect element and tell exactly how it works.

The server didn't have much of a design phase but I spent some time thinking about how it would be implemented. From the very beginning of the project it was decided the server would be implemented in Java as it was the main language I had learned and the only language I had ever implemented a server in previously. When I previously implemented a server I had done it using multithreaded socket programming so it was likely this would be the way I would be implementing the server once more. Based mostly on the ideas I came up with during the design phase my implementation would involve the web page I created from the above image which I would then have to put a trigger within, this would then be hosted on a server created in Java using multithreaded socket programming. The trigger itself I decided would either be a search term or button pattern as these seemed much simpler than any other trigger I may want to implement.

5. Implementation

5.1 - Early implementation

After the design phase was over implementation could begin. The web page that was created during the design phase was my early focus along with attempting to implement triggers in Javascript. Producing triggers in Javascript to test how the trigger would work was easy enough even though Javascript could not be used in the solution as it couldn't be hidden from the user. I created a button pattern in the navigation menu to work as a trigger where the user would have to click the home button twice, the news button three times, and then the about button once to activate the trigger. I then created a search bar trigger that required a user to input the term "secret" into the search bar to activate the trigger ignoring any other input. Although as previously stated these couldn't be used as the final trigger they were kept in the web page that was created but obfuscated Javascript through an online Javascript obfuscator. This meant if someone connected to the web page and wanted to view the Javascript they would have to deobfuscate the Javascript which is simple enough but would require them to take an extra step before knowing what the Javascript did and how the trigger worked.

The next step that was took after creating these basic triggers was to have them produce some sensitive information. The early attempts at implementing the secret page revolved around the simple attempt made of embedding an external page within another during the design phase. I added to the early Javascript some code to append an iFrame to the web page once either trigger was activated. Once this was implemented I attempted to see whether it was possible to make it so the external page didn't get saved to cache. The page didn't get saved into history which was good but would save everything into cache on all browsers. Looking at several discussions online the only suggestions were to append a random variable onto the URL of the external page which would cause it to not load from cache each time however the page was still being stored in cache.

After roughly a week of trying different ways to display an external page I was left with two options. Firstly the external page could be downloaded on the server and saved then displayed via an iFrame by the server, Alternatively my own page could be created and hosted on the server. Both options allowed for alterations to the HTTP header so that the secret page wouldn't be cached. As these two options were similar to each other (both just involving me hosting the secret page myself) I decided to go with the simpler method of just creating a page myself with information in it rather than downloading an external page. This did however mean that to change the information meant changing the page as a whole rather than just changing a url.

5.2 - Server

The next stage from this was to create the server this would be done in Java as it is the main language I have learnt and covers all aspects needed to create a server, particularly Java Socket programming would be helpful to use. In order to allow multiple users to access the server easily and quickly the server would have to be multithreaded, I had done a similar task from an assignment in the Networks module in the third year at University of Birmingham. The code for the server was based code from a website called Jenkov which had a tutorial showing how to implement a multithreaded server in Java.

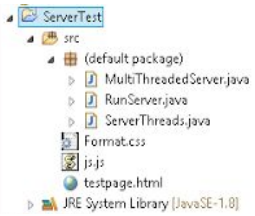

Image 3: A screenshot of the server files in eclipse

The server before the addition of the trigger worked through three java files. The first was the MultiThreadedServer file which takes a port as a parameter when created, this port would be 80 for the first server which is the port used most often for HTTP. Once created the MultiThreadedServer file creates a Java ServerSocket then once a connection was made would accept it as a ClientSocket and create a new ServerThreads object and pass it the ClientSocket to deal with the connection.

The ServerThreads file then given the ClientSocket gets the input stream from it (and the output stream to be used later) then using a BufferedReader reads the request in then checks for the requested file and if it exists returns a HTTP header:

```
"HTTP/1.1 200 OK: "+"\r\n"+
"Content-Type: "+fileType+"\r\n"+
"Date: "+ date+"\r\n"+
"Content length: "+fileInput.available()+
"\r\n";
```

Image 4: The code for the 200 HTTP header screenshotted from eclipse

It writes the above as the status and the requested file along with it through the output stream of the ClientSocket. If the file doesn't exist it returns the status HTTP/1.1 404 Not Found: [filename] Content-Type: text/html, along with a basic html page stating 404 file not found. Once it has finished doing everything above it closes everything and prints the filename and the status it responded with into the console.

```
status = "HTTP/1.1 404 Not Found: "+fileName+'\n'+"Content-Type: text/html"+"\r\n";
body = "<html><body><h1> 404 file not Found</h1><p>The requested resource doesn't exist.</p></body></html>";
```

Image 5: The code for the 404 HTTP header screenshotted from eclipse

```
public class RunServer
{
    public static void main(String[] args){
        MultiThreadedServer server = new MultiThreadedServer(8080); //create server
        new Thread(server).start();
        try{
            Thread.sleep(60000); //1 minute
        } catch(InterruptedException e){
            e.printStackTrace();
            server.stop(); //stop server
        }
        server.stop(); //stop server
    }
}
```

Image 6: A screenshot of the code for the RunServer.java file

The third file for the server is used to run the server itself this file was rather simple. The RunServer file creates the MultiThreadedServer with a desired port to be used for connections places it within a thread so the server can be run for a fixed amount of time then stops then server once it the time is up.

Once the server was running I checked whether it was connecting fine, once I knew it was I moved on to altering the HTTP header. Telling a browser to not cache a file is different across most browsers. In order to find the right header I searched online and found a stackoverflow discussion explaining the different headers needed. I then added these to the server:

```
"HTTP/1.1 200 OK: "+"\r\n"+
"Content-Type: "+fileType+"\r\n"+
"Date: "+ date+"\r\n"+
"Content length: "+fileInput.available()+"\r\n"+
"Cache-Control: no-cache, no-store, must-revalidate"+"\r\n"+
"Pragma: no-cache"+"\r\n"+
"Expires: 0"+
"\r\n";
```

Image 7: The code for the 200 HTTP header with the no cache headers screenshotted from eclipse

5.3 - Web pages

The next part of the implementation was creating the secret pages to be displayed. I created a copy of the server which used a different port(443). Since I couldn't display external pages I at least wanted the secret page to hosted on a separate server so that the secret page would have a different address needed to access it. As I don't have access to a second machine I couldn't achieve this so the secret page would have to be hosted on the same computer but a different port. Once that server was working I created the secret pages to be displayed with some basic information on STIs from the NHS.

I was then ready to implement the trigger used to display the hidden page. As the Javascript trigger wasn't appropriate it soon became apparent that having the trigger within the server would be the easiest way to implement a trigger that could be completely hidden. Because of this the idea of having the search term as the trigger was scrapped as the navigation could be implemented easier with less chance of anyone knowing there was any suspicious activity going on with the website. With the search term the input from the user would have to be sent to the server some how which may seem suspicious or reveal to others how to activate the trigger whereas with the navigation button the trigger could simply be within the server and would only require the loading of pages.

As the new trigger would be loading different pages the next thing implemented before the trigger was the pages it would load these were essentially all the same style as the one from the design phase but I added some dummy text and images into them. These were then placed in the same folder as the server code to then be sent to users when requested.

5.4 - Trigger

The final part of the implementation was to alter to server so that it included a trigger. The trigger would be similar to the navigation button pattern that was implemented in Javascript but this time would be based on the pages the user visited/loaded. Within the server there therefore would have to be some way to track the users visited page.

The way that was decided to track the user's visited pages was to save the user's IP address in a text document away from the hosted web pages on the server. Based on what page the user visited a number would be appended onto their IP in the text document. So visiting the home page would append plus one onto the IP, the news page would append plus 10, the contact page would reset the number to zero, and the about page would append plus one hundred. When a user connects to the server for the first time their IP will be added to the text document with a number based on their visited page. Everytime the user connects the number next to their IP is checked, if the number equals two hundred and thirty three they will be given a different HTML file which is the same as the home page without links to the other pages and a Javascript file different from the blank file normal requested when accessing the site. This alternative Javascript file is the same as the Javascript trigger from earlier where the user has to click on the home button twice, the news button three times, the contact button once and then the about button twice; this works as a second trigger. The alternative home page removes the links in the buttons in place

for functions used by the new Javascript file. Once they do this the secret page will be appended onto the home page in an iFrame along with a button on top in order for the user to return to the home page. Having the trigger require the user to load pages multiple time in any order means that while the pages are saved into history there is only a single entry saved for each page meaning anyone viewing the history and visiting the pages won't activate the trigger. After these triggers were roughly tested the code was cleaned and commented. The solution was then ready to be tested.

6. Testing

6.1 - Blackbox-testing

The black-box testing was all carried out using Chrome, Firefox, Internet Explorer, and Opera on my laptop connecting to another laptop running the server code.

Test #	Test Description	Expected Outcome	Actual Outcome
1	All pages can be loaded successfully.	All pages load fully without taking too long.	Pass- All four of the pages loaded correctly with images taking slightly longer to load on the page.
2	All links work on each page	When any link is clicked it loads the desired page fine	Pass- All pages were loaded when a link was clicked
3	The first trigger works	When the user visitings the home page three times, news page three times and about page twice the next link they click will load the alternate home page and Javascript	Pass- The obfuscated Javascript and the alternate Home page are loaded after the trigger is activated
4	Visiting the contact page resets the trigger progress	When the user visits the contact page the number corresponding to their trigger progress is reset to 0	Pass- The user's trigger number gets successfully reset to 0
5	The second trigger works	When the user has activated the first trigger and has the alternate home page and javascript they can activate the second trigger by clicking Home twice, News three times, Contact once, and About Us three times	Pass- The trigger is activated However with this one thing I noticed was that if a user clicked a button by mistake they would have to reload the page and activate the first trigger again, to prevent this I have added some Javascript that resets their trigger progress once the logo is clicked this was then tested next

6	Clicking the NotSuspicious logo resets the user's second trigger progress	Once the logo is clicked the variables keeping count of the button presses are reset	Pass- The second trigger gets reset successfully
7	The first trigger works after being reset	After the contact page is visited visiting the trigger pages the correct number of times activates the trigger	Pass- The trigger still works after being reset and still sends the alternate home page and Javascript
8	The second trigger works after being reset	After resetting the variables that keep count of the visited pages the trigger can then be activated	Pass- The trigger still activated fine after being reset
9	All the secret pages load	All the secret pages load within the iFrame once both triggers have been activated	Pass- All secret pages load when requested with all the links working fine also
10	The back button from the secret page works	When the Go Back button is pressed the user is returned to the home page as if the trigger was never activated	Fail- The button followed an old link to the wrong address, this is very simple to fix Pass- The link was changed to the correct one retested and worked as expected
11	The web pages can be accessed by people other than myself	This test was added when doing the questionnaire although I had checked the website on other computers I decided when I sent the Google form for people to fill out I would also check with them that they can access the website	Pass- 19 people filled out the questionnaire, all of them successfully accessed the website from their PCs using their prefered browser
12	Restoring a browser session doesn't load the secret pages back	If a user is on the secret page and the browser is unexpectedly closed and a user's session is restored the secret page isn't displayed	Pass- The correct home page is loaded again in place of the alternate home page and the Javascript isn't loaded as if the trigger was never activated

6.2 - History and cache

After roughly testing that everything worked as it should through black-box testing the next stages of testing could begin. Based on the aim of the project the tests would have to make sure that the website

was working as expected with history and cache of the most popular browsers. Therefore tests were done to check the state of the history and cache once a user activated the trigger and viewed the secret page. To check that the web page was working as expected with history and cache the server was ran on another laptop from my home then from my main laptop I connected to the server through Google Chrome, Mozilla Firefox, Microsoft Internet Explorer 11, and Opera. Once connected the trigger was activated and all secret pages were loaded, the history and cache of all the browsers were then checked.
Something to consider for these tests are that most browsers have some memory cache which is deleted once the browser window is closed and also have a disk cache. For the solution to work as expected the disk cache should be completely avoided, also if the user wants to keep their computer clear of any evidence of the secret page they will need to regularly close their browser.

For all these test the history and cache were cleared completely before accessing the server. All screenshots were then taken immediately after the triggers were activated and all the secret pages were visited.

6.21 - Chrome

The history of all the visited pages on Chrome only consists of the first pages hosted through port 80 on the server not the any of the secret pages. This works therefore as intended and means anyone looking at the user's history will only be shown these pages with no sensitive information. Within Chrome's cache there is no evidence of either the first set of pages or the second, this means that if a user connected via Chrome they can be safe in the knowledge that no evidence of the information they viewed is left on their computer.

Image 8: Screenshot of chrome's cache after the secret pages were visited

6.22 - Firefox

Firefox's history was the same as Chrome's in that it shows each of the first set of pages that were visited but none of the secret pages. Firefox's cache is different however as it contains two entries which start as "predictor-origin:" this includes "predictor-origin:http://79.78.234.126:443/" which is the address of the secret pages. Without the name of html documents with the sensitive information someone couldn't just access the secret pages but this still isn't ideal and means Firefox users accessing the site wouldn't be as secure. One fix for this would be to host all the web pages on one server therefore only saving the same address that is in history to cache.

Image 9: Screenshot of Firefox's cache after the secret pages were visited

6.23 - Internet Explorer

While Firefox's cache issue wasn't ideal Internet Explorer had a much more serious issue. After testing the trigger and secret page on Internet Explorer the history was checked and it showed both the regular pages and the secret ones. I believe the reason for it saving the secret pages to history is based on the way that the iFrame is appended to the html page. The web pages weren't saved in cache however similar to Chrome. As it stands if a user was to use Internet Explorer 11 they would have to manually delete the history entries of the secret page which in a way defeats the object of the solution.

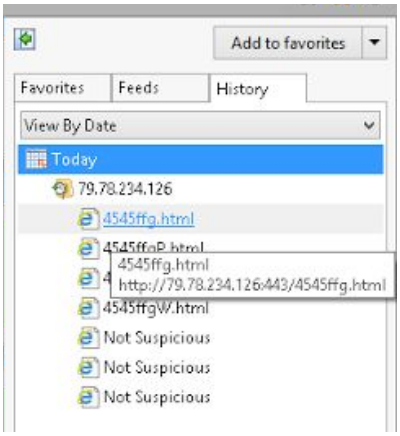

Image 10: Screenshot of IE's history after the secret pages were visited

6.24 - Opera

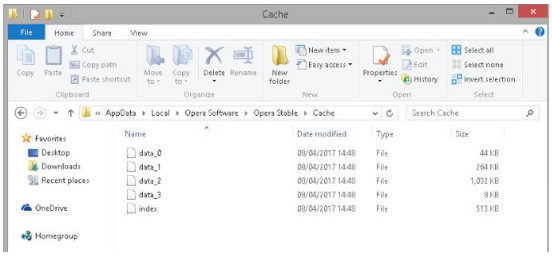

Image 11: Screenshot of Opera's cache after the secret pages were visited

Opera's worked essentially the same as Chrome with no pages ending up in either history or cache that weren't meant to be there. Opera and Chrome worked exactly as was specified in the specification phase. Firefox worked almost exactly as was specified however the two cache entries generated by visting the secret page weren't great but users could still connect to the server knowing they would leave almost no evidence. For the solution to work for anyone using Internet Explorer the solution would have to be changed.

Another idea that was tested after implementation was using a free domain site called Freenom to have a domain forwarded to the IP of the server. In practice a domain would have to be bought for the website so testing the server working with a domain was relevant. With all the browsers other than Internet Explorer when accessing the server through the domain “notsuspicious.ga” instead of having all the different pages saved to history there would only be one entry in history.

6.3 - User testing

The next important part of the solution is people's awareness of the trigger and secret page. Because of this the next part of the testing was to do with the aims from the specification, giving no way for a user who doesn’t know there is a trigger to find it or for a user without the trigger to view the secret page. To test this I created an extremely simple one question questionnaire on Google Forms and gave it to family and friends.

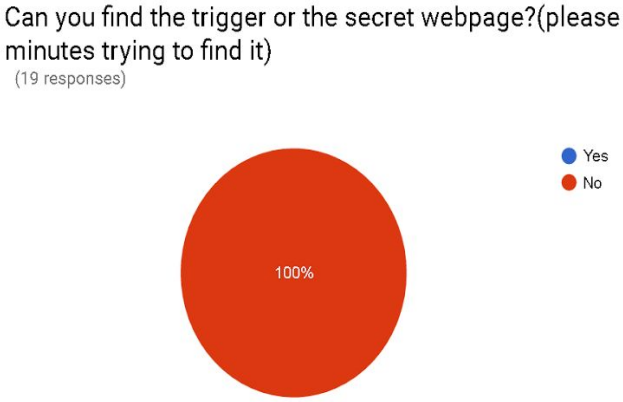

Image 12: Screenshot of the Google form responses

19 People filled in the questionnaire, all of them answering no to whether they could find the trigger or secret page. By accessing the site alone I cannot think of anyway for someone to find out how to activate the trigger. Only by viewing code for the server would someone be able to know how the trigger works and figure out how to access the secret page. With this test a large portion of those filling out the questionnaire were computer science students which served as a good test to check whether anyone adept with computers could access the secret page. Although some strange requests to the first server no one managed to make any requests to the second server holding the secret pages.

Something worth noting also is that someone could activate the trigger by accident even though the chances are quite slim if someone did it would not be difficult for them to figure out the second trigger by translating the Javascript so it was readable. Although this is possible it isn’t something that is a concern to the solution as it would be very unlikely to occur.

7. Project management

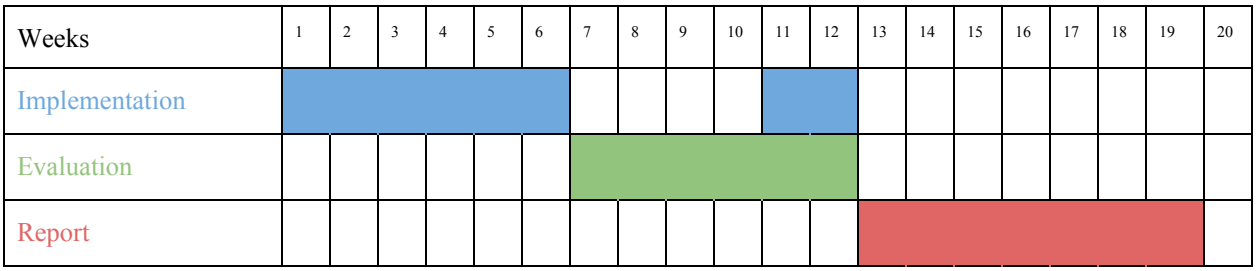

Figure 1: The original time plan from the project proposal

During the project proposal a rough time table plan was created. When I included this in the project proposal it was stated that this may later change, what wasn't realised however was just how much the time spent of these parts would change. I spent roughly double the time on implementation than I expected. During the project proposal the time that would be spent stuck wasn't took into account so anytime that I was stuck was time taken away from other part of the project, This then led to some later issues.

Throughout the beginning of the project time management wasn't an issue or so I believed. Nearing the end of the project I realised that time should have been managed better. From the start of the project I didn't set myself any deadlines based on exact dates but rather just thought about how long I believed parts of the project would take. Setting exact dates for example when to finish implementation or when begin the report would have been beneficial for the project.

Ideas that were suggested by Erik such as implementing the solution via a proxy instead of a website couldn't be done. This would involve a trigger working the same as it was currently after implementation but in place of the server would have a Java proxy server running which would take a list of URLs once the proxy server saw the user visited all the URLs it would send the user to a secret page. This would have been an interesting alternative to solve the problem but having not worked with proxies in Java before and with only around 2 weeks left from when this was suggested it couldn't be implemented. There were a few different situations like this near the end of the project where ideas were thought of that would benefit the solution such as looking into why Firefox was saving the URLs of the web pages but many couldn't be done through time restraints.

The main way to avoid these time restraints in the future would be to plan out the time of each stage better. Another focus in the future would be to spend more time on the design phase near the start so that different parts of the project could be planned out better. This showed near the start when ideas were tested that later proved not to useable such as external pages being used or the search bar. If the design had been slightly longer and covered more on how everything was going to implemented than time could have been saved later on.

8. Results and Evaluation

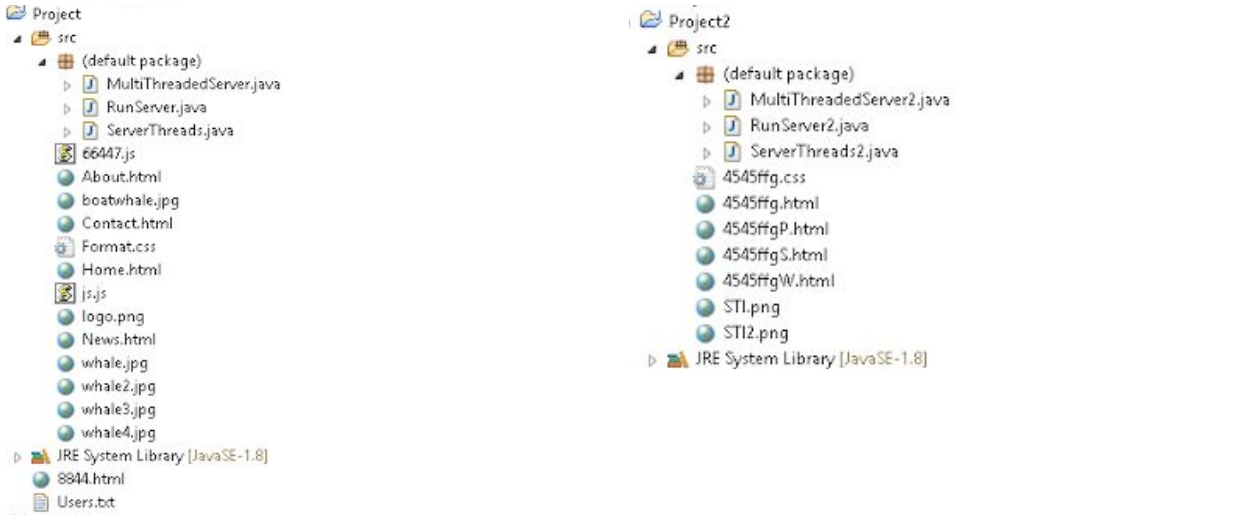

Image 13 & 14: (left)The files for the first server in eclipse (right) The files for the second server in eclipse

The final results of this project were that a server with a trigger in it was produced along with web pages that were hosted on the server, also a second server with it's own secret web pages. The first server contains four pages that get displayed to any user and an alternate home page displayed once the first trigger was activated. Also on the server is the image files the pages use, the CSS file which all the web pages use, and the Javascript used by the alternate home page. The user file also is in the same folder as the first server and is reset every time the server is ran. Below are the four web pages anyone can access without requiring knowledge of the trigger.

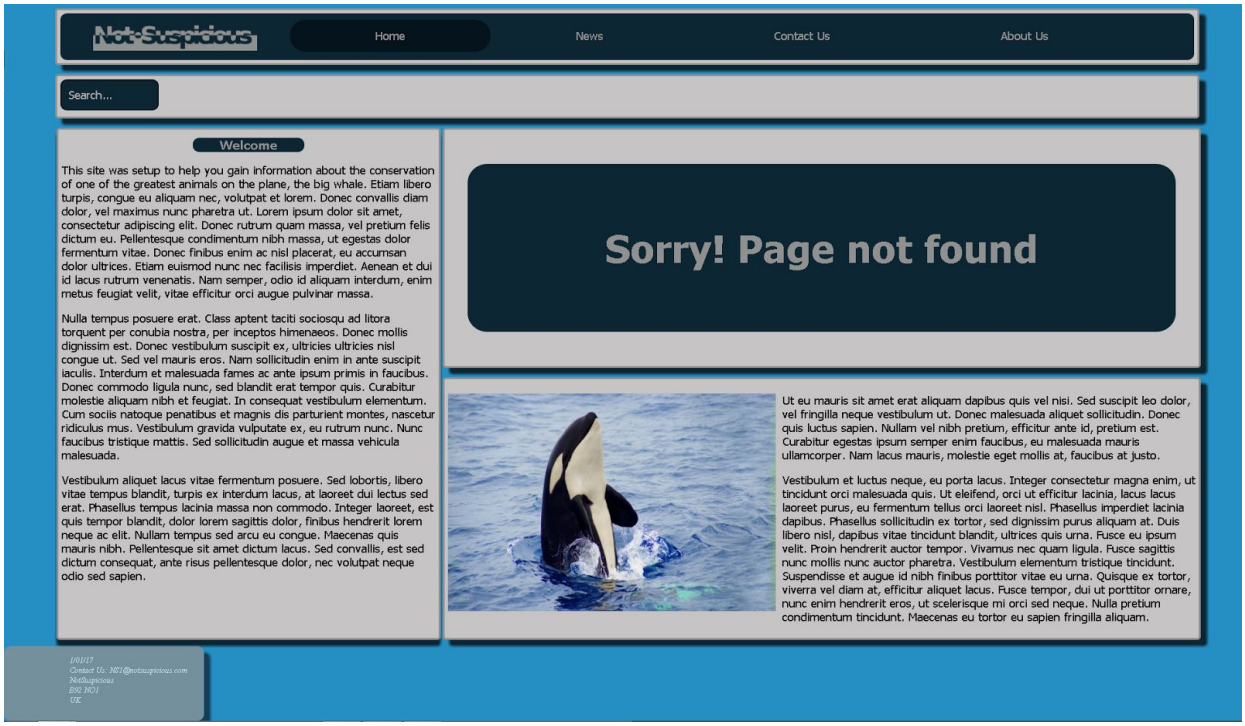

Image 15: A screenshot of the final implemented home page

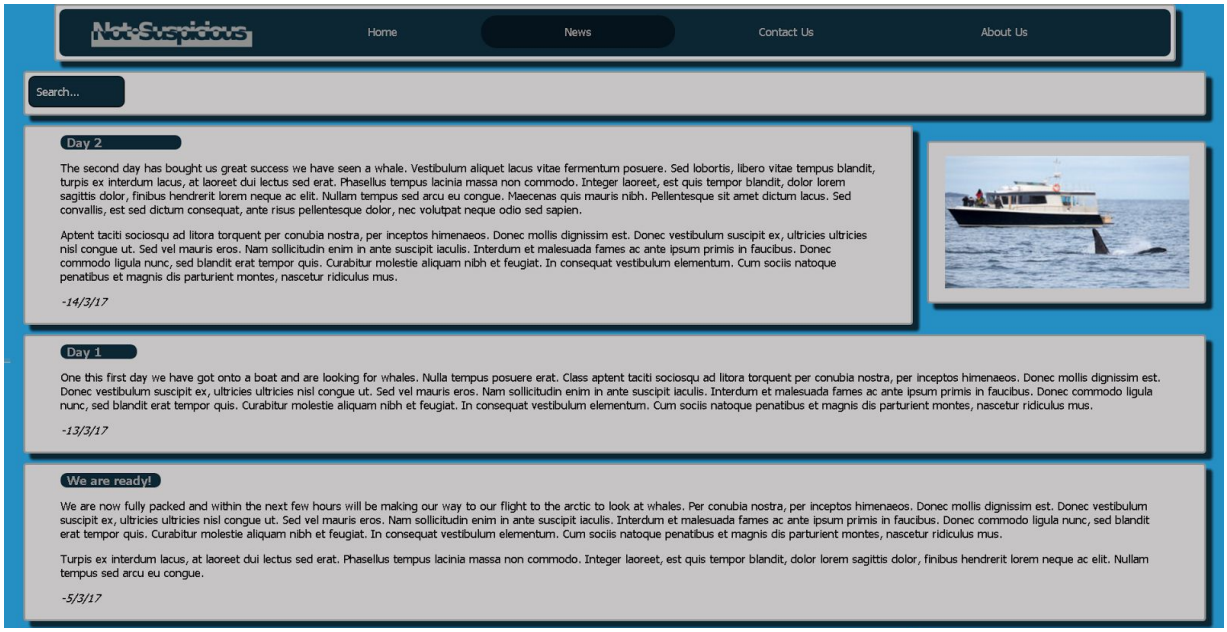

Image 16: A screenshot of the final implemented News page

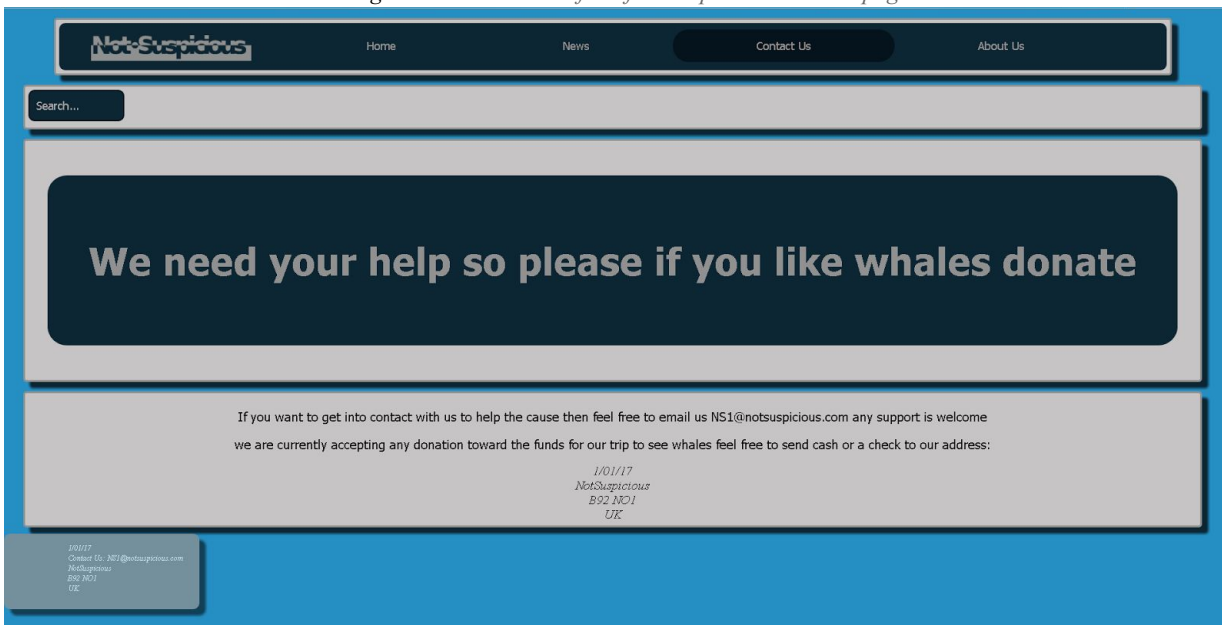

Image 17: A screenshot of the final implemented Contact page

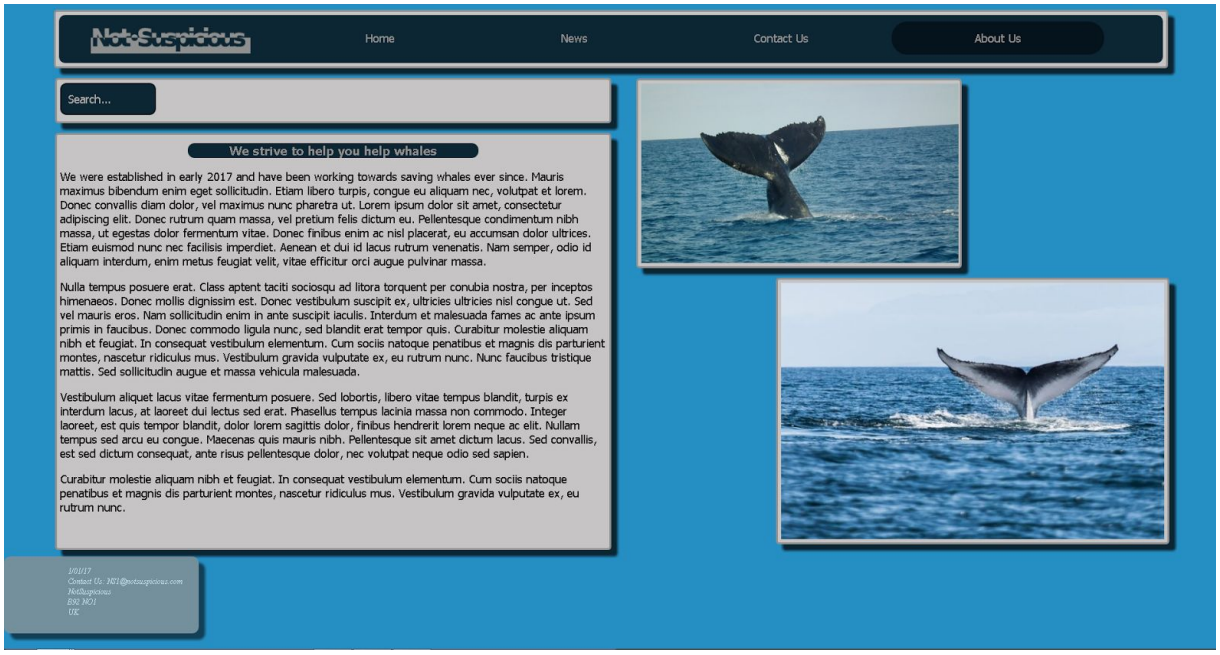

Image 18: A screenshot of the final implemented About Us page

The second server has the secret pages stored on it. The first page explains that the web page was setup to allow the user to browse sensitive topic privately and that anyone with access to their computer won't be able to find out they have accessed the page. The other pages give some facts on STIs, how to prevent getting STIs, and where to go for help. The information on these pages can be simply changed to information on any subject.

In practice a good idea would be to have multiple pages hosted on the first server which all were similarly generic and non suspicious which each have a trigger leading to pages on different subjects on a different second server. Alternatively having different button combinations lead to different pages could also be implemented.

Image 19: A screenshot of the first page of the secret pages with sensitive information

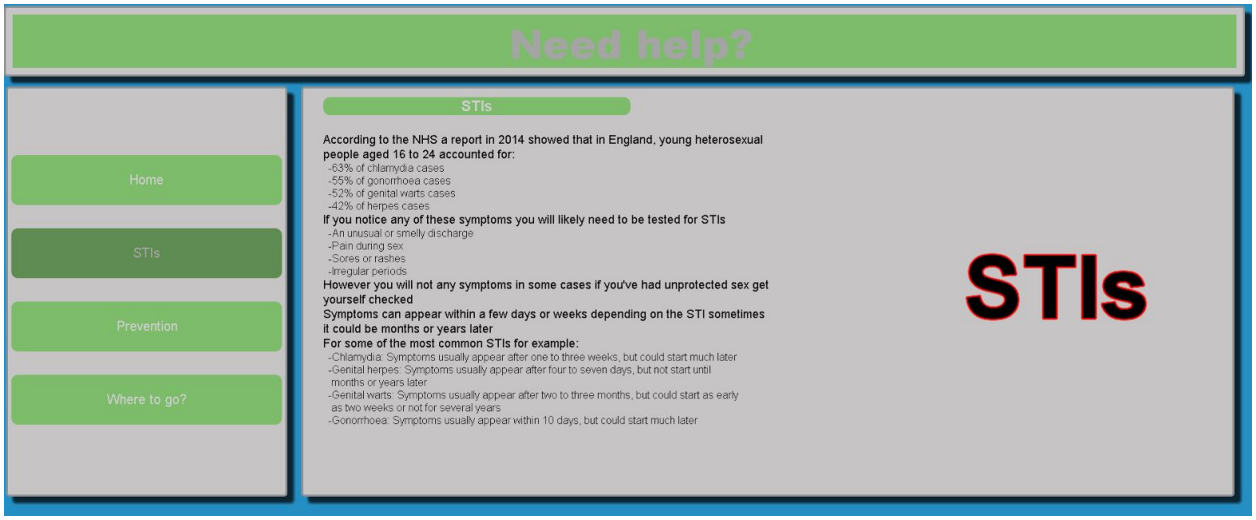

Image 20: A screenshot of the second page of the secret pages with sensitive information

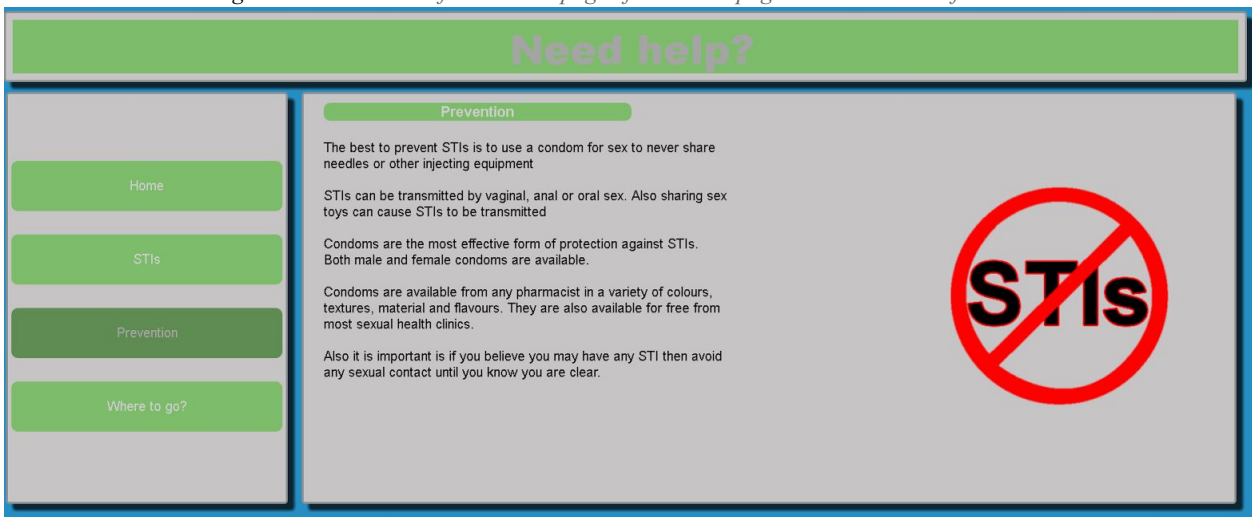

Image 21: A screenshot of the third page of the secret pages with sensitive information

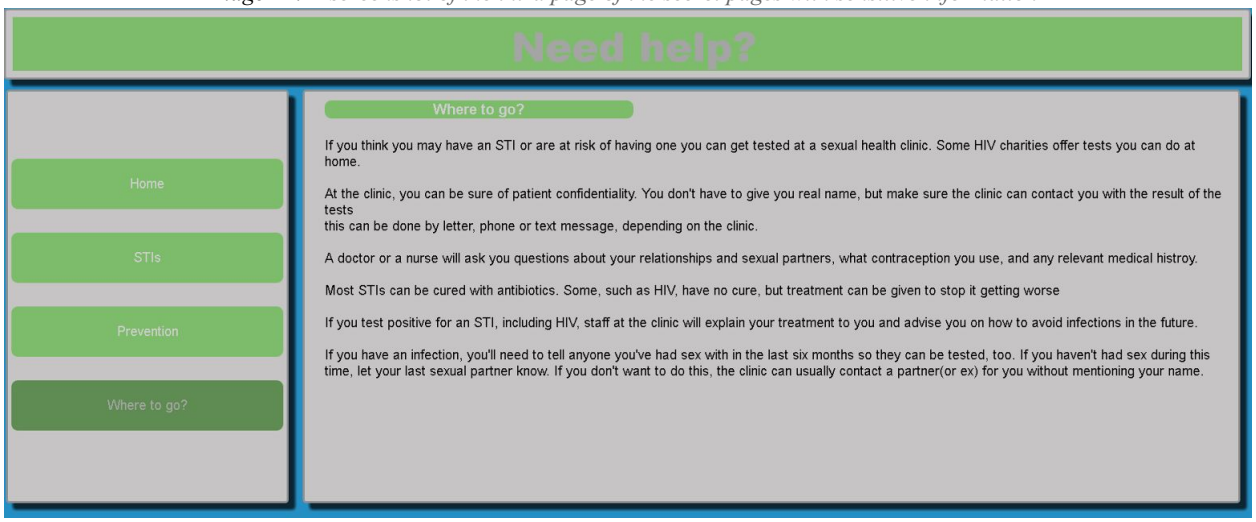

Image 22: A screenshot of the fourth page of the secret pages with sensitive information

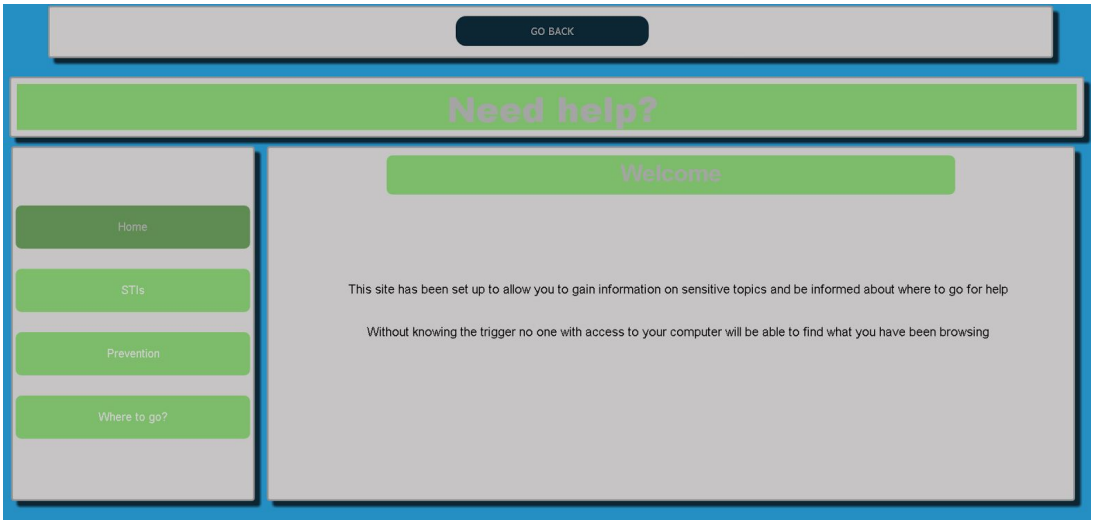

Image 23: The secret page loaded into the home page on Google Chrome

The final result of the project was as I expected. The server works fine loading all the web pages and the trigger activates then shows the secret page as expected. In terms of the specification most were met however some were slightly different as expected in the result. The first difference between the specification and the result is ensuring that the secret page isn't in cache or history.
Another aspect that is needed for consideration is to do with how the server tracks the user. Having evidence of the user visiting the site adds to the risks toward the user's privacy. Accessing this document would be difficult provided the server was hosted properly but still needs to be took into consideration. The document that keeps track of the users visited pages would have to be cleared daily which is simple enough and just involves the server being restarted which should only take up to 10 seconds.
The issue with Firefox saving the address of the secret page is an issue that preferably should be avoided however having looked online about the predictor-origin entries in cache it seems that these are unavoidable at the moment for anyone using Firefox. Although this is an issue the solution still works well enough in my opinion.

The biggest issue with cache and history is when the site is connected to via Internet Explorer. When the triggers are activated and the secret page is displayed it is saved to history on Internet Explorer. On all of the other browsers that were tested the secret page never got saved to history also when previously tested early on the page wasn't saved to history. Having looked at different ways to append the secret page onto the page all of them were also saved to Internet Explorer's history. With the time left for the project almost completely over this issue will have to stay in the solution. The only way that this issue could be avoided although not in any way ideal would be to in practice tell the user to delete the secret page entries from their history leaving no evidence in the cache and then in history when the entries are deleted.
The final result of the project solves to the problem I set out to solve up to a point. In practice the best course of action would be to host the secret page on the same server to avoid Firefox saving the address of the secret page. If this was done and the solution was then used as it currently is users would be able to view the sensitive information from the secret pages without the secret pages going into cache as was the aim of the project however in Internet Explorer they would have to remove the secret page from history. In comparison to the previously mentioned methods that are currently in place to allow private browsing of sensitive information the solution provided by this project works better in certain situations. These situations are when a user cannot access the other private browsing methods as they may be disabled. If a user needs information on some subject quickly and wants to keep their privacy this solution will be of use to them.

9. Discussion

Now that the project has come to an end based on what was originally specified some aims were met whilst other were not. In general the piece of software produced by this project met my expectations despite it's faults; these faults I would have preferred not to have in the final product but for the purpose of solving the problem I set out still mean that the final product solves the problem in nearly all cases. As the solution stands the main achievement has been creating a web page hosted on a server containing a trigger that once activated would display a secret page as stated in the specification. The achievement that then came from this being that anyone accessing the website with no knowledge of there being a secret page or trigger wouldn't be able to find the trigger or view the secret page as was the aim of the project.

Although these achievements were met some aims of the project were not met as well as would have been preferred. This is true for the part of the specification covering history and cache. While it was not specified how many browsers the solution should do this for having it work for as many browsers as possible would be the aim, so having Internet Explorer not support the solution is the main failure of the project. As well as this not specified was whether or not just having the page itself in the cache would be enough or in the case of Firefox whether having the address of the secret page in cache would still be fine.

The first improvement that would be made if given more time would be to fix the issues mentioned above. As previously mentioned the Firefox issue was unavoidable within the time left after implementation, however with some time researching Firefox's source code and trying to find out what leads to the cache entries being made the issue may be fixable. With Internet Explorer one fix that could be made in the future would be to not have the secret page be displayed in an iFrame but rather send it as the response once the first trigger is activated instead of the alternative home page and Javascript trigger. This means while it would appear in history it would appear as one of the other pages and so if someone visited the page from history would just load either the original home, news, contact, or about page. The Internet Explorer fix could also be implemented in a way where the server would only serve the secret page in this way to Internet Explorer whilst staying working the same for any other browser.

Once these issues currently present were fixed improvements could be made primarily by adding some aspects to the solution that time restraints didn't allow for. The first of these improvements would be attempting to try and implement the trigger using cookies; this was a suggestion that was made by Erik near the end of the project. As cookies are used commonly in websites to track the user's session it would have been an interesting alternative to track what pages the users had visited with cookies and activate the trigger this way. Another suggestion was an alternative to having a website and instead using a proxy server to track what pages the user visited and once they visited certain pages in a certain order display a page with sensitive information. Implementing a proxy server was not possible in the time left but would have been an interesting experiment if given more time.

The improvement that I would make if more time was available would be to change the way the secret page works. If given more time instead of creating the pages myself I would spend some time attempting to allow external pages to be shown. Having external pages be shown in place of my own pages would allow for a variety of different sensitive subjects to be shown through the website. Along with this implementing the trigger as suggested by Erik by having the user have to click an image in certain specific places to activate the trigger would have been an interesting and worthwhile improvement to make to the solution.

10. Conclusion

The majority of the main aims I had at the beginning of the project have been met. Even with some of the aims not being completely met the solution that has been produced solves the problem I set out in the introduction in the majority of cases. With the result of this project anyone who wishes to view some sensitive information can connect to the provided website then with knowledge of the trigger can access the page which contains the sensitive information they desire. Any user who uses the website will be able to do so without any worries about anyone else with access to their computer knowing about what they have been viewing. As users can do this the project has achieved what it set out to achieve which was solving the problem of private browsing in a way different from the current solutions in place.

References

1. Google Chrome (2017)
   https://www.google.com/chrome/ [Accessed 31st March 2017]
2. Firefox (2017)
   https://www.mozilla.org/en-GB/firefox/new/ [Accessed 31st March 2017]
3. Opera (2017)
   http://www.opera.com/ [Accessed 31st March 2017]
4. Microsoft Edge (2017)
   https://www.microsoft.com/en-gb/windows/microsoft-edge#D04R9fF7w5lOStJg.97
   [Accessed 31st March]
5. Google Incognito (2017)
   https://support.google.com/chrome/answer/95464 [Accessed 31st March 2017].
6. Firefox Private Browsing (2017)
   https://www.mozilla.org/en-GB/firefox/private-browsing/ [Accessed 31st March 2017]
7. Opera Private Browsing(2017)
   http://help.opera.com/Mac/12.10/en/private.html [Accessed 31st March 2017]
8. Microsoft Edge InPrivate Browsing (2017)
   https://support.microsoft.com/en-gb/instantanswers/34b9a3a6-68bc-510b-2a9e-833107495ee5/browse-inprivate-in-microsoft-edge [Accessed 31st March 2017]
9. Epic Privacy Browser (2017)
   https://www.epicbrowser.com/index.html [Accessed 1st April 2017]
10. Tor Browser(2017)
    https://www.torproject.org/about/overview.html.en [Accessed 2nd April 2017]
11. How to control web page caching across all browsers (2011)
    http://stackoverflow.com/questions/49547/how-to-control-web-page-caching-across-all-browsers
    [Accessed 16th January 2017]
12. Javascript Obfuscator (2017)
    https://javascriptobfuscator.com/Javascript-Obfuscator.aspx [Accessed 6th April 2017]
13. Jenkov Tutorials Multithreaded Server in Java (2014)
    http://tutorials.jenkov.com/java-multithreaded-servers/multithreaded-server.html
    [Accessed 5th January]
14. NHS Open your eyes to STIs (2015)
    http://www.nhs.uk/Livewell/Sexandyoungpeople/Pages/STIs.aspx
    [Accessed 23rd February 2017]
15. Freenom - A Name for Everyone (2017)
    http://www.freenom.com/en/index.html?lang=en [Accessed 8th April 2017]