FINITE STATE MACHINE FOR THE SOCIAL ENGINEERING ATTACK DETECTION MODEL: SEADM

Francois Mouton* , **Alastair Nottingham†** , **Louise Leenen‡** and **H.S Venter§**

* *Defence Peace Safety & Security, Council for Scientific and Industrial Research, Pretoria, South Africa E-mail: moutonf@gmail.com*

† *E-mail: anottingham@gmail.com*

‡ *E-mail: lleenen@csir.co.za*

§ *Department of Computer Science, University of Pretoria, Pretoria, South Africa E-mail: hventer@cs.up.ac.za*

Abstract: Information security is a fast-growing discipline, and relies on continued improvement of security measures to protect sensitive information. Human operators are one of the weakest links in the security chain as they are highly susceptible to manipulation. A social engineering attack targets this weakness by using various manipulation techniques to elicit individuals to perform sensitive requests. The field of social engineering is still in its infancy with respect to formal definitions, attack frameworks, and examples of attacks and detection models. In order to formally address social engineering in a broad context, this paper proposes the underlying abstract finite state machine of the Social Engineering Attack Detection Model (SEADM). The model has been shown to successfully thwart social engineering attacks utilising either bidirectional communication, unidirectional communication or indirect communication. Proposing and exploring the underlying finite state machine of the model allows one to have a clearer overview of the mental processing performed within the model. While the current model provides a general procedural template for implementing detection mechanisms for social engineering attacks, the finite state machine provides a more abstract and extensible model that highlights the inter-connections between task categories associated with different scenarios. The finite state machine is intended to help facilitate the incorporation of organisation specific extensions by grouping similar activities into distinct categories, subdivided into one or more states. The finite state machine is then verified by applying it to representative social engineering attack scenarios from all three streams of possible communication. This verifies that all the capabilities of the SEADM are kept in tact, whilst being improved, by the proposed finite state machine.

Key words: Bidirectional Communication, Finite State Machine, Indirect Communication, Social Engineering, Social Engineering Attack Examples, Social Engineering Attack Detection Model, Social Engineering Attack Framework, Unidirectional Communication.

1. INTRODUCTION

Protection of sensitive information is of vital importance to organisations and governments, and the development of measures to counter illegal access to such information is an area that continues to receive increasing attention. Organisations and governments have a vested interest in securing sensitive information and the trust of clients or citizens. Technology on its own is not a sufficient safeguard against information theft; staff members — often the weak link in an information security system — can be influenced or manipulated to divulge sensitive information that allows unauthorised individuals to gain access to protected systems.

The 'art' of influencing people to divulge sensitive information is known as social engineering, and the process of doing so is known as a social engineering attack (SEA). There are various definitions of social engineering, and a number of different models of social engineering attacks exist [1–5]. The authors of this paper considered different definitions of social engineering and social engineering attack taxonomies in a previous paper, *Towards an Ontological Model Defining the Social Engineering Domain* [6], and formulated a definition for both social engineering and a social engineering attack. In addition, the authors proposed an ontological model for a social engineering attack and defined social engineering as "the science of using social interaction as a means to persuade an individual or an organisation to comply with a specific request from an attacker where either the social interaction, the persuasion, or the request involves a computer-related entity" [6].

As clearly stated by various authors [7–10], the human component is one of the most vulnerable elements within security systems. Unfortunately it is the tendency towards cooperation and helpfulness in human nature that make people vulnerable to the techniques used by social engineers, as social engineering attacks exploit various psychological vulnerabilities to manipulate the individual to disclose the requested information [7, 10]. It is also the case that more and more individuals are exposed to electronic computing devices as the costs of these devices are decreasing drastically. Electronic computing devices have become significantly more affordable during the past few years and due to this nearly everyone has access to these devices. This provides the social engineer with more

Based on: "Underlying Finite State Machine for the Social Engineering Attack Detection Model", by F. Mouton, A. Nottingham, L. Leenen and H.S. Venter which appeared in the Proceedings of Information Security South African (ISSA) 2017, Johannesburg, 16 & 17 August 2017.

victims to target using skillfully crafted social engineering attacks.

The authors have previously performed research within the field of social engineering that focused on formalising and expanding upon concepts in the field. This research included proposing a social engineering attack framework, providing social engineering attack examples, considering ethical questions relating to social engineering, and both developing and revising the Social Engineering Attack Detection Model (SEADM) [11–14]. The previous iteration of the SEADM focused on covering all three different types of communication mediums for social engineering attacks [14]. Whilst using the SEADM to determine whether it is effective to detect social engineering attacks, it was noted that each set of questions focuses on a specific context. Also, the current iteration of the SEADM indicates that additional questions can be added to the model to address different implementation environments, but does not explicitly state how this should be done.

This paper focuses on addressing this problem by formalising the latest iteration of the SEADM into an abstracted deterministic finite state automata. The authors are not aware of similar approaches by other researchers. In its original form, SEADM was constructed as a non-deterministic flow chart that relied on general, qualitative sub-procedures to provide a model for detecting social engineering attacks. While effective as a procedure for reducing risk, the model made no provision and provided no guidance on how additional actions relevant in specific contexts and domains could be included, and at what points in the model these inclusions should be placed. Due to the inclusion of cycles in the SEADM model, the process was also non-deterministic, which added additional and unnecessary complexity in implementing the process.

This research aims to improve the extensibility of the SEADM, and to reduce its implementation complexity by restructuring the process to be cycle-free and deterministic. The extensibility of the model is addressed by replacing the qualitative sub-procedures with generalised states that better define the role of each sub-process, while treating questions posed in each state as general examples that may be expanded or removed, and not as a definitive collection of necessary queries. Organising the model as a finite set of generalised states provides a more concise representation of the process that encapsulates the broad set of questions into distinct units of related, deterministic, context specific states. This adjustment is intended to improve extensibilty, while simultaneously reducing the complexity of implementing the model as an organisational process or in software.

The remainder of the paper is structured as follows. Section II provides background information on the previous social engineering model and discusses the authors' previous work. Section III proposes the underlying deterministic finite state machine of the SEADM. Section IV provides a discussion on how each of the states were derived from the SEADM. Section V introduces the reader to social engineering attack templates, and evaluates the improved version of the SEADM against these templates. Section VI concludes the paper.

2. PREVIOUS ITERATIONS OF THE SOCIAL ENGINEERING ATTACK DETECTION MODEL

Many models and taxonomies have been proposed for social engineering attacks [1–6, 11]. The authors' ontological model depicts that a social engineering attack "employs either direct communication or indirect communication, and has a social engineer, a target, a medium, a goal, one or more compliance principles and one or more techniques" [6]. The ontological model clearly splits social engineering into three categories, namely bidirectional communication, unidirectional communication and indirect communication.

The initial SEADM was designed to cater specifically for social engineering attacks utilising bidirectional communication such as a call centre environment [13, 15]. This research was the first attempt to develop a detection model for social engineering attacks, as at the time of publishing the article there was still only limited research available in this field. Most of the research in this domain still centres around the training of users [7, 16, 17]. During the revision of the SEADM, the steps have been generalised to cater for all three communication categories, namely bidirectional communication, unidirectional communication and indirect communication. It has also been shown that the model is effective in detecting social engineering attacks by testing the model against known social engineering attack examples [14]. The previous iteration of the SEADM is depicted in Figure 1.

The following section discusses the representation of the SEADM as an abstracted finite state machine and provides a short discussion on each of the states.

3. UNDERLYING FINITE STATE MACHINE OF THE SEADM

A finite state machine (also known as finite state automaton) is an imaginary machine that embodies the idea of a sequential circuit. It has a finite set of states with a start state and accepting states, and a set of state transitions [18]. Finite state machines are commonly employed in the design and implementation of modern software and electronics, and range from simple and highly abstract models of computation or processing to complex and concrete executable mechanisms and physical circuitry.

Finite state machines can be deterministic or non-deterministic. A deterministic machine has exactly one path for every input-state pair. In a non-deterministic machine there may be multiple valid transitions for every input-state pair, and the chosen transition is not defined; any transition can be followed. A deterministic finite

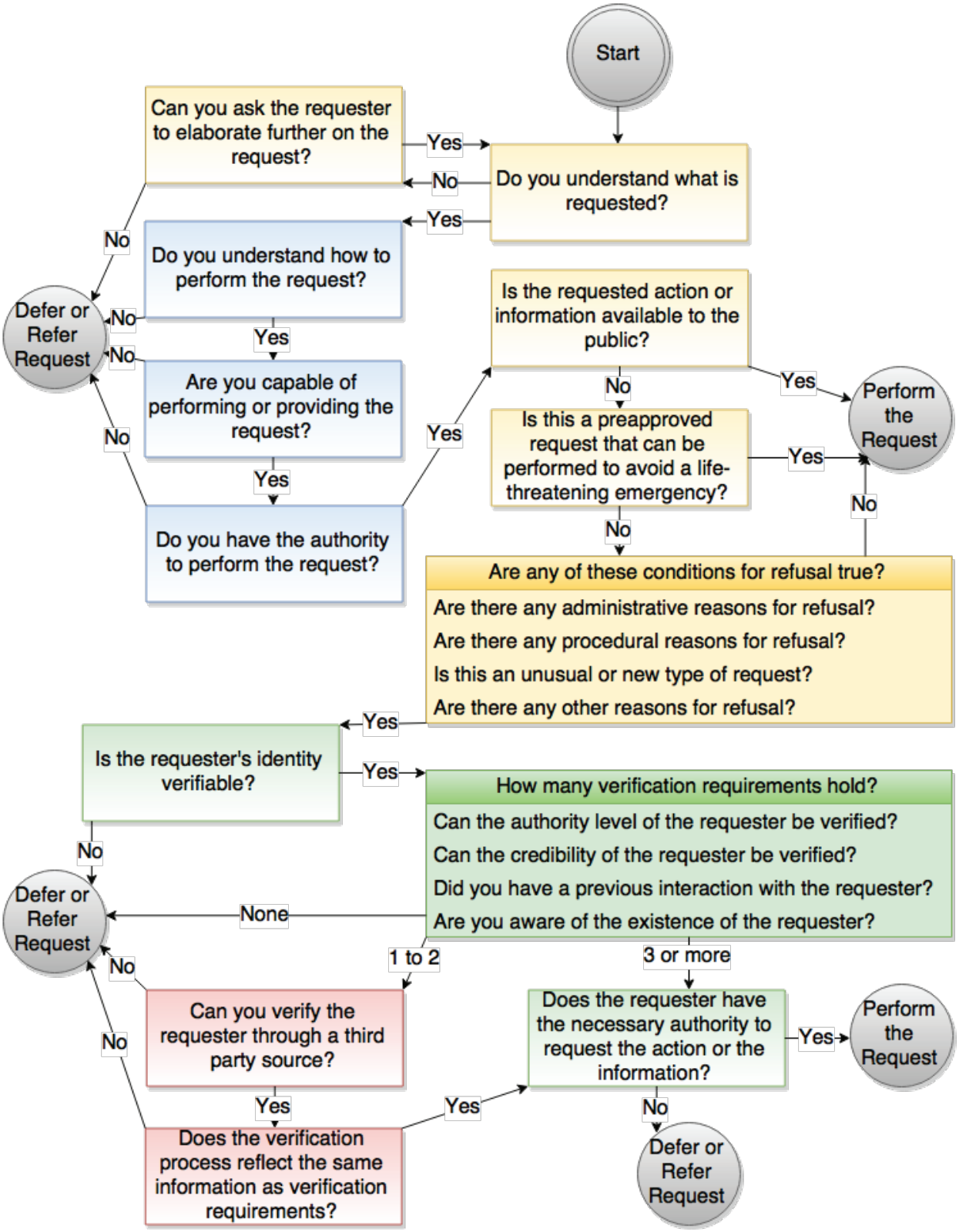

Figure 1: Social Engineering Attack Detection Model

state machine is a state machine that is guaranteed to complete for all inputs in a finite amount of time, while a non-deterministic finite state machine may execute indefinitely or fail to progress toward completion for certain input sets. A finite state machine is provably deterministic if and only if it is both free of cycles (that is, no state is ever revisited after being processed once) and defines a transition to a new state for each potential input in every state (that is, any valid input into a state results in a transition to a new state). These two properties together preclude the possibility of the state machine entering an infinite loop, either within a single state or between a collection of states, thereby ensuring that processing will always complete within a finite number of steps.

The finite state automaton described in this paper is an abstract or general model, and is intended to provide a structured but flexible deterministic high-level overview of the steps taken to mitigate a social engineering attack, improving upon the original SEADM flow-chart approach. While the SEADM flow-chart provides a static procedural template for implementing detection mechanisms for social engineering attacks, the finite state diagram provides a more accessible, abstract and extensible model that highlights the inter-connections between task categories associated with different scenarios. The abstract state-based model is intended to help facilitate the incorporation of domain, system or organisation specific extensions by grouping related activities into distinct categories, subdivided into one or more generalised nodes. Should a specific task, necessary in a particular domain, systems or organisational context not be included in the flow-chart, the state diagram may be used to infer the correct location within the model to incorporate the task. It further facilitates additional analysis on state transitions that are difficult to extract from the more verbose flow-chart.

The current iteration of the SEADM, as depicted in Figure 1, utilised four different state categories: the request, receiver, requester and third party. The request states, indicated in yellow, assesses information about the request itself. The receiver state, indicated in blue, considers the person handling the request and whether this person (the receiver) understands and is allowed to perform the request. The requester states, indicated in green, considers the requester and whether any information about the requester can be verified. The third party state, indicated in red, considers the involvement of a third party to support external verification of information supplied by the requester.

The same four categories and colour schemas are maintained in the state machine as they depict the primary topic that each specific state deals with, allowing one to better understand the attack detection model. The state machine is depicted in Figure 2. Each state has an associated letter which explains which condition needs to be met before the transition can be performed. As an example, a state can have an alphabet of Y and $\neg Y$. The symbol $\neg$ indicates negation, so $\neg Y$ is Not Y, or more accurately, the opposite of Y.

The states in Figure 2 are explained as follows:

- S_1 deals with the receiver's understanding the request. The request is either 'understood' (U) or 'not understood' ($\neg U$) by the receiver.
- S_2 deals with requesting more information from the requester by the receiver in an effort to properly understand the request. There is either 'sufficient information' (I) or 'insufficient information' ($\neg I$) for the receiver to understand and thus perform the request.
- S_3 deals with the capability of the receiver to perform the request. The receiver is either 'capable' (C) or 'incapable' ($\neg C$) of performing the request.
- S_4 deals with further verification requirements that may need to be met. The request either has further 'verification requirements' (R) or has 'no verification requirements' ($\neg R$). Additional verification requirements may be necessary in sensitive or secure contexts.
- S_5 deals with whether the receiver can verify and trust the identity of the requester, and how many of the verification steps hold. The verification steps hold to either a 'high amount' (V_H) where all or nearly all of the supplied information can be verified, a 'medium amount' (V_M) where the majority of information can be verified, or 'low amount' (V_L) where the majority of supplied information cannot be verified. These levels can be calibrated to be more or less restrictive, depending on the operating environment, and govern how the receiver should proceed.
- S_6 deals with trusted third party verification of information supplied by the requester. The requester is either 'verified' (T) by a third party or is 'not verified' ($\neg T$).
- S_7 deals with the authority of the requester. The requester either has 'sufficient authority' (A) or 'insufficient authority' ($\neg A$) for the particular request.
- S_F is an end state. This state indicates the failure state. The request should not be performed by the receiver, and should either be denied outright, or referred to a receiver with more knowledge or authority.
- S_S is an end state. This state indicates the success state. The request should be performed by the receiver.

The states are elaborated further in Section 4. A description of the full state machine in mathematical notation follows. The finite state machine is a 5-tuple consisting of the finite set of input alphabet characters Σ, the finite set of states Q, the start state S_0, a set of accepting states F, and a set of state transitions δ that contains 3-tuples representing state

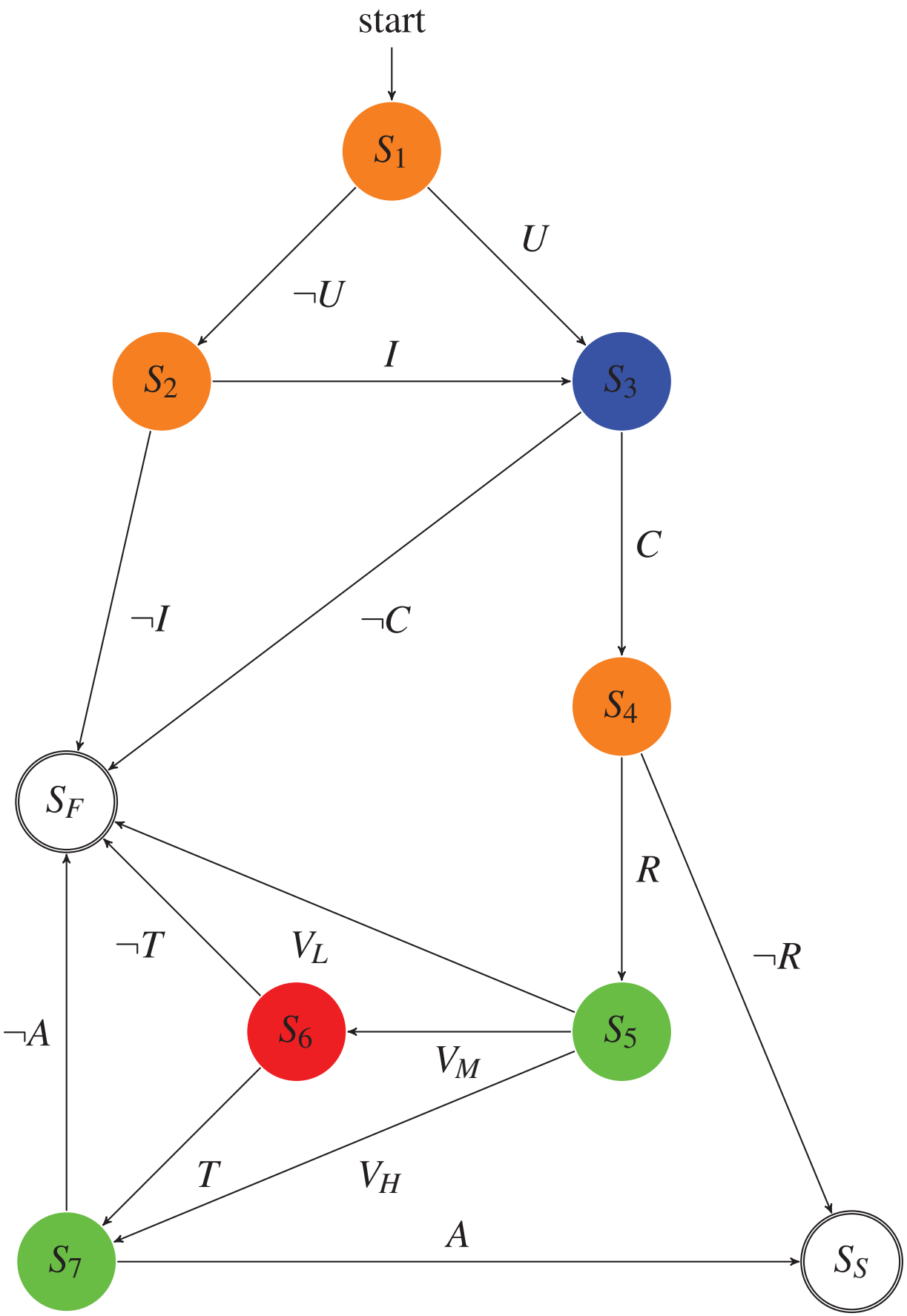

Figure 2: Underlying Finite State Machine of the SEADM

transitions. A 3-tuple in δ consists of a current state, a current input and the next state.

$$
\begin{aligned}
\Sigma &= \{U, \neg U, I, \neg I, C, \neg C, R, \neg R, V_L, V_M, V_H, T, \neg T, A, \neg A\} \\
Q &= \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_S, S_F\} \\
S_0 &= S_1 \\
\delta &= \{ \\
&\quad (S_1, U, S_3), (S_1, \neg U, S_2) \\
&\quad (S_2, I, S_3), (S_2, \neg I, S_F) \\
&\quad (S_3, C, S_4), (S_3, \neg C, S_F) \\
&\quad (S_4, R, S_5), (S_4, \neg R, S_S) \\
&\quad (S_5, V_L, S_F), (S_5, V_M, S_6), (S_5, V_H, S_7) \\
&\quad (S_6, T, S_7), (S_6, \neg T, S_F) \\
&\quad (S_7, A, S_S), (S_7, \neg A, S_F) \\
&\quad \} \\
F &= \{S_S, S_F\}
\end{aligned}
$$

Using both Figure 2 and the provided mathematical model it is straightforward to infer a state transition table. Table 1 depicts all the possible state transitions given a specific input for each state. For all input states, the output is either a terminal node or a node with a higher state index. This illustrates that the state machine is deterministic, eliminating cycles present in the original SEADM flowchart.

Table 1: State Transition Table for the SEADM

Input \ State	S_1	S_2	S_3	S_4	S_5	S_6	S_7
U	S_3	—	—	—	—	—	—
$\neg U$	S_2	—	—	—	—	—	—
I	—	S_3	—	—	—	—	—
$\neg I$	—	S_F	—	—	—	—	—
C	—	—	S_4	—	—	—	—
$\neg C$	—	—	S_F	—	—	—	—
R	—	—	—	S_5	—	—	—
$\neg R$	—	—	—	S_S	—	—	—
V_L	—	—	—	—	S_F	—	—
V_M	—	—	—	—	S_6	—	—
V_H	—	—	—	—	S_7	—	—
T	—	—	—	—	—	S_7	—
$\neg T$	—	—	—	—	—	S_F	—
A	—	—	—	—	—	—	S_S
$\neg A$	—	—	—	—	—	—	S_F

To further show that the state machine model is deterministic, resulting in a valid outcome of either success or failure for all given alphabet sequences, a transition table indicating all possible input alphabet sequences (paths) and their corresponding results are shown in Table 2. Each row in the table represents a path. Σ_i indicates the input character is the i-th character in the path. The symbol $\forall$ indicates no transition occurred in the i-th position of the path in the case of shorter paths. This table shows that for all possible paths, the state machine returns either success or failure in a finite number of steps, and is thus deterministic.

Table 2: State Transition Table for all Input Alphabets

	Input Alphabet							Output	
No	Σ_1	Σ_2	Σ_3	Σ_4	Σ_5	Σ_6	Σ_7	S_S	S_F
1	U	$\forall$	C	R	V_H	$\forall$	A	✓	—
2	U	$\forall$	C	R	V_M	T	A	✓	—
3	U	$\forall$	C	R	V_M	T	$\neg A$	—	✓
4	U	$\forall$	C	$\neg R$	$\forall$	$\forall$	$\forall$	✓	—
5	U	$\forall$	C	R	V_H	$\forall$	$\neg A$	—	✓
6	U	$\forall$	C	R	V_M	$\neg T$	$\forall$	—	✓
7	U	$\forall$	C	R	V_L	$\forall$	$\forall$	—	✓
8	U	$\forall$	$\neg C$	$\forall$	$\forall$	$\forall$	$\forall$	—	✓
9	$\neg U$	$\neg I$	$\forall$	$\forall$	$\forall$	$\forall$	$\forall$	—	✓
10	$\neg U$	I	C	R	V_H	$\forall$	A	✓	—
11	$\neg U$	I	C	R	V_M	T	A	✓	—
12	$\neg U$	I	C	R	V_M	T	$\neg A$	—	✓
13	$\neg U$	$\forall$	C	$\neg R$	$\forall$	$\forall$	$\forall$	✓	—
14	$\neg U$	I	C	R	V_H	$\forall$	$\neg A$	—	✓
15	$\neg U$	I	C	R	V_M	$\neg T$	$\forall$	—	✓
16	$\neg U$	I	C	R	V_L	$\forall$	$\forall$	—	✓
17	$\neg U$	I	$\neg C$	$\forall$	$\forall$	$\forall$	$\forall$	—	✓

Having considered the high-level state-based model for the SEADM, the following section elaborates on the purpose of each state and exactly how it was derived from the SEADM.

4. DISCUSSION OF EACH STATE

This sections explains how each of the states has been designed and integrated from the SEADM model. Throughout this discussion the alphabet of the states are provided. During the discussion on the states, it is also shown how each state relates to the SEADM. Each state has been generalised to such an extent that it can contain any number of questions required to achieve a specific transition result. This provides a rough guide for flexibility and extensibility, depending on the particular context the model is applied to.

4.1 State S_1: Understanding the Request

This state considers whether the receiver of the request fully understands the request in its entirety. This means that the requester should have provided all the information required to enable the receiver to perform the request in full. The question that was provided in the SEADM was "Do you understand what is requested?"

In the SEADM there was only one question asked here. This has been created as the start state as it is still important to fully understand what is requested before the request can be processed any further. In this state the alphabet is as follows:

- U represents that the request is understood in its entirety and that the receiver has all the necessary information in order to be able to perform the request.
- $\neg U$ represents that the request is not fully understood and that the receiver requires more information about the request.

This state can only transition in one of two ways and is depicted as follows:

- (S_1, U, S_3) if the request is fully understood.
- $(S_1, \neg U, S_2)$ if more information is required.

4.2 State S_2: Requesting information to fully understand the request

In the SEADM this state was previously grouped together with the previous question. This resulted in a loop, as the receiver could always ask the requester to elaborate further. At some point the requester would no longer be able to elaborate on the request as there would be no more additional information available, and the loop would terminate. At what point the loop would terminate was unclear, and could not be formalised into a deterministic state machine without additional complexity.

Representing this task as a distinct state more clearly defines this, as it deals directly with the information that is required to complete the request and not whether one can request more information. This state considers whether the requester can provide information to such an extent that the receiver is able to fully understand the request. A state transition occurs when either sufficient information is provided, or it is determined that the requester cannot provide sufficient information. Previously the question that was asked was as follows, "Can you ask the requester to elaborate further on the request?" In this state all questions should be aligned with whether the requested information, in the case that additional information is provided, allows the receiver to understand the request in full or not. In this state the alphabet is as follows:

- I represents that the requester can and has provided enough information for the request to be understood in its entirety by the receiver.
- $\neg I$ represents that the receiver is unable to understand the request in full. This may be because the requester could not provide more information, the requester could not be reached (in the case of remote communication), or that the information provided by the requester was insufficient or incomplete.

This state can only transition in one of two ways and is depicted as follows:

- (S_2, I, S_3) if the requester can provide sufficient information to understand the request.
- $(S_2, \neg I, S_F)$ if the requester is unable to provide sufficient information to understand the request, or cannot be reached.

4.3 State S_3: Does the receiver meet the requirements to perform the request?

State S_3 is used to determine whether the receiver meets all the requirements to perform the request. This state is associated with three questions in the SEADM. The questions are as follows:

- "Do you understand how to perform the request?"
- "Are you capable of performing or providing the request?"
- "Do you have the authority to perform the request?"

The goal of this state is to ensure that the individual who deals with the specific request has the necessary skill level and has the required authority to perform the request. Each question in this state deals directly with the role of the receiver and determines whether the request has been issued to the correct receiver. In this state the alphabet is as follows:

- C represents that the receiver has met all the requirements to be capable of performing the requested action or to provide the requested information.
- $\neg C$ represents that the receiver does not meet the requirements to be capable of dealing with the request.

This state can only transition in one of two ways and is depicted as follows:

- (S_3, C, S_4) if the receiver is able to service the request.
- $(S_3, \neg C, S_F)$ if the receiver is unable or incapable of servicing the request.

4.4 State S_4: Does the request have any further requirements that need to be met before the request can be serviced?

State S_4 deals with the the request itself and whether there are any special conditions or requirements, such as policies and procedures that need to be followed, associated with the request. Examples of special conditions include whether the request relates to information already in the public domain and is accessible to all, or whether the request is a life threatening emergency. If the request is

already in the public domain, for instance, there are no further requirements that need to be met. In the event of a life threatening emergency, the outcome depends on whether there are set policies in place to deal with such contingencies. For instance, if there is a policy in place that allows medical personnel to rather err on the side of caution in order to save an individuals life, there will be no further requirements that need to be met and the request may take place. The previous model dealt with these examples using the following questions:

- "Is the requested action or information available to the public?"
- "Is this a pre-approved request that can be performed to avoid a life-threatening emergency?"

In addition, this state also deals with any further requirements that need to be met. Examples of such requirements are any policies or procedures that are in place that require further verification of the identity or authority of the requester. This state can also cater for unusual requests. An unusual request is a request that is new to the receiver and/or is a request that the receiver does not usually deal with on a regular basis. By following the rest of the model, the receiver ensures that adequate information about the requester is obtained before the request is performed. It also allows the receiver time to think about the request and whether the request should be performed for the receiver. The previous model dealt with these examples using the following questions:

- "Are there any administrative reasons for refusal?"
- "Are there any procedural reasons for refusal?"
- "Is this an unusual or new type of request?"
- "Are there any other reasons for refusal?"

The goal of this state is to ensure that any request which is already public information should be immediately performed and that any request that requires verification should result in further interrogation of the requester. In this state the alphabet is as follows:

- R represents that the request has further verification requirements that need to be met in order to be able to perform the requested action or to provide the requested information.
- $\neg R$ represents that the request has no further verification requirements and that the requested action can be performed or that the requested information can be provided.

This state can only transition in one of two ways and is depicted as follows:

- (S_4, R, S_5) if further verification is required.
- $(S_4, \neg R, S_S)$ if no further verification is required.

4.5 State S_5: To what extent is the requester's identity verifiable?

State S_5 aims to address the question of the extent of verifiability of the requester's identity. The identity of the requester is determined to ensure that the requester has sufficient privileges to request the specific action or information. It is not always possible to verify the identity of the requester in full. The type of communication medium that is utilised by the requester usually will dictate to what extent the identity of the requester can be verified. Typically, if the requester makes the request in person one is able to verify significantly more information about the requester than would be possible over an e-mail or postal mail. It may also be the case that the request is performed over unidirectional communication and that the receiver is unable to receive any further communication from the requester.

The previous model provided a fourth question as to whether the requester's identity is verifiable at all. The state machine has combined not being verifiable and having a low level of verification as the same transition as both states lead to S_F. The questions that were previously used to perform the verification requirements are as follows:

- "Can the authority level of the requester be verified?"
- "Can the credibility of the requester be verified?"
- "Did you have a previous interaction with the requester?"
- "Are you aware of the existence of the requester?"

All of the questions catered for a single point of verification. It was also noted in the previous model that the level of verification required to transition to different states should be based on what type of environment the model is applied to. The state machine makes this more generalised by having three possible transitions where there is a low, medium or high level of verification. The threshold for low, medium and high must still be determined based on the environment or context, but the state diagram is more flexible when more questions are added. In this state the alphabet is as follows:

- V_L represents that there is a low level of verification as only a few of the verification elements could be verified.
- V_M represents that there is a medium level of verification as some of the verification elements could be verified, but not all of them.
- V_H represents that there is a high level of verification as all or most of the verification elements could be verified. Note that edge V_H should not be followed if any information provided in a request is inconsistent with, or differs substantially from, established or generally available information.

This state can only transition in one of three ways and is depicted as follows:

- (S_5, V_L, S_F) if only a few verification elements hold.
- (S_5, V_M, S_6) if a moderate amount of verification elements hold.
- (S_5, V_H, S_7) if all or most verification elements hold.

4.6 *State S_6: Can you verify the requester's identity from a third party source?*

State S_6 is only entered when there is a medium level of verification requirements that have been met. If the requester could not be fully verified directly, the third party source is utilised to determine whether the information provided by the requester was indeed truthful. The previous model did not elaborate much on the third party verification and only had two questions associated with it, as follows:

- "Can you verify the requester through a third party source?"
- "Does the verification process reflect the same information as the verification requirements?"

The supplied questions did not mention specifically which verification requirements needed to be verified. Utilising the extensibility and generality of the state machine, one could build intelligence into the model to only ask such questions when the verification requirements were obtained directly from the requester. In this state the alphabet is as follows:

- T represents that the verification requirements, as obtained from the requester, fully corresponds to the information that was obtained from the third party source.
- $\neg T$ represents that the verification requirements, as obtained from the requester, do not fully correspond to the information that was obtained from the third party source.

This state can only transition in one of two ways and is depicted as follows:

- (S_6, T, S_7) if information supplied by the requester is fully verified by the third party source.
- $(S_6, \neg T, S_F)$ if information supplied by the requester is not fully verified by the third party source.

4.7 *State S_7: Does the authority level of the requester provide them with sufficient rights to request the action or information?*

State S_7 utilises all the information obtained throughout the model and asks the receiver questions based on the information obtained. This state aims to determine whether the requester has the necessary authority and rights to request the action or the information. The previous model only asked "Does the requester have the necessary authority to request the action or the information?" This state elaborates on this by allowing the receiver to verify whether the requester has sufficient authority to gain access to the request or the information. In this state the alphabet is as follows:

- A represents that the requester has sufficient authority to be allowed to request the receiver to perform the action or to provide the information.
- $\neg A$ represents that the requester does not have sufficient authority and is thus not allowed to request the receiver to perform the action or to provide the information.

This state can only transition in one of two ways and is depicted as follows:

- (S_7, A, S_S) if the requester has sufficient authority for the request to be processed.
- $(S_7, \neg A, S_F)$ if the requester does not have sufficient authority for the request to be processed.

4.8 *State S_F: Halt the request*

This is the negative result state. In this state the request will be halted. In some environments one could opt to rather defer the request to a more authoritative receiver. Deferring the request may lead to the request never being performed, and can be considered as the request having been halted. In the case where the receiver is part of an organisation, there is the option to refer the request to a more authoritative person in the same organisation. This will allow someone else who may be better suited to determine whether to perform or halt the request.

4.9 *State S_S: Perform the request*

This is the positive result state of the model. In this state the receiver is allowed to perform the single request from the requester.

The next section briefly discusses social engineering attack templates, which are then tested against the social engineering attack detection model.

5. APPLICATION OF THE SOCIAL ENGINEERING ATTACK DETECTION MODEL

Previously, social engineering attack templates have been proposed to provide researchers with a set of social engineering attack examples that can be used to verify and make comparisons between models, processes and frameworks within social engineering [12]. All of the templates were derived from real-world social engineering attacks that have been documented in either news articles, technical reports, research reports, films or blogs. The news articles, technical reports, research reports or blogs did not always contain all of the information regarding the social engineering attack. This lack of information was addressed by proposing the templates as a more generalised form of the social engineering attacks provided in the literature [12].

Each template contains the full description of every phase and associated steps of the social engineering attack framework in such a way that each template will provide repeatable results when used for verification and comparison purposes. The templates are also kept as simple as possible so that they can be expanded upon to create more elaborate scenarios with similar principal structures. The templates can also be used to verify or compare other models, processes and frameworks without having to physically perform the attack and potentially cause harm to innocent targets [19]. The rest of this section is dedicated to testing the SEADM against the social engineering attack templates.

In the first scenario, a bidirectional communication template, the social engineer pretends to be someone who works on the management floor and has to convince a cleaner that he is indeed an employee [12]. He requests the cleaner to give him access to the management floor. In the second scenario, a unidirectional communication template, the social engineer attempts to obtain financial gain by sending out paper mail in which the letter requests a group of individuals to make a small deposit into a bank account owned by the attacker [12]. In the third scenario, an indirect communication template, the social engineer attempts to gain unauthorised access to a workstation in an organisation by using a storage medium device [12].

In each scenario the reader is provided with a generic description of the attack as taken from social engineering attack templates. This generic description is then populated with elements, both subjects and objects, from real-world examples of social engineering attacks, as provided in the discussion of the specific social engineering attack template. Using the generic description, the elements from the real-world examples and the fully detailed flow of the attack as provided in each phase and step of the social engineering attack framework, one is able to devise a social engineering attack scenario. This scenario is then reflective of a real-world example of which every phase and step is fully documented as per the social engineering attack framework. Using the proposed social engineering attack templates, one is able to formulate a social engineering attack scenario that always follows the same process, with regards to phases and steps, whilst the social engineering attack is still representative of a real-world scenario.

The remainder of this section is dedicated to mapping the social engineering attack templates to the states of the SEADM and verifying whether the social engineering attack detection model can assist in detecting these attacks. The following discussions will highlight at which points the SEADM could prevent the success of the social engineering attack if it were properly followed, but will make allowances for failure in following the SEADM to fully explore each scenario. Note that the discussion of each state makes implicit reference to the subsidiary questions defined and described for the state in Section 4.

5.1 Bidirectional Communication Scenario

The generic description for this scenario reads as follows: "This template illustrates an SEA where the attacker attempts to gain physical access to a computerised terminal at the premises of an organisation. The assumption is that when the attacker has once gained access to the computerised terminal, he/she is deemed to have been successful. The attacker is now able to install a backdoor onto the computerised terminal for future and further access from the outside." This scenario is populated with elements from a real-world example where the social engineer pretends to be someone who works on the management floor and convinces a cleaner of his supposed role. The goal of the attacker is to manipulate the cleaner into granting the social engineer access to the management floor. This allows the social engineer to gain physical access to the computerised terminals on the management floor [20, 21].

In this scenario a social engineer has to convince the cleaner, the receiver, to believe that he is indeed a staff member. In this scenario, the cleaners have full access to the building, yet, their security awareness is typically quite low. They are not trained to respond to unusual requests such as giving other employees access to the management floor. If the request is successful, access has been gained to the management floor, and a key logger is deployed onto a workstation. This attack is performed using bidirectional communication because the social engineer communicates with the cleaner and convinces him that the social engineer is allowed to have access to the management floor and the workstations.

S_1 **– Understanding the Request**: The request from the social engineer should clearly state that access needs to be gained to the management floor. The social engineer can also justify to the receiver why access is required to further allow the receiver to understand the request. If the receiver understands the request, edge U is followed to S_3. If the receiver does not initially understand the request, the SEADM would move to state S_2 via edge $\neg U$.

S_2 **– Requesting information to fully understand the**

request: In the unlikely event that the request is not initially understood, the social engineer would explain his/her request in more detail. As the request itself is straightforward, if unusual, it is assumed that it will ultimately be understood, resulting in a transition (S_2, I, S_3).

S_3 **– Does the receiver meet the requirements to perform the request?**: In this scenario, the receiver does not have the authority to grant access to the management floor. If the SEADM were followed, appropriate security policy were in place, and the cleaner had sufficient knowledge of said policy, the cleaner would decline the request and refer the social engineer to a more qualified individual, thwarting the attack. This would be reflected in the SEADM by a transition via edge $\neg C$ to the failure state, S_F. For the purposes of discussion, and assuming the cleaner is not properly trained and fears reprisals from a more senior employee, the transition (S_3, C, S_4) is followed.

S_4 **– Does the request have any further requirements that need to be met before the request can be serviced?**: In the scenario, only management and cleaners should have access to the management floor. The requested action is thus not available to the public, or in service of a medical emergency, and is subject to further verification of the requester's identity. If the SEADM were followed at this point, this would result in the transition (S_4, R, S_5).

S_5 **– To what extent is the requester's identity verifiable?**: As bidirectional communication is utilised, it allows for the receiver to communicate back via face to face communication and ask more questions to verify the requester. In this case, the authority principle is utilised and the social engineer mimics an authoritative figure who should have access to the management floor. The pretext utilised during this attack is that the social engineer is part of management and that he or she should have access to the management floor. The receiver is only able to verify the falsified authority level from the social engineer in this scenario. Since only a single verification requirement is met, transition (S_5, V_L, S_F) would be the most appropriate, resulting in the request being deferred to an individual with more authority, thus thwarting the attack. If the social engineer can provide additional false information, and the cleaner decides to err on the side of caution, the transition (S_5, V_M, S_6) may be followed instead. As the social engineer is not known to the cleaner and is not an actual employee, edge V_H should not be followed.

S_6 **– Can you verify the requester's identity from a third party source?**: The receiver will now have the ability to verify the information from another employee on the management floor. In the case that there are no other employees on the management floor, the transition $(S_6, \neg T, S_F)$ will be taken and the social engineering attack will be thwarted. If it is assumed that there are other employees on the management floor who can be contacted to verify the information, the receiver will be able to ask whether the authority level of the social engineer is indeed true. The other employee will deny this and thus the verification process will show that the information provided is not the same as the verification requirements. Consequently, the transition $(S_6, \neg T, S_F)$ will still be taken, and the social engineering attack will be thwarted.

S_7 **– Does the authority level of the requester provide them with sufficient rights to request the action or information?** Assuming the receiver is untrained, the SEADM is not followed, and the requester's identity is incorrectly assumed to be verified, their presumed authority would grant them access to the management floor, resulting in transition (S_7, A, S_S). If the SEADM were followed, however, the attack would move to the failure state S_F twice before this state is first reached.

In this scenario, the SEADM would have detected and thwarted the social engineering attack at two separate points before reaching the final state, S_7, in the model.

5.2 Unidirectional Communication Scenario

The generic description for this scenario reads as follows: "This template illustrates an SEA in which the attacker attempts to obtain financial gain by sending out paper mail. This letter requests a group of individuals to make a small deposit into a bank account owned by the attacker. In this template, the attacker develops a phishing letter that masks the attacker as a charity organisation requesting donations. Once the attacker has received the small deposit from the targeted individual, the SEA is deemed to be successful." This scenario is populated with elements from the real-world example where the social engineer performs a pretext using postal letters. The social engineer pretends to be various officials, internal employees, employees of trading partners, customers, utility companies or financial institutions, and the social engineer solicits confidential information by using a wide range of persuasive techniques [22].

In this scenario, the attacker will develop a phishing letter that masks the attacker as a charity organisation requesting donations. The phishing letter contains the contact details, the logo and the purpose of the charity to improve the authenticity of the letter. This attack uses unidirectional communication and thus the receiver is not able to communicate with the attacker. The rest of this section maps the scenario to the model.

S_1 **– Understanding the Request**: The letter from the social engineer should clearly state that a receiver is requested to make a donation to the specific charity. The letter will include all the required details because this receiver cannot communicate with the social engineer. In the SEADM, this should result in the transition (S_1, U, S_3). In the unlikely event that the receiver does not understand the request, the transition $(S_1, \neg U, S_2)$ will be followed.

S_2 **– Requesting information to fully understand the request**: The social engineer would have tried to ensure that the targeted individual fully understands the request.

As the attack uses unidirectional communication, the receiver is unable to request further information. As a result, in the SEADM, the only available state transition is $(S_2, \neg I, S_F)$, thwarting the attack.

S_3 **– Does the receiver meet the requirements to perform the request?**: The receiver is the owner of their bank account and has the required information to make the deposit. Given that their understanding of the request has been ascertained, and assuming they have a bank account with an available balance and are both capable and authorised to deposit their money into another bank account, the transition (S_3, C, S_4) will be followed.

S_4 **– Does the request have any further requirements that need to be met before the request can be serviced?**: The requested action is to make a deposit into the bank account of the requester. This action is only available to receiver, and is not in service of preventing a life-threatening emergency. Furthermore, this request can be seen as either unusual or new as the requester would not usually receive this specific type of letter from the charity. It is additionally likely that the requester feels uneasy about, or suspicious of, the request, which is itself a reason for refusal. The transition S_3, R, S_4 should thus be followed.

S_5 **– To what extent is the requester's identity verifiable?**: Since unidirectional communication is utilised, the receiver can only verify the identity of the requester using the information provided in the received letter. While the receiver may be aware of the existence of the charity organisation, if the letter provides no additional information for verification purposes (such as contact details), or if information in the charity's official correspondence or online presence is inconsistent with information provided in the fabricated request, the SEADM indicates that the transition (S_5, V_L, S_F) should be followed, thwarting the attack. If the letter contains the actual contact details of the charity organisation, thereby providing some additional verifiable information, the transition (S_5, V_M, S_6) may be followed instead. As the request is unidirectional and does not specify the charity's actual bank details, the transition (S_5, V_H, S_7) should not be followed; edge V_H indicates very high confidence, which should not be followed when a request is unusual and provides limited verifiable information.

S_6 **– Can you verify the requester's identity from a third party source?**: As the charity is a well known charity, the request may be verified by contacting the charity directly. The receiver will make a phone call to the charity to verify the information. If the charity cannot be reached, the transition $(S_6, \neg T, S_F)$ should be followed, thwarting the attack. If the receiver is able to contact the charity directly, the receiver will be able to ask the organisation whether such a letter has in fact been sent out. The charity organisation will deny this and thus the verification process will show that the information provided is not the same as the verification requirements. Consequently, the transition $(S_6, \neg T, S_F)$ will be followed and the social engineering attack will be thwarted.

S_7 **– Does the authority level of the requester provide them with sufficient rights to request the action or information?** Assuming that the SEADM was not followed up to this point, and the receiver believes the information provided in the fabricated request, the receiver may consider the requester to have sufficient authority to make the request. If this is the case, the transition (S_7, A, S_S) may be followed, allowing the attack to succeed. If the SEADM was followed, however, state S_7 would not have been reached, preventing this transition from occurring.

While thwarting this form of attack requires some diligence on the part of the receiver, following the SEADM should prevent the SEA from succeeding.

5.3 Indirect Communication Scenario

The generic description for this scenario reads as follows: "This template illustrates an SEA in which the attacker attempts to gain unauthorised access to a workstation within an organisation by using a storage device. Once the target has plugged the storage device (in this case a USB flash drive) into the targeted workstation, the SEA is deemed to be successful; the attacker is now able to install a backdoor onto the workstation via the storage device. The social engineer can then proceed to use this workstation as a pivot point for any further attacks on the organisation." This scenario is populated with elements from the real-world example where the social engineer attempts to gain unauthorised access to a workstation in an organisation by using a storage medium device [23,24]. This attack is also depicted in a popular television series about penetration testing, Mr. Robot [23].

In this scenario, the organisation does not have a company policy in place that disallows employees plugging storage devices into their workstations. The social engineer will leave the device outside the organisation's building to be found by an employee. The device will be infected with a trojan so that when it is plugged into the workstation, it opens a backdoor for the social engineer to connect to the system remotely. As the storage device is left unattended, this attack utilises indirect communication. The rest of this section maps this scenario to the model.

S_1 **– Understanding the Request**: The storage medium device planted by the social engineer should be marked clearly to indicate that it contains important and/or confidential information. The social engineer expects that the receiver – the employee who finds this device – will either try to return the device to its rightful owner (a benevolent target), or attempt to access the contents of the drive (a malevolent target). The social engineer may attempt to manipulate a specific employee into finding the device, or may settle for any employee. As it is an inherent request, the request should be easily understandable to the target, and the transition (S_1, U, S_3) should be followed in the SEADM.

S_2 **– Requesting information to fully understand the request**: This state would not typically be entered, as the social engineer would have made certain that the storage medium device is deployed at such a location that only individuals who have access to a workstation and who understand how such devices work would find the device. If this is not the case, the attack runs the risk of immediate failure, as the receiver may take the device to a lost-and-found where the attack is indefinitely halted, or to the IT department where the attack may be detected. Either of these cases would result in the transition $(S_2, \neg I, S_F)$, a failure state.

S_3 **– Does the receiver meet the requirements to perform the request?**: As it is assumed that the organisation does not prohibit the use of external USB drives on company workstations, and the receiver has access to a workstation that supports the USB attachment, the receiver is capable of performing the request. In most instances, this would be sufficient for the requirements of performing the implicitly requested action, and the transition (S_3, C, S_4) would be followed. If the drive is labelled as confidential, however, the receiver may be considered to not have sufficient authority to attach the drive. In this instance, the transition $(S_3, \neg C, S_F)$ would be followed in the SEADM, resulting in the attack being halted or thwarted.

S_4 **– Does the request have any further requirements that need to be met before the request can be serviced?**: The implicit request is directed at the receiver to manipulate them into attaching the device to a company worstation, either to return it to its rightful owner or to determine its contents. This action is not available to the public, or in service of preventing a life-threatening emergency. While there are no administrative or procedural reason for refusal due to a lack of specific company policy, the request would be considered unusual, and so the transition (S_4, R, S_5) would be followed.

S_5 **– To what extent is the requester's identity verifiable?**: Since indirect communication is utilised, the only piece of information the receiver has is the physical storage device, which does not provide any external verification information. Following the SEADM, this would result in a transition (S_5, V_L, S_F), resulting in the implicit request being either halted or detected by a more knowledgeable individual who is allowed to safely, and on a secure workstation, verify the contents of the storage device and potentially contact the rightful owner. Neither V_M or V_H should be followed in this instance.

S_6 **– Can you verify the requester's identity from a third party source?**: While this state should not be reached, there is no available third-party to provide verification of the device. The only option in this instance would be to follow transition $(S_6, \neg T, S_F)$, similarly resulting in the failure of the attack.

S_7 **– Does the authority level of the requester provide them with sufficient rights to request the action or information?** If this state were hypothetically reached, the lack of specific organisational policy governing the use of external storage media would provide sufficient authority for the implicit request to be performed, resulting in a transition (S_7, A, S_S) and allowing the attack to succeed. Similar to S_6, however, this state should not be reached if the SEADM is properly followed.

This demonstrates how the SEADM may be successfully utilised to prevent an attack when SEAs use indirect communication.

The preceding three sections have shown, through the use of attack templates, how the SEADM may be applied to help detect and prevent several social engineering attack vectors that utilise bidirectional, unidirectional and indirect communication approaches. The SEADM is, however, only a tool, and is best utilised in conjunction with security awareness and robust security policy to reinforce and guide its appropriate application and use, largely dependent on the security context.

The following section concludes the paper with a summary of the advantages of the underlying finite state machine and how it was still able to thwart social engineering attack templates.

6. CONCLUSION

The protection of information is extremely important in modern society and even though the security around information is continuously improving, a weak point is still human actors who are susceptible to manipulation techniques. This paper explored social engineering as a domain and social engineering attack detection techniques as a process within this domain. To this end, both the previous papers by the authors, *Social Engineering Attack Detection Model: SEADM* [6] and *Social Engineering Attack Detection Model: SEADMv2* [14] were revisited.

Both of the previous iterations of the SEADM focused on expanding the capability of the accuracy of detection. The models were populated with questions catering for attacks utilising either bidirectional communication, unidirectional communication or indirect communication. Previous work did mention that the model is extensible with additional questions, but was unclear on how and where additional questions should be added.

This paper improves on the SEADM by providing the underlying finite state machine, which allows researchers to better understand and utilise the SEADM. Representing the SEADM as a finite state machine allows one to have a more concise overview of the process that is followed throughout the model. The model provided a general procedural template for implementing detection mechanisms for social engineering attacks. The state diagram provides a more abstract and extensible model that highlights the inter-connections between task categories associated with different scenarios. This paper also shows that the finite state machine is both deterministic and

correct for all possible input alphabets, simplifying the process of implementing the SEADM model either as a process or in software. The state diagram is currently implemented as a mobile application as part of a social engineering prevention training tool [25].

The improved SEADM was further taken and tested against social engineering attack templates, showing that the SEADM remains capable of thwarting social engineering attacks. Adapting the SEADM to a finite state machine improved the extensibility of the model without negatively impacting on detecting social engineering attacks.

The SEADM, with the underlying finite state machine, can be used as a general framework to protect against social engineering attacks. Even if the model is not adhered to in respect of every request, it will cause one to think differently about requests — and this is already a step in the right direction.

REFERENCES

[1] D. Harley, "Re-floating the titanic: Dealing with social engineering attacks," in *European Institute for Computer Antivirus Research*, 1998, pp. 4–29.

[2] L. Laribee, "Development of methodical social engineering taxonomy project," MSc, Naval Postgraduate School, Monterey, California, June 2006.

[3] K. Ivaturi and L. Janczewski, "A taxonomy for social engineering attacks," in *International Conference on Information Resources Management*, G. Grant, Ed. Centre for Information Technology, Organizations, and People, June 2011, pp. 1–12.

[4] F. Mohd Foozy, R. Ahmad, M. Abdollah, R. Yusof, and M. Mas'ud, "Generic taxonomy of social engineering attack," in *Malaysian Technical Universities International Conference on Engineering & Technology*, Batu Pahat, Johor, November 2011, pp. 1–7.

[5] P. Tetri and J. Vuorinen, "Dissecting social engineering," *Behaviour & Information Technology*, vol. 32, no. 10, pp. 1014–1023, 2013.

[6] F. Mouton, L. Leenen, M. M. Malan, and H. Venter, "Towards an ontological model defining the social engineering domain," in *ICT and Society*, ser. IFIP Advances in Information and Communication Technology, K. Kimppa, D. Whitehouse, T. Kuusela, and J. Phahlamohlaka, Eds. Springer Berlin Heidelberg, 2014, vol. 431, pp. 266–279.

[7] J. W. Scheeres, "Establishing the human firewall: reducing an individual's vulnerability to social engineering attacks," Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, March 2008.

[8] K. D. Mitnick and W. L. Simon, *The art of intrusion: the real stories behind the exploits of hackers, intruders and deceivers.*, W. Publishing., Ed. Indianapolis: Wiley Publishing, 2005.

[9] J. Debrosse and D. Harley, "Malice through the looking glass: behaviour analysis for the next decade," in *Proceedings of the 19th Virus Bulletin International Conference*, September 2009.

[10] G. L. Orgill, G. W. Romney, M. G. Bailey, and P. M. Orgill, "The urgency for effective user privacy-education to counter social engineering attacks on secure computer systems," in *Proceedings of the 5th Conference on Information Technology Education*, ser. CITC5 '04. New York, NY, USA: ACM, 2004, pp. 177–181. [Online]. Available: http://doi.acm.org/10.1145/1029533.1029577

[11] F. Mouton, M. M. Malan, L. Leenen, and H. Venter, "Social engineering attack framework," in *Information Security for South Africa*, Johannesburg, South Africa, Aug 2014, pp. 1–9.

[12] F. Mouton, L. Leenen, and H. Venter, "Social engineering attack examples, templates and scenarios," *Computers & Security*, vol. 59, pp. 186 – 209, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404816300268

[13] M. Bezuidenhout, F. Mouton, and H. Venter, "Social engineering attack detection model: Seadm," in *Information Security for South Africa*, Johannesburg, South Africa, August 2010, pp. 1–8.

[14] F. Mouton, L. Leenen, and H. S. Venter, "Social engineering attack detection model: Seadmv2," in *International Conference on Cyberworlds (CW)*, Visby, Sweden, October 2015, pp. 216–223.

[15] F. Mouton, M. Malan, and H. Venter, "Development of cognitive functioning psychological measures for the seadm," in *Human Aspects of Information Security & Assurance*, Crete, Greece, June 2012, pp. 40–51.

[16] D. Gragg, "A multi-level defense against social engineering," SANS Institute InfoSec Reading Room, Tech. Rep., December 2002.

[17] R. Bhakta and I. Harris, "Semantic analysis of dialogs to detect social engineering attacks," in *Semantic Computing (ICSC), 2015 IEEE International Conference on*, Feb 2015, pp. 424–427.

[18] S. S. Epp, *Discrete Mathematics with Applications*, 4th ed. Brooks/Cole Publishing Co., 2010.

[19] F. Mouton, M. M. Malan, and H. S. Venter, "Social engineering from a normative ethics perspective," in *Information Security for South Africa*, Johannesburg, South Africa, August 2013, pp. 1–8.

[20] L. Janczewski and L. Fu, "Social engineering-based attacks: Model and new zealand perspective," in *Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on*, Oct 2010, pp. 847–853.

[21] T. Dimkov, A. van Cleeff, W. Pieters, and P. Hartel, "Two methodologies for physical penetration testing using social engineering," in *Proceedings of the 26th Annual Computer Security Applications Conference*, ser. ACSAC '10. New York, NY, USA: ACM, 2010, pp. 399–408. [Online]. Available: http://doi.acm.org/10.1145/1920261.1920319

[22] M. Workman, "A test of interventions for security threats from social engineering," *Information Management & Computer Security*, vol. 16, no. 5, pp. 463–483, 2008.

[23] S. Esmail, "eps1.5_br4ve-trave1er.asf," June 2015, mr. Robot: Season 1, Episode 6. [Online]. Available: http://www.usanetwork.com/mrrobot/episode-guide/season-1-episode-6-eps15br4ve-trave1erasf

[24] M. Jodeit and M. Johns, "Usb device drivers: A stepping stone into your kernel," in *Computer Network Defense (EC2ND), 2010 European Conference on*, Oct 2010, pp. 46–52.

[25] F. Mouton, M. Teixeira, and T. Meyer, "Benchmarking a mobile implementation of the social engineering prevention training tool," in *Information Security for South Africa*, Johannesburg, South Africa, Aug 2017, pp. 106–116.