UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Department of Computer Science

# Computer Science Honours Final Paper 2016

| | |
|---|---|
| Title: | Testing an Android Implementation of the Social Engineering Protection Training Tool |
| Author: | Marcel Teixeira |
| Project abbreviation: | SEPTT |
| Supervisors: | Tommie Meyer<br>Francois Mouton (external supervisor) |

| **Category** | **Min** | **Max** | **Chosen** |
|---|---|---|---|
| Requirement Analysis and Design | 0 | 20 | - |
| Theoretical Analysis | 0 | 25 | - |
| Experiment Design and Execution | 0 | 20 | 15 |
| System Development and Implementation | 0 | 15 | 10 |
| Results, Findings and Conclusion | 10 | 20 | 20 |
| Aim Formulation and Background Work | 10 | 15 | 15 |
| Quality of Paper Writing and Presentation | 10 | | 10 |
| Quality of Deliverables | 10 | | 10 |
| Overall General Project Evaluation (this section allowed only with motivation letter from supervisor) | 0 | 10 | - |
| **Total marks** | | **80** | **80** |

# Testing an Android Implementation of the Social Engineering Protection Training Tool

Marcel Teixeira
Department of Computer Science
University of Cape Town
marceltex@gmail.com

## ABSTRACT

As the nature of information stored digitally becomes more important and confidential, the security of the systems put in place to protect this information needs to be increased. The human element, however, remains a vulnerability of the system and it is this vulnerability that social engineers attempt to exploit. The Social Engineering Attack Detection Model version 2 (SEADMv2) has been proposed to help people identify malicious social engineering attacks. Prior to this study, the SEADMv2 had not been implemented as a user friendly application or tested with real subjects. This paper describes how the SEADMv2 was implemented as an Android application. This Android application was tested on 20 subjects, to determine whether it reduces the probability of a subject falling victim to a social engineering attack or not. The results indicated that the Android implementation of the SEADMv2 significantly reduced the number of subjects that fell victim to social engineering attacks. The Android application also significantly reduced the number of subjects that fell victim to malicious social engineering attacks, bidirectional communication social engineering attacks and indirect communication social engineering attacks. The Android application did not have a statistically significant effect on harmless scenarios and unidirectional communication social engineering attacks.

## Keywords

Android, Social engineering, Social Engineering Attack Detection Model, Social Engineering Attack Framework

## 1. INTRODUCTION

Social engineering refers to various techniques that are utilised to obtain information through the exploitation of human vulnerability in order to bypass security systems [10]. Social engineers exploit the helping and trusting nature that most humans inherently have. Social engineers also prey on the fact that most people never expect to be a victim of social engineering and are rather careless at times [11].

This thesis was completed as part of a Bachelor of Science Honours degree in Computer Science at the University of Cape Town in 2016. The Android application developed for this research was published to the Google Play Store and can be accessed at https://play.google.com/store/apps/details?id=za.co.social_engineer.www.socialengineer. All source code for this project was made available on a public git repository and can be accessed at https://github.com/marceltex/Social-Engineer. Raw data obtained from the experiment performed can be accessed at http://www.social-engineer.co.za/data.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Successful social engineering attacks have proven to be extremely expensive. In the UK, for example, it is estimated that identity theft[1] related crimes cost the UK economy around 1.2 billion pounds in 2009 [16]. Losses from phishing[2] were around 23.2 million pounds in 2005. This is almost double the amount lost due to phishing in 2004, which was 12.2 million pounds [16]. In 2004, the US Department of Justice concluded that one in three people are likely to become a victim of social engineering in their lifetime [18]. Therefore, it is essential that an effective social engineering protection tool be implemented, tested and made available to the public to save individuals and corporations from losing millions.

There are limited techniques available to detect social engineering attacks. Some of the detection mechanisms, that have been proposed, assist the user to identify whether they are the victim of a social engineering attack or not, while other mechanisms use an automated system to detect social engineering attacks. The Social Engineering Attack Detection Model version 2 (SEADMv2), proposed by Mouton *et al.*, [12] is the most prominent social engineering attack detection mechanism and therefore it was used for this study. The SEADMv2 achieves detection of social engineering attacks by using a series of states that require a user to provide yes/no answers for each question in the state. Based on the user's answers to the questions, the SEADMv2 gives the user an indication as to whether the provided scenario is a social engineering attack or not. The SEADMv2 was chosen since it can be used to detect both textual and verbal social engineering attacks. In addition, it can be used to detect unidirectional, bidirectional or indirect social engineering attacks, making it a very well-rounded and versatile social engineering attack detection model.

There are currently limited resources available to aid in the detection and prevention of social engineering attacks. In addition, there is a general lack of knowledge about social engineering and the techniques used by attackers to manipulate their target. As a result, the probability of successful social engineering attacks is relatively high. The research question, of this study, is formally stated as follows: *Can an Android application that implements the SEADMv2 reduce the probability of a subject falling victim to a social engineering attack?* I hypothesise that an Android application that implements the SEADMv2 will reduce the probability of a

[1]The fraudulent acquisition and use of a person's private identifying information, usually for financial gain.
[2]The activity of defrauding an online account holder of financial information by posing as a legitimate company.

person falling victim to a social engineering attack, since it is less likely that a person will fall victim to an attack with the aid of a model that serves its intended purpose correctly.

This paper will describe how the SEADMv2 [12] was implemented as an Android application called Social Engineering Protection Training Tool (SEPTT). It also describes the experiment that was carried out to test whether the Android implementation of the SEADMv2 serves its intended purpose and improves people's ability to detect malicious social engineering attacks correctly. The results obtained from performing the experiment were analysed, using appropriate statistical measures, to ensure that the results are statistically significant. The Android implementation of the SEADMv2 was found to significantly reduce the number of subjects that fell victim to social engineering attacks, the only exceptions are that the model did not significantly reduce the number of people who fell victim to unidirectional communication social engineering attacks and harmless scenarios.

The remainder of this paper is structured as follows: Section 2 provides a brief background of social engineering, the SEADMv2 [12], the Social Engineering Attack Framework (SEAF) [15] and the classification of social engineering attacks. Section 3 describes how the Android application, which was developed for this experiment, was designed and implemented. Section 4 describes how the experiment was conducted and the scenarios used for the experiment. Section 5 analyses and discusses the results obtained. Section 6 draws conclusions from the results and discusses these conclusions. Section 7 suggests improvements that could be made to the SEADMv2 and the SEPTT Android application and Section 8 acknowledges everyone who helped to make this research project a success.

## 2. BACKGROUND

Social engineering is defined as the techniques used to exploit human vulnerability to bypass security systems in order to gather information [10]. In social engineering attacks, the vulnerability of the system is considered to be the human element. The attacker exploits the trusting nature that most humans inherently have, in order to get the information they desire. It is common for attackers to pose as an authoritative figure, such as a manager or IT support, in order to make the receiver of the call more inclined to provide them with the information they desire [5].

In order to implement the SEPTT Android application, it was essential to review all social engineering detection models that have been proposed in literature. Five such detection models were found and compared. It was also vital to analyse the various social engineering attack frameworks that have been proposed. It is important that the social engineering attack scenarios, generated for the experiment, adhere to a social engineering attack framework. In addition, social engineering attacks can be classified into three distinct categories, namely unidirectional communication, bidirectional communication and indirect communication. The differences between these categories have to be properly comprehended in order to classify the social engineering scenarios, generated for the experiment, into one of the three categories.

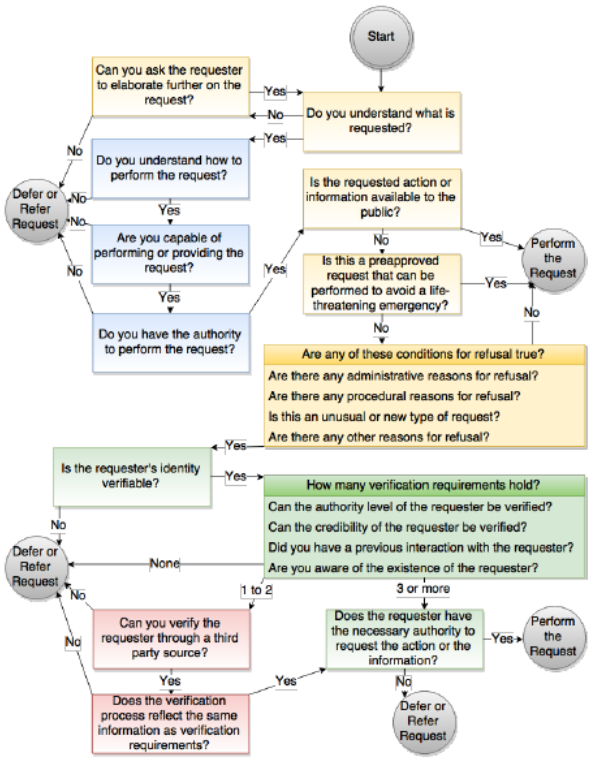


**Figure 1: Social Engineering Attack Detection Model version 2 (SEADMv2)**

### 2.1 Social Engineering Detection Models

Multiple social engineering detection models have been proposed in literature. These detection models include, the Social Engineering Attack Detection Model (SEADM) [1], the Social Engineering Attack Detection Model version 2 (SEADMv2) [12], detecting social engineering attacks using neural networks [16], the Social Engineering Defense Architecture (SEDA) [5] and detection of social engineering using natural language processing [17]. Table 1, provides a summary of the advantages and the disadvantages of each of these social engineering detection models.

The SEADMv2 was chosen to be implemented, in an Android application, and tested. Figure 1 provides an illustration of the SEADMv2. It illustrates the states that the user is required to progress through in order to reach a final state. There are two possible final states a user can end up in. One final state indicates that the user can trust the requester, i.e. *Perform the Request*. The other final state indicates that the user is the potential victim of a social engineering attack, i.e. *Defer or Refer Request*.

The colours used for the states in Figure 1 are to differentiate the different types of states supported by the model. Yellow states are request states and they deal with the request itself. Blue states are receiver states and they deal with whether an individual understands what is requested or not. The green states deal with the requester and any information that can be determined about the requester and the red states are third party states and refer to whether the requester can be verified using a third party [12].

In addition to having coloured states, the SEADMv2 can

| Detection Model Used: | Advantages: | Disadvantages: |
|---|---|---|
| SEADM | Modular design. | Requires user to determine own emotional state.<br>Only caters for bidirectional communication. |
| SEADMv2 | Colour codes to differentiate types of states.<br>More state transitions than the SEADM.<br>More modular design than the SEADM.<br>Caters for bidirectional, unidirectional and indirect communication. | No states to examine the emotional state of the user. |
| Social Engineering Detection using Neural Networks | Accurate at detecting attacks. | Never been tested in a real world scenario.<br>Tedious for the user to enter values into the input nodes. |
| SEDA | No user interaction required.<br>Prevents same social engineer targeting different employees. | Social engineer could trick the system by using voice recordings of the person they are imitating.<br>Only works for verbal social engineering attacks. |
| Social Engineering Detection using Natural Language Processing | No user interaction required.<br>Processes text rapidly.<br>Accurate at detecting attacks. | Only works for textual social engineering attacks. |

**Table 1: Comparison of Social Engineering Detection Models**

be used to detect both verbal and textual social engineering attacks. It can also be used for bidirectional communication, unidirectional communication and indirect communication social engineering attacks.

## 2.2 Social Engineering Attack Frameworks

A social engineering attack framework provides all the components required to generate a social engineering attack. In addition, a social engineering attack framework adds temporal data such as flow and time [15], making the scenario easier for the reader to relate to. It is for these reasons, that it was essential to use a social engineering attack framework when generating social engineering attack scenarios for the experiment. Various authors have proposed social engineering attack frameworks, each with different phases that make up an attack. Some of the proposed social engineering attack frameworks include, the Social Engineering Attack Cycle (SEAC) proposed by Mitnick [10], the Social Engineering Attack Phases proposed by Laribee [6], Harley's Social Engineering Attack Mechanism [4] and the Social Engineering Attack Framework (SEAF) proposed by Mouton [15].

It was decided to use the SEAF [15] to generate social engineering attack scenarios for the experiment. The SEAF improves on the SEAC [10] by including more phases to make generation of social engineering attacks more structured. The phases included in the SEAF are as follows, attack formulation, information gathering, preparation, develop a relationship, exploit the relationship and debrief.

In the attack formulation phase, the attacker identifies their target. This is usually based on the knowledge the target possesses or the information which the target has access to. During the information gathering phase, the attacker gathers as much information about their target as possible [14]. Social networks have made this phase particularly easy for social engineers, as people tend to share quite a substantial amount of personal information on social networks. During the preparation phase, the attacker analyses the information they have gathered about their target and formulates a scenario that would most likely result in the target giving the attacker the desired information. The next phase involves developing a relationship with the target, this is either achieved through technological means or by meeting the target in person and building a relationship. Once the attacker has established a relationship with their target, the next phase involves exploiting this relationship. In this phase, the attacker starts to probe the target for the information they require. The debrief phase is aimed at returning the target to a "stable" emotional state. The purpose of this phase is to reassure the target that they were not under attack and that everything is fine.

The SEAF was used when generating the social engineering scenarios for the experiment. It was essential to ensure that each scenario went through all the phases of the SEAF. This made the scenarios more believable and easier for the test subjects to relate to.

## 2.3 Classification of Social Engineering Attacks

The type of communication used in a social engineering attack is classified into one of three categories, namely unidirectional communication, bidirectional communication and indirect communication. It was important to understand each of these communication types, to ensure that at least one example of each communication type was included in the scenarios used for the experiment.

Unidirectional communication is a type of direct communication that occurs when the conversation is one-way only, i.e. from the attacker to the target [13]. Typical unidirectional communication social engineering scenarios include, when an attacker sends a paper letter to their victim with no return address. Phishing attacks are also considered to be unidirectional communication social engineering attacks.

Bidirectional communication is a type of direct commu-

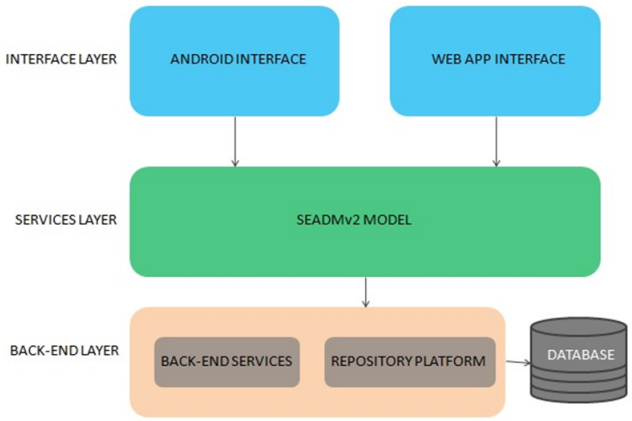


**Figure 2: High Level Overview of SEPTT System**

nication in which both parties participate in the conversation, i.e. the attacker and the target communicate with one another [13]. Examples of bidirectional communication include, when an email is sent form the attacker to the target and the target replies to the email or when the attacker phones the target and has a telephonic conversation with the target.

Indirect communication occurs when there is no actual communication between the attacker and the target. Indirect communication occurs through a third party medium [13]. An example of communication occurring through a third party medium, is when an attacker infects a USB flash drive and leaves it in a public place to be found by a random target.

## 3. SYSTEM DESIGN AND IMPLEMENTATION

Figure 2 illustrates a high level overview of the system that was built. The system was designed using a three-layered architecture. In a three-layered architecture, the system is divided into three distinct layers that each serve a well-defined purpose. This results in the system achieving faster responses and processing of users' requests [7]. The interface layer consists of both the Android application, implemented and tested by myself, as well as the web application, implemented and tested by Michael Pepper (my partner for this project), and it is the layer which the users will interact with. The services layer contains the logic of the system and it is where the SEADMv2 model will be implemented. This layer was tested thoroughly during development, since it is vital to the validity of the experiment. If this layer does not implement the SEADMv2 correctly, the entire experiment will be invalid. The back-end layer stores user analytical data obtained from users using the models through either the web or Android applications. It also provides an interface for an administrator to update the SEADMv2 model, should any changes need to be made to the model in the future.

Figure 3 is a screenshot of the Android application that was developed and used for the experiment. The application was developed using the Rapid Application Development (RAD) methodology. This methodology uses minimal planning in favour of rapid prototyping [9]. Using this methodology ensured that a working prototype of the application was always available. The visual design and user interface of the application went through numerous iterations in order to optimise the user experience. In the early iterations of the application, the entire interface was completely textual with no colours or any indication of progression through the SEADMv2. Figure 3 is a screenshot of the final version of the application and it is clear that improvements were made to the user interface. The *Yes* and *No* buttons were made green and red respectively. This change made it easier for users to differentiate the buttons, without necessarily reading the text on the buttons. A state progression bar was added to the top of the interface. This progression bar gives the users a sense of how far they are from reaching a final state. In addition, it gives the user a sense of progression, since new chevrons are highlighted as the user answers questions. A, readily available, help button was added to the bottom of the interface. This button uses a question mark icon to indicate that it is intended to help the user. Upon touching the button, the user is provided with a more thorough description of the question currently displayed. Haptic feedback (vibration) was also implemented in the application. Upon reaching a final state, the user's device vibrates for 100ms drawing the user's attention to the fact that they have reached a final state. The application was completed on 3 October 2016. It was released, as a free application, on the Google Play Store, listed as Social Engineering Training[3], with the intention that it will be used as a cybersecurity educational tool.

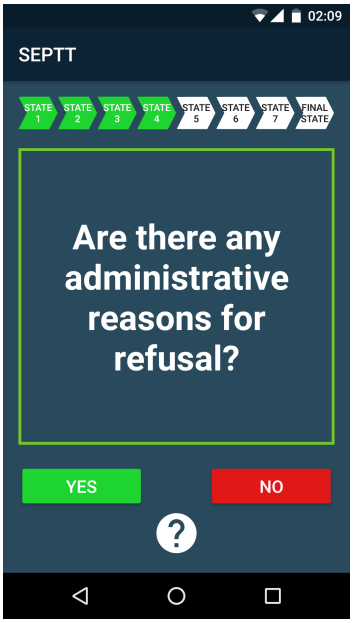


**Figure 3: Screenshot of the SEPTT Android Application**

[3]https://play.google.com/store/apps/details?id=za.co.social_engineer.www.socialengineer

# 4. METHODOLOGY

## 4.1 Experiment Design

Before describing how the experiment was conducted, a few details to note about the experiment, are as follows. This experiment consisted of a single factor (a controlled independent variable) and it had a single measure (dependent variable which is measured). The factor was the use of the model, which has two levels (without model and with model). This factor was a within-subjects factor, meaning that each subject is tested at each level of the factor. The measure of this experiment was the number of errors made by the subjects when answering the scenarios with and without the aid of the model.

This experiment was performed on 20 subjects. Since the only factor was a within-subjects factor, all 20 of the subjects were required to use both levels of the factor. This experiment contained both a fixed effect and a random effect. The fixed effect was the use of the model and the random effect was the subjects. The levels of the subject (age, gender, faculty of study, etc) do not really influence the outcome of this experiment and therefore it is classified as a random effect.

Initially, the subjects were planned to be recruited by means of sending out a bulk email to the entire University of Cape Town (UCT) community. This email would have requested that any interested students sign up and take part in this experiment. I was hoping to recruit approximately 50 subjects using the convenience random sampling technique. Convenience random sampling is the basic random sampling technique where a group of subjects are selected, for a study, from a larger group[8]. However, due to the shutdown that occurred on the UCT campus from 16 September 2016 until 17 October 2016, alternative plans had to be made to recruit subjects. I instead opted to use a snowball sampling technique, which is a non-probability sampling technique where study subjects recruit future subjects from among their acquaintances[2]. I initially approached a small group of my peers and asked them to, voluntarily, be subjects in my experiment. I also asked that they suggest one of their friends that I could approach and I continued doing this until I managed to get 20 subjects to complete the experiment. Due to the slow nature of snowball sampling, as opposed to convenience random sampling, the amount of subjects was reduced, from the initial goal of 50 subjects, to 20 subjects. Using 20 subjects was enough to obtain sufficient results, which could be analysed to test the hypothesis.

This experiment was conducted by giving each subject a total of 10 different scenarios. Eight of the scenarios were malicious social engineering attacks and the remaining two scenarios were harmless scenarios. The scenarios were presented to each subject in a completely random order to eliminate any ordering effects. Ordering effects refer to the differences in research participants' responses that result from the order in which experimental materials are presented to them[3]. Each scenario had four possible answers associated with it and the order in which these answers were provided to the subjects was also randomised. An electronic questionnaire, which was created using Google Forms, was used. Google Forms made it easy to randomise the order in which the scenarios were provided to the subjects, as well as randomise the order in which the answers to each scenario were provided. Google Forms also made it easy to capture the results from the experiment.

Every subject was given each of the 10 scenarios twice. The first time they were required to select an answer using their own intuition, without the aid of the model. The second time they were presented with the same 10 scenarios, but this time they were also provided with the Android implementation of the SEADMv2 to assist them in selecting the most correct answer. The order in which the subjects answered the scenarios was fixed, i.e. the subjects always first answered the scenarios without the aid of the model first and then with the aid of the model. The purpose of this, was to eliminate any lasting effects the use of the model could have on the subjects which would influence their answers without the use of the model.

Even though the entire questionnaire was available available online[4] and the Android application was published to the Google Play Store, I was still present during each experiment. This was to ensure the subject understood exactly what was expected of them, as well as to answer any questions the subject might have while doing the experiment. The measure of this experiment was the total number of errors made by each subject. This was measured by comparing a subjects answers to the correct answers. If the subject got an answer incorrect, that would be counted as an error. It stands to reason that the less errors a subject made, the better they did at correctly identifying the social engineering attacks.

Ethical clearance was applied for and obtained from the UCT Science Faculty Research Ethics Committee. Each participant was also required to sign a consent form before partaking in the experiment, agreeing that they are voluntarily partaking in the experiment and that their results will be kept completely anonymous and only used for the purpose of this experiment.

## 4.2 Summary of Scenarios Used

The scenarios, provided to the subjects in the experiment, were generated using the SEAF. An effort was made to ensure that there were scenarios from each of the three social engineering attack communication categories, namely bidirectional communication, unidirectional communication and indirect communication. In Section 2, it was specified that the only two possible final states provided by the SEADMv2 are *Perform the Request* or *Defer or Refer Request*. *Performing the Request*, could either mean perform the request without asking questions or perform the request with caution. Similarly, *Deferring or Referring the Request* could either mean to not perform the request at all or refer the request to someone else. Therefore, it was decided that there would be two possible correct answers for each scenario. A short description of the scenarios, used for the experiment, are provided below.

### 4.2.1 Bidirectional Communication Scenarios

Four of the ten scenarios provided to the subjects, during the experiment, were bidirectional communication scenarios. All four of the bidirectional communication scenarios were malicious social engineering attacks. Therefore, the correct answers for these scenarios was either to refer the attacker's request to someone who has the authority to handle such a request or to refuse to help the attacker entirely. Two of

[4]Questionnaire available at https://goo.gl/forms/H0Z8PPnJR72rqJim1

the bidirectional scenarios were phone calls from an attacker who impersonated someone who has the authority to obtain the requested information from the victim. The phone call in one of the scenarios took place over the Christmas holiday. The attacker was targeting the victim's emotional state over this period, since over the holidays people do not want to be bothered with work related problems and will give away sensitive information, less reluctantly, just to end the phone call. Another of the bidirectional communication scenarios was a conversation over a popular instant messaging platform, Facebook Messenger. The attacker and the victim have never met in person, so there is no way the victim can be sure the attacker is who they say they are. The attacker first builds a virtual relationship with the victim and then exploits this relationship requesting sensitive information, namely the victim's Facebook login credentials. The last bidirectional communication scenario, used in the experiment, was an attacker posing as a student and asking another student, the victim, to swipe him into the computer lab, since he had lost his student card.

#### 4.2.2 *Unidirectional Communication Scenarios*

Five of the ten scenarios, used for the experiment, were unidirectional communication scenarios. Two of these scenarios were harmless scenarios, while the remaining three were malicious social engineering attack scenarios. The two harmless scenarios both entailed an email sent from one party to another. In the one scenario, a recruiter from LinkedIn sent an email and in the other scenario a new employee at an accounting firm sent an email. In both these scenarios, the two people had not met in person. However, both emails came from domains owned by the two companies that the recruiter and the employee at the accounting firm worked for respectively. This is already a strong indication that the people sending the emails are who they say they are. Another indication was the signatures of the emails contained both the senders' positions at their companies as well as their contact details. Therefore, it would be relatively easy to verify that both the people who sent the emails are who they say they are.

The three unidirectional malicious social engineering attack scenarios entailed emails sent from unverifiable domains as wells as a guest lecturer requesting sensitive information. In the one scenario an email was sent from a lecturer to a student, however, the lecturer's email was not sent from the typical Vula (the online teaching system used at UCT) announcement system and the domain of the lecturer's email address was not the university's domain. As a result, there is no way for the student to verify that this email was in fact sent from their lecturer. Another of the unidirectional malicious scenarios, involved a student emailing their incomplete assignment to another student, requesting some assistance. Both students had never met in person and there was, once again, no way to verify the person sending the email. The last unidirectional malicious social engineering attack, involved a guest lecturer requesting students to complete forms. However, these forms required the students to provide sensitive information, such as their identity numbers. Since there is no way of verifying the guest lecturer's identity, the correct answer was to refuse to complete that section of the form (deferring the request) or telling the guest lecturer to obtain that information from the university's records (referring the request).

#### 4.2.3 *Indirect Communication Scenarios*

There was only one example of an indirect communication scenario, in the ten scenarios used for the experiment. Indirect communication scenarios are more difficult to generate, due to the fact that the attacker and the victim do not have a direct channel of communication, but rather communicate through a third-party medium. The indirect communication scenario, used in the experiment, involved the victim picking up a USB flash drive, lying unattended on a university campus. The USB flash drive is not labelled with any name or contact details, so there is no way for the victim to identify the owner of the USB flash drive, without inserting the USB flash drive into a computer and inspecting the files. The SEADMv2 has a question that asks the user whether or not they have the authority to perform the action. Since you do not have the authority to plug someone else's USB flash drive into your computer, without their permission, the correct action to take would be to leave the flash drive where you found it (deferring the request) or taking the flash drive to lost and found (referring the request).

## 5. RESULTS AND DISCUSSION

The results obtained from the 20 subjects was exported as a CSV file[5] from Google Forms. This CSV file was rather verbose and as a result, a Python script was written to extract the required information from it. Four different sets of results were extracted from the raw data, namely how the subjects' answers changed when they answered the scenarios with the aid of the model opposed to when they answered the scenarios without the aid of the model. How the number of errors made without the aid of the model compare to the number of errors made with the aid of the model. How the number of errors made with and without the aid of the model for malicious social engineering attacks compare to the number of errors made with and without the aid of the model for harmless social engineering attacks. Lastly, how the number of errors made with and without the aid of the model for unidirectional communication scenarios, bidirectional communication scenarios and indirect communication scenarios compare.

### 5.1 Comparison of Changes to Answers

The possible changes to answers were grouped into four categories, namely a subject's answer changes from incorrect without the model to correct with the aid of the model, a subject's answer changes from correct without the model to incorrect with the aid of the model, a subject's answer remains incorrect with and without the aid of the model, and a subject's answer remains correct with and without the aid of the model. It should be obvious, that if the subject's answer changes from incorrect without the model to correct with the aid of the model or if their answer remains correct with and without the model, this is seen as positive and the model is serving its intended purpose. Similarly, if a subject's answer changes from correct without the model to incorrect with the aid of the model or if their answer remains incorrect with and without the model, this is seen as negative and the model is not serving its intended purpose. To obtain these results, the raw data CSV file was manipulated by a

[5]Raw data CSV file available at http://www.social-engineer.co.za/data/SEPTTOutputOriginalAnswers.csv

| Description of Change: | Number of Changes: | Percentage: |
|---|---|---|
| Incorrect to correct | 57 | 28.5% |
| Correct to incorrect | 25 | 12.5% |
| Remained correct | 80 | 40% |
| Remained incorrect | 38 | 19% |

**Table 2: Changes to Answers**

Python script to produce a new CSV file[6]. This CSV file contained columns that tally the number of times a subject's answer changed from incorrect to correct or vice versa or remained either correct or incorrect.

Table 2 tabulates the results obtained, describing the number of changes in each category as well as the percentage associated with each category. It is clear that the use of the model had a significant affect in changing the answers positively. The use of the model improved the subject's answer from incorrect to correct 28.5% of the time. The model also kept the subject's answer correct 40% of the time. This means the model served its intended purpose more than two thirds of the time (68.5%). The remaining 31.5% of the time the model did not serve its intended purpose. However, this is a much smaller fraction of the time, compared to when the model did serve its intended purpose. The times when the model failed to serve its purpose can also be attributed to the subjects not understanding the wording used in the model and subjects getting tired during the experiment (both these factors were observed while watching subjects complete the experiment).

## 5.2 Comparison of Total Errors Made

The comparison of the total errors made with and without the aid of the model, is the measure that gives the best indication of whether the model is serving its intended purpose or not. Another Python script was used to tally the total errors made by each subject across all the scenarios and create a CSV file[7] of the results. An error, in this context, refers to when a subject chose an incorrect answer for a particular scenario. These results were analysed to test for statistical significance. For this experiment, the significance threshold was set at 0.05. This means that any p-value obtained that is less than 0.05 is statistically significant, any other p-value is not statistically significant.

First the distribution of data needed to be determined, since this would give an indication of which statistical tests could be used to analyse the data. It was suspected that the data would follow a Poisson distribution, since errors and error rates usually tend to follow this sort of distribution. To determine the distribution of the data, a Chi-Squared goodness of fit test was run on both the number of errors with and without the aid of the model. The Chi-Squared goodness of fit indicated that the errors made with the aid of the model did follow a Poisson distribution, since $p > 0.05$ and therefore it does not significantly deviate from a Poisson distribution. However, running the same test on the data without the aid of the model, produced $p < 0.05$, which means the data does significantly deviate from a Poisson distribution and therefore it does not follow a Poisson distribution.

[6]CSV file available at http://www.social-engineer.co.za/data/SEPTTSummaryAnalysis.csv

[7]CSV file available at http://www.social-engineer.co.za/data/SEPTTErrorRate.csv

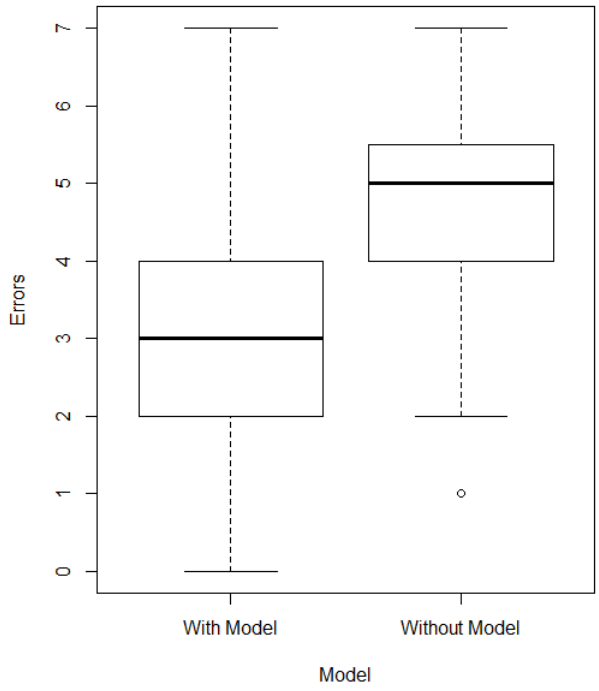


**Figure 4: Box plots of errors with and without the model**

The two data sets did not both follow the same distribution and therefore a non-parametric statistical test had to be used to test for significance. This experiment contained one within-subjects factor with exactly two levels (with model and without model) and, as a result, a Wilcoxon signed-rank test was run to determine if there is a significant difference between the results. It was found, that there is a significant difference between the number of errors made with the aid of the model and the number of errors made without the model ($p < 0.05$). The box plots of the errors made with and without the aid of the model, illustrated in Figure 4, shows this significant difference clearly. From Figure 4, it is clear that the first quartile of the number of errors made without the model is equal to the third quartile of the number errors made with the model. This strengthens the result obtained with the Wilcoxon signed-rank test. It is clear that the use of an Android application that implements the SEADMv2 model does significantly decrease the number of errors made when identifying social engineering attacks.

## 5.3 Comparison of Errors Made Based on Type of Scenario

As an additional comparison, it was decided to separate the scenarios into two categories, namely attack and harmless scenarios. Attack scenarios are scenarios in which an attacker is trying to manipulate the victim with malicious intent, while harmless scenarios are those scenarios in which there is no malicious intent. This comparison was performed to provide insight into whether the SEADMv2 is actually reducing the number of errors in malicious social engineering attacks or if the reduction of errors is only occurring in harmless scenarios. A Python script was again used to separate the scenarios into attack scenarios and harmless scenarios

and two new CSV data files[8] were created for analysis.

Similar to the total errors, the distribution of the data had to first be determined to provide an indication of which statistical tests can be used for analysis. After running the Chi-squared goodness of fit test on the results with the model and without the model for malicious attack scenarios. It was found that the the errors with the aid of the model followed a Poisson distribution ($p > 0.05$), while the errors without the aid of the model did not follow a Poisson distribution ($p < 0.05$). Since the data does not follow the same distribution and there is one within-subjects factor with exactly two levels (with model and without model), a Wilcoxon signed-rank test was the most appropriate statistical test to use. Since this is a post hoc test, an adjustment needed to be made to the original significance threshold to account for the additional hypotheses that were introduced. This adjustment is known as the Bonferroni correction and it is introduced to avoid the null hypothesis (a default hypothesis that claims there is no relationship between the two measurements) being rejected incorrectly. The Bonferroni correction adjusts the original significance threshold as follows, $\alpha/\kappa$, where $\alpha$ is the original significance threshold and $\kappa$ is the number of hypotheses introduced in the post hoc test. In this case there were two hypotheses introduced, namely harmless scenarios and attack scenarios. Using the Bonferroni correction on the original significance threshold, we get an adjusted significance threshold of $0.05/2 = 0.025$. After running the Wilcoxon signed-rank test, it was found that there is a significant difference between the number of errors made with the aid of the model and the number of errors made without the model for attack scenarios ($p < 0.025$).

The same procedure was followed for the errors with and without the model for harmless scenarios. First the distribution was determined using a Chi-squared goodness of fit test. The errors with the aid of the model followed a Poisson distribution ($p > 0.05$), while the errors without the aid of the model did not follow a Poisson distribution ($p < 0.05$). Once again, a Wilcoxon signed-rank test was used to determine statistical significance. However, it was found that there is no significant difference between the number of errors made with the aid of the model and the number of errors made without the model for harmless scenarios ($p > 0.025$). The box plots of the errors made with and without the aid of the model for harmless scenarios, in Figure 5, supports this result. In Figure 5, the medians are identical, this is a strong indicator of no significant difference.

Since there was a significant difference between the number of errors with and without the model overall and there was also a significant difference between the number of errors with and without the model for attack scenarios. It stands to reason, that the SEADMv2 is helping subjects detect malicious social engineering attacks well, but not having much of an effect on the way subjects react to harmless scenarios. This is acceptable, since the purpose of the SEADMv2 is to prevent users from falling victim to malicious social engineering attacks, which it has proven to do correctly. Figure 6 is a stacked graph, which clearly illustrates how the number of errors made in harmless scenarios with and without the model is practically the same. Figure 6 also illustrates that there is a big difference, overall, between the number of errors made with and without the model and this difference is clearly caused by the attack scenarios and not the harmless scenarios.

[8]CSV files available at http://www.social-engineer.co.za/data/SEPTTErrorRateAttack.csv & http://www.social-engineer.co.za/data/SEPTTErrorRateHarmless.csv

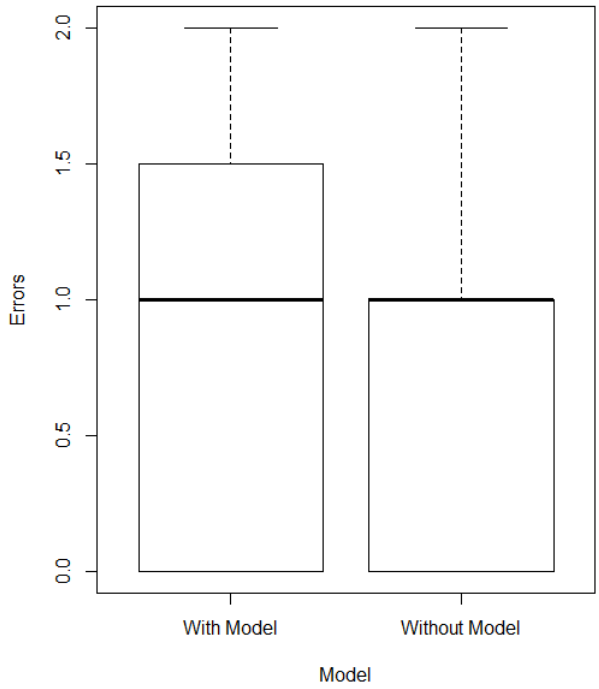


**Figure 5: Box plots of errors with and without the model for harmless scenarios**

## 5.4 Comparison of Errors Made Based on Communication Category of Scenario

The last comparison performed was to test whether the SEADMv2 works better for a given communication category of scenarios. Similar to before, a Python script was used to separate the scenarios into bidirectional, unidirectional and indirect communication and three new CSV data files[9] were created for analysis. Since this is another post hoc test, the Bonferroni correction had to be applied to the original significance threshold. This time there were three hypotheses introduced and the Bonferroni correction adjusted the significance threshold to be $0.05/3 = 0.017$. This adjusted significance threshold was used to determine statistical significance for the three communication categories described below.

### *5.4.1 Bidirectional Communication Scenarios*

The errors with and without the model for the bidirectional communication scenarios were first analysed to see if a common distribution could be found using the Chi-squared goodness of fit test. The number of errors with the model did fit a Poisson distribution ($p > 0.05$), however, the number of errors without the aid of the model did not fit a Poisson distribution ($p < 0.05$). Therefore, the Wilcoxon signed-rank test would have to be used to test for statistical significance

[9]CSV files available at http://www.social-engineer.co.za/data/SEPTTErrorRateBi.csv; http://www.social-engineer.co.za/data/SEPTTErrorRateUni.csv & http://www.social-engineer.co.za/data/SEPTTErrorRateIndirect.csv

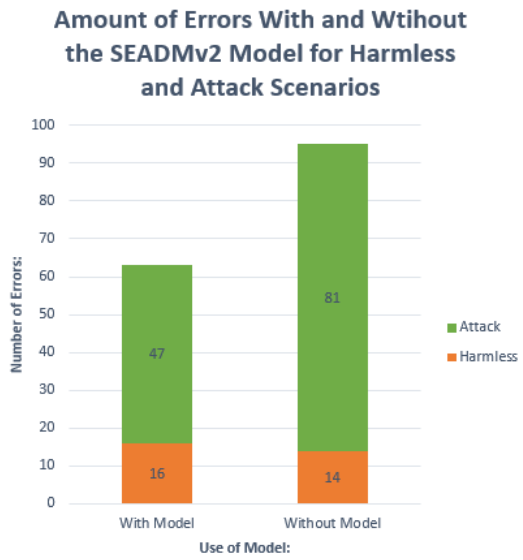


**Figure 6: Stacked bar graph showing the number of errors with and without the model for harmless and attack scenarios**

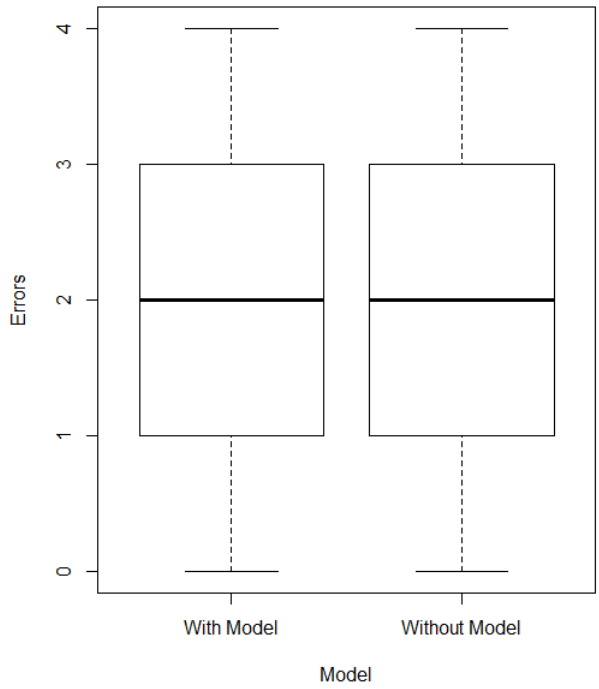


**Figure 7: Box plots of errors with and without the model for unidirectional communication scenarios**

between the number of errors with and without the model. There was found to be a significant difference between the number of errors with and without the model for bidirectional communication scenarios ($p < 0.017$). This indicates that the model serves its intended purpose and significantly reduces the number of errors for bidirectional communication scenarios.

### 5.4.2 *Unidirectional Communication Scenarios*

Similar to bidirectional communication, the number of errors with and without the model for the unidirectional communication scenarios were first analysed, to see if a common distribution could be found using the Chi-squared goodness of fit test. However, in the case of the unidirectional communication scenarios a rather interesting observation was made. The number of errors with and without the model were identically distributed. This is illustrated in Figure 7, which shows the box plots for number of errors with and without the use of the model for unidirectional communication scenarios. Since the distributions of the number of errors with and without the model are identical, it would be unnecessary to run any statistical tests. It is quite clear that there is no difference between the number of errors with and without the model for unidirectional scenarios. Therefore, the SEADMv2 does not serve its intended purpose for unidirectional scenarios.

### 5.4.3 *Indirect Communication Scenarios*

Lastly, the number of errors with and without the model, for indirect communication scenarios, were analysed to see if a common distribution could be found using the Chi-squared goodness of fit test. The number of errors with the model did fit a Poisson distribution ($p > 0.05$), however, the number of errors without the aid of the model did not fit a Poisson distribution ($p < 0.05$). Therefore the Wilcoxon signed-rank test would have to be used to test for statistical significance between the number of errors with and without the model for indirect communication scenarios. There was found to be a significant difference between the number of errors with and without the model for indirect communication scenarios ($p < 0.017$). This indicates that the model serves its intended purpose and significantly reduces the number of errors for indirect communication scenarios.

## 6. CONCLUSIONS

The results analysed in Section 5 can be used to draw vital conclusions. The most important result obtained, is that an Android implementation of the SEADMv2 does significantly reduce the number of errors made by subjects when identifying social engineering scenarios, opposed to when no model is used. This answers the research question of this paper and agrees with the hypothesis.

The model was also shown to be statistically significant at reducing the number of errors made for malicious attack scenarios. However, the model did not have any significant affect on reducing the number of errors made for harmless scenarios. The model was also statistically significant at reducing the number of errors made for both bidirectional communication and indirect communication scenarios. However, the model did not have any significant effect on the number of errors made for unidirectional communication scenarios. Therefore, it can be concluded that an Android implementation of the SEADMv2 is effective at reducing the probability of a subject falling victim to malicious attack scenarios, bidirectional communication scenarios and indirect communication scenarios. However, an Android implementation of the SEADMv2 is not effective at reducing the probability of a subject falling victim to unidirectional communication scenarios and harmless scenarios.

The Android implementation of the SEADMv2 was also

good at positively influencing the subjects' answers. It changed the subjects' answers from incorrect to correct 28.5% of the time and kept the subjects' answers correct 40% of the time. This means 68.5% of the time, the model influenced the subjects positively. This is more than the 12.5% of the time that the model changed the subjects' answers from correct to incorrect or the 19% of the time when the model kept the subjects' answers incorrect. This leads us to the conclusion that an Android implementation of the SEADMv2 influences a subjects' answers positively more often than when it influences a subjects' answers negatively.

On the whole, the Android implementation of the SEADMv2 did have a statistically significant positive influence on the detection of social engineering attacks. Therefore,it can be concluded that the Android implementation of the SEADMv2 does reduce the probability of a subject falling victim to a social engineering attack.

# 7. FUTURE WORK

This paper described the first experiment with the SEADMv2 on actual subjects. As a result, a few aspects of the model, which may have worked well in theory, were not as effective in reality. The most notable aspect is the wording of some of the questions in the model. Numerous subjects either asked for clarification on what a question meant or made use of the help provided in the SEPTT Android application, to clarify some of the questions in the model. The wording of some of the questions should be revised in future iterations of the SEADM. In addition, the model was found to not decrease the number of errors for unidirectional communication scenarios and harmless scenarios. This should definitely be investigated and future iterations of the SEADM should also significantly decrease the number of errors for unidirectional communication scenarios and harmless scenarios.

The Android application that was developed for this experiment is fully functional and was tested thoroughly. However, an aspect that bothered some subjects during the experiment, is that the progress bar, visible at the top of the interface, does not accurately indicate a user's distance from reaching a final state. For example, if a user had to answer *No* to the first two questions, the progress bar would jump from *state 2* to the *final state*. This is due to the way the underlying model is designed. Future work on the Android application could involve writing an algorithm to enable the state progress bar to more accurately indicate the user's distance from reaching a final state.

# 8. ACKNOWLEDGEMENTS

I would like to thank my supervisors Prof. Tommie Meyer and Francois Mouton for all their help and guidance throughout this project. I doubt this project would have been as successful as it was, without their input. I would also like to express my gratitude towards Dr Brian DeRenzi, who helped me tremendously, by suggesting appropriate statistical tests that I could use to analyse my results.

I would also like to thank my project partner, Michael Pepper. Although we worked on separate parts of this project and no collaboration was required, his encouragement and help contributed to the successful completion of this project.

# 9. REFERENCES

[1] Bezuidenhout, M., Mouton, F., and Venter, H. S. Social engineering attack detection model: Seadm. In *Information Security for South Africa (ISSA), 2010* (2010), IEEE, pp. 1–8.

[2] Biernacki, P., and Waldorf, D. Snowball sampling: Problems and techniques of chain referral sampling. *Sociological methods & research 10*, 2 (1981), 141–163.

[3] Gass, S. M., and Torres, M. J. A. Attention when?: An investigation of the ordering effect of input and interaction. *Studies in Second Language Acquisition 27*, 01 (2005), 1–31.

[4] Harley, D. Re-floating the titanic: Dealing with social engineering attacks. *London: EICAR* (1998), 13.

[5] Hoeschele, M., and Rogers, M. Detecting social engineering. In *Advances in Digital Forensics*. Springer, 2005, pp. 67–77.

[6] Laribee, L. *Development of methodical social engineering taxonomy project*. PhD thesis, Monterey, California. Naval Postgraduate School, 2006.

[7] Liu, D. T., and Xu, X. W. A review of web-based product data management systems. *Computers in industry 44*, 3 (2001), 251–262.

[8] Marshall, M. N. Sampling for qualitative research. *Family practice 13*, 6 (1996), 522–526.

[9] Martin, J. *Rapid application development*, vol. 8. Macmillan New York, 1991.

[10] Mitnick, K. D., and Simon, W. L. *The art of deception: Controlling the human element of security*. John Wiley & Sons, 2011.

[11] Mouton, F., Leenen, L., Malan, M. M., and Venter, H. Towards an ontological model defining the social engineering domain. In *ICT and Society*. Springer, 2014, pp. 266–279.

[12] Mouton, F., Leenen, L., and Venter, H. Social engineering attack detection model: Seadmv2. In *2015 International Conference on Cyberworlds (CW)* (2015), IEEE, pp. 216–223.

[13] Mouton, F., Leenen, L., and Venter, H. Social engineering attack examples, templates and scenarios. *Computers & Security 59* (2016), 186–209.

[14] Mouton, F., Malan, M. M., Kimppa, K. K., and Venter, H. S. Necessity for ethics in social engineering research. *Computers & Security 55* (2015), 114–127.

[15] Mouton, F., Malan, M. M., Leenen, L., and Venter, H. S. Social engineering attack framework. In *Information Security for South Africa (ISSA), 2014* (2014), IEEE, pp. 1–9.

[16] Sandouka, H., Cullen, A., and Mann, I. Social engineering detection using neural networks. In *CyberWorlds, 2009. CW'09. International Conference on* (2009), IEEE, pp. 273–278.

[17] Sawa, Y., Bhakta, R., Harris, I. G., and Hadnagy, C. Detection of social engineering attacks through natural language processing of conversations. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)* (2016), IEEE, pp. 262–265.

[18] Workman, M. Gaining access with social engineering: An empirical study of the threat.

*Information Systems Security 16*, 6 (2007), 315–331.