



UNIVERSITY *of* LIMERICK

OLLSCOIL LUIMNIGH

*SOCIAL ENGINEERING AS A
METHOD OF DEPLOYING
MALWARE ON ANDROID MOBILE
DEVICES*

*A practical study of the general population's knowledge of Social
Engineering and the security of their private information*

Author

James Dilleen

Supervisor

Dr. Chris Exton

APRIL 14, 2016

*B.Sc. Computer Games Development
Department of Computer Science and Information Systems
University of Limerick
Ireland*

Abstract

This project is a practical study of the general population's knowledge of Social Engineering and the security of their private information, specifically on their Android mobile devices.

An Android application was developed to demonstrate the effectiveness of Social Engineering in successfully breaching a random user's private information. A remote web server was developed to showcase the resulting data obtained from this breach and to educate the user as to the various methods of Social Engineering that could be used against them. A user study was then carried out where the affected users were asked a series of questions for statistical analysis.

The results of the user study carried out as part of this project are available both in this report, and online, [here](https://jdilleen.me/fyp/statistics/) (https://jdilleen.me/fyp/statistics/).

Declaration

This final year project is presented in part fulfilment of the requirements for the degree of Bachelor of Science in Computer Games Development. It is entirely my own work and has not been submitted to any other University or higher education institution, or for any other academic award in this University.

Where there has been use made of the work of other people, it has been fully acknowledged and fully referenced.

All brands, product names, or trademarks are properties of their respective owners.

James Dilleen

14th April 2016

Acknowledgements

I would like to thank my supervisor Dr. Chris Exton for agreeing to supervise this project and allowing me the freedom to develop the various aspects of this final year project to the standards I needed. His input and advice during the duration of this project has been tremendous. His assistance with obtaining ethical approval for the user study aspect of this project was very much appreciated.

I would like to thank all my friends for their overwhelming support and input throughout the project. I'd like to thank my wonderful girlfriend, Louise, for being there for me at any time and helping me through stressful times throughout the development of this project.

I would also like to thank every participant in my user study, without all of you, the development of the Android application and web server would be for nothing. The feedback and data resulting from your participation has proven to be incredibly interesting and certainly eye opening.

Lastly and most importantly, I would like to thank my family: my grandfather for his patience and understanding when I would not return home for weeks on end due to work load and my grandmother for always lending an ear when I needed to talk about whatever was causing me stress. Without the both of you, I would not be in the position I am today and I will be forever grateful.

Table of Contents

Abstract.....	i
Declaration.....	ii
Acknowledgements.....	iii
Table of Figures	vii
1. Introduction.....	1
1.1 Objectives.....	2
1.2 Motivation	3
2. Research.....	5
2.1 Social Engineering	5
2.2 The Target Breach.....	6
2.3 Dendroid	7
2.4 Environment.....	9
2.4.1 Digital Ocean vs. AWS or other options	9
2.4.2 Programming Languages & Technologies Researched	9
3. Implementation	11
3.1 List of Implementation Technologies	11
3.1.1 Virtual Private Server	11
3.1.2 Apache2	11
3.1.3 HTML	12
3.1.4 CSS	12
3.1.5 JavaScript.....	12
3.1.6 PHP	12
3.1.7 PostgreSQL.....	13
3.1.8 Java	13
3.1.9 Git	13
3.1.10 External Resources.....	13

3.2	Environment Set-up.....	15
3.2.1	Linux Virtual Machine.....	15
3.2.2	Windows Machine	16
3.2.3	Virtual Private Server	16
3.3	Website Development	17
3.3.1	Database.....	19
3.3.2	Login	20
3.3.3	Generating a device.....	22
3.3.4	Dashboard	22
3.3.5	Device	23
3.3.6	Application Image Scraping	24
3.3.7	Statistics	25
3.3.8	API.....	26
3.4	Android Application Development	27
3.4.1	AppScraper	27
3.4.2	NetworkOperations	28
3.4.3	PermissionsCheck	28
3.4.4	PhoneScraper	29
3.4.5	MainActivity	29
3.4.6	Game Menu & Breach Activity	30
4.	Evaluation	32
4.1	Testing.....	32
4.1.1	Website Testing	32
4.1.2	Android Application Testing	33
4.2	Problems Encountered.....	33
4.2.1	Android to Server communication.....	33
4.2.2	Ethical Approval	34

4.3	User Study	34
4.3.1	Statistics	35
5	Conclusion	40
	References	41

Table of Figures

Figure 1: Anatomy of the Target Breach	6
Figure 2: Dendroid Control Panel.....	8
Figure 3: Debian Virtual Machine (Left) running on Windows (Right)	15
Figure 4: Standard Web Page Layout	17
Figure 5: Login Check and Error message	18
Figure 6: Database Schema.....	19
Figure 7: Login Form.....	20
Figure 8: reCAPTCHA Request	21
Figure 9: Randomly generating a Device	22
Figure 10: Device Page.....	23
Figure 11: Google Play Logo Scraping	24
Figure 12: Statistics Page.....	25
Figure 13: AppScraper Function.....	27
Figure 14: NetworkOperations Class.....	28
Figure 15: PhoneScraper Class	29
Figure 16: Main Menu and Breach Screen	31
Figure 17: Chrome Dev Tools in action	32
Figure 18: Social Engineering Awareness	35
Figure 19: Permission Acceptance.....	36
Figure 20: Anti-Virus Installed.....	36
Figure 21: Third Party Apps	37
Figure 22: Run-time Permissions.....	37
Figure 23: Security of Data.....	38
Figure 24: Will Install Anti-Virus.....	38
Figure 25: Third-Party Source Vigilance.....	39
Figure 26: Permissions Vigilance	39

1. Introduction

This project is a practical study into the effectiveness and various uses of Social Engineering as a method of deploying malware, in this case, on Android mobile devices. The aim of this study is to educate the general public as to the various different methods of exploitation attainable through the use of Social Engineering by malicious third parties. Once the study has been completed, the participants will be educated as to what to look for, what to avoid and how to protect themselves against these kinds of attacks.

Social engineering is a non-technical method of intrusion hackers' use that relies heavily on human interaction and often involves tricking people into breaking normal security procedures. (Rouse, 2014)

The effectiveness of social engineering is dependent on the knowledge of the victim the attacker plans on exploiting. An organization can spend tens and thousands of euros on keeping their networks malware free, but there is very little they can do to protect the users from their own stupidity.

There is three main aspects to this project: the Android application, the remote Command and Control (CnC) webserver and finally, the user study.

The Android application will pose as a game, a basic menu system that fools the user into believing they have installed a simple game to play. The underlying maliciousness of the application will not be revealed until the success of a breach has been confirmed. This will be confirmed by the use of a visual cue; once the user has installed the application and accepted the permissions requested, they will press the play button which is show some basic "Loading..." dialog. The application will instead be running a function that will be uploading a list of the devices installed applications to a remote web server. Once the list of applications has been uploaded, the application will switch its view to show that a breach has occurred.

The CnC server will be a Digital Ocean "droplet", a virtual private server that will be running the Debian operating system. This server will be hosting an Apache2 web server with an authenticated Web API exposed and listening for requests from the Android application. The CnC will also serve a Web UI that can be accessed to view the data that has been uploaded by the application. This will be accomplished via

HTML, CSS, and JavaScript for the front-end UI with PHP and PostgreSQL running the back-end of the server. All traffic will be encrypted end to end via a forced SSL connection and the site will also have a valid registered domain name associated with it.

The final aspect of the project was the user study. The study has been approved by the ethics council based on some requirements that have been met. One such requirement is that the Android application cannot be installed on any device other than my development test device and it must be programmed in such a way that if it was installed on another device, it would not upload data. The user study comprised of asking random users to “test a game” that had been developed for a final year project, having them go through the entire download and installation on a test device, opening the application, accepting the permissions and attempting to play the game. It was then revealed to the participant that the application had been malicious all along. The user was then shown the Web UI where they could clearly see the application had contacted the server moments ago and the list of applications installed on the device was present and searchable. Once this had taken place the user was then asked to complete a short survey on their thoughts about the application and the events that had just unfolded.

The survey provided great insight into the mind-set of users that are susceptible to Social Engineering attacks. It also provided some insight into the general public’s knowledge of Androids application permissions system, the security of their private data and their feelings toward anti-malware solutions and their importance.

1.1 Objectives

The main objective of this FYP is the create a fully functional Android application for the purpose of deceiving a user into believing the application is a game, when it is a personal data scraping tool.

A website must be created for the displaying of this information to show the affected user that the applications intended purpose of breaching private information has been successful. This Web UI will also serve as a web form survey for the participants to answer questions on their experience, which, when completed, will also display statistics about all returned results in a graphical manner.

Along with the Web UI, an authenticated PHP API must be developed for the Android application to communicate with the remote web server and upload its data successfully.

1.2 Motivation

For my co-operative education I was lucky enough to be hired by the world leader in cyber security, FireEye. Whilst having no experience in the cyber security sector, I became enthralled by the vast amount of methods that are leveraged by small time hackers to large and sophisticated collectives, the most famous being Anonymous. During my time on co-op, I began to fall in love with all facets of cyber-security, from anti-malware to breach mitigation and incident response.

From an outside perspective, many people wouldn't even realise the need for strong cyber security, let alone the vast amount of resources that need to be protected. From network traffic, email, file storage and mobile security solutions, I got to experience it all while on co-op.

It was during a particular presentation by some colleagues that had travelled from the United States to showcase the company's new mobile threat prevention system that I had an idea for my Final Year Project. They had said that intrusion via mobile devices was on the rise, as once a device is outside the company's corporate network (while the employee is travelling or just at home) there was very little that could be done to protect it. That infected device could then be brought back inside the company's internal network and malware could propagate through the network.

FireEye employees, myself included, are targets for malicious attacks, mainly due to the fact that if a FireEye employee was breached, a hacker might be able to access the company's internal network, which would be a death sentence for any cyber-security company. At least two to three times a week I received bogus emails attempting to phish private information from me, this is when I decided to devise a Final Year Project that aims to study and therefore educate users on how to avoid these kinds of attacks.

While this project was not influenced or requested by my employer, I feel as if something like my application could be a great selling point for any company offering mobile threat prevention services. Simply walk into a room of potential customers, have

one download the application, show them what is really happening and how easy it was to gain access to their private data. My application could be expanded upon to gather SMS messages, photos, contacts and emails but for ethical approval reasons, I stuck with a mundane list of applications that the victim has installed.

With my newfound interest in cyber security, I decided to propose my own project and was lucky enough to find a supervisor that had also shown an interest in the cyber security field while they visited me on co-op.

In my opinion, I believe that the best protection from social engineering attacks comes from security through education of the end user. Educating is far simpler than investing in security solutions that can be bypassed by exploitation of an individual's human nature.

2. Research

2.1 Social Engineering

Social Engineering is so broad that it takes on many different forms, all of which are proving to be incredibly effective.

- **Phishing**

Phishing is the practice of sending emails appearing to be from reputable sources with the goal of influencing or gaining personal information.

- **Vishing**

Vishing is the practice of eliciting information or attempting to influence action via the telephone.

- **Impersonation**

This is the attack method that my research relies on, it is the practice of pretexting another person with the goal of obtaining information or access to a person, company or computer system.

Some of the statistics on the above attack methods are staggering. Phishing is incredibly popular mainly due to the amount of emails being sent worldwide each day. In 2014, there was 294 billion emails sent daily, with 90% of emails being spam or malicious in intent. Phishing itself represented 77% of all socially-based attacks with 33.7 million users reporting phishing attacks in 2014.

Impersonation proved to be some of the most effective methods of social engineering with the average age of a victim being 41.7 years old and an average monetary loss to that person, or that person's organization being \$4,187. 80% of all thefts of data involved disabling or bypassing security controls by attackers impersonating an organizations internal log on system, with 88% of all stolen assets being reported as personal data. (The Social Engineer, 2014)

The vast amount of data that has been stolen by non-technical attacks, as you can see in the previous paragraph, is staggering and that is exactly why a firmer stance on security through education should be taken by a lot of companies in order to protect both themselves and their employees.

Unlike common technical attacks, social engineering attacks cannot be prevented by security tools and software. Instead of attacking a network directly, a social engineer exploits human psychology in order to coerce the victim to inadvertently divulge sensitive information. (Snyder, 2015)

2.2 The Target Breach

Between the 27th of November 2013 and the 15th of December 2013, the US retail chain store “Target” was compromised in one of the biggest data breaches in US history. The plan of attack was very simple to attackers; they targeted a third party contractor with phishing emails and obtained their password for Targets air conditioning systems. Target failed to ever implement a segmented network, meaning that the air conditioning systems are all on the same network as every single point-of-sale credit card reader across all of their stores in the United States. The attackers installed malware that would proceed to scrape every credit or debit card that was used at any of the stores. This attack was also very meticulously planned, as it fell right on Black Friday, some of the busiest days for US retail stores.

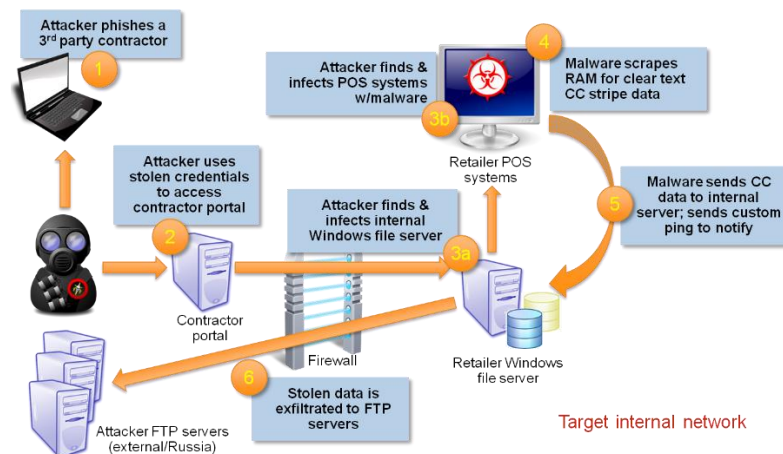


FIGURE 1: ANATOMY OF THE TARGET BREACH

To put the significance of this breach, executed with a single basic phishing email, here are some statistics:

- 40 million credit and debit card numbers were stolen between the 27th of November and the 15th of December.
- 70 million records that included the names, address, email address and phone number of Target shoppers.
- To reissue stolen 21.8 million of the cards that were affected, banks and financial institutions incurred a cost of \$200 million.
- Zero card readers were Chip-and-PIN enabled across any of the Target stores.
- The attacker(s) generated an estimated \$53.7 million on the sale of stolen cards on the black market before the cards were cancelled by banks.

(Krebs, 2014)

2.3 Dendroid

Dendroid is a particularly malicious strain of Android malware that made the headlines during 2015. Dendroid is a Remote Access Trojan (RAT), this means that any device infected with the Dendroid malware could be remotely controlled and accessed from anywhere in the world by the hacker that infected the device.

What is so special about Dendroid? Google performs anti-malware checks on any application submitted to the Google Play Store, this means that in theory, every application on the Play Store is certified malware free by Google or has been deemed safe in nature for the general public to install on their phone. However, Dendroid managed to escape Google's anti-malware checks and made its way onto the Play Store via carrier applications. The author of Dendroid promised that the code would go undetected and could be inserted into any Android application for your own personal use. Dendroid remained undetected on the Play Store until a security researcher noticed some odd behaviour on his device and decided to look into it.

Dendroid was incredibly advanced, so advanced that it could intercept and block SMS messages being received by the target device, it could download pictures from the device, it could take pictures or video with the device, it could download the devices web history and bookmarks, download account (email and social media) information, send text messages from the device, record ongoing calls from the device and even open prompts to ask for passwords from the victim.

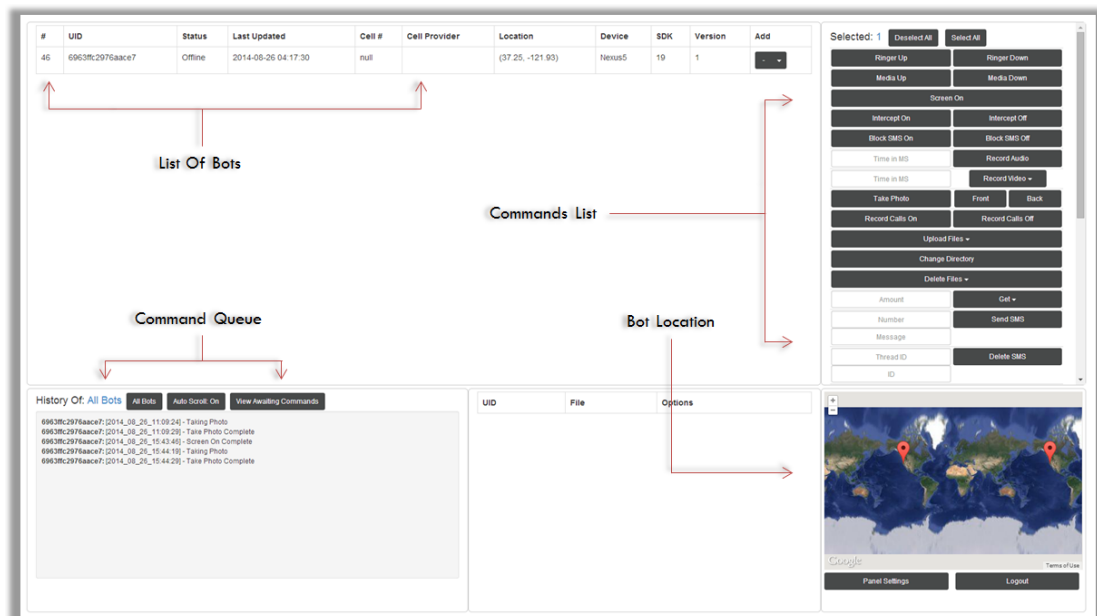


FIGURE 2: DENDROID CONTROL PANEL

The developer of an app capable of such malicious and sophisticated behaviour must be a seasoned and mature hacker, right? Wrong. Dendroid was created by Morgan Culbertson, a 20 year old University student, who has seen been arrested, charged and pleaded guilty to federal charges involving the creation and sale of Dendroid malware and it's source code. He was selling the injection kit for \$300, payable in bitcoin of course and if you were so inclined, you could purchase the source code for the entire kit for a small sum of \$65,000 in bitcoin.

The most interesting facet of this story is that Culbertson happened to be a former intern at FireEye, using his time at the world leader in cyber security to learn about bypassing their methods of detection, assuming if he can bypass the most advanced anti-malware system in the world, he could bypass Google.

Dendroid could even detect if its host application was being run in an emulated environment, often used in anti-malware systems (and indeed, used by Google's "Bouncer" automated malware detection system) and therefore not run any suspicious functions until the application was being hosted on a physical device. (Rogers, 2014)

2.4 Environment

2.4.1 Digital Ocean vs. AWS or other options

Digital Ocean is a cloud platform for software developers, it acts a hosting service that allows users to create "droplets" (servers) running the operating system of their choice. The droplet the user creates is a virtual private server and allows full control to the user to use it for whatever they want. For this project Digital Ocean has been utilised as an Apache2 web server with a PostgreSQL database on the backend. Digital Ocean also offers student developers \$50 worth of free credit to be used however the student wishes, this equates to 10 months of a low tier server for free.

Other solutions to webhosting such as Namecheap, GoDaddy and Amazon Web Services (AWS) also exist and each have their own benefits. Each option was researched and ultimately, Digital Ocean was chosen as it offered 10 months of free hosting and also offered root access to the server that you rent, meaning you have complete control over what technologies you install on the server itself.

2.4.2 Programming Languages & Technologies Researched

This project uses a vast amount of different languages to accomplish the tasks needed. For the Web UI this project uses HTML, CSS and JavaScript to build and style the front-end look and feel of the website. This project also makes use of jQuery, a cross platform JavaScript library designed to simplify the client-side scripting of HTML.

On the back end side of the web server, this project uses PHP to execute scripts that are needed for the site to attain data for pages by accessing the database. PHP also runs some unique scripts that are called via front-end jQuery AJAX calls to show/update data in real time. The project also uses a PostgreSQL database in which all user data, device data and survey data is stored in encrypted form.

For all web development, the IDE used was Sublime Text with the Emmet plugin which allows for long winded code samples to be generated programmatically by using a

certain syntax, this proved to be a huge time saver, especially for the web form page that was using for obtaining the results of the survey.

The Android application for this project was written in Java using Android Studio as the integrated development environment. The ease of use of Android studio proved to be the deciding factor in its inclusion in this project. Android Studio provides easy access to all Google Android SDK and provides images of each Android version for use in emulated environments for real time testing during the development of the application.

Aside from the chosen technologies, Python and C# were researched for use on the web side of the project, however, given a lack of experience and the vast amount of time it would take to learn these new technologies, implementing them was decided against.

3. Implementation

3.1 List of Implementation Technologies

- Virtual Private Server (VPS) running Linux (Debian) - (Digital Ocean, 2011)
- Apache2 (Apache Software Foundation, 1995)
- HTML (W3C; WHATWG, 1993)
- CSS (Håkon Wium Lie; Bert Bos; W3C, 1996)
- JavaScript (Brendan Eich; Netscape Communications Corporation; Mozilla Foundation; Ecma International, 1995)
- PHP (Zend Technology, 1995)
- PostgreSQL (PostgreSQL Global Development Group, 1996)
- Java (Oracle Corporation, 1995)
- Git (Torvalds, 2005)
- Third party extensions / resources
 - Bootstrap (Otto & Thornton, 2011)
 - Bootswatch (Park, 2014)
 - jQuery (Resig, 2006)
 - Alertify.js (Younes, 2014)
 - Highcharts (Highsoft AS, 2009)
 - DataTables (SpryMedia Ltd, 2011)
 - LetsEncrypt (LetsEncrypt, 2015)
 - GitHub (GitHub, 2008)

3.1.1 Virtual Private Server

This project uses a virtual private server (VPS) running the Debian distribution of Linux. This VPS is hosted in London and has been assigned its own unique static IP address to which there is a domain name associated. To view the homepage of the VPS, you may browse to: <https://jdilleen.me>.

3.1.2 Apache2

Apache2 is the web server platform that was chosen to host the web site associated with this project. Apache2 was chosen because of its flexibility and ability to force users to access the website with SSL encrypted connections by default. Apache2 can be used to restrict access to certain web folders or password protect some folders that you may not

want publically accessible. Apache2 also happens to be the best choice of web server configuration for the implementation of one of the other technologies used, LetsEncrypt.

3.1.3 HTML

Hyper Text Mark-up Language (HTML) has been the bread and butter of the World Wide Web since its release in 1993. Combined with CSS and JavaScript, HTML makes up barebones of every website you visit. This technology is used to develop the web pages that are shown to the victim when the user study has been completed.

3.1.4 CSS

Cascading Style Sheets (CSS) are used to style a web page, you can use them to alter the look of text, the colour of the background or the size of images. Without including CSS in a web page, the page will look incredibly basic and you will not be able to alter the look and layout of the page very much.

3.1.5 JavaScript

While HTML and CSS are merely mark-up languages and style sheets, JavaScript as a high-level, dynamic, untyped and interpreted programming language. The use of JavaScript in web pages provides a huge array of possibilities to the developer, such as form validation, error checking and real time requests without reloading the entire page. The use of JavaScript in this project is limited due to the fact jQuery was included, resulting in much simpler methods of completing functions JavaScript would normally be used for.

3.1.6 PHP

PHP is a server side scripting language that can be used in conjunction with HTML, PHP is used in conjunction with HTML throughout the web aspect of this project where database connection is required a HTML page to display results. The benefit of PHP is that the code is entirely server side and therefore not available to the client, so passwords to databases etc. can be stored in PHP scripts without a user ever obtaining access to them. PHP can be used in conjunction with jQuery to provide JSON encoded results to a jQuery script that has triggered an AJAX request. In this project, for example, all graphs and statistics on the Statistics page and AJAX controlled requests to PHP scripts which then retrieve data from the database and return it to the client.

3.1.7 PostgreSQL

PostgreSQL is a database management system, it stores data securely and is a great alternative to MySQL as it released under a different licence, making it completely free for all use, even commercial use. PostgreSQL offers great supporting documentation and allows for databases, tables, users etc. to be set up in a matter of minutes for an experienced user.

3.1.8 Java

Java is the main programming language that is used in the development of Android applications. Java is an all-round programming language, it will make up the bare-bones part of the application, taking the devices list of apps and then it will establish a connection to the VPS' API and transmit them via HTTP requests. It will also be used to make the basic game menu that will be used as the dummy part of the application.

3.1.9 Git

Git is a widely used open source, source code management system that is used extensively in software development. Git allows for the creation of repositories that enable a full-fledged history and version tracking capabilities.

3.1.10 External Resources

3.1.10.1 Bootstrap

Bootstrap is a free and open-source collection of tools for creating websites, of which this project will be using their CSS and JavaScript elements to ease in the development of the Web UI. Bootstrap CSS provides an extensive stylesheet that can be used to create beautiful front-end application. Their JS library provides functionality for their CSS and HTML dropdown systems as well as modal pop-ups etc.

3.1.10.2 Bootswatch

Bootswatch is a free resource that enables a developer to change the default look of Bootstrap if the original is not to their liking. The Bootswatch theme used in this project to replace the default Bootstrap CSS is called "United".

3.1.10.3 jQuery

jQuery is a JavaScript library that is designed to simplify the client side scripting of HTML. It makes changing the look and feel of your web page on the fly very easy.

jQuery can also be used to trigger AJAX requests in a much simpler syntax to JavaScript. This project is using the latest version of jQuery and it is a requirement for the Bootstrap JavaScript component to run.

3.1.10.4 Alertify.js

Alertify.js is a JavaScript library that provides extra functionality to a browser's default alert, prompt and confirm dialogs with the ability to customise them as needed. It also provides support for pop up success, warning and error messages that are not intrusive and can be triggered very easily with some basic scripting. This project uses Alertify.js in the Web UI as a means of notifying the user of any errors regarding login parameters i.e. username / password fields left blank before clicking submit on the login form.

3.1.10.5 Highcharts

Highcharts is an extensive JavaScript graphing library that utilizes HTML 5 canvas tags to display interactive charts. Highcharts is free for personal use but requires a licence for commercial use. This project uses Highcharts to draw all graphs on the Statistics page.

3.1.10.6 DataTables

DataTables is a JavaScript plugin that can be used to convert a standard HTML table into a sortable, filterable and searchable table that updates in real time. In this project, DataTables is used for the application list on the unique device page and for the device list on the Dashboard page. DataTables also has a separate stylesheet plugin for use with Bootstrap, ensuring the table styling is uniform across the website.

3.1.10.7 LetsEncrypt

LetsEncrypt is a service that allows for the generation of free SSL/TLS certificates for the encryption of end to end communication between the client using a website and the server itself. This project utilises LetsEncrypt to enable and force an SSL connection to any resource that requests access to the server, this includes the Android applications API requests.

3.1.10.8 GitHub

GitHub is a web based Git repository hosting service. GitHub allows for code repositories to be stored on the cloud and accessed from anywhere or any device that

needs access to the repository. This project used GitHub for both web site repositories and Android application repositories.

3.2 Environment Set-up

Two different development environments were used throughout the development of this project; a Linux (Debian) virtual machine, for all web development, and Windows for the use of Android Studio and the development of the Android application. Finally, the live environment was enabled for public access to the website via a Digital Ocean VPS.

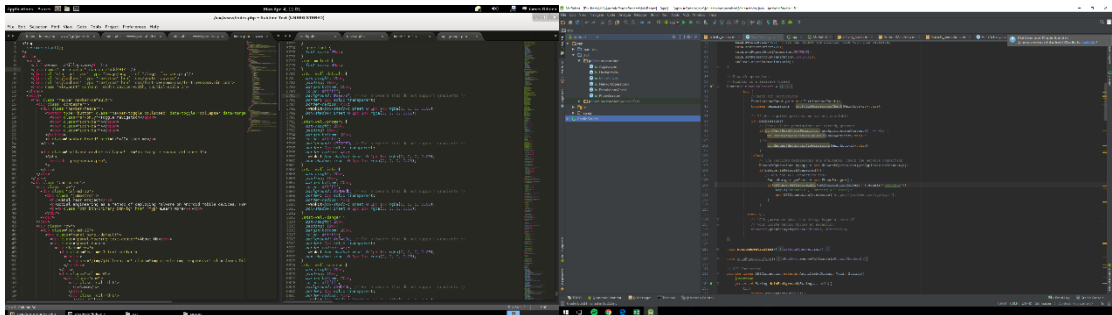


FIGURE 3: DEBIAN VIRTUAL MACHINE (LEFT) RUNNING ON WINDOWS (RIGHT)

3.2.1 Linux Virtual Machine

The development of the web aspects of this project were conducted with the aid of a Linux virtual machine, running the Debian operating system. This kind of set up was chosen for a very specific reason, the layout would match that of the VPS being used to host the live version of the web site. This meant that any configuration file, database or third party command line tools (LetsEncrypt) would behave exactly the same on both systems. The output of the development on the virtual machine would be identical to the output on the VPS once the latest Git commit had been pushed from local and pulled to the live environment. To enable development on the virtual machine, the following was installed: Apache2, PHP, PostgreSQL, Git and Sublime Text. These tools then allowed for the website to be developed in an offline environment and once the build was ready, Git allowed for commits and pushes to take place.

3.2.2 Windows Machine

The Windows aspect of this project was used for Android development. The installation of Android Studio was required and once that was installed, the various different SDKs available from Google for each Android API version were downloaded. The application was aimed at running on all Android phones that were Android Ice Cream Sandwich (4.0) or greater. The inclusion of version control software support in Android studio made it incredibly easy to set up Git to work with the Android Studio project. The Git binary was downloaded and Android Studio automatically located it and enabled the use of Git for version control of the application.

Accessing the VPS from Windows is different than on Linux, as Linux has native ssh protocols included, meaning you can ssh into the VPS terminal and work from there while using your Linux machine. On Windows, the program Putty was downloaded which acts as an ssh client for Windows to Linux communication and provides a terminal interface to administrate the server it connects too.

3.2.3 Virtual Private Server

The implementation of the VPS for this project was relatively straightforward. Once signed up on Digital Ocean it is only a few clicks required to initialize the server and have terminal access. Once the server is initialized, Putty is used to connect via SSH and administrate. The server is set up almost identically to Debian virtual machine, as mentioned before, the only difference is that the LetsEncrypt certificate is now for the registered Namecheap domain that was also included in the GitHub student developers pack. Once LetsEncrypt was installed and configured, the website was accessible via the domain <https://jdilleen.me> and was forcing SSL encrypted end-to-end connections for all clients. Git was also installed on the server and the GitHub web repository was cloned into the /var/www/ directory, this meant that any changes that were made on the local system could be pushed to GitHub and then pulled to the Digital Ocean server.

3.3 Website Development

HTML, CSS, JavaScript, jQuery and PHP were used to create the website aspect of this project. This project used PHP files to serve the user a webpage instead of HTML files, this was done because it allowed for the use of PHP scripts inline to the HTML code.

The base layout of every page was developed first, including a dynamic header based on `$_SESSION` variables and a base container layout for content to slot into. The dynamic header was accomplished via a PHP script that checked if the user on the website was logged in, if they were it would output some more HTML list `` items than if they weren't, giving logged in users more access to certain areas. Below (Figure 4) is an example page that shows the general layout of the site when a user visits a page, a simple header navigation bar followed by a container for all page content to be displayed.

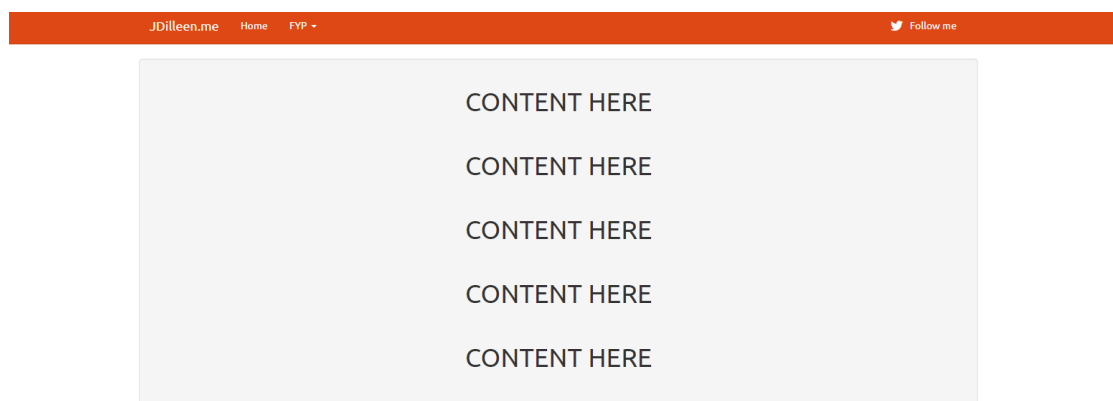
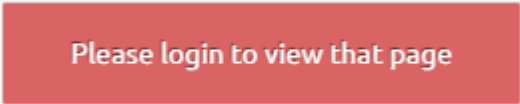


FIGURE 4: STANDARD WEB PAGE LAYOUT

Security of the website was paramount, being a security based project, so every page a user visits has a check included in its header, this is accomplished via a function to check if the page that the user is visiting requires that user being logged in to view, if the user isn't logged in the script will automatically redirect the user to the login page with an appropriate error message. Below (Figure 5) is a code sample showing how this function was developed and how the errors would be displayed to the user when they were not logged in.

```
function permissions($access_level){  
    // Start a session  
    session_start();  
  
    // Check if logged_in is set, if not - redirect the user  
    if(!isset($_SESSION["logged_in"])){  
        header("Location: /fyp/?error=login_needed");  
    }else{  
        if($_SESSION["access"] < $access_level){  
            header("Location: /fyp/?error=access_level");  
        }  
    }  
}
```



Please login to view that page

FIGURE 5: LOGIN CHECK AND ERROR MESSAGE

Once the basic layout of the website was laid out and the security measures to ensure that unauthorized access to the sensitive pages was not possible, the other aspects of the site began their development.

3.3.1 Database

The database was developed first in order for sample data to be inserted and tested with the layout of the website, once that had been developed. The schema to the database is as shown in the diagram below:

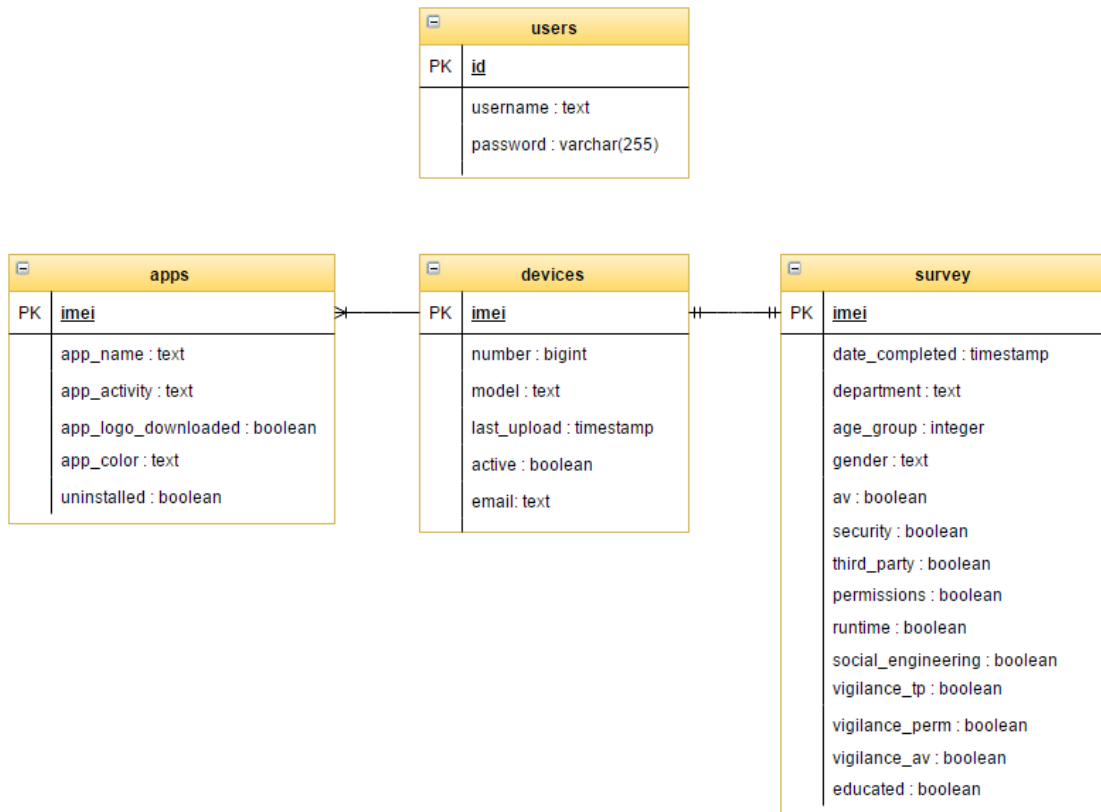


FIGURE 6: DATABASE SCHEMA

Figure 6 clearly shows the schema of the PostgreSQL database, with its relations highlighted via entity relationship indicators.

As can be seen in Figure 6, the IMEI of a device is the primary key in each table. The IMEI can be used to join all the tables and it has a relationship between them. The relationship between the devices table and the apps table is a one-to-many, meaning that each IMEI in the devices table must be unique but the apps table will have many entries for each IMEI (one for each application on the device.) The relationship for the survey table is a one-to-one, meaning that each IMEI in the survey table must also be unique, this ensures that each participant can only return one survey.

3.3.2 Login

The login system for this project utilises PHPs new “password_hash” function, this allows for hashing and salting of passwords without any user interference and is much more secure than the outdated MD5 and SHA-1 algorithms.

The login uses a simple HTML form with three inputs, one text input for the user to enter their username in clear text, a password field which masks the users input so their password is hidden to anyone watching and finally, a Google reCAPTCHA to ensure that bots cannot try to brute force a login on the site, this reCAPTCHA requires user input every time a user wishes to log in. The reCAPTCHA then uses complex algorithms to decide whether it thinks the user is a bot or a real human and if it’s unsure, it will ask the user to select a number of pictures from a grid of options telling the user to “Select all pictures that contain trees” or something similar.

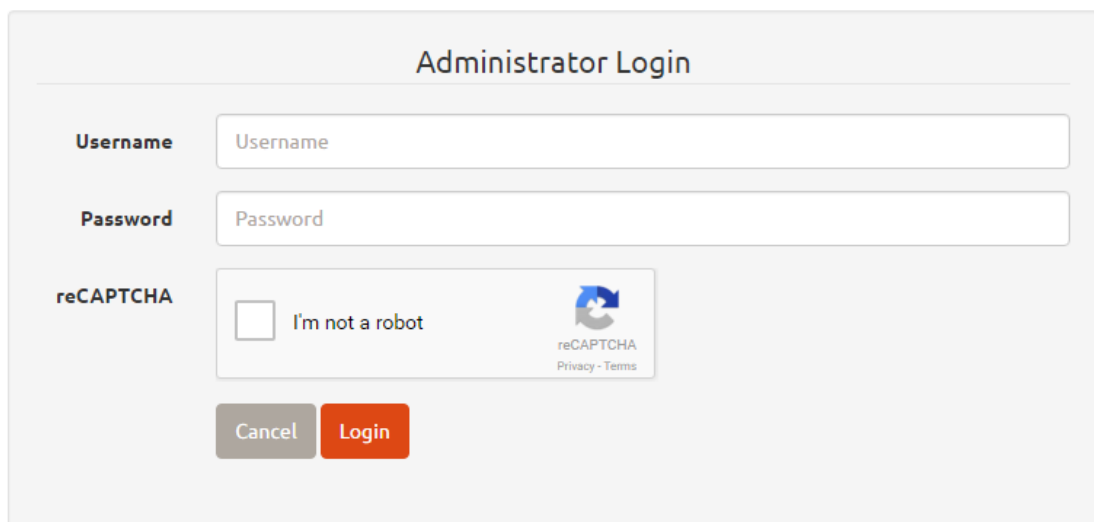
The image shows a web form titled "Administrator Login". It contains three main input sections: "Username" with a text input field, "Password" with a password input field, and "reCAPTCHA" which includes a checkbox labeled "I'm not a robot" and the reCAPTCHA logo. Below these inputs are two buttons: a grey "Cancel" button and an orange "Login" button.

FIGURE 7: LOGIN FORM

Once the user has entered details and submits the form by clicking the “Login” button, the form submission is cancelled by some jQuery on the page and the input fields are inspected. The jQuery checks that a username and password has been entered, that the username and password are not just spaces and that the reCAPTCHA has been completed. Once the jQuery has completed its checks, it will return a “true” variable to the function and it will submit the form via a POST request to a PHP script.

The first thing this script does is checks the validity of the reCAPTCHA response, this is accomplished by sending a request to the Google reCAPTCHA servers and the server will return the results of the reCAPTCHA back.

```
// Try and verify the reCAPTCHA
$url = "https://www.google.com/recaptcha/api/siteverify";
$data = array(
    "secret" => " ",
    "response" => $_POST["g-recaptcha-response"],
    "remoteip" => $_SERVER["REMOTE_ADDR"]
);

$options = array(
    "http" => array(
        "header" => "Content-type: application/x-www-form-urlencoded\r\n",
        "method" => "POST",
        "content" => http_build_query($data)
    )
);

// Send the data for verification
$context = stream_context_create($options);
// Get the results back
$result = file_get_contents($url, false, $context);
// Assign our success code to a variable
$response = json_decode($result)->success;
```

FIGURE 8: reCAPTCHA REQUEST

Once the reCAPTCHA has returned a successful result, the script will then retrieve the users username from the POST request and query the database to ensure that a user with this username exists, if the user exists, it pull the users hashed password from the database. With the hashed password and the plaintext password sent through the POST request, the script makes use of PHPs password_verify function where a developer can pass a clear text password and a hashed password as parameters and the function will return true if the hashed version of the clear text password matches that of the existing hashed password.

Once the script has verified the reCAPTCHA is successful, the user exists and the password they have provided is correct, it will start a session and start initializing session variables that are needed for logged in users, the most important being “\$_SESSION[“logged_in”]” as this variable is what the script in Figure 5 validates to ensure access to pages that need a login. After the script has assigned all the necessary variables, it triggers a header function to redirect the user to the Dashboard page, or if the login fails, back to the login page once again.

3.3.3 Generating a device

The original plan for this project involved the installation of the Android application on as many participants' devices as possible. Unfortunately this idea was rejected by the Universities ethics committee and compromises had to be made. The result was each participant would be given a test device so that their personal data would not be compromised. To allow for these restrictions, a separate function was developed to allow for the generation of fake devices, so that the participants could then answer the survey questions.

This was accomplished with a basic web form which took the participants name and the type of device the participant used (Android, iPhone, etc.). Once the form was submitted, a PHP script generated random details for the IMEI, email and phone number. This allowed for the project to adapt to the changes requested by the ethics committee without having to re-write a lot of how the system functioned.

```
// IMEI
$imei = mt_rand(1,9);
for($i = 0; $i < 14; $i++){
    $imei .= mt_rand(0, 9);
}

// Phone number
$number = 3538;
for($i = 0; $i < 8; $i++){
    $number .= mt_rand(0, 9);
}

// Email
$email = strtolower($_POST["name"])."."."@generated.com";
```

FIGURE 9: RANDOMLY GENERATING A DEVICE

3.3.4 Dashboard

The Dashboard page is the first page a user is greeted with when they log into the site. This is a simple overview of some statistics of current devices (there will only be one real device and the rest will be generated). It allows for the user to check if surveys have been completed for each device and view more details on the device if they are available. This page also shows a brief overview of the IMEI, phone number, model of phone, email address, activity, number of applications and the last time the device contacted the server and uploaded its list of application.

3.3.5 Device


The device page is where the user can view and search what applications have been installed on the device, when it last uploaded its details, toggle the activity of the device on or off, obfuscate the records or delete the records completely.

The option to toggle on and off the activity of the device was implemented before the ethics committees decision, this would mean that if a user study participant had installed the app on their device and had forgotten to uninstall it, the Web UI user could manually toggle the phones ability to upload to the server.

The option to obfuscate is also redundant because of the ethics committee's decision as this was implemented for users who decided they did not feel comfortable with their information being available to the website users and this function would completely randomise all details apart from the primary key, the IMEI.

Nexus 5 (358240056853377)

Information
Phone Number: +353851176871
Model: Nexus 5 (358240056853377)
Owner: dilleen.james@gmail.com

Active

Last Upload: 4 days ago
[Survey completed](#)

Actions
[Toggle activity off](#)
[Obfuscate records](#)
[Delete all records](#)

Installed Applications (77)

Table View [Grid View](#)

Options
☒ Show all applications
☐ Show installed applications
☐ Show uninstalled applications

Show entries Search:






App Logo	App Name	App Activity
	3Plus	ie.three.threeplus
	8 Ball Pool	com.miniclip.eightballpool
	A Dark World	com.EamonMallon.FYP
	AIB Mobile	aib.ibank.android
	AllCast	com.koushikdutta.cast

FIGURE 10: DEVICE PAGE

The Device page allows for the user to search through the list of applications the device has installed via a DataTables initialized table in which all fields are sortable and searchable. It also allows for a grid view of applications to be viewed which is just a graphical representation of the applications installed on the device.

This page will also highlight applications that the victim has uninstalled from their device since the last time the device contacted the web server. These applications are highlighted with a yellow tint in the application table list.

3.3.6 Application Image Scraping

For the Android application to be considered a successful malicious application, the application should leave as little of a footprint on the device as possible. To show the list of devices in a graphical way on the Web UI, it would be best to show the applications logo, however, uploading a mass of images from a device would leave a footprint on the user's data usage and also specifically point out that the game application was using a lot of data in particular.

To combat this, this project only uploads the application name and the application activity from the device to the web server in a JSON object so that data usage is minimal. To show the application images on the web server, this project uses a PHP function that automatically scrapes the Google Play Store for the applications store page, finds the application logo and downloads that logo to the web server to display whenever needed. This automated process is accomplished via the use of a CRON task, where the PHP script is called upon every minute. The script checks if there is any applications in the database that do not yet have a logo present on the server for them and if there is it launches a request to the Google Play Store to download the application image.

```
// Append our app name to the full play store URL
$full_url = $play_store_url.$app_activity;

// If getting the contents was successful
if(get_http_response_code($full_url) == "200"){
    // Get the contents of this URL
    $html = file_get_contents($full_url);

    // Create a new DOM Document
    $doc = new DOMDocument();
    @$doc->loadHTML($html);

    // Get all the image data on the page
    $tags = $doc->getElementsByTagName("img");

    // Get the first element, this is always the chosen apps image
    foreach($tags as $tag){
        if($tag->getAttribute('class') == "cover-image"){
            $cover_image = $tag->getAttribute('src');

            // Break the foreach as we only need to get the first cover-image element
            break;
        }
    }
}
```

FIGURE 11: GOOGLE PLAY LOGO SCRAPING

3.3.7 Statistics

The statistics page is used for viewing all the results of the participant's surveys that were returned during the user study phase of this project.

This page is a collection of graphs and statistical information displayed in a graphical format. The rendering of these graphs were made possible by the use of the Highcharts plugin. This plugin allows for the customization of multiple parameters for different types of graphs depending on what the project needs.

The best aspect of Highcharts is the ability to trigger an AJAX request to a PHP script that will return a JSON encoded array full of details for Highcharts to draw the graph.

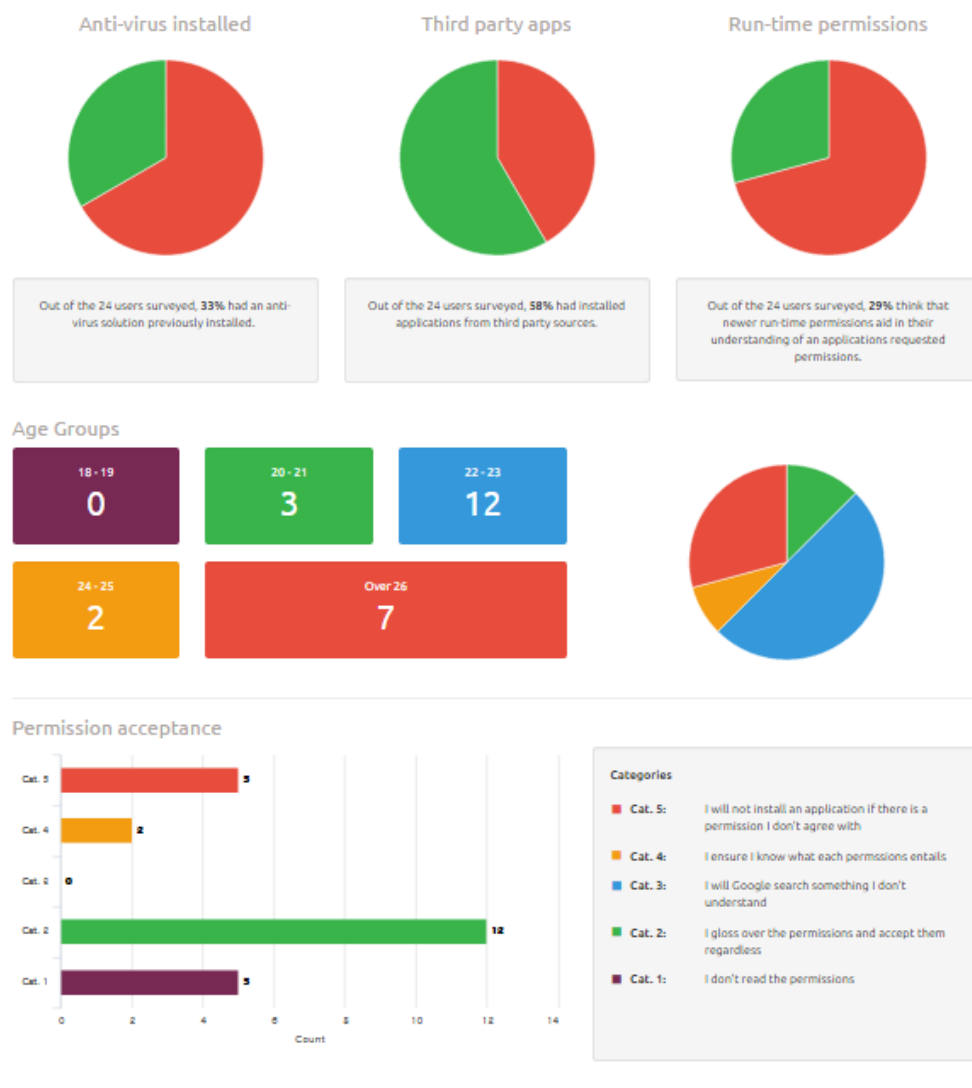


FIGURE 12: STATISTICS PAGE

3.3.8 API

The API for the Android application to communicate with the server is relatively simple aspect of the project. It is a single PHP script that has three functions; **addnew** for registering and adding new devices to the database, **verify** for checking if the Android device that is uploading is marked as an active device and **upload** for uploading the list of applications from the device and inserting them in the database.

The API functions by using the HTTP query string function. This means that to call the **addnew** function, “`?action=addnew&imei=12345678`” is appended to the default API URL. The PHP script parses the query string into two parts, the action and the IMEI number of the device.

All data sent from the Android application is sent via a `JSONObject` which is received by the PHP script by the `php://input` function. This can then be decoded via the `PHP json_decode` function and will result in an accessible array.

Depending on what action parameter is passed in the query string, the following functions are called:

addnew – The script will perform a check on the database to ensure the device is already in the database. If the device isn’t in the database, a simple `INSERT` query is run with the scraped IMEI, phone number, email and phone model included and inserted into the database. If the device is already in the database, the script will return a “**verify**” message to the Android application signalling the application to continue to the next step.

verify – The **verify** function will query the database for the activity flag for the device that is currently trying to communicate, this activity flag is a simple Boolean. If the flag is true, the API will return a “**upload**” message to the application and if the flag is false, it will return an “**inactive**” message. This means that if the device has been set to inactive through the Web UI, it will no longer upload its list of installed applications.

upload – Assuming the application is active, the API will then accept a `JSONObject` which will be decoded and looped through to perform multiple `INSERT` queries, one for each installed application. This function accounts for apps already in the database as to not have duplicates.

3.4 Android Application Development

The Android application is designed to look like a game, only showing a message saying “Breach Detected” when the application has uploaded the list of installed applications from the device. To accomplish this, a basic menu was developed with all scraping and uploading functionality running in the background, unknown to the participant.

3.4.1 AppScraper

The AppScraper class is used to provide a function that returned a JSONObject of all installed applications on the device. To accomplish this, the function needs to take the applications current Context as an argument in order for a PackageManager object to be created.

Once the object is created, the function creates a JSONObject and starts a for loop, adding each applications LoadLabel (Name) and package name into a pair to the JSON object. It then simply returns the object.

```
public static JSONObject ListAllInstalledApps(Context c){
    // Create a PackageManager with the given context
    PackageManager pm = c.getPackageManager();

    // Create a list of all installed applications
    List<ApplicationInfo> apps = pm.getInstalledApplications(PackageManager.GET_META_DATA);

    JSONObject obj = new JSONObject();

    // For each app, check if they have:
    // a) Launcher intent - they are launchable apps (not background processes)
    // b) The app is not flagged as a system app
    // c) The app is not flagged as a system updated app
    for(ApplicationInfo app : apps){
        if(pm.getLaunchIntentForPackage(app.packageName) != null) {
            if((app.flags & ApplicationInfo.FLAG_UPDATED_SYSTEM_APP & ApplicationInfo.FLAG_SYSTEM) != 1) {
                try {
                    obj.put(app.loadLabel(pm).toString(), app.packageName);
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    // Return the arraylist of apps
    return obj;
}
```

FIGURE 13: APPSCRAPER FUNCTION

3.4.2 NetworkOperations

The NetworkOperations class provides access to a function for the MainActivity class to use to verify if the network is connected on the device. If the network is not connected, then the device will not be able to upload any data and will throw an error and crash the application. This check ensures that an upload is not attempted without first verifying that the network is connected.

```
public class NetworkOperations {
    // Set a tag for debugging
    private static final String TAG = "NetOps";

    Context mContext;

    // Pass context to this class
    public NetworkOperations(Context mContext) { this.mContext = mContext; }

    // Check if the network is connected
    public boolean isNetworkConnected() {
        ConnectivityManager connMgr = (ConnectivityManager)mContext.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        if (networkInfo != null && networkInfo.isConnected()) {
            Log.d(TAG, "Network is connected");
            return true;
        } else {
            Log.e(TAG, "Network is not connected");
            return false;
        }
    }
}
```

FIGURE 14: NETWORKOPERATIONS CLASS

3.4.3 PermissionsCheck

There is two permissions that are needed for this application to run. In order for the application to retrieve the devices IMEI number, the Read State permission is needed and in order for the application to retrieve the Google Account email that is associated with the device, it needs the Get Accounts permission.

To accomplish the upload, the application must retrieve data if and only if the permissions that are required have been accepted by the user, otherwise the functions that scrape data will throw errors and crash the application. For the application to check if the permissions have been accepted, the applications current context is passed to functions contained within the PermissionsCheck class and these functions simply return a Boolean value depending on the status of that permissions acceptance. These Boolean values are used in the MainActivity to trigger permission requests (Android Lollipop 5.0+).

3.4.4 PhoneScraper

The PhoneScraper class is responsible for three aspects of the applications functionality, it retrieves the devices IMEI, the phone number associated with the device and the email associated with the device.

For these functions to be carried out, the applications current context must be passed to each of the functions, the functions will then return the requested aspect.

```
public class PhoneScraper {

    public static String GetDeviceIMEI(Context c){
        TelephonyManager tm = (TelephonyManager)c.getSystemService(Context.TELEPHONY_SERVICE);
        return tm.getDeviceId();
    }

    public static String GetDeviceNumber(Context c){
        TelephonyManager tMgr = (TelephonyManager)c.getSystemService(Context.TELEPHONY_SERVICE);
        return tMgr.getLine1Number();
    }

    public static String GetDeviceEmail(Context c){
        String possibleEmail = "unavailable";
        Pattern emailPattern = Patterns.EMAIL_ADDRESS; // API level 8+
        Account[] accounts = AccountManager.get(c).getAccounts();
        for (Account account : accounts) {
            if (emailPattern.matcher(account.name).matches()) {
                possibleEmail = account.name;
                break;
            }
        }

        return possibleEmail;
    }

}
```

FIGURE 15: PHONESCRAPER CLASS

3.4.5 MainActivity

The MainActivity class is responsible for a lot of the applications background processes. The first thing the class does is creates the screen to display and changes some of the TextViews to a console like font, to suit the theme of the application.

The repeating network task then begins, but before anything can be uploaded the application must pass all of the checks described in the previous sections. The permissions are first to be checked, this is a new behaviour due to how Android has updated their permissions procedure. Before, an application would list all required permissions before the app was installed, now, the application requests the permissions as run-time when it requires them. If the required permissions are not present, the

application will trigger a dialog asking the user to accept whatever permission the application requires.

If the applications required permissions have been accepted, the application then checks if the network is connected in order to upload the data required. Once these two requirements have been met, the application begins an AsyncTask on a separate thread (as required by Android for network operations).

The application then makes a total of three requests, once request to the addnew function of the PHP API, checking if the application has already registered the device in the remote database. This function will register the phone in the database if it has not already done so or it will tell the application to continue to the next step. The next step is a call made to the verify function of the API, this returns true or false based on whether the devices activity has been toggled on or off, if the activity returns as false, the application doesn't continue to the next step, if the activity function returns true, the application continues to the final step. The application now creates an object of the AppScraper class and runs the ListAllInstalledApplications function to retrieve the JSONObject. It then proceeds to send this object to the upload function of the API.

In order to execute HTTP requests to an external API the MainActivity class has a function specifically for building these requests and executing them. The requestBuilder function relies on Android built in URL and HttpURLConnection classes to make the necessary HTTP calls to the API.

3.4.6 Game Menu & Breach Activity

The main purpose of the Android application in this project is to deceive the participant into believing the application is an actual game and therefore the application needed to be developed with that in mind.

The application itself is called "Mr. Robot" a homage to the USA network TV show about a hacker who works at a cyber-security company and is therefore designed with a terminal look in mind.

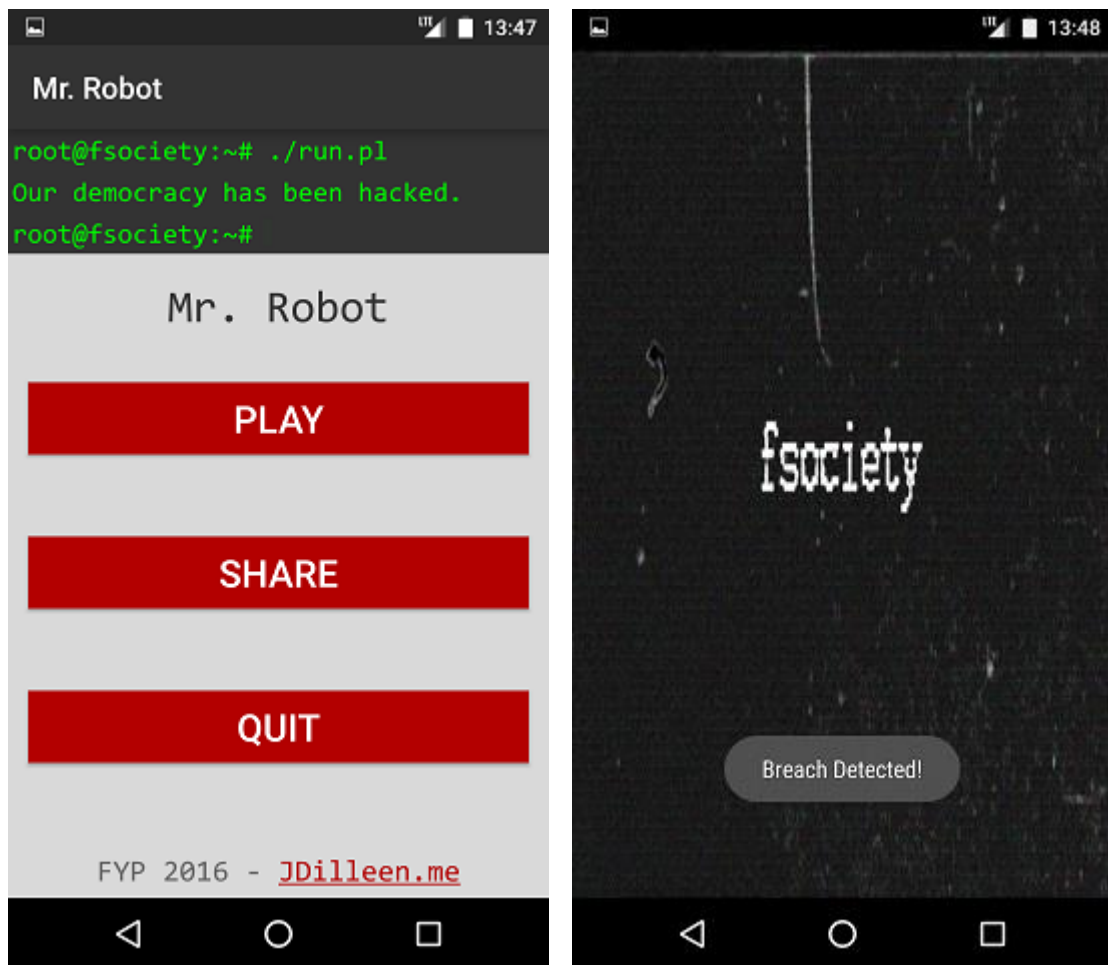


FIGURE 16: MAIN MENU AND BREACH SCREEN

As seen in Figure 16, the main menu consists of three buttons that actually have no purpose. The play button will only check for permissions and will trigger a permission request if a permission required is not valid. It will also load some fake Toast dialogs stating that the game is “Loading”, “Gathering resources” and “Executing commands” which are all meaningless and are just present to distract the user as to what is happening with the application in the background. The share button isn’t linked to any listener and is 100% useless. The quit button will actually quit the application, if the user hits this quit button instead of their home button, the applications process will be ended and the application will not continue to upload until it is restarted. This button was added as a method of ensuring the application was not running in the background unexpectedly.

Once the application has successfully uploaded the list of installed applications on the device, it will launch a new activity that shows a graphical animation of the “fsociety” collective (inspired by Anonymous) from Mr. Robot with a “Breach Detected” Toast message shown to the user.

4. Evaluation

4.1 Testing

4.1.1 Website Testing

After any change was made to the website in the local development environment, it would be extensively testing to ensure that all new functions were working as intended. Once these changes were tested sufficiently, the live server would be updated through the use of a git pull.

To test any defects of the website, Google Chromes developer tools were used to display a console output of any errors that were occurring on the current page that was being developed.

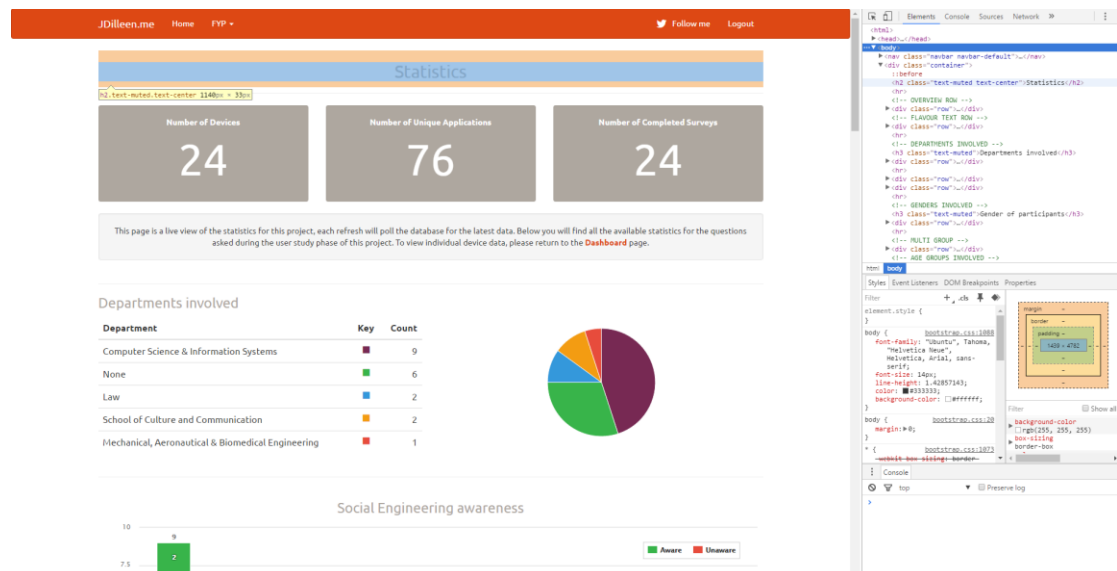


FIGURE 17: CHROME DEV TOOLS IN ACTION

These developer tools made developing the website much easier as the console displayed any errors and the inspect feature can help with CSS changes tremendously. As seen in Figure 17, Chrome Dev Tools can be used to highlight elements on the current page to view details about attributes like padding and margins etc.

To ensure the website functioned as expected, a small heuristic evaluation was conducted to highlight any usability issues that may have arisen during development.

4.1.2 Android Application Testing

While in development, the Android Studio debug console proved to be an invaluable tool to aid in error checking and general debugging. Android Studio can display messages that have been logged programmatically in the code, displaying a user defined tag and a message and/or variable that the user wants to log.

This debug console was used extensively during the HttpURLConnection development of the application, where application crashes and broken return results from the server were commonplace. Using the console to quickly identify the location and cause of a fatal crash proved to be useful in fixing these app-breaking issues.

The benefit of developing with Android Studio was that an emulation environment was available whenever one was needed. This meant that a physical device connected to the development machine was not always a requirement as it was easy to choose a device of any make and model to test the applications functionality on. Android Studio also provides an easy to understand interface for connecting a physical device to the development machine and running the application on that device.

4.2 Problems Encountered

4.2.1 Android to Server communication

For the development of the Android applications network communication capability, it was not possible to test the application with an internal local IP, this meant that any changes that were made to the API would have to be committed, pushed and then pulled to the live remote server. To test the communication, the application would also need to be launched on a physical device and restarted after each change made to the applications source code.

With all these factors combined, it made for testing the applications communication a very troublesome, tedious and frustrating task.

To effectively troubleshoot any errors arising from either aspect (server or application) PHP's "echo" function was used to return results to the application which then logged any results and output to the debug console for easy to read errors.

4.2.2 Ethical Approval

Given the potentially malicious behaviour of this application, ethical approval was needed solely on the grounds that the application would be deceiving the user study participants.

Soon after the form was submitted for ethical approval I received an email saying that the request had been denied and that certain changes needed to be made to allow for approval to be granted. Fortunately, the changes that were requested were simple; do not install the application on any other device but the test device and if the application was installed on any other device, it would not function as intended.

Initially, the website was developed in a way that in order for a user to be surveyed, they would need to have run the application on their own device so that it registered an entry in the database for that individual. Once the ethics request was denied, this system needed to be changed in order to obtain approval. To accommodate the requests of the ethics committee, the “Generate a device” feature was added. This feature generates a dummy entry for a device that doesn’t exist and allows a user to be tied to that dummy device for the purposes of the user study survey.

Making changes to the Android application to ensure that the scraping functionality would not be called on any device other than the project test device was very simple. As the application already needs to scrape the devices IMEI number in order to run HttpURLConnection requests, a simple if statement was put in place that would simple check if the current devices IMEI was equalled to that of the project test devices IMEI, and if it was not, no data scraping would take place.

4.3 User Study

The user study took place over the course of two weeks where 24 randomly selected participants were selected to take part in the study. These users were told that the user study they were partaking in was for the purpose of testing a game made as part of a final year project and their feedback was welcomed.

Users were given the project test device on a download page and asked to touch the application apk file in order to install it. The device then ran an anti-virus scan via Lookout mobile security, a sophisticated Android anti-malware application, and the

scan always returned “app_1.2.apk is safe”. The user was then asked to open the application and accept the permissions as they were needed to run the game, not one user questioned the permissions once they were told they were required, users simply pressed “Allow” on both permission request dialogs.

As expected, each user then believed the Toast dialogs they were presented with stating the game was “Loading...” etc. Not until the application triggered the Breach Activity did people question what was happening. They were then told what the purpose of the application was and they had been deceived. Every user was very surprised at how easy it was to breach their personal data, most said they would have installed the application on their device if they were asked too.

The users were then asked to complete a short survey, the results of the survey will be expanded upon and explained in section 4.3.1.

4.3.1 Statistics

Over the course of two weeks 24 users were successfully breached in the user study, all statistics in this section are available to view in detail on the projects website [here](https://www.jdilleen.me/fyp/statistics/). (<https://www.jdilleen.me/fyp/statistics/>)

4.3.1.1 Social Engineering Awareness

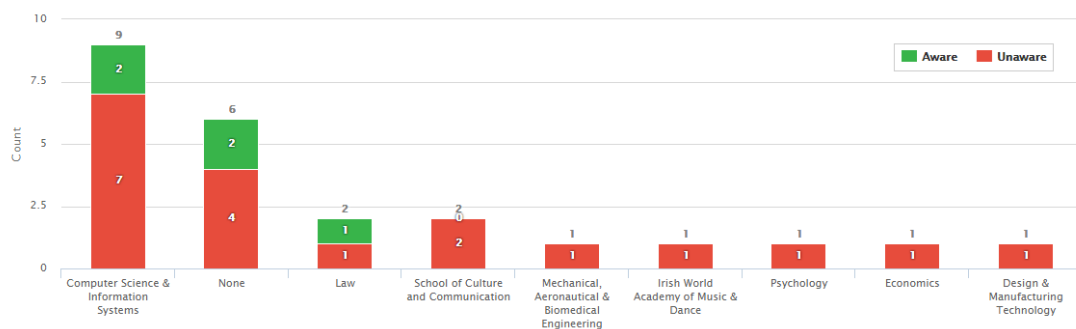


FIGURE 18: SOCIAL ENGINEERING AWARENESS

Out of the 24 participants, a total of 5 users were already aware of what Social Engineering was before they undertook the user study. This equates to a measly 21%.

9 of the 24 participants had come from the Computer Science and Information Systems department, yet only 2 of those 9 were previously aware of what social engineering was or entailed. This is a surprising statistic as common sense would dictate that technology savvy individuals would be aware of these kinds of affairs. This shows that Social

Engineering is effective against any individual, no matter what their background is, as it is a non-technical form of exploitation. Surprisingly, 5 out of the 5 individuals who were previously aware of what Social Engineering was had learned about it through the TV show that the Android application was inspired by, Mr. Robot.

4.3.1.2 Permission Acceptance

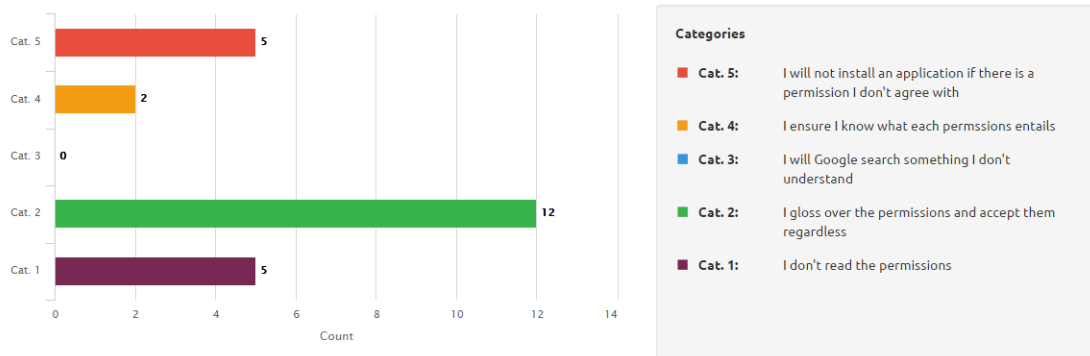


FIGURE 19: PERMISSION ACCEPTANCE

Another interesting statistic to come out of the user study is the amount of participants that accept permissions regardless of what they are, or don't read permissions whatsoever. A staggering 17 of the 24 total participants fell into one of those two categories with not one user saying they would Google search a permission they did not understand. The disparity in this question arises when 2 users state they ensure they know what each permission entails and 5 users say they will not install an application if there is a permission they don't agree with. This was surprising seeing as those users had also just blindly accepted permissions a few moments before this survey on the premise of them being told too.

4.3.1.3 Anti-virus previously installed

In this pie chart, the green segment accounts for the portion of users with an anti-virus solution already installed on their device. Out of the 24 users surveyed, only 33% of them had an anti-virus solution installed on their device. Most users cited anti-virus as being large and resource intensive / battery draining.



FIGURE 20: ANTI-VIRUS INSTALLED

4.3.1.4 Installed apps from Third Party sources

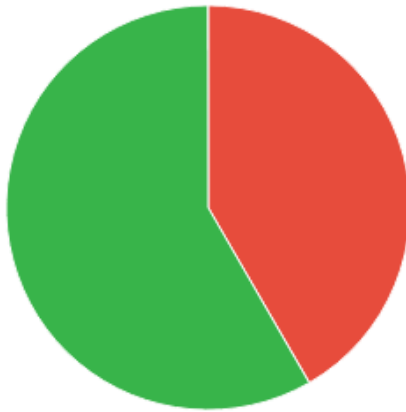


FIGURE 21: THIRD PARTY APPS

In this pie chart, the green segment accounts for the portion of users who had installed applications to their device from Third Party sources (unofficial sources) in the past. Out of the 24 participants, 58% of them had installed applications from unofficial sources at some point in the past. Many of these users were unaware that applications acquired from unofficial sources are not certified or vetted by Google. The participants were then made aware of the vast amount of security implications that arise when installing

applications from a third party source and that the personal data on their device would be very easy to compromise without an the application being certified by Google.

4.3.1.5 New run-time permissions

In this pie-chart, the green segment accounts for the portion of users who believe Google's new method of requesting permissions is better than the old method. In the past (pre-Android Lollipop 5.0), when installing an application, your device would list all permissions that the application would require and let the user decide whether or not they wanted to install that application. The new system for requesting permissions is very different, instead of listing all permissions before the application is

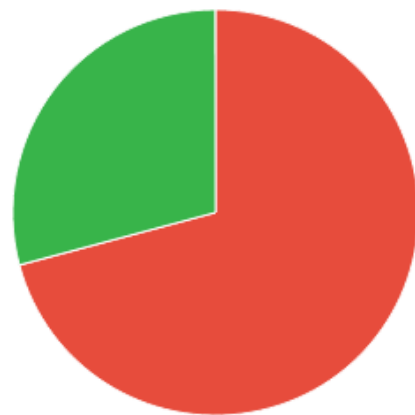


FIGURE 22: RUN-TIME PERMISSIONS

installed, the application will prompt a user to accept a permission at run-time, when the application needs access to this permission for operational purposes. This means that the user would have a chance to become familiar with the application and may be surprised by a permission they would rather not accept. The user study asked users if they prefer the old system over the new system and a staggering 71% answered "No", the old system was much better.

4.3.1.6 Security of data

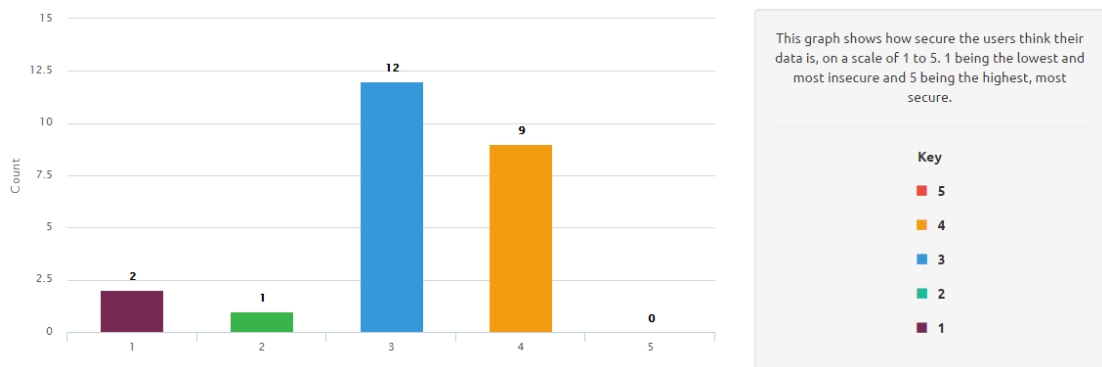


FIGURE 23: SECURITY OF DATA

During the survey, participants were asked how secure they thought their data was on their device on a scale of 1 to 5, 1 being very insecure and 5 being very secure. As shown in Figure 23, many 21 of the 23 users settled for a 3 to 4 rating. No user said their data was very secure while 3 users said they felt their data was not very secure on the device. The amount of users on the secure end of the scale is surprising given the amount of media any large scale data breach gets these days.

4.3.1.7 Will install an anti-virus solution

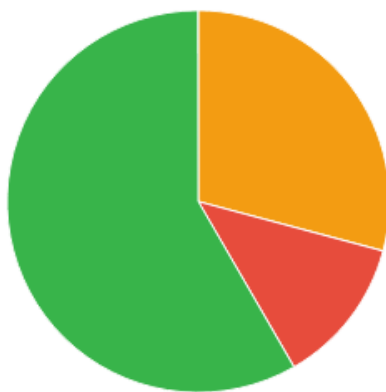


FIGURE 24: WILL INSTALL ANTI-VIRUS

In this pie-chart, the green segment represents users who will install anti-virus software on their mobile device after the study. The orange segment represents the portion of users who already had anti-virus installed before the study and the red segment represents users who would not install anti-virus after the user study. Out of the 24 users surveyed, 88% would install, or already had installed, an anti-virus software solution on their device after completing the study. Although the application used during the study

was cleared as safe by a well renowned anti-virus solution, if an application was uploading more than just a list of applications, any mediocre anti-virus solution would flag the app as dangerous. The surprising aspect of this question was the 12% of users who would not install an anti-virus solution, even after just being shown how it is to breach any user's private information.

4.3.1.8 Vigilance regarding Third-party sources

In this pie-chart, the green segment represents the portion of users who would now be more vigilant when installing applications from third-party sources after the conclusion of the user study. Of the 24 users surveyed, 79% said they would now be more vigilant. This is a great result as most malware on Android devices comes from third-party sourced apps, this is one of the downsides of the Android operating system being so open to developers. With users opting to be more vigilant when it comes to

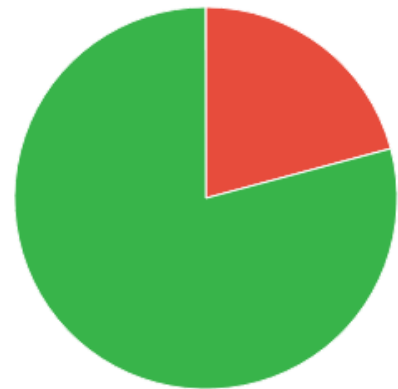


FIGURE 25: THIRD-PARTY SOURCE VIGILANCE

third-party sources, this would mean that the user's personal information is all the more safe and that companies could rely on their employees not to bring infected devices into a corporate network where they might have the chance to propagate.

4.3.1.9 Vigilance accepting permissions



FIGURE 26: PERMISSIONS VIGILANCE

In this pie-chart, the green segment represents the portion of users that would be more vigilant when accepting permissions from any application. Out of the 24 users surveyed, 75% of them answered "Yes" to the question regarding if they would be more vigilant when accepting permissions requested by any application, not just applications from third-party sources. This shows that the study had an effect on users, where accepting two basic permissions allowed for the retrieval of their entire list of installed applications on their device.

Users were surprised at how easy it was to obtain a list of applications and even more surprised when told that the retrieval of the list of applications themselves required no permissions whatsoever. The only aspects of the applications that required permissions was obtaining the devices IMEI number and retrieving the Google account email address that was set as the primary address for the device.

5 Conclusion

In conclusion, the purpose of this study was to raise awareness of the various aspects of Social Engineering that could be used to exploit a person's human nature. The idea of security through education focuses on showing users, first hand, how easy it is to breach their personal information and it employs the use of a scare tactic mentality to push the point.

The very last question of the user study survey asked the participants if, after the study, they felt more educated as to what Social Engineering is and how it can be used to exploit you to which the result was 100% "Yes". This proves the study was a success and that if conducted on a broader scale could educate a much larger group of people against one of the most effective yet hidden hacking methods being used today.

"The greatest threat to an organization's information security is the manipulation of employees who are too often the victims of ploys and techniques used by slick con men known as social engineers." (Brody, et al., 2007)

References

Apache Software Foundation, 1995. *HTTP Server Project*. [Online]

Available at: <https://httpd.apache.org/>

[Accessed 2015].

Brendan Eich; Netscape Communications Corporation; Mozilla Foundation; Ecma International, 1995. *JavaScript.com*. [Online]

Available at: <https://www.javascript.com/>

[Accessed 2015].

Brody, R., Brizzee, W. & Cano, L., 2007. Flying under the radar: social engineering. *International Journal of Accounting & Information Management*.

Digital Ocean, 2011. *Simple Cloud Hosting, Built for Developers..* [Online]

Available at: <https://www.digitalocean.com/>

[Accessed 2015].

GitHub, 2008. *GitHub - How people build software*. [Online]

Available at: <https://www.github.com>

[Accessed 11 April 2016].

Håkon Wium Lie; Bert Bos; W3C, 1996. *World Wide Web Consortium*. [Online]

Available at: www.w3.org

[Accessed 2015].

Highsoft AS, 2009. *Highcharts: Interactive JavaScript charts for your webpage*.

[Online]

Available at: <http://www.highcharts.com/>

[Accessed 2015].

Krebs, B., 2014. *Krebs on Security*. [Online]

Available at: <http://krebsonsecurity.com/2014/05/the-target-breach-by-the-numbers/>

[Accessed 11 April 2016].

LetsEncrypt, 2015. *Let's Encrypt - Free SSL/TLS Certificates*. [Online]

Available at: <https://letsencrypt.org/>

[Accessed 11 04 2016].

Oracle Corporation, 1995. *java.com: Java + You*. [Online]

Available at: <http://java.com/en/>

[Accessed 2015].

Otto, M. & Thornton, J., 2011. *Bootstrap - The world's most popular mobile-first and responsive front-end framework*. [Online]

Available at: <http://getbootstrap.com/>

[Accessed 2015].

Park, T., 2014. *Bootswatch: Free themes for Bootstrap*. [Online]

Available at: <https://bootswatch.com/>

[Accessed 2015].

PostgreSQL Global Development Group, 1996. *PostgreSQL: The world's most advanced open source database*. [Online]

Available at: <http://www.postgresql.org/>

[Accessed 2015].

Resig, J., 2006. *jQuery*. [Online]

Available at: <http://jquery.com/>

[Accessed 2015].

Rogers, M., 2014. *Dendroid malware can take over your camera, record audio, and sneak into Google Play*. [Online]

Available at: <https://blog.lookout.com/blog/2014/03/06/dendroid/>

[Accessed 2015].

Rouse, M., 2014. *What is Social Engineering?*. [Online]

Available at: <http://searchsecurity.techtarget.com/definition/social-engineering>

[Accessed 2015].

Snyder, C., 2015. *Handling Human Hacking: Creating a Comprehensive Defensive Strategy Against Modern Social Engineering*. s.l.:s.n.

SpryMedia Ltd, 2011. *DataTables Table plug-in for jQuery*. [Online]

Available at: <https://datatables.net/>

[Accessed 2015].

The Social Engineer, 2014. *The Social Engineering Infographic*. [Online]

Available at: <http://www.social-engineer.org/social-engineering/social-engineering-infographic/>

[Accessed 10 April 2016].

Torvalds, L., 2005. *Git --fast-version-control*. [Online]

Available at: <https://git-scm.com/>

[Accessed 11 April 2016].

W3C; WHATWG, 1993. *World Wide Web Consortium*. [Online]

Available at: <http://www.w3.org/>

Younes, M., 2014. *ALERTIFY JS*. [Online]

Available at: <http://alertifyjs.com/>

[Accessed 2015].

Zend Technology, 1995. *PHP: Hypertext Preprocessor*. [Online]

Available at: <https://secure.php.net/>

[Accessed 2015].