



AFRL-RH-WP-TR-2020-0130

CONTINUOUSLY HABITUATING ELICITATION STRATEGIES FOR SOCIAL-ENGINEERING ATTACKS (CHESS)

Tiffany Kim
HRL Laboratories, LLC
3011 Malibu Canyon Road
Malibu, CA 90265

June 2020

Final Tech Report

Distribution A. Approved for public release; distribution unlimited.

**AIR FORCE RESEARCH LABORATORY
711th HUMAN PERFORMANCE WING
AIRMAN SYSTEMS DIRECTORATE
WARFIGHTER INTERFACE DIVISION
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the AFRL Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RH-WP-TR-2020-0130 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

HANSEN.ERIC
.G.1249715502

Digitally signed by
HANSEN.ERIC.G.1249715502
Date: 2021.04.28 16:34:59
-04'00'

ERIC HANSEN, DR-III
Mission Analytics Branch
Airman Systems Directorate
711th Human Performance Wing
Air Force Research Laboratory

MURDOCK.WILLI
AM.P.1048742161

Digitally signed by
MURDOCK.WILLIAM.P.1048742
161
Date: 2021.04.28 16:49:24 -04'00'

WILLIAM P. MURDOCK, DR-IV, Ph.D.
Chief, Mission Analytics Branch
Airman Systems Directorate
711th Human Performance Wing
Air Force Research Laboratory

CARTER.LOUISE
ANN.1230249128

Digitally signed by
CARTER.LOUISE.ANN.123024912
8
Date: 2021.05.28 13:14:47 -04'00'

LOUISE A. CARTER, DR-IV, Ph.D.
Chief, Warfighter Interaction and Readiness Division
Airman Systems Directorate
711th Human Performance Wing
Air Force Research Laboratory

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE				<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 30-06-2020		2. REPORT TYPE Final		3. DATES COVERED (From - To) 28 September 2018 – 29 May 2020	
4. TITLE AND SUBTITLE Continuously Habituating Elicitation Strategies for Social-Engineering Attacks (CHESS)				5a. CONTRACT NUMBER FA8650-18-C-7877	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Tiffany Kim				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER H0X3	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) HRL Laboratories, LLC 3011 Malibu Canyon Road Malibu, CA 90625				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Air Force Materiel Command Air Force Research Laboratory 711 Human Performance Wing Airman Systems Directorate Warfighter Interactions & Readiness Division Wright-Patterson AFB OH 45433				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-RH-WP-TR-2020-0130	
12. DISTRIBUTION / AVAILABILITY STATEMENT Distribution A. Approved for public release; distribution unlimited.					
13. SUPPLEMENTARY NOTES AFRL-2021-1280; Cleared 27 Apr 2021					
14. ABSTRACT This effort summarizes the CHESS system that is a partial Technical Area 2 (TA2) solution with emphasis on eliciting information from social engineering attackers and engaging with them by prolonging conversations. This report was developed under a contract for Active Social Engineering Defense (ASED).					
15. SUBJECT TERMS CHESS, MLN, CM					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER PAGES	19a. NAME OF RESPONSIBLE PERSON Eric Hansen
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (include area code)
			SAR	74	

TABLE OF CONTENTS

LIST OF FIGURES.....	iii
LIST OF TABLES	iv
1.0 SUMMARY	1
2.0 INTRODUCTION.....	3
2.1 Human Subjects Experiments.	4
2.2 Threat-Actor Organization	4
2.3 High-Level Action Spaces & Game Tree Generation	4
3.0 CHESS SYSTEM OVERVIEW AND ASSUMPTIONS	6
3.1 Assumptions.....	6
3.2 System Overview.	6
4.0 THREAT-ACTOR IDENTIFICATION USING MLN	9
4.1 Layer Development and Training.....	9
4.1.1 Message ID.	9
4.1.2 Topic Modeling	11
4.1.3 Sentiment Analysis.....	12
4.1.4 TF-IDF	12
4.1.5 Readability Test	12
4.2 Mapping and Embedding Incoming Messages to Inner Product Space	12
4.3. Clustering	13
4.4. Evaluation.....	13
4.4.1 Initial Evaluation.....	13
4.4.2 Addressing Dataset Limitations	14
4.4.3 MLN vs. TRIAGE Evaluation	15
4.4.4 Limitations	16
4.4.5 Future Work	16
5.0 TRUST AND DECEPTION INFERENCE USING LIAR’S DICE EXPERIMENT	17
5.1. Background in Liar’s Dice	17
5.1.1 Liar’s Dice Description	18
5.1.2 Opponent Strategies.	18
5.1.3 Incentives.	18
5.1.4 Hypotheses.	19
5.1.5 Opponent Strategies.	19
5.1.6 Incentives.	19
5.1.7 Hypotheses.....	20
5.2 Study Procedure on Amazon Mechanical Turk.....	20
5.2.1 Participant Recruitment.....	21

5.2.2	Recruitment Demographics.....	21
5.3	Results and Discussion	22
5.3.1	Discussion	26
6.0	ATTACKER’S SIMULATION ENVIRONMENT USING CM.....	27
6.1	Three Cognitive Models Using ACT-R	28
6.1.1	ACT-R.....	28
6.1.2	ToM CM.	29
6.2	Attacker’s Next Action Simulation using CM	29
6.2.1	Initialization	30
6.2.3	Prediction	32
6.2.4	CM Outputs.....	32
6.3	Evaluation.....	34
6.3.1	CM’s Computational Overhead	34
6.3.2	Action-Space Distribution.....	35
6.3.3	Prediction Accuracy	36
6.3.4	Discussion	37
7.0	OPTIMAL STRATEGY PRESCRIPTION USING BEHAVIORAL GAME THEORY	40
7.1	Strategic Response Prescription	40
7.1.1	Action Space Definition.....	40
7.1.2	Game Tree Generation	41
7.1.3	Utility Computation	42
7.1.4	Game Tree Pruning	42
7.2	Evaluation.....	43
8.0	PERFORMANCE EVALUATION FOR PHASE 1 WORKSHOPS	45
8.1	Helping Human Operators with Strategies.....	45
8.2	Helping Chatbots with Strategies	47
8.2.1	Higher Compliance Rate with CHESS Recommendations.....	47
9.0	CONCLUSIONS.....	49
10.0	REFERENCES.....	51
11.0	LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS	54
	APPENDIX A – Opponent Strategy Specifications in Liar’s Dice.....	55
	APPENDIX B – Liar’s Dice Interface and Survey Questions	57
	APPENDIX C – Liar’s Dice Cognitive Model	65

LIST OF FIGURES

Figure 1.	CHESS Architecture Overview.	1
Figure 2.	CHESS Technical Approach Overview.....	3
Figure 3.	High-Level Action Spaces.	5
Figure 4.	A Top-Level Architecture Data Flow Diagram Detailing the Interface Points of the CHESS System.	7
Figure 5.	High-Level Overview of MLN Procedure.....	9
Figure 6.	Message ID Clustering Output.	10
Figure 7.	Threat-Actor and Message ID Relationship Analysis using March 2019 Dryrun Results.	11
Figure 8.	Mean Cost of the Evaluation.	13
Figure 9.	Our MLN-Based Threat-Actor Co-Clustering Algorithm (left) vs. TRIAGE Algorithm (right).	15
Figure 10.	Email Content for the Cluster in Figure 12.....	16
Figure 11.	TRIAGE Message Example in a Threat-Actor Set.....	16
Figure 12.	Example Overlapping Threat-Actor Sets Analyzed by MLN and TRIAGE using JPL Abuse Dataset.	16
Figure 13.	Overview of Liar’s Dice Study Protocol.	21
Figure 14.	RT of Participants who Called (left) or Rolled (right) across Opponent Strategies and Point Incentives (n=429).	24
Figure 15.	RT of Participants who Called (left) or Rolled (right) in the First Ten Rounds (first half of the experiment, n = 429).....	24
Figure 16.	RT of Participants who Called (left) or Rolled (right) in the Last Ten Rounds (last half of the experiment, n = 429).	25
Figure 17.	Challenge Probability of Human Opponents in the First Half (left) vs the Second (right) Half of the Match.	25
Figure 18.	Histograms of Claim Increases that Participants in each of the Experiment Groups Exhibited.	26
Figure 19.	Prediction Capabilities of Participants in the 1st (left) and 2nd (right) Half of the Experiment.	27
Figure 20.	Data Flow Diagram for CM Module.	28
Figure 21.	Example Representation of an Event/Instance in Memory for ACT-R-based CM Module in CHESS.	30
Figure 22.	State-Action Mapping Diagram.....	32
Figure 23.	ToM CM Model Processing Time per Message (in seconds).....	35
Figure 24.	ToM CM Model Processing Time per Message (in seconds) for only the First 2,000 Messages.	36
Figure 25.	Mean Number of Messages per Starting Scenario from Model Predictions of Human Behavior.	36

Figure 26.	Mean State Values of the Attacker (left) and Defender (right) across Messages in the Example Scenario that Starts with an Attacker Action of Redirect-Request-Act and Defender Response of Challenge-Deny.....	37
Figure 27.	Predictive Accuracy of CMs across 28 Threads, after Collapsing across “Request”Actions.	38
Figure 28.	Example Game Tree Generation.....	43
Figure 29.	Game Tree Generation and Utility Computation.....	44
Figure 30.	Attacker-Defender Action Space Prescription Matrices for Different Tree Search Depth (i.e., number of next action spaces to explore) and Pruning Threshold (i.e., pruning actions with probabilities less than the indicated threshold).	45
Figure 31.	Information Captured using CHESS’s Recommendations during August 2019 Evaluation.	45
Figure 32.	Average Dialog Turns of CHESS’s Human/Bot Responses inside PANACEA and RECURSE Systems.	46
Figure 33.	The Distribution of Defender Actions Recommended by the CHESS System for all Published Threads across All Teams from the Winter 2020 Evaluation.	47
Figure 34.	The Distribution of Defender and Attacker Actions from all Published Threads across all Teams from the Winter 2020 Evaluation.....	48
Figure 35.	Compliance Alignment Rates given Search Depth of 2 vs. 4 on the CHESS System.	48

LIST OF TABLES

Table 1.	MLN and TRIAGE Performance Comparison using JPL Abuse Dataset.....	16
Table 2.	Demographic Distribution of our New Test Populations	23
Table 3.	Mean Number of Messages and Mean Changes in Attacker State Attributes for each Possible Starting Scenario.	39
Table 4.	Frequency of Attacker Responses to Defender Actions.	40
Table 5.	Frequency of Defender Responses to Attacker Actions.	40

1.0 SUMMARY

This report summarizes the Continuously Habituating Elicitation Strategies for Social-Engineering-Attacks (CHESS) system that is a partial Technical Area (TA) 2 solution with emphasis on eliciting information from social engineering attackers and engaging with them by prolonging conversations. CHESS is composed of three modules with the following capabilities:

- Multi-Layer Network (MLN) module is capable of extracting hidden attacker information from the incoming messages and identifying common threat-actor organizations using co- clustering algorithms.
- Cognitive Model (CM) module builds a forward- simulation environment to estimate an attacker's reasoning abilities and attack strategies.
- Game Theory (GT) module closely interacts with CM to build a game tree to estimate rewards for various sequences of attacker-defender interactions, and selects an optimal strategy that results in the attacker's compliance and engagement in a conversation.

Figure 1 is an overview of the CHESS architecture that highlights the message flows inside the CHESS system and across external collaborator's systems.

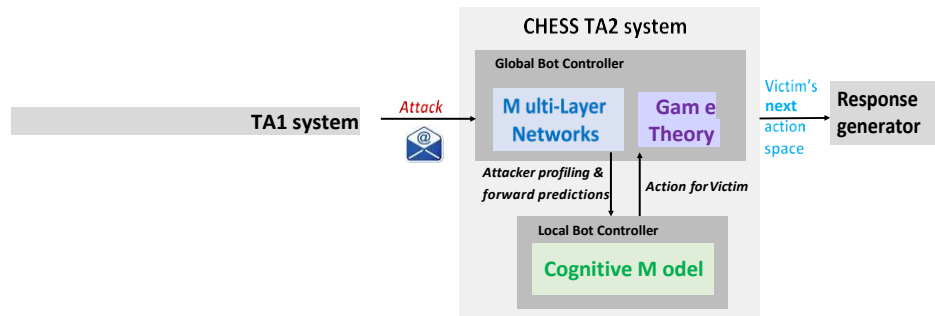


Figure 1. CHESS Architecture Overview.

Given an attack message from a collaborating TA1 system, MLN module processes the message content and header, if available, to extract hidden attacker's information, such as clusters of threat-actor organizations. CM module then builds a simulation environment using Adaptive Control of Thought – Rational (ACT-R), and GT module interacts with CM to check the attacker's reactions for possible victim's responses. In return, GT identifies the optimal strategy that provides the highest rewards and forwards the strategy to a collaborator's response generator.

To understand attacker's mental models and cognitive states, we also conducted an online human subjects experiment using a modified version of the classic Liar's Dice game to portray similar characteristics of phishing attacks, including deception and trust tactics that our cognitive model is based upon, as well as different incentive mechanisms for attacker role players vs. defender role players. Our experimental results confirm that (1) players adapt their deception tactics as they learn their opponent's persona and strategy, (2) players, especially defenders, heavily favor low-risk actions, adapting to in-game rules and structures, (3) players tend towards opportunistic deception when the projected risk is low and they tend to stay honest/loss-averse, and (4) metrics of trust (general and specific to the opponent) change as the player has different gameplay experiences within the construct of Liar's Dice.

As a partial TA2 system, we focused on developing CHESS to be modular such that integrating with others' systems is smooth. We successfully integrated to our collaborators' systems in all Phase-1 evaluations. According to the evaluation results and post-evaluation analysis in Phase 1, CHESS has promising directions in recommending strategies to help a defender system. In

particular, CHESS can ease the cognitive overload of human operators who must interact with multiple attackers. Even for the defender systems without humans in the loop, CHESS recommendations could have improved the compliance rate from attackers, acquiring additional information from them or having them perform certain actions as we intended. Specifically, the CHESS system consolidated more of its recommendations in alignment with successful strategies undertaken by the systems during the evaluation by 31%, resulting in 67% of all actions undertaken by performer systems that would lead to an attacker compliance.

2.0 INTRODUCTION

The prevalence of social engineering attacks continues to rise, underscoring the need for human-centric approach to cybersecurity. According to the 2019 Data Breach Investigation Reports [1], the top threat was the phishing attack, comprising 32% of confirmed breaches and 78% of cyber-espionage incidents. Furthermore, 28% of breaches involved malware infections and 29% involved stolen credentials, both of which are frequently accomplished through phishing attacks.

While the majority of existing work focuses on preventing users from becoming victims of such attacks [2], [3] [4], Defense Advanced Research Projects Agency's (DARPA) Active Social Engineering Defense (ASED) program aims at engaging with attackers, similar to scambaiters [6], and obtain information from the attackers. As a performer to address TA2 challenges of eliciting information from and engaging with attackers, we have developed a system called CHESS that combines the benefits of network analysis, cognitive modeling, and behavioral game theory to build optimal strategies on behalf of defenders as follows:

1. Since information about the attacker is likely to be limited and mostly fake, CHESS extracts hidden information about the attacker and identifies common threat organizations using multi-layer networks and clustering.
2. Since the attacker's true reasoning abilities and attack strategies are unknown, CHESS builds a forward-simulation environment of the attacker's behavior, strategy, and game environment using Adaptive Character of Thought - Rational (ACT-R)-based cognitive modeling.
3. Together with the forward-simulation environment, CHESS predicts attackers' strategies, and builds optimal strategies for defenders to induce the attacker to provide information and prolong the conversation using behavioral game theory.

Figure 2 illustrates an overview of CHESS's technical approach.

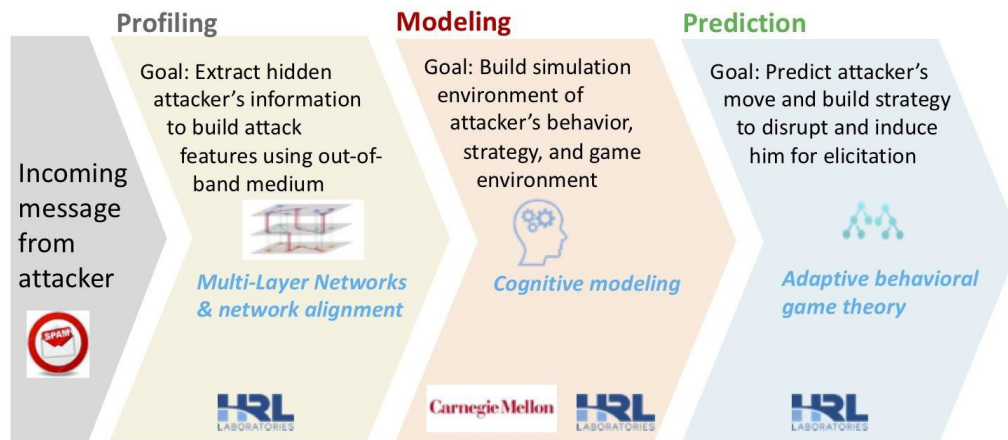


Figure 2. CHESS Technical Approach Overview.

Upon receiving a message from an attacker identified by a TA1 collaborator, CHESS first processes the message content and header, if available, to extract the hidden information about the attacker, such as clusters of threat-actor organizations. CHESS then builds a simulation environment using ACT-R cognitive modeling, and based on the predicted attacker's moves, CHESS identifies the optimal strategy that provides the highest rewards and forwards the strategy to a natural language generator.

2.1 Human Subjects Experiments.

Understanding attacker’s mental models and their cognitive states is crucial to accurately model the social engineering attackers and generate effective counter-strategies. As a result, one major challenge of the ASED program was collecting good representation of multi-turn attacker-victim interactions in realistic social-engineering attack settings. To address this, we put significant effort in designing human subjects experiments to identify trust and deception tactics that people use to achieve specific goals in mind using the Liar’s Dice game. Initially, we designed our experiment using the phishing-attack scenarios, asking a subset of study participants to mimic phishing attackers. However, the phishing attack by itself has many uncontrollable variables, including a wide variety of messages and responses from an attacker that can contain significantly different information from user to user. Furthermore, recruited participants were not likely to be experienced in launching phishing attacks nor properly trained to become attackers, having a high chance of experiencing controllability, complexity, and credibility issues. Liar’s Dice, on the other hand, portrays similar characteristics of phishing attacks, including deception and trust tactics that our cognitive model is based upon, without additional complexities in the game environment since a player has a fixed set of moves that he/she can take. This aspect removes many of the semantic ambiguities that arise when attempting to communicate to machines via natural language. Furthermore, Liar’s Dice supports modeling the attacker- victim environment using different incentive structures. We recruited participants using Amazon Mechanical Turk (MTurk) and observed that (1) players adapt their deception tactics as they learn their opponent’s persona and strategy, (2) players, especially defenders, heavily favor low-risk actions, adapting to in- game rules and structures, (3) players tend towards opportunistic deception when the projected risk is low and they tend to stay honest/loss-averse, and (4) metrics of trust (general and specific to the opponent) change as the player has different gameplay experiences within the construct of Liar’s Dice. Detailed study setting and results are in Section 5.0, Trust and Deception Inference using Liar’s Dice Experiment.

2.2 Threat-Actor Organization.

Multiple, distinct online identities, used by either one or more attackers, can belong to the same threat organization. Identifying such a cluster can help CHESS enhance the defense strategies. For example, CHESS can save defender’s resources by coordinating responses and asking for distinct types of information from attackers within the same organization or providing distinct strategies to prevent the threat-actor organization from noticing that it is being counter-attacked by the CHESS system. CHESS utilizes multi-layer co-clustering algorithms to detect various types of similarities from incoming messages and groups message senders into clusters. We performed evaluations using the TA3 performer’s data as well as a publicly available dataset to demonstrate the capability of our multi-network technology and algorithms for automated threat actor identification. Details about our algorithms and evaluation results are in Section 4.0, Threat-Actor Identification using Multi-Layer Network (MLN).

2.3 High-Level Action Spaces & Game Tree Generation.

CHESS is a partial TA2 solution that performs attacker analyses using high-level action spaces without Natural Language Processing (NLP) and Natural Language Generation (NLG) capabilities. Analyzing messages using the high-level action spaces have the following benefits: (1) As our core goal is to come up with the optimal strategy to interact with the attacker, mapping complex messages to simpler high-level action spaces empower CHESS to explore a game tree with various depths to come up with the best response given an objective; (2) High-

level action spaces empower CHESS to forward-simulate attacker’s responses and estimate levels of trust and engagement. For computational feasibility, generating a set of high-level action spaces that cover a wide range of messages without overlaps is a challenge, and we have identified ten action spaces for attackers and ten for defenders, as shown in Figure 3. Details on how cognitive modeling and game theory modules utilize these high-level action spaces to build a game tree and the evaluation results are summarized in Sections 6.0, Attacker’s Simulation environment using Cognitive Modeling and 7.0, Optimal Strategy Prescription using Behavioral Game Theory, respectively.

Attacker’s Action Space	Defender’s Action Space
REQUEST_INFO	REQUEST_INFO
REQUEST_ACT	REQUEST_ACT
ENGAGE	ENGAGE_WASTE_TIME
	ENGAGE_BUILD_TRUST
CHALLENGE_DENY	CHALLENGE_DENY
CHALLENGE_DISENGAGE	CHALLENGE_DISENGAGE
COMPLY_DISHONEST	COMPLY_DISHONEST
COMPLY_HONEST	COMPLY_HONEST
REQUEST_HANDOFF	REQUEST_HANDOFF
IGNORE	IGNORE
UNKNOWN	

Figure 3. High-Level Action Spaces.

CHESS uses high-level action spaces to predict attacker’s responses and build a game tree to select a strategy that is highly likely to result in the highest rewards.

2.4 Collaboration Efforts.

As a partial TA2 solution, CHESS has collaborated with other performers and complemented each other’s approaches as follows. (1) When our collaborator system identifies that the incoming message is from a social engineering attacker, we map the messages into one of the abovementioned attacker’s action spaces. (2) CHESS outputs the optimal action space on behalf of the defender and forwards it to our collaborator’s NLG module, which can be either a human mediator or a chatbot, and (3) the NLG module translates our action space into a response message to the attacker. Our evaluation results in Phase 1 confirm that CHESS provides cues that can guide the NLG modules from crafting messages in order to increase the chance of prolonging the conversation with the attacker and obtain identifiable information from the attacker. Details about our own system architecture and collaboration environment are described in Section 3.0, CHESS System Overview and Assumptions.

3.0 CHESS SYSTEM OVERVIEW AND ASSUMPTIONS

As a TA2 performer, our goals are to (1) elicit information from social-engineering attackers, (2) engage with the attacker and waste their time in order to prevent them from contacting additional victims, and (3) analyze the organizational structure of the online attacker identities to deduce if any of them belong to the same threat-actor organization. While achieving these goals, our desired security and performance properties include (1) protecting private information of defenders and (2) optimizing possibly limited amount of resources for handling multiple attackers. Example resources include fictitious accounts, attachments, or Uniform Resource Locators (URL) to passively collect information about their computing environment (e.g., Internet Protocol (IP) addresses, locations, browsers, etc.). Metrics for evaluation include (1) the number and types of information or “flags” that are captured from attackers for information elicitation, and (2) the number of dialog turns for the attacker engagement.

3.1 Assumptions.

As a partial TA2 solution, the core focus of which is to provide optimal strategies to interact with attackers while protecting victims, we make the following assumptions:

- We assume that our system is invoked with messages from attackers with high confidence (i.e., it is not our task to identify whether an incoming message is from a social-engineering attacker).¹
- Since CHESS performs using high-level action spaces, we assume that the incoming messages are mapped to one of the attacker action spaces, as shown in Figure 3. We also assume that our recommended action-space output is forwarded to an external NLG module that is capable of crafting a response guided by our recommended action-space output.
- We assume that a pool of available resources (e.g., fake uniform resource locators [URL], account information, etc.) exist that can be utilized as part of our recommended strategies (e.g., request an attacker to click a link to obtain browser information, request an attacker to fill in a document to obtain the address).

To meet these assumptions, we collaborated with other teams in the ASED program and we were integrated into their system architectures to participate in the Phase 1 evaluations.

3.2 System Overview.

Figure 4 describes the complete data flow within the CHESS system as well as the collaboration environment. Attacker messages identified by TA1 systems are submitted to the /message endpoint of the CHESS Representational State Transfer (REST) Application Programming Interface (API). The endpoint persists the data in the internal datastore and pushes the data to an internal publication-subscribe messaging service topic for incoming messages. Two parallel components listen to and pull data from the ta1.message topic of the messaging service:

¹ CHESS can be extended to identify an attacker’s message using the MLN feature that analyzes various aspects of message contents.

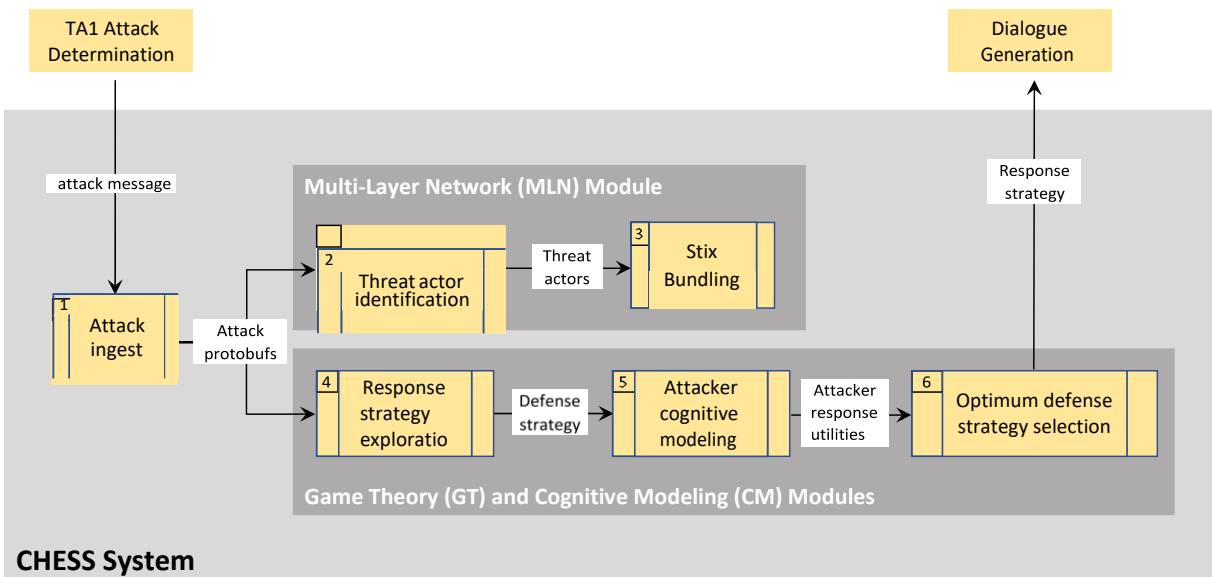


Figure 4. A Top-Level Architecture Data Flow Diagram Detailing the Interface Points of the CHES System.

1. MLN module identifies the hidden attributes about the attacker in terms of threat actors, and puts them into the Structured Threat Information Expression (STIXTM) bundle format and submits the STIXTM bundles to a STIXTM endpoint.
2. GT and CM modules build a game tree as part of the response strategy exploration, and prescribe the optimal response on behalf of the victim/defender. The trust and engagement estimations from the CM and the optimal response action are used to filter and rank a set of goals to pursue at this turn of the conversation. The strategy and goal list are then made available via the /responses endpoint of the REST API for use by an external dialogue generator.

The REST API serves a single external interface point to the CHES system for high-level strategy and goal recommendations. REST is a ubiquitous standard for interoperability between services that external collaborators can submit data to and asynchronously query for responses. With the default exploration parameters, the CHES system takes approximately one second to generate responses. The REST API schemas are automatically generated and documentation is published in Swagger User Interface (the root endpoint) and ReDoc (the /doc endpoint) so that external users can easily integrate the CHES recommendation system into their system pipelines.

Internally, the CHES system leverages a microservice architecture to allow for flexible configurations and horizontal scalability. The CHES system performs inter-module data passing using data structures specified in the Google Protocol Buffers proto3 format. Data structure definitions are kept in a central repository which the continuous integration system monitors for changes and triggers compilation using the Google-provided proto3 compiler

whenever modifications are made to the repository. The compilation artifacts for all languages used in the CHESS system are archived and hosted for incorporation into CHESS module builds so that all modules maintain a consistent internal data structure specification.

Data passing between modules is brokered by an internal RethinkDB database instance. RethinkDB provides a change notification mechanism by which clients can define topics at the table, record, or field level and subscribe to those topics to be notified of relevant changes. This allows us to both persist relevant data as well as create a publication-subscribe mechanism by which multiple modules can be notified of new incoming data and pass their results on to other components of the CHESS system. The data processing pipeline for the CHESS system is therefore inherently flexible and accommodates a variety of configuration possibilities. The main data pipelines in the system are detailed in Figure 4.

In the following sections, we describe the core modules in detail. In Section 4.0, we delineate our MLN module's co-clustering algorithms for threat-actor identification, and in Section 5.0, we introduce our human subjects experiments to gain insights on trust and deception tactics that people use to achieve specific goals in mind as a proxy to understand unobservable attackers' mental models. In Section 6.0, we describe the CM module to predict attacker's responses, and in Section 7.0, we describe how our GT module interacts with CM to build a game tree and prescribe the optimal strategy. We evaluate CHESS's performance during Phase 1 evaluations in Section 8.0, Performance Evaluation for Phase 1 Workshops.

4.0 THREAT-ACTOR IDENTIFICATION USING MLN

To infer the hidden attacker profiles and identify the threat-actor organizational structures behind incoming messages from attackers, we designed and developed an MLN that consists of layers, each of which represents a particular attribute that can be inferred solely from incoming messages. In general, nodes and edges are defined in a heterogeneous manner from one layer to another, and we construct a network for each layer and accurately reflect the similarity and relatedness of the corresponding feature characteristics.

As shown in Figure 5, we selected five features that can be extracted from incoming messages and built a layer for each: (1) message identification (ID), (2) topic modeling, (3) sentiment analysis, (4) term frequency-inverse document frequency (TF-IDF), and (5) readability test. Note that the message ID layer is email-specific as we can get the message IDs only from the email headers, but the rest of the layers are agnostic to message types, covering emails, short message services (SMS), and LinkedIn messages to name a few. After building and training each layer, we map each incoming message to the network and embed the output into an inner product space. Based on the closeness or the distance among the outputs in the inner product space, we cluster them to identify the common threat-actor organizations.

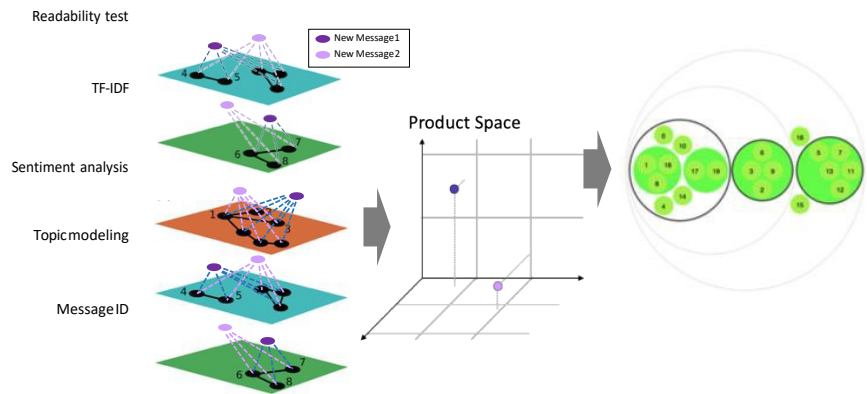


Figure 5. High-Level Overview of MLN Procedure.

We train five layers based on similarities in message ID, topic modeling, sentiment analysis, TF-IDF, and readability test. Upon receiving new messages, we map them to the trained layers and embed them into an inner product space. Then we cluster them to identify threat-actor organizations.

4.1 Layer Development and Training

We first describe five layers that comprise our MLN module and present how we combine their outputs to infer common threat-actor organizations based on incoming attacker messages.

4.1.1 Message ID.

A unique field in the email header is a message ID field (e.g., 201704241508.v3OF8Y3D029698@mail.jpl.nasa.gov). In practice, message IDs are *statistically unique*, and different mail platforms use different algorithms for generating them. As such, we may be able to gain insights about the source of each email using the message ID field. For instance, some automatic mail platforms might contain the strings JavaMail, Rafael.Alanis (JIRA), and other message ID generation algorithms choose to begin IDs with a string of either letters or numbers. These differences allow message IDs to be clustered, even though they contain randomness. As an

example of the latter case, the message ID <CAB6t5t- mEB5Bm6FT0o6rQOzufVmc UcArPJLugP4AALLL =agxA@mail.gmail.com> will likely be in a different cluster than <20190305232717.E2D28877E4@gitbox.apache.org>

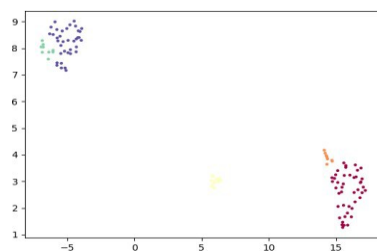


Figure 6. Message ID Clustering Output.

Using message IDs from the emails in JPL Social_Engineering and Recon datasets, we identified two clusters that may represent two distinct attacker personas.

To train this layer, we extracted message IDs from emails, and clustered them using Levinshstein distance. As we receive new message IDs, they can be assigned to existing clusters, or we can re-cluster. We evaluated this idea using two datasets from Jet Propulsion Laboratory (JPL): Social_Engineering and Recon datasets that were distributed in early Phase 1. We used Uniform Manifold Approximation and Projection (UMAP) to create lower dimensional embeddings, but care must be taken since UMAP does not preserve density.

Manual inspection of cluster results seems to show reasonable results, and two example clusters from the subset of JPL datasets are shown in Figure 6. Below is the list of unique Message-IDs in each cluster.

Cluster 1:

15a38021a90-314e-5457@webprd-a32.mail.aol.com
15d13b352df-ea3-15510@webprd-a20.mail.aol.com
df19c05dc45e43d3a06845c46e95d455@AP-EHUB-SP01.RES.AD.JPL
15ad14b9556-2fb-3a9@webprd-a22.mail.aol.com

Cluster 2:

1436074201.3854768.1497901304636.JavaMail.zimbra@mediassistindia.com
1829060328.20641570.1491042650003.JavaMail.root@giga.ocn.ne.jp
863071590.8992607.1493036175983.JavaMail.root@embargmail.com
1436074201.3854768.1497901304636.JavaMail.zimbra@mediassistindia.com
1831004574.1094096.1486389358500.JavaMail.zimbra@brisyarlah.co.id

We further analyzed threat-actor and message-ID relationships and Figure 7-(a) shows the threat actor- message ID relationships from the entire dry run results and Figure 7-(b) shows an isolated view of some relationships in Figure 7-(a). As Figure 7 shows, multiple threat actors are bundled possibly to indicate that they collaborate.

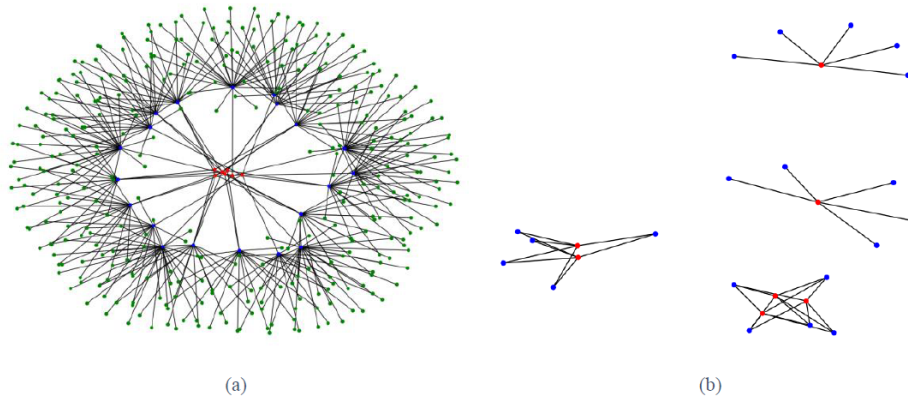


Figure 7. Threat-Actor and Message ID Relationship Analysis using March 2019 Dryrun Results.

Green, blue, and red nodes represent observed data, intrusion-set, and threat actor relationships, respectively. (a) Overall relationship. (b) Isolated view of some relationships in (a). As Figure (b) shows, some threat actors are “linked” together.

4.1.2 Topic Modeling.

A topic modeler, which was part of the Memex program², uses Probabilistic Latent Semantic Analysis (PLSA) [7] [8] [9], and this framework requires the number of topics to be chosen a priori. We used eight topics from the Jet Propulsion Laboratory (JPL) datasets. The topics tend to split emails up on subject matter and some example topic words are as follows:

1. payment, court, legal, case, make, pick, charges, employer, step, order
2. money, transaction, trust, late, chemical, country, sum, deposit, fund, cleaning
3. e-mail, intended, information, error, attachments, prize, prohibited, confidential, action, received
4. required, toy, account, data, information, security, file, officially, writing, pay
5. consignment, personal, summit, attendees, family, hotel, world, provide, receive, details
6. forwarded, life, mail, real, private, contact, years, country, day, sending
7. received, people, spam, stop, fund, payment, decided, office
8. gov, soldier, active, service, respond, involved, safe, stolen, content

To train a topic model, the plain-text bodies are extracted from a corpus of attack emails, and the extracted bodies act as inputs to the PLSA framework, which further extracts topic words and generates a probabilistic model of the form $P(w|z)$ where z is a topic, and w is a word.

² <https://github.com/mitll/topic-clustering>.

When a new message arrives, it can be assigned a topic distribution, which can be interpreted as probability of topic membership. Constructing the topic distribution is performed by the following equation:

$$P(z_i) = \exp \left[\frac{1}{N} \sum_j^N a_j \log P(w_j|z_i) \right]$$

where a_i is a weight for word i , and N is the total number of words. If a word is not present in the evaluation message, its weight is set to zero.

4.1.3 Sentiment Analysis

The sentiment network is trained by evaluating a sentiment trajectory for each message in the training corpus. A sentiment trajectory is the sentiment score for a document using a sliding window along the document content. By default, we used a window size of five, which is then interpolated into a sequence of ten bins. This ten-element vector represents the sentiment for that message. The network is made of messages as nodes and links according to distance between trajectory vectors.

New messages are evaluated according to their trajectory distances.

4.1.4 TF-IDF.

TF-IDF determines similarity between content of a set of documents. To train this layer, TF-IDF is run on the training corpus, resulting in a network with messages as nodes and links according to the document similarity. A “TF-IDF Matrix” is also generated, which is used to map new messages into the trained model.

4.1.5 Readability Test.

Each message is scored using the Flesch-Kinkaid readability index. The network is created using a node for each message in the training corpus, with links given by difference in readability score.

4.2. Mapping and Embedding Incoming Messages to Inner Product Space

When a new message arrives, it can be assigned to a topic distribution, which can be interpreted as probability of topic membership. Constructing the topic distribution is performed by the following equation:

$$P(z_i) = \exp \left[\frac{1}{N} \sum_j^N a_j \log P(w_j|z_i) \right]$$

where a_i is a weight for word i , and N is the total number of words. If a word is not present in the evaluation message, its weight is set to zero.

Embedding a new message requires evaluating it with respect to each network, producing a new node in that network, then computing the distance from the evaluated node to other nodes. This produces a set of distance vectors for each network. Computing the outer product of all vectors places the new message into a higher dimensional space as a single point. This process is outlined in detail below.

Given a set of training objects θ and evaluation objects Φ , we wish to use training objects to create a

set of networks $\mathcal{G} = \{G_1, \dots, G_n\}$. Each network is created using a generator function

$$(\theta) \mapsto (V_i, E_e),$$

where V_i is a vertex set with m_i vertices and E_i is an edge set.

A second generator function operates on evaluation objects $\varphi_j \in \Phi$ such that

$$h_i(G_i, \varphi_j) \mapsto v_i \in \mathbb{R}^{m_i},$$

where v_i is a representation of φ_j in relation to G_i . The exact relation depends on the properties of h_i , but it is assumed to be akin to a similarity measure between φ_j and elements of V_i .

In order to embed all v_i into one space, we construct a new vector using a sequence of tensor products

$$z_j = \bigotimes^n p_i,$$

where z_i We are now free to cluster $z_j \forall \varphi_j \in \Phi$ using any metric-based clustering technique.

4.3 Clustering

Clustering in the product space is performed in two steps. Since the product space is high dimensional, and it is unlikely that all of the dimensions are important, performing the dimensionality reduction technique first can improve the clustering results. The technique used to reduce dimensions is UMAP [8]. After this step, clustering is performed using Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [9].

Upon applying the abovementioned steps to messages, our MLN module groups messages with similar traits, indicating that those messages in one group originate from the same threat-actor organization with high probability.

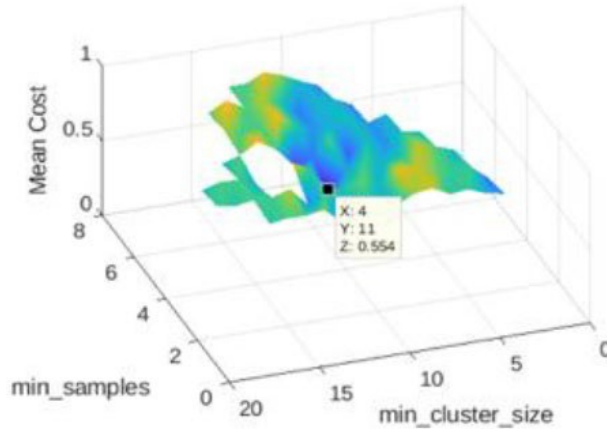


Figure 8. Mean Cost of the Evaluation.

Assigning clusters to threat actors is evaluated using linear sum assignment (Munkres) and the mean cost of that assignment is plotted for many choices of clustering parameters.

4.4 Evaluation

4.4.1 Initial Evaluation.

We used an attacker-email dataset from the TA3 performer, along with the groundtruth information for our initial assessment of the accuracy of the claimed multi-layer network- based attacker organization identification. For this initial evaluation, we used message ID and topic modeling layers only. The dataset consist of 77 email messages, and our goal is to match our identification results with the groundtruth representation provided by the government team. Due to the nature of the small data size, most of the analyzed clusters only contained one message, resulting in the average F-score of .667. If a cluster contains more than one message, our approach achieves the F-score of .923. Determining an alignment between clusters and attackers is not trivial. To create a mapping and evaluate performance, a linear sum assignment algorithm (Munkres) was applied to the two sets.

Performance was measured by taking the mean of the resulting assignment costs.

As shown in Figure 8, a sweep of clustering parameters shows a relative optimum combination at a minimum cluster size of 11 and a minimum number of samples of four. These two parameters are specific to the HDBSCAN implementation, which imposes the restriction that the minimum number of samples is less than or equal to the minimum cluster size. Our evaluation indicates that our system is capable of clustering and identifying attackers that belong to the same threat organization given sufficient amount of data. Hence, acquiring good-quality datasets with accurate groundtruth would be a must for further evaluation.

4.2.1 Addressing Dataset Limitations.

One programmatic challenge is acquiring datasets with groundtruth to evaluate our algorithm; although some types of messages from cyber attackers (e.g., phishing emails) are publicly available, no groundtruth information is available that identifies which message senders belong to the same organization. To address this major limitation, we attempted to build our own groundtruth dataset, following the previously-published approach [10]. More specifically, we built our own (1) web scraper to download emails from www.419scams.org, (2) parser to extract features, and (3) algorithms to aggregate emails from the common threat actors.

1. **Scraper:** To the best of our knowledge, www.419scams.org prevents downloading more than 1 email at a time. Since we need to gather a large dataset, we investigated the pattern of the website and wrote a recursive web crawler that identifies the relevant links and recursively clicks on those links to access the actual emails. Our scraper downloads the emails, with a unique ID per email. We successfully scraped 390,000 malicious emails spanning over the past ten years.
2. **Parser:** Since all the scraped emails were in the html format, we wrote an html parser and extracted the following features:
 - “From” field
 - “Reply to” field
 - “Date” field
 - “Subject” field

- Body of the email
- Any phone number and/ or fax number in the body
- Any email address found in the body
- Any URL found in the body, along with the domain information.

We converted the extracted features into the JavaScript Object Notation (JSON) format and insert them into our RethinkDB database for further analysis.

3. **TRIAGE Algorithm:** To aggregate the emails from the common threat actor, we started implementing the TRIAGE algorithm [11] [12]. TRIAGE involves constructing a graph with weighted edges for each feature and aggregating the graphs. For aggregation functions, we implemented Ordered Weighted Average (OWA) and Choquet Fuzzy Integral (CFI), which is considered as one of the best integrals.

4.2.2 MLN vs. TRIAGE Evaluation.

We compared the accuracy of our current MLN clustering approach with the state-of-the-art TRIAGE approach to identify common threat actors behind multiple messages. Figure 9 shows the comparison of two approaches. Both approaches attempt to identify the common attack sources using different axes; our approach uses similarities among message topic, sentiment, and metadata to create a threat-actor hierarchy, while TRIAGE uses similarities among message content, such as phone numbers or URLs to create sets of messages from common threat actors.

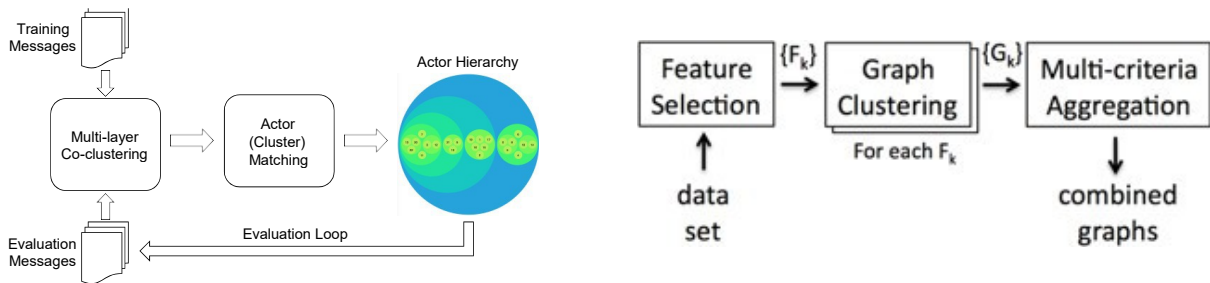


Figure 9. Our MLN-Based Threat-Actor Co-Clustering Algorithm (left) vs. TRIAGE Algorithm (right).

Both approaches attempt to identify the common attack sources using different axes.

Our approach uses similarities among message topic, sentiment, and metadata to create a threat-actor hierarchy, while TRIAGE uses similarities among message content, such as phone numbers or URLs to create sets of messages from common threat actors.

We ran each algorithm using the same JPL abuse dataset and we aligned the resulting actor sets in order to check for the overlap between the results from two approaches. As summarized in Table 1, we observed a good amount of overlap, which is interesting considering how different the two approaches are.

Compared to MLN, TRIAGE algorithm seems to be more conservative according to the number of identified actor sets.

Table 1. MLN and TRIAGE Performance Comparison using JPL Abuse Dataset.

	MLN	TRIAGE
Number of actor sets	135	26
Messages in actor sets	985 (~7.3/set)	329 (~12.7/set)
Overlapping sets	8	
Messages in overlapping sets	69 (7.0%)	83 (25.2%)

We further investigated specific sets for manual comparison. Figure 12 is an example of overlapping threat-actor sets that were identified by MLN and TRIAGE, and Figure 10 is the email content for the message IDs in Figure 12. Note that TRIAGE does not use message ID as a feature and two non-matching message IDs do not appear to be part of the same actor set.

Unlike TRIAGE, our MLN technique uses topic modeling as one of the features, resulting in distinctions. In another set with high overlap between TRIAGE and MLN, all messages that TRIAGE selected is shown in Figure 11. In addition to the message shown in Figure 11, MLN also put the message shown in Figure 13 in the same cluster.

From: IHS Markit [mailto:ihsemail@ihs.com]
Subject: IHS Markit Courtesy Notice

Your privacy and email preferences are important to IHS Markit

IHS Markit is a business information and analytics company focused on key industries and issues shaping today's global landscapes. IHS Markit offers critical information to enable our customers' success in these key sectors...

Figure 10. Email Content for the Cluster in Figure 12.

From: Haley Hoffmann <hhoffmann@keyence.com>
Subject: Quick survey with gift

Hi Annette,
We need your help to better understand the microscope market and how it relates to our latest digital microscope...

Figure 11. TRIAGE Message Example in a Threat-Actor Set.

MLN	TRIAGE
004867924106\$ccb2a66b5d2bb299e5@khtte	1bc386527c394c679d1ed9739f6a0f35@1770935282
1fdde33c9fc3413387aa623f18bf989f@1770935282	1fdde33c9fc3413387aa623f18bf989f@1770935282
681941740a3f4302890e90e8430dab22@1770935282	681941740a3f4302890e90e8430dab22@1770935282
7eec8ff7710c41c0acda7f5d844d450e@1770935282	6c4760d78d424e6391dc29728d0bc927@1770935282
93a3adf186ee4219801be4555386701d@1770935282	7eec8ff7710c41c0acda7f5d844d450e@1770935282
aab6529d3675bd877963a652d.ec1d263f8c.	93a3adf186ee4219801be4555386701d@1770935282
20170508170046.631315718f.3df76b23	aa11082522234b10a8ec26dd45777daf@1770935282
@mail121.at1111.rsgsv.net	adb98cd0bf1247fb837540d3b6268573@1770935282
adb98cd0bf1247fb837540d3b6268573@1770935282	ef795d7dc9e3421c8dc61a020b3bb748@1770935282
ef795d7dc9e3421c8dc61a020b3bb748@1770935282	

Figure 12. . Example Overlapping Threat-Actor Sets Analyzed by MLN and TRIAGE using JPL Abuse Dataset.

From: Pat Smith [mailto:pat.smith@writingassistance.com]
Subject: Technical Writers, Copywriters, Editors | On or Off-Site |

Dear Ronald,
Writing Assistance, Inc. is considered a nationwide leader in providing
Technical Writers and Trainers to thousands of companies in many different
industries... If you would like to learn more about our services or discuss
the specifics of what you need, please let me know a convenient time to
schedule a call...

Figure 13. MLN Message Example in a Threat-Actor Set.

4.2.3 Limitations.

This approach has two major limitations. (1) We attempted to use TRIAGE output as groundtruth.

However, TRIAGE has many proprietary parameters that need to be properly configured for given datasets, and we did not have access to the proprietary information. Hence, verifying TRIAGE output for correctness is non-trivial. (2) The data was not clean as it came from the abuse emails forwarded by JPL employees. In many cases, emails contained text from the employee without the original headers.

4.2.4 Future Work

Leaveraging the TRIAGE algorithms, augmenting MLN with phone numbers and URL layers would help MLN behave similar to TRIAGE. Given the lack of groundtruth, one approach to resolve such a challenge is turning both MLN and TRIAGE at the same time to yield a maximum overlap and coverage and hence to acquire promising results.

5.0 TRUST AND DECEPTION INFERENCE USING LIAR'S DICE EXPERIMENT

Theory of Mind (ToM) relates to the cognitive capability to mentally model others' intents, in order to plan one's own actions accordingly [13]. In the context of the ASED program, ToM plays an important role as attackers have a mental model of victims when they build their strategies. In order to better understand and reveal what influences the attacker's mental model of the victim, with special emphasis on trust and deception signal detection, and how attackers use it to develop their attack strategy, we designed and conducted a human subjects experiment.

Initially we developed protocols that ask experiment participants to play the roles of attackers and victims and exchange emails, in line with the ASED environment. However, major risks were involved due to the complexity of the protocols and uncontrollable variables, including a wide variety of messages and responses that participants may generate. Furthermore, as the recruited participants are unlikely to be experienced in launching phishing attacks nor properly trained to become attackers, we decided to reserve the complex protocols and instead develop a streamlined experimental approach using the abstraction of the CHESS task as a game theoretic model using Liar's Dice. Liar's Dice portrays similar characteristics of social engineering attacks, including deception and trust tactics without adding additional complexities in the game environment, as a player has a fixed set of moves that he/she can take. Most specifically, Liar's Dice features the same tradeoffs as the CHESS task between near-term incentives (i.e., comply and tell the truth to avoid being caught lying) and long-term consequences (i.e., cumulatively reveal too much information and give the game away). Furthermore, we can model the attacker-victim environment in Liar's Dice using different incentive structures.

In Section 5.1, we provide background information of the Liar's Dice game that we used in the study, along with the hypotheses that we aimed at testing, and in Section 5.2, we describe our study procedure. In Section 5.3, we analyze the results and discuss insights that we gained from this experiment.

5.1. Background in Liar's Dice

Games of dice have been frequently used in experiments to measure team performance and interpersonal trust for many decades in psychological experiments due to clear structure and easy adaptation to a variety of scenarios. In this study, we used our own custom version of Liar's Dice in order to gain insights about human behavior within the context of ASED.

In Liar's Dice, players alternate the opposing roles of "Roller" and "Caller," the former rolling a set of dice at the beginning of each round, making a numeric claim. This claim must be the actual value of the dice roll or less to be considered truthful; a claim larger than the dice roll is untruthful. The Caller then has an option of challenging the Roller's claim, or accepting it; accepting the claim advances to the next round where the roles of Roller and Caller are reversed and the steps repeated, and no points are awarded. If the claim is instead challenged, the value of the dice roll is revealed and the truthfulness of the claim is determined. Should the claim be revealed to be truthful, the Roller wins, points are awarded to the Roller, and the match ends.

Should the claim be untruthful, the Caller wins, the points are awarded to the Caller, and the match ends. Thus, each match may consist of a variable number of rounds, terminating with the assignment of points to one of the players. A rigorous definition of these roles can be found below.

5.1.1 Liar's Dice Description.

Liar's Die is a two-player, constant-sum game of bluffing. In one of its most basic version [14], there are two players, $I = \{I, II\}$, and two roles, $R = \{pr, pc\}$ called the roller and the caller. Let $\theta : R \times Z^+ \rightarrow I$ be a function that specifies the player associated with the role in the first argument during the round indicated in the second argument.³ Let $X = \{2, 3, \dots, 12\}$ be the outcomes of 2 fair dice.

The game begins in round $t = 1$ with (say) player I as the roller, $\theta(pr, 1) = I$, and player II as the caller, $\theta(pc, 1) = II$. Player I observes the outcome $x_1 \sim X$ of a fair die, and then chooses a $y_1 \in X$ and makes the claim that x_1 is at least as large as y_1 . Player II, who does not know the value of x , must either accept or challenge player I's claim. If player II challenges the claim then the game is over and player II wins if $x_1 < y_1$, and I wins if $x_1 \geq y_1$. If player II accepts the claim then the game continues to round $t = 2$ where the roles are reversed; $\theta(pr, 2) = II$ and $\theta(pc, 2) = I$. In round 2, player II rolls the die and observes x_2 and then chooses a $y_2 \in X$ such that $y_2 > y_1$ and claims that $x_2 \geq y_2$. Now, player I can either accept or challenge player II's claim where, as before, a challenge ends the game with player I as the winner if $x_2 < y_2$ and player II as the winner if $x_2 \geq y_2$. The game proceeds with alternating roles until either player claims $y_t = 12$, at which time the player in role pc automatically challenges the claim and the outcome is determined.

5.1.2 Opponent Strategies.

In order to observe (1) how participants form game strategies and (2) changes in participants' trust as the game proceeds, we developed strategies for the computer agent that varied the gameplay across participants. We implemented two strategies: aggressive (risk-seeking) vs. defensive (loss-averse).

- Aggressive/risk-seeking strategy allows greater probability to return an untruthful claim, or challenge study participants' claims.
- Defensive/loss-averse strategy results in less probable deceptive claims and a decreased chance to challenge study participants' claims.

In order to identify changes in behavioral outcomes as the opponent strategy changes, we designed the computer agent to start with an initial strategy and then switch to the opposite strategy midway through the experiment. For example, an agent may start playing aggressively and switch to be defensive/loss-averse, and vice versa. We can then examine what occurs with human behavior and strategies as this transition occurs. Detailed strategy specifications are in Appendix A.

5.1.3 Incentives.

In the ASSED environment, attackers and defenders have different incentives as their interactions proceed. Attackers aim to lure defenders into their social engineering attacks and exploit defenders *quickly* (e.g., get information from victims, install malware in victims' organization when they click an attachment, etc.). Defenders also aim at ultimately exploiting attackers, but in order to achieve their ultimate goal, they are more likely to start building relationships with attackers first and waste their time. In our study, we wanted to observe these different behaviors and strategies when participants play the role as an attacker vs. defender. To meet this objective, we altered the point structure as follows:

³ Below, we use the shorthand $\theta_r^t \equiv \theta(pr, t)$ and $\theta_c^t \equiv \theta(pc, t)$ to denote the players occupying the respective roller and caller roles in round t .

- The attacker paradigm incentivizes quick gameplay, and the participant earn more points if they manage to end matches quickly (in few rounds). A maximum of five points can be earned, with one point removed each round for a minimum of one point.
- The defender paradigm incentivizes dragging on gameplay, awarding more points if the participant can win matches over many game rounds. The participant begins with one point possible, and gains an additional point for each round for a maximum of five points.

Unbeknownst to participants, our computer opponent always operates according to the opposite point incentives, i.e., when the participant is given the incentive structure of an attacker, the computer opponent is given the incentive structure of a defender, and vice versa.

5.1.4 Hypotheses.

Using Liar's Dice, we aimed at measuring the following:

- Does the incentive structure have an effect on the study participant's performance?
- Does a shift in strategy from the computer opponent result in reduced prediction capability or performance on the part of the study participant?
- Does the participant's ability to learn the game (i.e., slope of performance) depend on the incentive structure or the opponent strategy?
- Do prediction or task behaviors correlate with known trust scales?

5.1.5 Opponent Strategies.

In order to observe (1) how participants form game strategies and (2) changes in participants' trust as the game proceeds, we developed strategies for the computer agent that varied the gameplay across participants. We implemented two strategies: aggressive (risk-seeking) vs. defensive (loss-averse).

- Aggressive/risk-seeking strategy allows greater probability to return an untruthful claim, or challenge study participants' claims.
- Defensive/loss-averse strategy results in less probable deceptive claims and a decreased chance to challenge study participants' claims.

In order to identify changes in behavioral outcomes as the opponent strategy changes, we designed the computer agent to start with an initial strategy and then switch to the opposite strategy midway through the experiment. For example, an agent may start playing aggressively and switch to be defensive/loss-averse, and vice versa. We can then examine what occurs with human behavior and strategies as this transition occurs. Detailed strategy specifications are in Appendix A, Opponent Strategy Specifications in Liar's Dice.

5.1.6 Incentives.

In the ASED environment, attackers and defenders have different incentives as their interactions proceed. Attackers aim to lure defenders into their social engineering attacks and exploit defenders *quickly* (e.g., get information from victims, install malware in

victims' organization when they click an attachment, etc.). Defenders also aim at ultimately exploiting attackers, but in order to achieve their ultimate goal, they are more likely to start building relationships with attackers first and waste their time. In our study, we wanted to observe these different behaviors and strategies when participants play the role as an attacker vs. defender. To meet this objective, we altered the point structure as follows:

- The attacker paradigm incentivizes quick gameplay, and the participant earn more points if they manage to end matches quickly (in few rounds). A maximum of five points can be earned, with one point removed each round for a minimum of one point.
- The defender paradigm incentivizes dragging on gameplay, awarding more points if the participant can win matches over many game rounds. The participant begins with one point possible, and gains an additional point for each round for a maximum of five points.

Unbeknownst to participants, our computer opponent always operates according to the opposite point incentives, i.e., when the participant is given the incentive structure of an attacker, the computer opponent is given the incentive structure of a defender, and vice versa.

5.1.7 Hypotheses.

Using Liar's Dice, we aimed at measuring the following:

- Does the incentive structure have an effect on the study participant's performance?
- Does a shift in strategy from the computer opponent result in reduced prediction capability or performance on the part of the study participant?
- Does the participant's ability to learn the game (i.e., slope of performance) depend on the incentive structure or the opponent strategy?
- Do prediction or task behaviors correlate with known trust scales?

5.2 Study Procedure on Amazon MTurk.

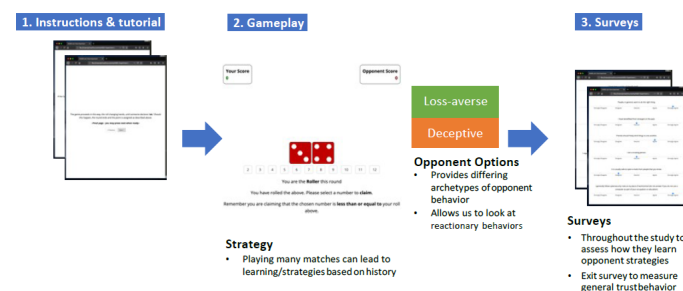


Figure 14. Overview of Liar's Dice Study Protocol.

- (1) When participants join, we introduce the Liar's Dice game and provide three sample rounds to help them understand the game. (2) Participants proceed to the actual study that consists of 20 match rounds against our computer opponent that switches its strategy after the 10th round. (3) For every match, we ask participants who become rollers to predict whether or not their opponent will accept or challenge. After 5th, 10th, and 15th round, we also display the 1-page survey to assess how they learn the opponent's strategies. When they are done with 20 rounds, we display an exit survey that measures general trust behavior.

We designed our study to consist of 20 matches of Liar's Dice as described previously. Participants started with instructions and three practice matches before proceeding to the actual study. Each participant was randomly assigned to a different incentive structure to mimic the behavior of an attacker or a defender, and the reward structure was displayed accordingly. The opposite reward structure was assigned to the computer opponent without being displayed to the participant, and the computer opponent's initial strategy was randomly assigned as either aggressive or defensive. Within each match, each participant was assigned to be a Roller or a Caller, and the roles passed between the human and computer players as described above. After playing as a Roller, participants were asked to predict how the opponent player would respond (i.e., challenge or accept). Further, after completing every five matches, participants were asked about their beliefs about their opponent's strategy. At the end of the study, participants completed a survey about general trust and deception and to get feedback about the game itself, in addition to the standard demographic questions. Participants earned points when they won the match according to the incentive structure for their role. Figure 14 is an overview of the study protocol, and sample interfaces and survey questions are in Appendix B, Liar's Dice Interface and Survey Questions.

Along with the challenging/accepting behavior and survey questionnaire answers, we also recorded the amount of time that participants took before they proceeded to the next page. This timing information may help us analyze the relationship between deception and incentives/player roles.

5.2.1 Participant Recruitment.

In order to collect results from a large participant pool, participants were recruited to complete an online experiment via Amazon MTurk. Qualification flags were specified to ensure that results were of high-quality: (1) the location of Turkers was limited to US only, (2) the Human Intelligence Task (HIT) approval rate must be greater than 95%, and (3) the number of prior HIT approvals must be greater than 5000. Upon joining the task, we displayed the consent form that Turkers must agree in order to proceed. The study was designed to be completed in less than 30 minutes, and participants were paid a base rate of \$7.00 and we provided \$3.00 as a performance bonus for earning above 25 points.

5.2.2 Recruitment Demographics

A survey was performed on each participant following the experiment in order to gain demographic/ethnographic information about our study population. The demographics of our study population is summarized in Table 2. Our algorithm assigned appropriate numbers to each experiment group and demographics were distributed evenly across each experiment condition. Because the population of the group consists of active Amazon MTurks, the gender distribution skews male, and the age range is relatively young; participants 25-34 and 35-44 years of age comprised the majority of our study population.

Table 2. Demographic Distribution of our New Test Populations

		Aggressive Attacker	Loss-Averse Attacker	Aggressive Defender	Loss-Averse Defender	Total
Gender	Male	59	71	74	71	275
	Female	57	52	51	50	210
Age	18-24	5	3	3	3	14
	25-34	34	42	55	55	186
	35-44	43	42	32	36	153
	45-54	20	22	17	14	73
	55-64	10	10	17	10	47
	65+	4	5	1	3	13
Education	Less than HS	0	0	0	0	0
	Highschool	12	14	25	17	68
	Some College	42	39	27	30	138
	Bachelors	53	55	59	59	226
	Post-Graduate	9	16	14	15	54
Income	0-25 K	21	27	16	16	80
	25-50 K	32	46	37	37	152
	50-75 K	29	23	35	25	112
	75-100 K	20	13	22	23	78
	100-150 K	11	10	11	15	47
	150 K+	3	5	4	5	17
Marital	Single/Never Married	44	58	57	55	214
	Married/Partnership	60	54	56	59	229
	Widow/Widower	0	2	2	0	4
	Divorced	12	10	9	6	37
	Separated	0	0	1	1	2
Employment	Employed	73	81	86	88	328
	Self-employed	19	29	21	20	89
	Looking for work	5	4	6	2	17
	Not looking for work	3	0	4	2	9
	Homemaker	10	3	2	4	19
	Student	2	0	1	2	5
	Military	0	0	0	0	0
	Retired	3	2	5	1	11
	Unable to work	1	5	0	2	8
Ethnicity	White/Caucasian	87	101	92	91	371
	Hispanic/Latino	9	7	8	6	30
	African American/Black	5	7	11	8	31
	Indigenous Peoples	0	0	0	2	2
	Asian/Pacific Islander	14	7	11	8	40
	Other	1	2	3	6	12

5.3 Results and Discussion

Out of 720 people recruited, 526 completed the task, and the average time per study was 25 minutes. Among those who completed the task, 97 participants had irregularities within their results, and 429 provided perfectly clean data. For this analysis, we used only the clean data, and

our results are divided into four experimental groups (based on initial opponent strategy & participant incentive structure): defensive & attacker, aggressive & attacker, defensive & defender, aggressive & defender ($n = 115$, $n = 110$, $n = 105$, $n = 99$, respectively). Each participant earned \$7.00 base rate, with a bonus of \$3.00 if they scored above 25 points.

We first examined the effects of opponent type and role type on reaction times across experiment groups. We use reaction time (RT) as one potential analog of strategic conflict, as more complex evaluation typically requires additional processing and hence extended RT. In Figure 15, the reaction times are tabulated for each of our four experiment groups for each role type, “caller” or “roller”. The results show that, for both Caller and Roller roles, reaction times are greater when under an attacker incentive structure than a defender incentive structure, regardless of opponent strategy. This observation reflects that the defender incentive structure is a fairly straightforward choice to minimize one’s own bids and avoid challenging the opponent’s bids.

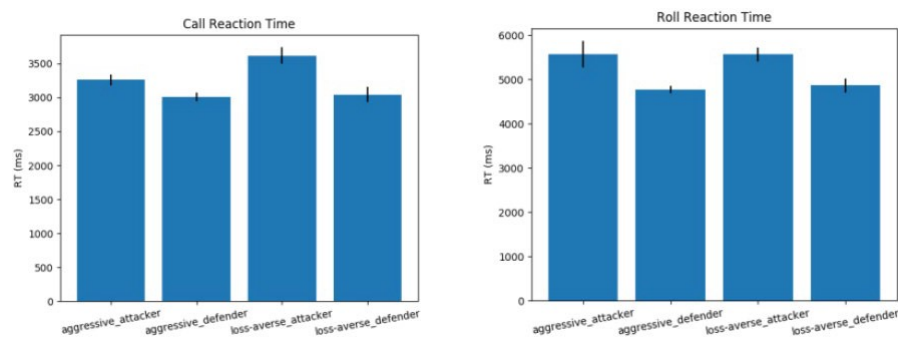


Figure 15. RT of Participants who Called (left) or Rolled (right) across Opponent Strategies and Point Incentives (n=429).

Participants generally have increased RT when placed in the attacker point structure: $F=187.7$, $p<0.001$ for Call RT (left), $F=120.0$, $p<0.001$ for Roll RT (right).

We also analyzed how changing the opponent strategy midway through the experiment affects participants’ response time. For the first ten matches, RT is generally higher for all participants, possibly because they were learning the mechanics of the game (Figure 16). On the other hand, based on the changes in RT, we noticed some differences in the participants’ calling behavior during the second half of the experiment. More specifically, participants’ RT increased against computer agents with aggressive roles (i.e., aggressive attackers and aggressive defenders), as shown in Figure 17. From this result, changing the opponent strategy seems to affect the user behavior, even over the relatively short time course of the experiment. Specifically, facing an aggressive attacker when one only has experience facing a defensive one raises challenging, risky decisions that require additional reflection.

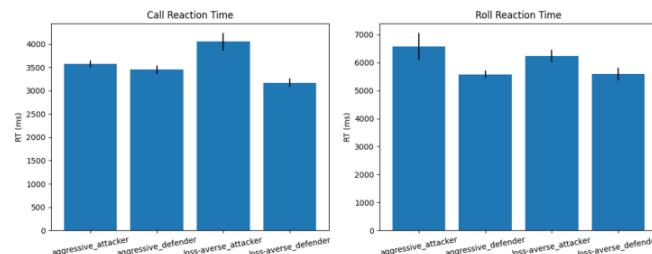


Figure 16. RT of Participants who Called (left) or Rolled (right) in the First Ten Rounds (first half of the experiment, n = 429).

These numbers appear similar to the global results above, but with overall higher RT in general.

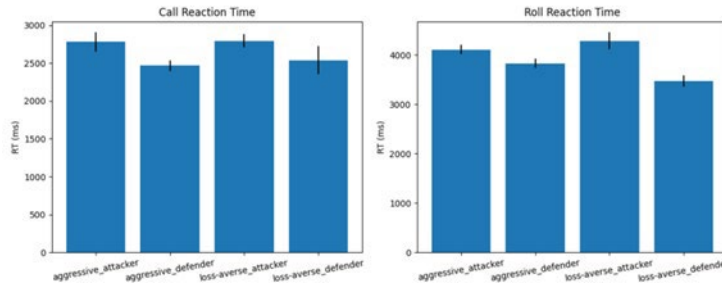


Figure 17. RT of Participants who Called (left) or Rolled (right) in the Last Ten Rounds (last half of the experiment, n = 429).

Differences in RT observed in the first half of the experiment are reduced, such as the RT of aggressive attackers compared to aggressive defenders.

These results fall in line with our expectations, as ending a match quickly requires careful strategy while delaying one is (in most circumstances) a matter of deliberately issuing low claims that are generally truthful, and hence, defenders have a simpler task. As a caller, it is in the defender's best interests to blindly accept low claims as the point structure rewards additional rounds of play. More specifically, attackers have fewer examples to work from, and each round is more critical, which we observed as increased RT.

Interestingly, the changes in the opponent behavior seems to reverse this trend of increased RT – the trends seen in the first match are less

extreme, which could indicate that some difference has been noticed by the player, resulting in the differing gameplay. A similar effect can be seen when examining the likelihood that a player challenges the computer opponent's claim; although the challenge probability is higher for participants playing against the aggressive opponent

as expected (i.e., aggressive opponents take more risks), participants change their behavior by the end of the match based on the changing opponent strategies (Figure 18).

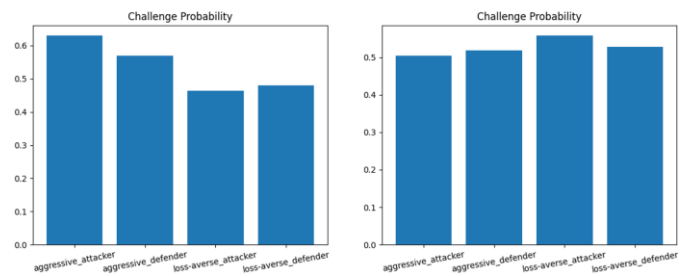


Figure 18. Challenge Probability of Human Opponents in the First Half (left) vs the Second (right) Half of the Match.

As expected, we observed a large tendency to challenge aggressive opponents initially. However, once the opponent strategy changes, participants adjust to the opponent's new strategy.

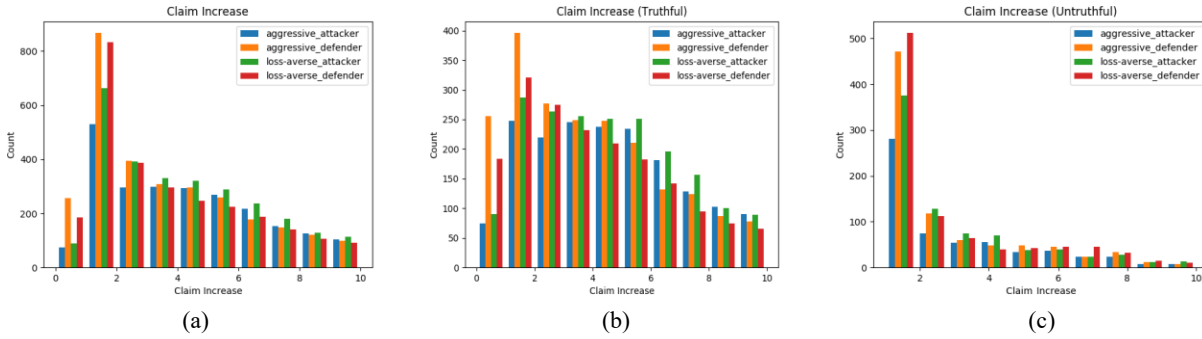


Figure 19. Histograms of Claim Increases that Participants in each of the Experiment Groups Exhibited.

(a) The left shows the overall, (b) middle shows the distribution for all truthful claims, and (c) the right shows the distribution for all untruthful claims. Note that each match may have a differing number of rounds; as a result, n in groups are not equivalent. Results show that on the whole, human participants gave large truthful claims, with the defender groups issuing deceptive claims far more frequently for low-risk deceptive claims.

Additionally, as the veracity of a claim is only known after a challenge is issued, at least a subset of human participants seems to be challenging the opponent sufficiently to obtain some predictive power over their responses. This tradeoff between exploration and exploitation is a trademark of human decisions under uncertainty. This observation provides some good evidence that users can infer their opponents' behavior, allowing us to study how this confidence may degrade as the game context and opponent behavior changes in the middle of gameplay.

We investigated more specific behaviors by examining how participants increase their claims over the course of the match. As shown in Figure 19-(a), incremental increases in claim (increase of 0 in starting bids or 1 in other bids) are most common in the defender point structure, while similar spikes are not seen for other groups, which falls in line with our point-structure strategy. Further examination of the claim behavior by separating true and false claims reveals that defender role-playing participants tended to give the lowest-risk deceptive claims while the attacker role-playing participants did not exhibit the same behavior, as shown in Figure 19-(b) and Figure 19-(c), respectively. According to our game point structure, attackers are rewarded for short matches, which incentivizes making higher claims in early rounds to bring the game to a quick conclusion, while defenders more often find themselves with having to make untruthful claims because of high bids earlier by aggressive opponents. This behavior is shown in Figure 19-(b), in which large, truthful claim increases (e.g., > 5) seem to be preferred by attacker players rather than defender players. This observation was not the case for deceptive responses – neither the attacking nor defending group seemed to favor large, deceptive claim increases.

To understand the subjective perspective of the human participants' predictability of the computer agent's strategy, we directly posed a survey response after each round asking whether participants felt that their opponents' actions were predictable. As shown in Figure 20, the groups with members with good prediction capability ($R = 0.8$) is initially limited to participants of either role facing an aggressive opponent, or a defender playing against a defensive opponent in the first ten rounds. However, when we investigate the latter ten rounds of the match, almost all groups have equivalent probability of an accurate prediction. Although anti-correlated predictions are reduced in the second half of the match, the (formerly) aggressive-opponent facing groups have greater likelihood of making anti-correlated predictions. By contrast, the

participants facing (formerly) loss-averse opponents have greater likelihood of making highly correlated predictions. This could imply significance in the order of opponents faced and needs additional scrutiny.

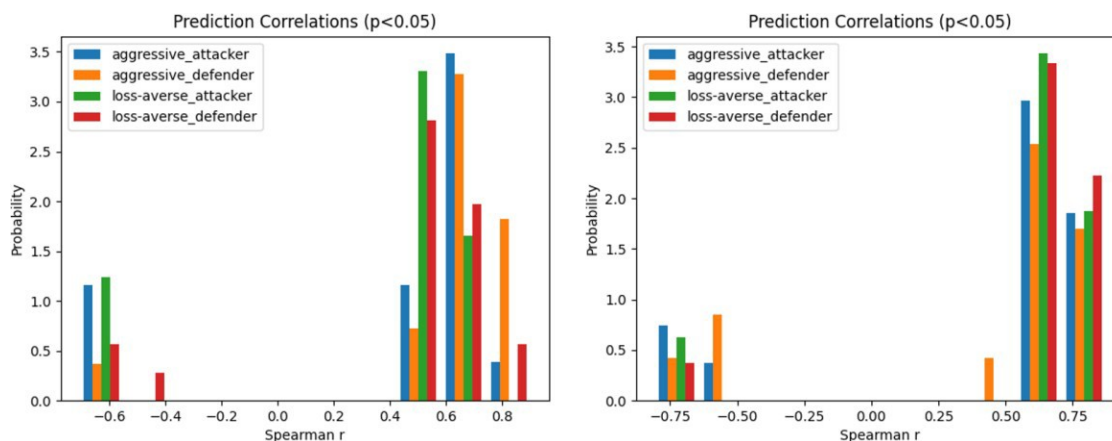


Figure 20. Prediction Capabilities of Participants in the 1st (left) and 2nd (right) Half of the Experiment.

Probability indicates the likelihood that a given participant in a group falls into one of the Spearman r score bins ($p < 0.05$) which in turn indicates the correlation between user prediction and opponent action. Negative r values indicate anti-correlation.

5.3.1 Discussion.

In summary, our experimental results confirm that (1) players adapt their deception tactics as they learn their opponent's persona and strategy, (2) players, especially defenders, heavily favor low-risk actions, adapting to in-game rules and structures, (3) players tend towards opportunistic deception when the projected risk is low and they tend to stay honest/loss-averse, and (4) metrics of trust (general and specific to the opponent) change as the player has different gameplay experiences within the construct of Liar's Dice.

These real-world behaviors can help us to inform our cognitive models, as we have attempted to replicate real-world examples of changing user behaviors triggered by cyberattacks, which we framed as truth/deception in a variety of task contexts. In addition to the quantitative data presented here, these results can be unified with correlates derived from subjective measures in order to connect the user's internal perceptions and states to external behaviors and performance metrics that would ultimately determine the efficacy of an attack. In Appendix C, Liar's dice Cognitive Model, we provide how these results have been applied to a cognitive model for the Liar's Dice game.

6.0 ATTACKER'S SIMULATION ENVIRONMENT USING CM

Modeling a social engineering attacker's mental and cognitive states is extremely beneficial in building defender strategies. Our CM module attempts to estimate the cognitive state of the attacker during their interactions with CHESS, and this is key for CHESS agents to forward-simulate the continued interactive game play in conjunction with the GT module.

We have investigated two core components that can approximate attacker's cognitive states in the context of ASER:

- *Level of trust* approximates how much the current attacker is likely to trust the victim. High level of trust could indicate that the attacker may behave in an unlikely manner considering that he/she is an attacker. Examples include providing requested information that can be sensitive (e.g., attacker's bank account number) or performing requested action that can be risky (e.g., open an attachment with an embedded IP tracer).
- *Level of engagement* approximates how much the current attacker is engaged in the victim's conversation. High level of engagement could imply that the attacker is attentive and likely to respond fast.

A safe strategy for a defender is to increase both levels of trust and engagement, and high levels of trust and engagement would be a good indicator that defenders can try more aggressive strategies to extract information from the attacker. On the other hand, high level of trust but low level of engagement could provide additional time for the victim to generate strategies for this attacker or to engage with other attackers without worrying about losing contact with the current attacker. Conversely, low level of trust but high level of engagement may indicate that the attacker has a high chance to behave dishonestly, helping the victim judge the credibility of the attacker's responses. In addition, high level of engagement indicates that the victim can practice a strategy to communicate more frequently to build trust with the attacker.

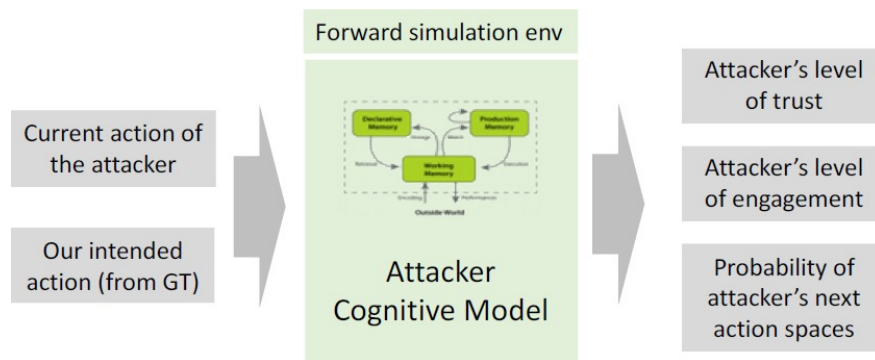


Figure 21. Data Flow Diagram for CM Module.

Given the incoming action of the attacker, along with our intended action (from our Game Theory module) in response to that attacker's incoming action, CM predicts (1) attacker's level of trust, (2) attacker's level of engagement, and (2) probability distribution of this attacker's next action.

CM acts as the attacker simulation environment for CHESS, and actively interacts with GT as shown in Figure 21. More specifically, given the incoming action of the attacker, along with our intended defender action (from our GT module) in response to that attacker's incoming action,

CM predicts (1) attacker's level of trust, (2) attacker's level of engagement, and (3) probability distribution over possible next-actions for this attacker.

We developed three versions of the attacker's cognitive model using ACT-R [15]. We introduce three CM versions in Section 6.1. In Section 6.2, we describe how CM predicts attacker's next actions given a defender's response. In Section 6.3, we provide evaluation results.

6.1. Three Cognitive Models Using ACT-R

We developed (1) baseline, (2) baseline+learning, and (3) ToM cognitive models using ACT-R. We provide brief background information about ACT-R, and highlight three model's characteristics in this section.

6.1.1 ACT-R

ACT-R has been used to create a wide variety of highly accurate models of human cognition.⁴ ACT-R provides a theoretical framework to model complex human cognition and processes such as memory retrieval, pattern matching, and decision making [15]. ACT-R integrates symbolic knowledge and sub-symbolic computations that result in statistical adaptivity that reflects individual differences and emergent cognitive biases, and accounts for training effects by modeling learning processes [16].

ACT-R operates over prior experiences or instances that are stored in memory; for every decision made, the model stores a trace of its decision and future decisions are made by retrieving information from these instances. Instances include the contextual features of the decision, the action taken, and the utility (or outcome) of the action. ACT-R also supports a functionality is called blending, which produces continuously varying answers by retrieving not just one but an aggregate of a set of related facts.

In the cyber domain, cognitive models have been shown to accurately predict human attacker behavior [17] [18] [19]. Additionally, using techniques such as model-tracing [20] [21] that use the real history of attacker behavior to make personalized predictions, the cognitive model can be run online, alongside human attackers to make predictions of their behavior and adapt defenses in real-time. This form of personalized, adaptive defense shows improvement compared to traditional, static defenses [19].

Three versions of cognitive models. In ASED, we extended this prior research of adaptive defense to help build richer and more sophisticated predictions of attacker actions that also include representations of mental states (e.g., trust and engagement levels). We developed three versions of the CM as follows:

1. **Baseline Cognitive Model.** The baseline model predicts what action to take by retrieving (via blending) an action given the opponent's prior action and the attacker's current state. The baseline model does not learn anything, as it receives no feedback regarding the groundtruth actions of the defender and attacker.
2. **Baseline+Learning CM.** Built on top of the baseline model, the

⁴See <https://act-r.psy.cmu.edu/publication/>

baseline+learning model learns patterns of play as attacker-defender-attacker action sequences. The baseline+learning model retrieves an action given the proposed defender action and the prior attacker action. If there is no matching chunk in declarative memory (i.e., retrieval fails), the model defaults to the baseline model and retrieves an action via blending. The model will efficiently learn attacker patterns, as previously demonstrated in other models of adversarial interactions, while constantly adapting to changes in attacker strategies.

3. **ToM CM.** ToM is a theory to attribute and understand the mental states (e.g., beliefs, desires, intentions, perspectives) of others. As an input, ToM CM takes as inputs the actions of both attacker and defender agents, reasons about the likely defender response to potential actions, and their associated utility, to make decisions, and outputs probability distributions over the possible actions and state dynamics for trust and engagement level.

6.1.2 ToM CM

To better predict the next action of an agent (i.e., attacker) based on the current state of the agent and the observed action of the other agent (i.e., defender), we developed an Instance-Base Learning (IBL) CM that utilizes ToM processing and is built in the ACT-R cognitive architecture [22] [15]. ToM CM takes as inputs the actions of both attacker and defender agents and outputs probability distributions over the possible actions and state dynamics for trust and engagement level.

According to IBL, decisions are made by generalizing across past experiences [22]. ACT-R provides a framework for simulating human cognitive processes, such as retrieval, that operate over these experiences, or instances, stored in memory. That is, for every decision made, the model stores a trace of its decision and future decisions are made by retrieving information from these instances. Instances include the contextual features of the decision, the action taken, and the utility (or outcome) of the action. The CM uses ACT-R's blending mechanism [16] to retrieve an expected outcome from these past instances to make a decision, which produces continuously varying answers by retrieving not just one but an aggregate of a set of related facts.

15e-f39...	redirect-request-act	ignore	0.78	0.92	redirect-request-info	0.87	0.56	Challenge-deny	0.56
Attack-id	Prior-action-self	Action-other	Trust	Engage-ment	Action-self	Trust-other	Engage-ment-other	Response-other	Utility

Event Representation (Context)

Figure 22. Example Representation of an Event/Instance in Memory for ACT-R-based CM Module in CHEAD.

Figure 22 shows an example representation of an instance in memory in the ToM CM for CHEAD. The contextual features of the phishing scenario for the attacker decisions include: a unique identifier of the attack thread, the attacker's prior action, the defender's current action, the attacker's current trust and engagement levels, the attacker's action, the predicted defender trust and engagement levels, the defender response to the attacker's action, and the utility of the attacker's action based on the defender's response. The representation for the Baseline and Baseline+Learning models contain a subset of this representation: namely the Attack-id, Action-Other, Trust, Engagement, and Action-Self. The utility refers to the value of the information that would be gained if the opponent performed a certain action.

6.2. Attacker's Next Action Simulation using CM

We describe the CM process using the ToM version in this section. Using the action spaces that we defined in Figure 3, we defined an attacker's possible responding actions to a victim's action, as well as how the victim's actions affect attacker's trust and engagement levels. An example of this process is shown in Figure 23.

6.2.1 Initialization.

The state-action mappings were used to initialize the model with a set of knowledge that represents the possible responses that could be made based on the agent's current state (engagement level and trust level) and the opponent's action. The result is an initial memory of 124 instance from which the model can base its initial decisions. These initial decisions can be quickly overcome by actual experiences as the model learns because, like humans, memories decay with time and play less of a role in retrieval processes after the model gains experience from feedback of actual human decisions. These initial instances do not contain information about the opponent's state, their actions, or the utility of those actions. Another ten instances define the utility of each possible defender action. Therefore, the model initially only knows the actions that could be taken given its current trust and engagement levels an opponent action and the utilities of each opponent action. However, with this knowledge the model can infer the other knowledge described in Figure 22, such as the predicted defender trust and engagement, through ToM processing.

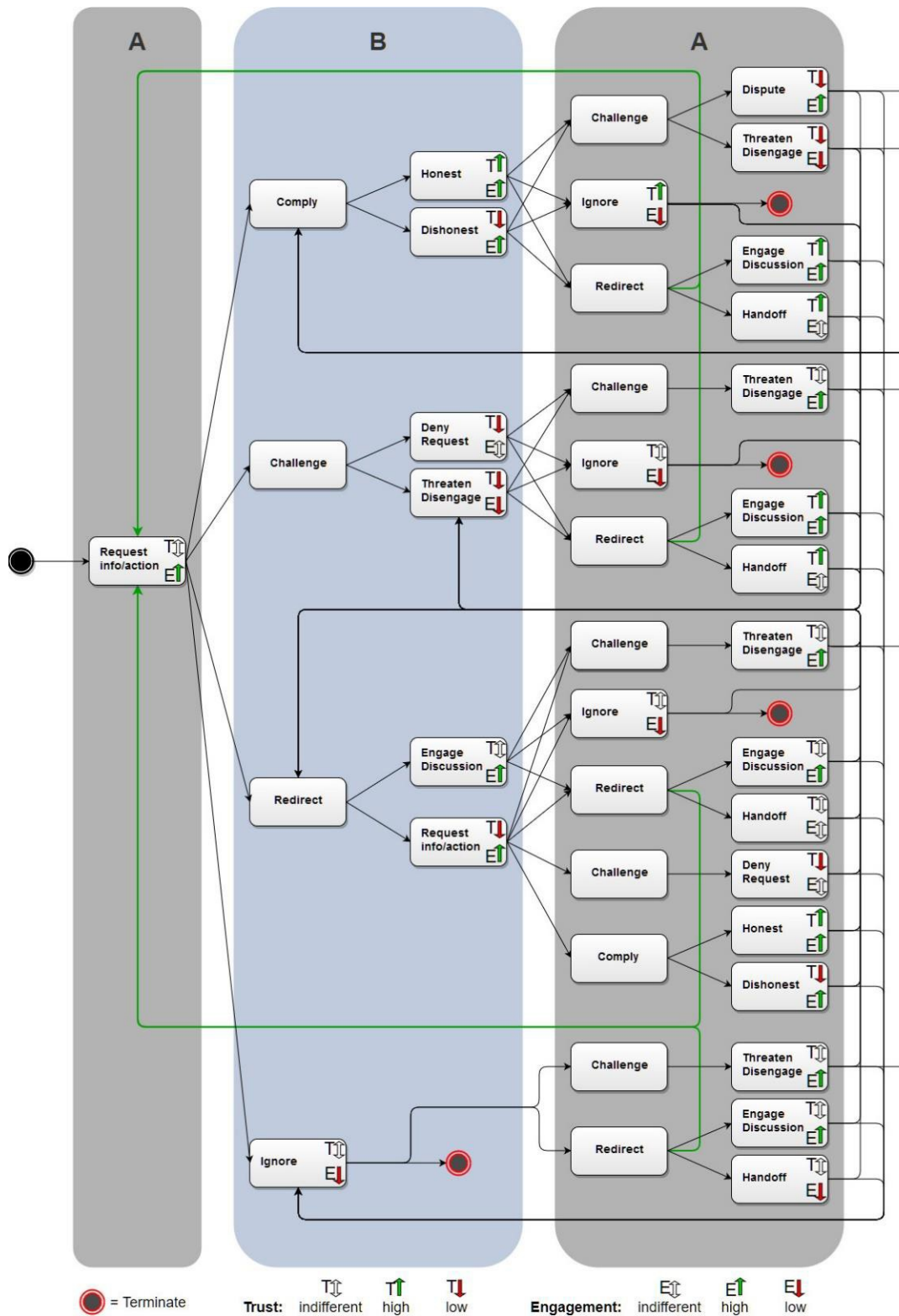


Figure 23. State-Action Mapping Diagram.

The arrows indicate the current states of the agents required to perform that action; red is 'low' while green is 'high' and white is 'no change'.

6.2.3 Prediction.

To predict the attacker's actions, the model takes as input the prior attacker action, a possible defender response to that action, and the current attacker trust and engagement levels (which are derived from the change the defender response has on the prior trust and engagement levels). The model then infers the utility of opponent responses to its own actions to choose the best action.

- In *Step One*, the model infers the current state of the defender given the defender's action by introspection of what its own state would be had it made that decision. ("If this was the response I made to that action, what would have been my state?"). The model reverses its roles during memory retrieval: it enters its own previous action as the 'opponent-action' and the defender's action as the 'action' and retrieves, via blending, an estimate of the defender's trust and engagement. While the most active instance is usually retrieved, a blending process [16] can also be applied that returns an aggregate value reflecting the similarity between the values of the content of all chunks, weighted by their retrieval probabilities reflecting their activations and partial-matching scores, and is computed by the following equation.

$$V = \underset{i=1}{\operatorname{argmin}} \sum_{i=1}^n P_i \cdot (V_t, v_{it})^2$$

The value, V , is an interpolated value based on matching chunks ii , weighted by their retrieval probability P_i . The retrieval probability is influenced by power laws of frequency and recency, according to ACT-R theory of memory retrieval [15]. The similarity function, $(V_t, v_{it})^2$, is used to compare memory chunks v_{it} and candidate consensus values V_t . In the simplest case, where the values are numerical and the similarity function is linear, the process simplifies to a weighted average by the probability of retrieval:

- In *Step Two*, the model cycles through each possible response to the defender's action, inferring the defender's response and an associated utility. First, a possible attacker action is retrieved from memory (without blending). Combined with the opponent's estimated current state derived from step one, the model retrieves, via blending, an opponent response. Next, the model retrieves, via blending, the utility value of that opponent response given the attacker action and predicted defender response. The possible utilities span from 1 to -1 and are static and computed ad hoc. In the future, it may be possible to elaborate on the utilities based on the actual information the defender gives to the attacker, rather than the action, and variable depending on the current situation and other factors. The current method, however, provides a reasonable representation from which the model can learn. After generating an expected utility for each possible response, given the defender action, the action with the highest expected utility is selected.

6.2.4 CM Outputs.

After making a prediction, the model produces output that includes (1) the attacker's own state (engagement and trust levels), (2) its estimate of the opponent's current state, and (3) the set of

own action, most likely opponent reaction, and expected utility. Example model output in List Processing LISP notation is as follows:

```
( (ACTION_DISTRIBUTION
  (DISTRIBUTION
    ((UNKNOWN NA 0.0)
      (REDIRECT-ENGAGE-DISC-WASTE REDIRECT-ENGAGE-DISC-WASTE -0.063639484)
      (DISENGAGE REDIRECT-ENGAGE-DISC-TRUST 0.21119438)
      (COMPLY-HONEST CHALLENGE-THREATEN-DISENGAGE -0.147798)
      (REDIRECT-ENGAGE-DISC-TRUST REDIRECT-ENGAGE-DISC-TRUST 0.014095526)
      (COMPLY-DISHONEST REDIRECT-REQUEST-INFO -0.058835577)
      (CHALLENGE-DENY REDIRECT-REQUEST-INFO -0.015272696)
      (REDIRECT-REQUEST-INFO REDIRECT-REQUEST-INFO 0.04303847)
      (REDIRECT-HANDOFF REDIRECT-HANDOFF -0.061705478)
      (CHALLENGE-THREATEN-DISENGAGE CHALLENGE-THREATEN-DISENGAGE -0.25461653)
      (REDIRECT-REQUEST-ACT REDIRECT-REQUEST-ACT -0.062362887)
      (IGNORE IGNORE -0.15715441))
    (SPACE . SPACE_ATTACKER))
  (REQUEST_UID . 86323)
  (STATE_SELF (ENGAGEMENT . 0.8403535) (TRUST . 0.2958767))
  (STATE_OTHER (ENGAGEMENT . 0.6535555) (TRUST . 0.4716667)))
```

The model learns over time. When the attacker makes a new action, that action is encoded in the model's memory, along with the attacker's current state values, the prior defender action, and the prior attacker action. These memories therefore have an effect on retrieving possible defender actions in the future.

Over time, the model learns what actions the defender is likely to take given an attacker action. After making predictions of the attacker, the game tree uses that information to inform its recommendations for defender actions. After the defender makes an action, the model is provided ground truth feedback and an instance is stored that represents the attacker's action, defender's response, and the associated utility of that response for the attacker. These instances influence future retrievals when generated expected utilities of the predicted defender responses to the possible attacker actions. Additionally, the attacker's trust and engagement levels are updated when a defender makes an action and is based on the influence the defender's action has on the attacker's prior trust and engagement levels, as defined by the state-action mapping.

More specifically, the states are updated according to the following function:

Given the current state S of trust or engagement, one of the following equations define the new state S' depending upon the effect the opponent's action has on the current state, as gleaned from the action-state mappings (the effect on the current state can either increase I , decrease D , or stay the same S – this course-grained distinction could be made finer as needed):

- $S'_I = S + (random(1 - S)) + / - (random 0.2)$
- $S'_D = S - (random S) + / - (random 0.2)$
- $S'_S = S + / - (random 0.2)$

The *random* function generates a value between 0 and the number given. S' is limited between 0.0 and 1.0. Additionally, S' is then decreased by an exponential factor if the current action is the same as the last n actions:

$$○ S'' = S' - (S' * ((1 - e^{-n}) - \sqrt[n]{e^{-n}})^2)$$

6.3. Evaluation

We emphasized two performance criteria: (1) computational load over time should be reasonable, and (2) CM should not output unexpected results.

6.3.1 CM's Computational Overhead.

Since ToM CM builds a memory over time, it was critical to stress- test CM and examine the computational load over time. To stress-test the ToM attacker CM, the model was pitted against a simulated defender. The baseline defender model is similar to the ToM attacker CM; however, the model is much simpler in how it generates actions; it simply retrieves (via blending) an action given the attacker's prior action and the defender's current state. Like was done for the attacker model, an action-state mapping was defined for the possible defender actions.

The ToM CM of the attacker was run against the defender baseline model for 894 interactions, and a combined 14,480 messages. The first attacker action and defender response were provided as input for the start of each interaction. Then the attacker model began by simulating an action, followed by the defender, and so on until the attacker model decided to disengage or its level of engagement fell below 0.1. The total number of attacker messages produced by the model was 6,672, excluding the initial messages provided (not simulated). The computer processing time was captured for each attacker message produced. The simulation was run on a MacBook Pro 2016, intel i7-7700HQ 2.8 Ghz, 16GB RAM, Windows 10 Pro.

Figure 24 shows the processing time across all attacker messages, and Figure 25 zooms in and shows the processing time across only the first 2,000 messages. With more messages, processing time increases as seen by the general linear trend in processing time across messages. The spikes are indicative of individual attack threads. Within a thread, the processing time is much faster for the first message, then becomes much slower with more messages. When starting a new attack thread, the initial processing time drops again but is longer than the initial time of the previous thread.

After 14,480 messages, the simulation crashed from a heap exhaustion. An easy remedy is to simply increase the heap size, but regardless, the model shows that its performance is very fast even with thousands of messages processed. From a computational perspective, the ToM CM is rather lightweight.

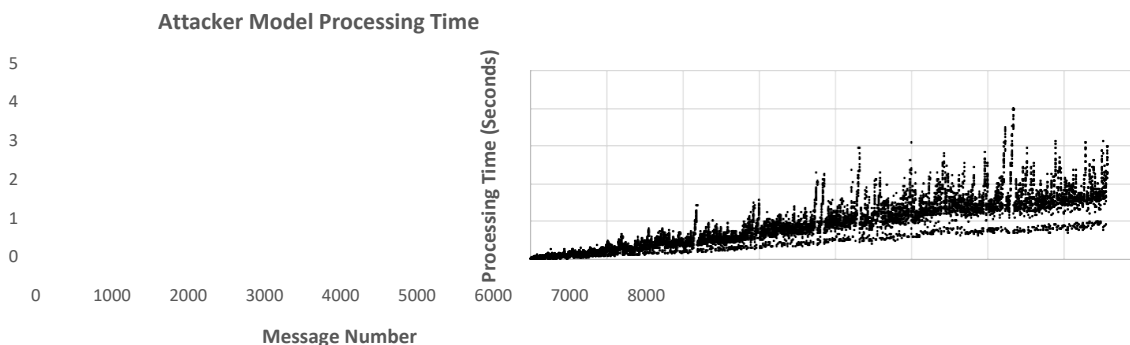


Figure 24. ToM CM Model Processing Time per Message (in seconds).

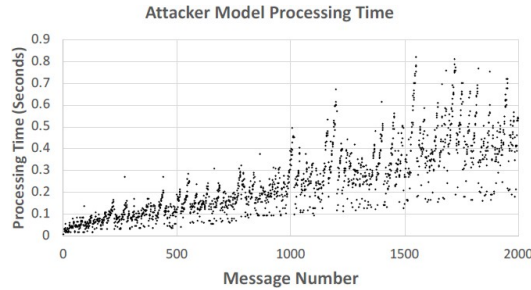


Figure 25. ToM CM Model Processing Time per Message (in seconds) for only the First 2,000 Messages.

6.3.2 Action-Space Distribution.

We measured the distribution of ToM CM’s action spaces, along with anticipated duration of conversations when we place ToM CM model against a simulated defender (i.e., the same as the stress-test condition above). The attacker ToM model and defender baseline model played against each other for 10,000 runs under various starting scenarios. That is, the attacker model is fed as input an initial attacker move and defender response, and then the attacker model starts the interaction, followed by the defender move, and so on. The interaction ends when the attacker model decides to disengage and end the conversation (i.e., 19 additional initial instances are added to memory to represent the possible conditions, given the attacker’s state and defender action, in which the attacker might disengage from an interaction).

Table 3, presented at the end of the current section, shows the mean number of interactions for each starting scenario, as well as the initial, final, and changes in the attacker’s trust and engagement levels. The cognitive model predicts that attackers will disengage after about 4-6 messages depending on the initial situation. Figure 26 shows the mean number of messages per starting scenario from Table 3. Some interactions are predicted to be very short, while some are predicted to be greater than 50 exchanges.

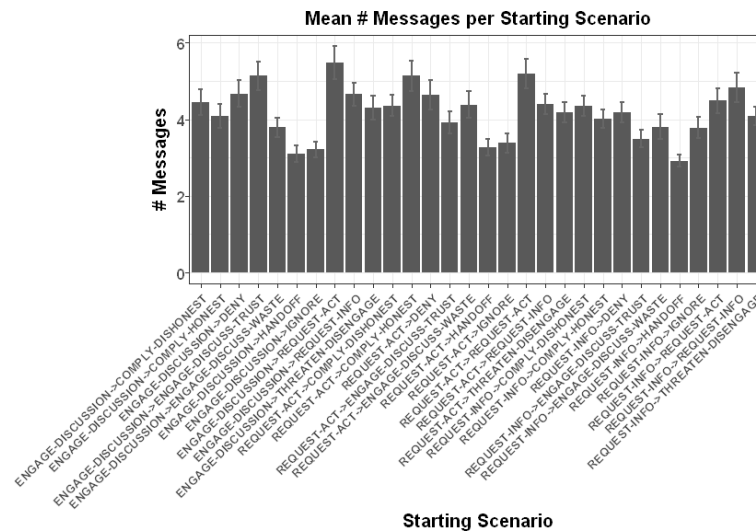


Figure 26. Mean Number of Messages per Starting Scenario from Model Predictions of Human Behavior.

The attacker's initial trust and engagement values vary depending on the initial situation, but it is typical for the trust to increase over the course of the interaction while the engagement level decreases. Figure 27 shows an example of the trust and engagement levels across message numbers for the scenario that includes an initial attacker action of redirect-request-act and defender response of challenge-deny.

Table 4 and Table 5, presented at the end of the current section, show the number of attacker responses given a defender action and number of defender responses given an attacker action, respectively. Table 4 indicates that the attacker responds mostly with the action of request-info or request-action, while it will eventually decide to disengage on all interactions. In response to a defender request-info or request-action, the model gives a very distributed set of responses, with more comply-honest than comply-dishonest, and the fewest proportion of threats to disengage. Also, the attacker model does not respond with comply-honest, comply-dishonest, or deny-request to any defender actions except for request-info or request-action. The attacker tends to disengage most often given a defender ignore action, but typically only when the ignore message comes later in the sequence, while early in the sequence an ignore action elicits many other types of responses from the attacker.

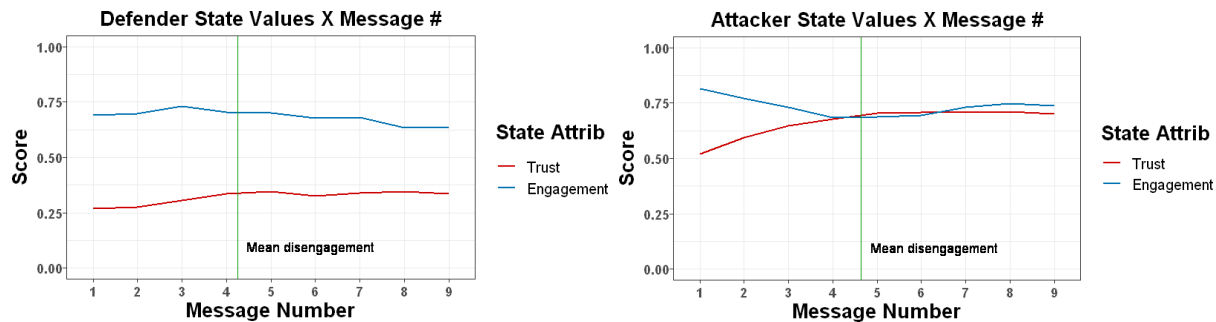


Figure 27. Mean State Values of the Attacker (left) and Defender (right) across Messages in the Example Scenario that Starts with an Attacker Action of Redirect-Request-Act and Defender Response of Challenge-Deny.

The green line indicates the mean disengagement.

Table 5 indicates that the defender model will mostly request-info or request-action to any attacker action. However, engaging in discussion to build trust is another popular defender action. Currently the defender model favors ignoring more than anticipated. However, these results shed light on how we might adjust the attacker and defender models, particularly the utility function, state change function, and initialization of declarative memory.

6.3.3 Prediction Accuracy.

The predictive accuracy of the ToM CM was examined against the groundtruth data from human attackers that interacted with human end-users and defenders during the Winter 2020 evaluation. Our cognitive model takes as input the first attacker action and the defender response and predicts the subsequent attacker action. The model ingests the messages in order presented in the dataset and learns over time by storing decisions as instances in memory. The model was run 1,000 times to generate stable predictions of model accuracy.

Because there were only 28 threads and most were only of three turns, the model did not have much data to learn from, and the threads were also from different defenders with different strategies. Figure 28 summarizes the predictive accuracy of the three versions of the cognitive

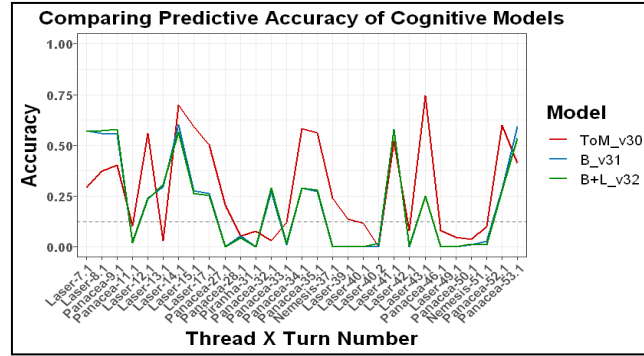


Figure 28. Predictive Accuracy of CMs across 28 Threads, after Collapsing across “Request” Actions.

model: ToM, Baseline, and Baseline+Learning. The y-axis shows the accuracy and the x-axis shows the thread and turn number (turn being the attacker action being predicted). The dashed line shows the probability of chance, or 1/8. Overall, the ToM model was the most accurate, 28.7%, performing above chance, while the Baseline and B+L models were 20.8% and 20.6%, respectively.

6.3.4 Discussion.

The ToM CM is a robust model of an attacker that can be leveraged to evaluate and predict outcomes for a diverse range of scenarios. Introspection of the CM provides underlying statistics regarding decision probabilities, expected utilities, and state information, in addition to action predictions. These predictions inform the Game Tree design and, therefore, the more accurate the model the better the strategy recommendation can be. Future work will attempt to better understand the differences between model and human behavior and use these results to improve the CM predictions.

Table 3. Mean Number of Messages and Mean Changes in Attacker State Attributes for each Possible Starting Scenario.

Starting Scenario	# Sims	Mean # Messages	Min # Messages	Max # Messages	Attacker Initial Trust	Attacker Initial Engage	Attacker Final Trust	Attacker Final Engage	Trust Δ	Engage Δ
ENGAGE-DISCUSSION->COMPLY-DISHONEST	311	3.45	0	48	0.82	0.83	0.84	0.67	0.03	-0.16
ENGAGE-DISCUSSION->COMPLY-HONEST	329	3.09	0	52	0.82	0.82	0.85	0.69	0.03	-0.13
ENGAGE-DISCUSSION->DENY	353	3.68	0	56	0.83	0.82	0.85	0.69	0.02	-0.13
ENGAGE-DISCUSSION->ENGAGE-DISC-TRUST	316	4.14	0	61	0.60	0.89	0.70	0.64	0.11	-0.25
ENGAGE-DISCUSSION->ENGAGE-DISC-WASTE	341	2.79	0	31	0.16	0.49	0.49	0.56	0.34	0.08
ENGAGE-DISCUSSION->HANDOFF	355	2.10	0	37	0.09	0.44	0.32	0.44	0.24	0.00
ENGAGE-DISCUSSION->IGNORE	336	2.21	0	36	0.16	0.48	0.37	0.47	0.22	-0.01
ENGAGE-DISCUSSION->REQUEST-ACT	299	4.48	0	72	0.57	0.90	0.74	0.65	0.18	-0.25
ENGAGE-DISCUSSION->REQUEST-INFO	327	3.66	0	33	0.57	0.88	0.73	0.68	0.16	-0.20
ENGAGE-DISCUSSION->THREATEN-DISENGAGE	309	3.31	0	45	0.16	0.90	0.45	0.68	0.29	-0.22
REQUEST-ACT->COMPLY-DISHONEST	361	3.36	0	43	0.23	0.82	0.63	0.66	0.39	-0.16
REQUEST-ACT->COMPLY-HONEST	341	4.13	0	72	0.74	0.90	0.82	0.72	0.08	-0.18
REQUEST-ACT->DENY	332	3.64	0	68	0.52	0.81	0.68	0.67	0.16	-0.15
REQUEST-ACT->ENGAGE-DISC-TRUST	346	2.92	0	42	0.74	0.45	0.78	0.53	0.03	0.09
REQUEST-ACT->ENGAGE-DISC-WASTE	334	3.38	0	50	0.22	0.46	0.53	0.57	0.31	0.11
REQUEST-ACT->HANDOFF	338	2.28	0	29	0.23	0.48	0.39	0.46	0.16	-0.02
REQUEST-ACT->IGNORE	321	2.39	0	34	0.53	0.46	0.69	0.46	0.16	0.00
REQUEST-ACT->REQUEST-ACT	296	4.19	0	40	0.52	0.89	0.72	0.69	0.20	-0.20
REQUEST-ACT->REQUEST-INFO	340	3.40	0	43	0.52	0.87	0.70	0.67	0.17	-0.21
REQUEST-ACT->THREATEN-DISENGAGE	350	3.19	0	46	0.49	0.89	0.65	0.64	0.16	-0.26
REQUEST-INFO->COMPLY-DISHONEST	340	3.35	0	33	0.25	0.81	0.55	0.68	0.31	-0.13
REQUEST-INFO->COMPLY-HONEST	355	3.02	0	31	0.76	0.91	0.81	0.70	0.05	-0.21
REQUEST-INFO->DENY	326	3.18	0	32	0.52	0.82	0.68	0.66	0.16	-0.16
REQUEST-INFO->ENGAGE-DISC-TRUST	346	2.49	0	30	0.71	0.43	0.76	0.56	0.06	0.12
REQUEST-INFO->ENGAGE-DISC-WASTE	318	2.81	0	44	0.25	0.49	0.50	0.54	0.25	0.04
REQUEST-INFO->HANDOFF	342	1.92	0	19	0.25	0.47	0.35	0.44	0.10	-0.03
REQUEST-INFO->IGNORE	342	2.78	0	38	0.52	0.48	0.64	0.53	0.13	0.05
REQUEST-INFO->REQUEST-ACT	323	3.49	0	45	0.46	0.89	0.67	0.66	0.21	-0.23
REQUEST-INFO->REQUEST-INFO	348	3.84	0	83	0.49	0.90	0.67	0.73	0.18	-0.17
REQUEST-INFO->THREATEN-DISENGAGE	325	3.08	0	22	0.50	0.91	0.62	0.67	0.12	-0.24

Table 4. Frequency of Attacker Responses to Defender Actions.

		Defender Action										Total
		REQUEST -INFO	REQUEST -ACTION	COMPLY- HONEST	COMPLY- DISHONES T	DENY- REQUE ST	THREATE N- DISENGA GE	ENGAGE- DISCUSS- TRUST	ENGAGE- DISCUSS- WASTE	REDIRECT - HANDOFF	IGNORE	
Attacker Response	REQUEST-INFO	1497	1328	263	268	251	241	1173	279	264	1325	6889
	REQUEST-ACTION	1257	1266	250	233	241	283	1087	265	274	1183	6339
	COMPLY-HONEST	1172	1076	0	0	0	0	0	0	0	0	2248
	COMPLY-DISHONEST	844	778	0	0	0	0	0	0	0	0	1622
	DENY-REQUEST	823	768	0	0	0	0	0	0	0	0	1591
	THREATEN-DISENGAGE	516	532	70	62	61	40	393	50	67	495	2286
	ENGAGE-DISCUSSION	1065	1012	204	213	197	229	886	214	264	1006	5290
	REDIRECT-HANDOFF	785	750	71	73	78	96	651	75	81	654	3314
	IGNORE	997	857	78	78	80	116	712	69	101	0	3088
	DISENGAGE	986	982	89	87	107	105	1273	117	128	1773	5647
Total		9942	9349	1025	1014	1015	1110	6175	1069	1179	6436	

Table 5. Frequency of Defender Responses to Attacker Actions.

		Defender Response										Total
		REQUEST -INFO	REQUEST -ACTION	COMPLY- HONEST	COMPLY- DISHONES T	DENY- REQUE ST	THREATE N- DISENGA GE	ENGAGE- DISCUSS- TRUST	ENGAGE- DISCUSS- WASTE	REDIRECT - HANDOFF	IGNORE	
Attacker Action	REQUEST-INFO	2114	2060	355	341	328	339	1658	342	368	879	8784
	REQUEST-ACTION	1940	1813	341	362	334	364	1525	351	355	855	8240
	COMPLY-HONEST	1040	971	0	0	0	0	227	1	0	9	2248
	COMPLY-DISHONEST	470	464	0	0	0	13	85	6	18	566	1622
	DENY-REQUEST	512	434	0	0	0	14	75	0	8	548	1591
	THREATEN-DISENGAGE	692	685	0	0	0	4	403	6	5	491	2286
	ENGAGE-DISCUSSION	1439	1338	329	311	353	311	1152	345	355	1209	7142
	REDIRECT-HANDOFF	723	717	0	0	0	43	543	7	41	1240	3314
	IGNORE	1012	867	0	0	0	23	507	11	29	639	3088
	Total	9942	9349	1025	1014	1015	1111	6175	1069	1179	6436	

7.0 OPTIMAL STRATEGY PRESCRIPTION USING BEHAVIORAL GAME THEORY

One of the core challenges in the ASED environment is operating in a state of uncertainty both with respect to the attacker's aims and abilities. To reduce strategic uncertainty and exploitative actions for extracting information and effort from the attacker, we applied the class of *extensive form games with incomplete information*. This class of games includes two key elements of the ASED scenario: sequential interactions with the attacker and uncertainty about the current state of the game as it pertains to the attacker's intentions. Whereas this class of games has traditionally been studied in the context of a perfectly rational attacker, however, our needs required that we extend the framework to allow for more realistic and varied behavioral characteristics from the attacker. This behavioral challenge is compounded by the fact that an inherent uncertainty exists to determine exactly which behavioral type applies to a given attacker. To overcome these challenges, we integrated a cognitive model of the attacker into a fictitious play learning algorithm, and used subsequent interactions with the attacker to revise the behavioral model and update our strategy as needed, ultimately increasing attacker's likelihood of prolonging the ongoing conversation and disclosing identifying information.

In Section 7.1, we describe definitions of the strategic game between an attacker and a defender, the game tree formation, and utility computation. In Section 7.2, we provide some evaluation results.

7.1. Strategic Response Prescription

Representing the “game” our system is playing with the attacker presents another key challenge for two reasons: in an explicit representation of the extensive form game, decision trees grow exponentially with the duration (number of interactions) of the game; more importantly, it may not be entirely clear to learn exactly *which game* the attacker has initiated. To overcome these challenges, we used the attacker's CM as a compact representation of the extensive form game tree. CM provides the likelihood attacker's current game as the interaction with the attacker unfolds. With the reduction in strategic uncertainty provided by CM, CHESS's GT module aims at prescribing adaptive and increasingly effective actions against the attacker as the interaction plays out.

7.1.1 Action Space Definition.

After several iterations, we identified the action spaces for the attacker and defender to encapsulate the exchange between the two parties. The actions are specified so that, roughly, each message that one party sends to the other constitutes a discrete action. As introduced in Figure 3, our action spaces are capable of capturing categorical actions at the top level and tactical actions at a lower level in a hierarchical manner as follows:

Attacker's actions in a hierarchy:

1. **Compliance** with a request for information or action from the Defender
 - a. Perform dishonestly (provide fake information)
 - b. Perform honestly (provide real information)
2. **Challenge** to the Defender's previous action
 - a. Explicitly deny request
 - b. Threaten to disengage

3. **Engage** in discussion
4. **Request** for the Defender to furnish information or to perform some activity
 - a. Request information
 - b. Request performance of activity
 - c. Handoff to third-party
5. **Ignore**
6. **Unknown**

Defender's actions in a hierarchy:

1. **Compliance** with a request from the attacker
 - a. Perform dishonestly (provide fake information)
 - b. Perform honestly (provide real information)
2. **Challenge** to the attacker's previous action
 - a. Explicitly deny request
 - b. Threaten to disengage
3. **Engage** with the attacker
 - a. Engage in to waste time
 - b. Engage to build trust
4. **Redirection** of the current discourse/exchange with the attacker, which may involve making a request or forwarding the attacker into a new exchange with a "new" defender.
 - a. Request information
 - b. Request performance of activity
 - c. Handoff to third-party
5. **Ignore**

Such a hierarchical representation of action spaces enable controlling the granularity of tactical actions for each category, which can be beneficial for building a game tree as we explain next.

7.1.2 Game Tree Generation.

Using the action spaces as defined above, CHESS builds a game tree that is rooted at the last observed move by the attacker and projects the next few alternating defender and attacker actions. The depth of the game tree is determined either by computational constraints (fixed depth) or the "natural" end of the game as indicated by the attacker disengaging and thereby ceasing subsequent interactions with the defender.

Figure 29 is an illustration of a game tree for depth 2:

1. Assume that the current-input action-space from an attacker is request to perform an action or RA, which is denoted in a black node at the root of the tree (i.e., tree depth = 0).
2. CHESS's GT module queries CM with a potential strategy that a defender can take (e.g., comply honestly [CH]). This step corresponds to tree depth = 1.
3. After running the simulation, CM provides the probabilities of the attacker's

possible reaction spaces for the queried defender's strategy (e.g., CH). CM also provides projected attacker's trust and engagement states. This step corresponds to tree depth = 2.

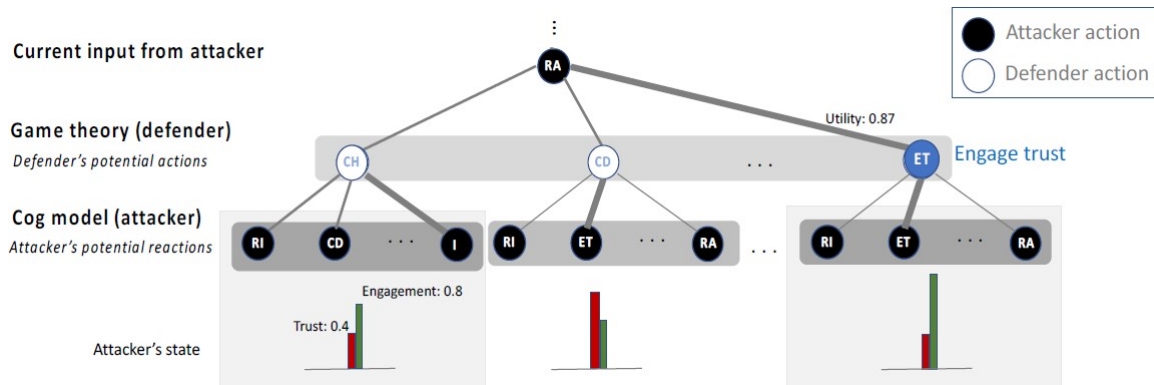


Figure 29. Example Game Tree Generation.

Given a current input action space from an attacker (black node on top), CHESS's GT module builds a game tree for defender's possible responding action spaces (white nodes) by simulating attacker's potential reactions and cognitive states using the CM).

4. GT repeats steps 2-3 for other possible strategies for defenders (e.g., comply dishonestly or CD, engage to build trust or ET, etc.) at tree depth = 1 & 2.
5. GT selects the strategy that results in the highest utility in the root's immediate subtree based on the outputs from CM.

If the depth of the game tree is more than 2, GT repeats step 2 on those possible reaction spaces from step 3 to further explore the subtrees below depth 2.

7.1.3 Utility Computation.

As shown in Figure 30, the root of the subtree that generates the highest expected utility is defender's action that is ultimately selected. To compute the utility value for each subtree, leaves of the subtree are evaluated with respect to the reported state of the cognitive model's *trust* and *engagement* values as they can be a good indicator of the attacker's likelihood to comply honestly and either provide requested information or perform requested action. In other words, higher trust and engagement values generate greater utility than lower values. In addition, the length of the path from the root to the leaf node is added to the utility value for the leaf, which is meant to favor "long" interactions with the attacker over "shorter" ones (in terms of the number of exchanges). The values of the leaves are propagated up the subtree; as the value propagates it alternates between defender nodes and attacker nodes (corresponding to defender and attacker actions, respectively); at an attacker node, the value is discounted by the probability that the attacker will take the corresponding action assuming that future play reaches that point (as estimated by the cognitive model).

At a defender's decision point in a subtree, only the defender action with the highest expected future utility (as determined by the subtree below the point) is propagated up. This algorithm terminates when the decision point is at the root of the original game tree at which point the defender action with the highest expected utility is selected and returned by the algorithm.

7.1.4 Game Tree Pruning.

One practical concern with the game-tree development is the computational overhead, in particular to generate a game tree with a large number of depths. Hierarchical representation of the action spaces can ease the computational overhead, for example using the top-level categorical actions for building trees of a large number of depths. However, this approach does not capture tactical distinctions between action spaces.

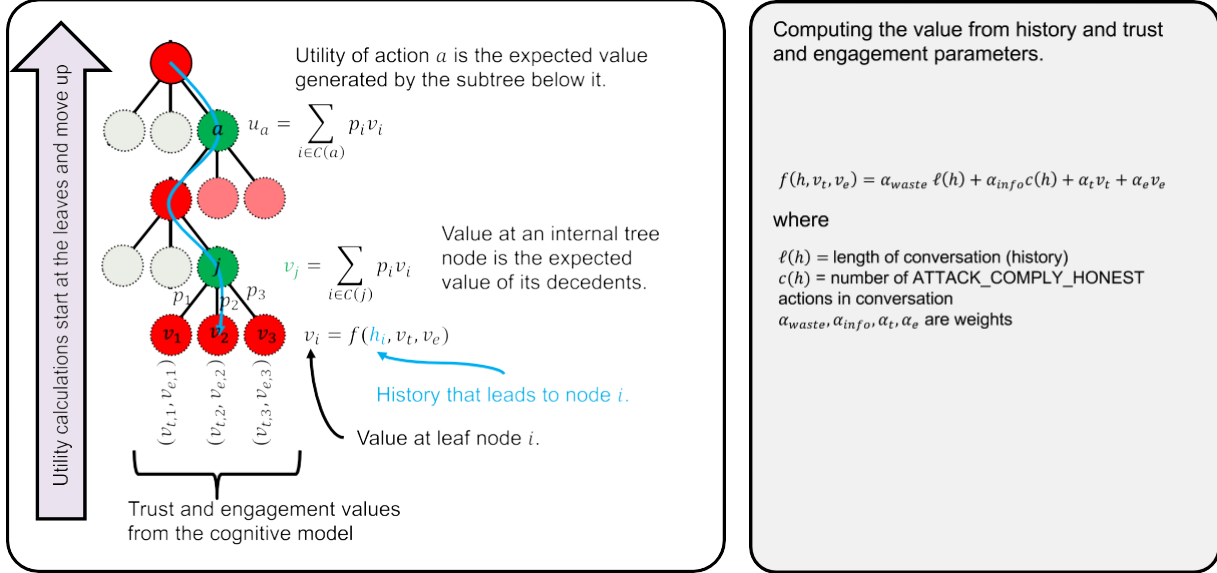


Figure 30. Game Tree Generation and Utility Computation

Red nodes represent the projected trust and engagement values of the attacker from (CM, and the game tree selects the action that results in the highest utility based on the values generated by the subtree below it.

To address this concern, we developed a pruning capability that allows the game-tree algorithm to ignore entire branches of the search tree that it deems improbable to reach over the course of a game. More specifically, given a pruning threshold, GT stops exploring subtrees for the node whose utility is less than the pruning threshold. Pruning the tree significantly reduces the number of queries to CM, speeding up the computation of defender actions.

7.2. Evaluation

We used TA3 Campaign datasets (May - July, 2019) to evaluate GT's output recommendations. Figure 31 shows the recommended defender action distribution against the input attacker action using 57 email exchanges from the TA3 Campaign datasets. Varying pruning thresholds resulted in different distribution of recommended action spaces. For example, increasing the pruning threshold to 0.1 resulted in varying distributions in the types of recommended action spaces. On the other hand, increasing the search depth of the game tree did not seem to provide additional benefits. However, analysis of the Winter 2020 evaluation data did show improvements when search depth was increased (see section 8.2). For all of the Phase 1 evaluations, we used the search tree depth of 2 and the pruning threshold of 0.

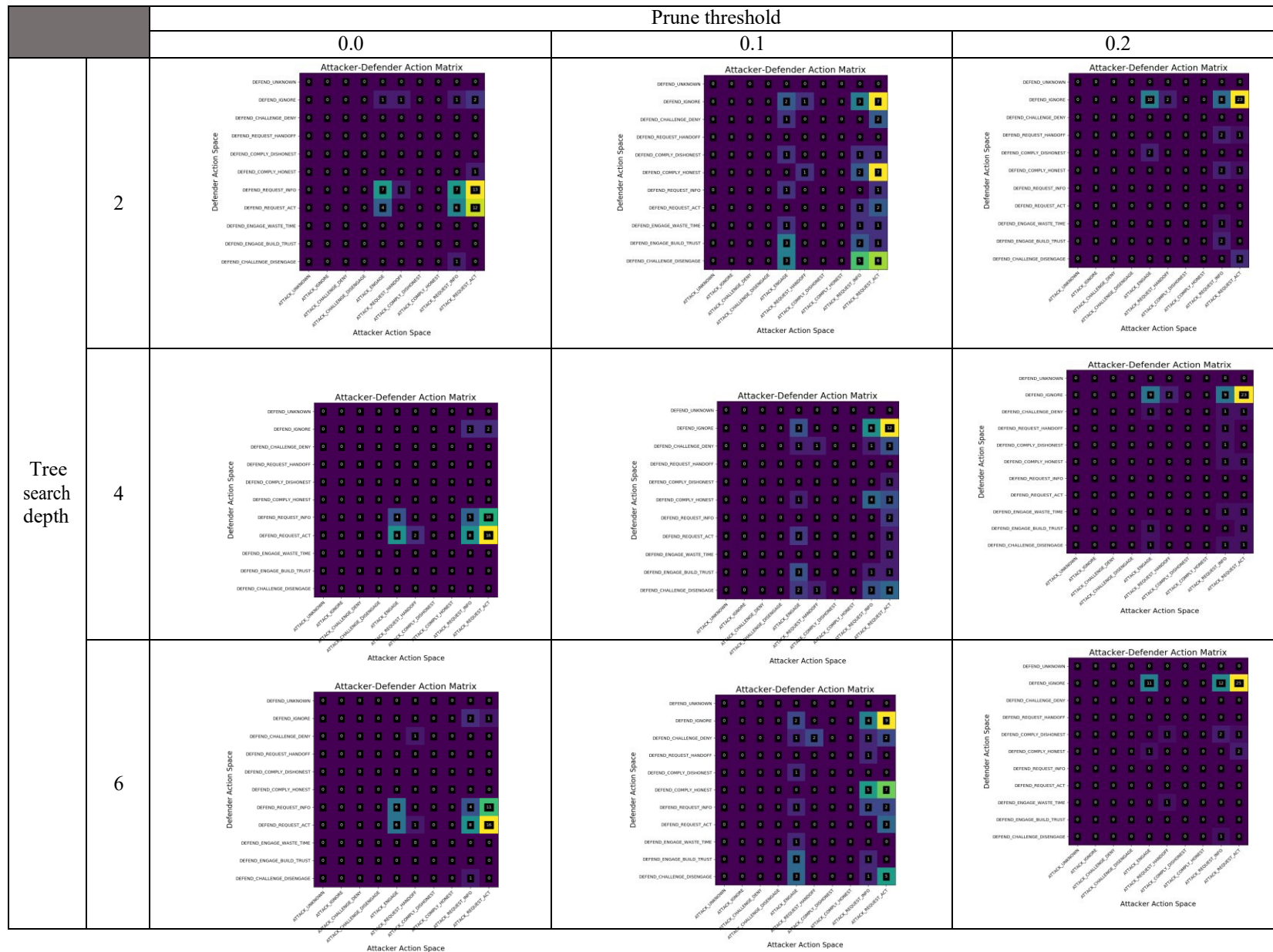


Figure 31. Attacker-Defender Action Space Prescription Matrices for Different Tree Search Depth (i.e., number of next action spaces to explore) and Pruning Threshold (i.e., pruning actions with probabilities less than the indicated threshold).

8.0 PERFORMANCE EVALUATION FOR PHASE 1 WORKSHOPS

We successfully integrated with our collaborators' systems during all of the Phase 1 evaluations. For the August 2019 evaluation, we interacted with TA3 attackers within PANACEA, and RECURSE systems with bots and human operators who translated CHESSE's suggestions into messages, and for the Winter 2020 evaluation, we were integrated within the PANACEA's system but CHESSE's capabilities were turned off during the actual evaluation.

In this section, we focus on highlighting (1) how CHESSE's recommended actions helped human operators in strategizing their responses during the August 2019 evaluation, and (2) what the projected outcome would have been using CHESSE's recommendations during the Winter 2020 evaluation.

8.1. Helping Human Operators with Strategies

To evaluate the capability of prolonging conversations with attackers, CHESSE was integrated with our collaborators' systems: PANACEA and RECURSE. Upon receiving a message from an attacker, our collaborator's system would run CHESSE and obtain the recommended strategic action that would guide their human operators in generating response messages.

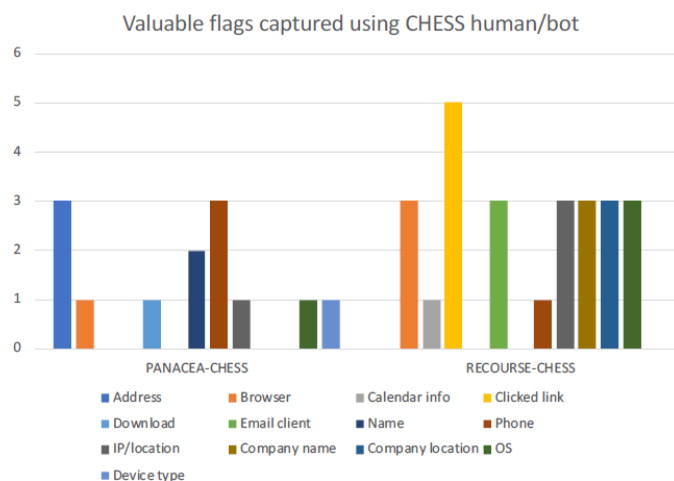


Figure 32. Information Captured using CHESSE's Recommendations during August 2019 Evaluation.

Overall, we captured 14 flags from the PANACEA-CHESSE collaboration system, and 25 flags from the RECURSE-CHESSE collaboration system.

Figure 32 and Figure 33 highlight how CHESSE can help human operators generate strategy-guided and context-specific dialog to elicit information and prolong conversations, respectively.

More specifically, Figure 32 summarizes a variety of information that were captured using CHESSE's recommendations during the August 2019 evaluation. This graph includes additional flags that were not reflected in the JPL's official report, such as the attacker's network, browsing, and device information (e.g., IP address, browser version, device type, email client, operating system (OS) version, etc.). Such information can be useful in correlating attackers and inferring attacker organizations.

Team	Bot Response	Avg Turns	Avg Rating	Human/Bot Response	Avg Turns	Avg Rating	Human Response	Avg Turns	Avg Rating
NEMESIS	20/27	1.2	1.0	--	--	--	3/3	4.0	3.0
PIRANHA	15/15	6.5	1.6	6/6	4.8	2.6	9/9	8.1	2.9
PANACEA	15/15	4.8	1.9	6/6	8.8	2.7	9/9	8.2	2.9
RECOURSE	24/24	3.0	1.8	3/3	3.7	2.7	3/3	6.3	2.4
LASER	18/21	8.6	1.1	--	--	--	8/9	2.6	1.9
Average		4.5	1.4		6.2	2.6		6.1	2.6

1. Response columns show [number of responses] / [number of attack messages]
2. Avg Turns = average number of turns of dialogue per conversation
3. Avg Rating = average Turing Test ratings for all responses where [3 = human | 2 = uncertain | 1 = bot]

Figure 33. Average Dialog Turns of CHESS's Human/Bot Responses inside PANACEA and RECOURSE Systems.

Given that the dialog turns were as good as Human responses, - only solutions, CHESS has a potential to aid Bot-only solutions to perform like human operators.

Figure 33 highlights that the dialog turns in the CHESS-PANACEA's Human/Bot responses were better than any of the Human-only or Bot-only responses. Such results indicate the strength of CHESS's approach that can aid bots to behave like human operators in future performances.

We also received promising feedback from human operators who incorporated our recommendations during the evaluation:

- "Because it kept prompting me to request a real-world action like send a link/file, we ended up getting more clicks on our fingerprinting site with the CHESS-protected account than with any other. The chart in the evaluation report does not include flags captured via things like our honeypot – currently they only show things TA3 explicitly sent in messages."
- "CHESS-protected account was the most successful at capturing flags, more so than the plain human and *definitely* more so than our bots."
- "Having strategic suggestions are valuable to both a human-in-the-loop and bots that can take suggestions as to actions to take."
- "As an operator trying to manage even a handful of threads, having actionable suggestions is really important."
- "Guiding aspect of the strategy was helpful. The human only did better possibly because they could be more creative, but a template-based response system (bots) would benefit from the strategy guides."

Considering an operational environment where human operators are involved, we believe that CHESS's strategy recommendations can become particularly beneficial when human operators need to handle multiple attackers, which can overload human operators and possibly lose track of the game states. Along with the action-space recommendations, CHESS has a capability to provide a list of available mediums to exploit the attackers and which goals to seek out in a given turn of an exchange (e.g., ask the attacker to click a link using SMS).

8.2. Helping Chatbots with Strategies

Unlike the August 2019 evaluation, the Winter 2020 evaluation's core focus for TA2 was evaluating the chatbot's NLG capabilities. Although we were integrated and deployed into PANACEA's main system for the evaluation, CHESS was never turned on during the actual evaluation due to technical problems on the PANACEA side. Hence, we conducted post-evaluation analysis using the released data to assess the impact of CHESS's recommendations.

8.2.1 Higher Compliance Rate with CHESS Recommendations.

We compiled all the message data provided by JPL and Thomson Reuters Special Services (TRSS) and organized them according to thread, team, and attack or defense turn. We then manually labeled each message with our action-space classification as shown in Figure 3. The distributions of action spaces are shown in Figure 35.

Once the data from the evaluation had been compiled, we fed them through the CHESS system to compare the actual defender actions taken to what defender actions our CHESS system would recommend for each thread.

Figure 34 shows the recommended defense-action distribution provided by the CHESS system for all Winter 2020 evaluation threads. We first wanted to see how much the defender actions taken by the overall participant systems aligned with the CHESS recommendations, and 7% of the overall actions taken during the evaluation by defender systems were the same.

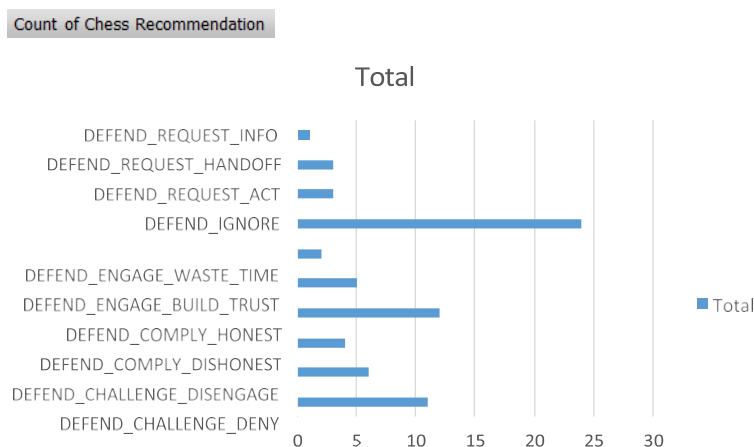


Figure 34. The Distribution of Defender Actions Recommended by the CHESS System for all Published Threads across All Teams from the Winter 2020 Evaluation.

We further analyzed the specific threads that lead the attacker to a compliance action (i.e., giving requested information to the defender, fulfilling a request for action by the defender, etc.). We then checked if our strategy recommendations matched those undertaken by the defender system to receive a compliance response. We divided the total number of times our recommendation was in alignment with the compliance-eliciting defender system by the total number of compliances in the dataset. This metric gave us an idea of how successful our strategy recommendations would have been in known conversations with good outcomes across all defender systems in the evaluation. We found that the CHESS recommendation aligned with the actions taken by the defender 33% of the time. The improved alignment on successful interactions suggests that if defender systems follow our high-level action recommendations, there is a possibility to improve their success rate.

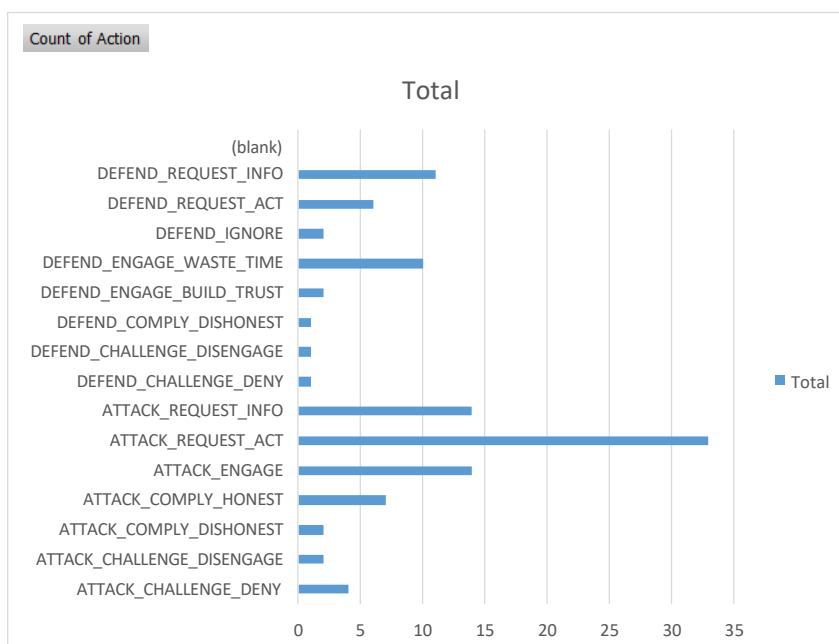


Figure 35. The Distribution of Defender and Attacker Actions from all Published Threads across all Teams from the Winter 2020 Evaluation.

8.2.1.1 Game-Tree Search Depth Analysis.

We also tried different game-tree search depths to analyze the attacker’s compliance rate. We again looked at defender conversation turns where the next attacker response was a compliance action.

With search depth 2 (i.e., CHESS predicts the next attacker move for each possible defender action), our recommendations aligned with 36% of the strategies that lead to a compliance action by the attacker. At search depth 4 (i.e., CHESS predicts two rounds of attacker-defender interactions in advance), the CHESS system consolidated more of its recommendations in alignment with successful strategies undertaken by the systems during the evaluation by 31%, resulting in 67% of all actions undertaken by performer systems that would lead to an attacker compliance. The compliance-alignment enhancement results are summarized in Figure 36.

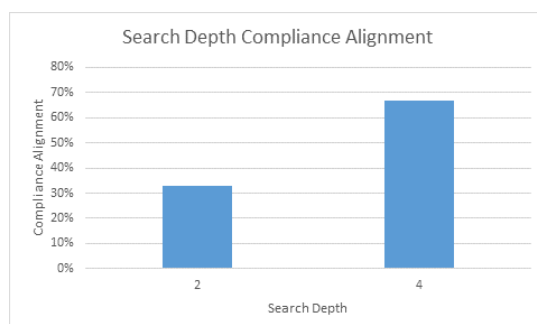


Figure 36. Compliance Alignment Rates given Search Depth of 2 vs. 4 on the CHESS System.

9.0 CONCLUSIONS

Predicting an attacker's mind and next game moves is a non-trivial task, especially in an online environment. Yet it is crucial to create effective strategies to cope with the attacker and influence him/her to achieve our goal. During the Phase 1 of the ASED program, we developed a novel system called CHESS that has the following capabilities:

- CHESS supports the forward-simulation of an attacker's mental and cognitive processes for each of the possible defender strategies using Theory of Mind (i.e., what are the likely responses from the attacker given a particular action from the defender?).
- CHESS is capable of building a game tree, where each branch represents the sequence of attacker-defender moves, and computing rewards for each branch.
- Based on the game tree, CHESS prescribes the optimal strategy with the highest rewards.
- CHESS is capable of simulating multiple interactions at once to predict the *sequence of interactions* between an attacker and a defender.
- CHESS is capable of identifying common threat-actor organizations behind seemingly distinct attackers or an attacker that uses multiple sender personas.

According to the evaluation results and post-evaluation analysis, CHESS has promising directions in recommending strategies to help a defender system involving human operators. In particular, CHESS can ease the cognitive overload of human operators who must interact with multiple attackers. CHESS has automated capabilities for threat actor identification, using technologies to extract signatures from messages and combine them with diverse information sources to identify individual actors and groups working together. As CHESS is a learning system, human operators can provide positive or negative feedback with respect to the recommendations to fine-tune parameters for both cognitive model and game theory modules to update the current game state and update strategies accordingly. Even for the defender systems without humans in the loop, CHESS recommendations could have improved the compliance rate from attackers, acquiring additional information from them or having them perform certain actions as we intended. Specifically, the CHESS system consolidated more of its recommendations in alignment with successful strategies undertaken by the systems during the evaluation by 31%, resulting in 67% of all actions undertaken by performer systems that would lead to an attacker compliance.

While we exercised CHESS to defend against victims in the ASED environment, we envision how CHESS can be applied to the attacker's side, as well as to other domains. More specifically, one major challenge was understanding attacker's strategies, which could have been acquired from meaningful and a rich set of multi-turn attacker-victim interactions. We attempted to address this challenge with our human subjects study. On the other hand, other prior work looked at the mental states of the *victims* that attackers exploit in the ASED environment [23] [24]. Plus, anyone can be exposed to social engineering attacks and it is more trivial to interview them, rather than interviewing social-engineering attackers.

Hence, the current CHESS system can be tweaked into an adversarial system on behalf of TA3 and help the red team generate strategies and assess TA1/TA2 performers.

Furthermore, the current CHESS system is reactive; we need to wait until an attacker approaches us. Instead, we can turn such a reactive CHESS system into a proactive solution such that we can strategically exploit attackers' associates first and deter them from attacking others. As for other

domains, many terrorist organizations attempt to recruit agents using online forums. As a mitigation solution to prevent the terrorist/attacker organizations from influencing innocent members of the forum, multiple CHESS-supported bots can be deployed to actively interact with the hosting attackers/terrorists with well-developed, possibly coordinated strategies and disrupt their operations.

10.0 REFERENCES

- [1] Verizon, "2019 Data Breach Investigation Report," <https://enterprise.verizon.com/resources/reports/2019-data-breach-investigations-report.pdf>, 2019.
- [2] J. Hong, "The State of Phishing Attacks," *Communications of the ACM*, vol. 55, no. 1, pp. 74-81, 2012.
- [3] S. Sheng, B. Magien, P. Kumaraguru, A. Acquisti, L. F. Cranor, J. Hong and E. Nunge, "Anti- Phishing Phil: The Design and Evaluation of a Game That Teaches People Not to Fall For Phish," in *Proceedings of the 3rd Symposium on Usable Privacy and Security (SOUPS)*, Pittsburgh, PA, 2007.
- [4] Federal Trade Commision, "How to Recognize and Avoid Phishing Scams," May 2019. [Online]. Available: <https://www.consumer.ftc.gov/articles/how-recognize-and-avoid-phishing-scams>. [Accessed 18 May 2020].
- [5] Z. Kleinman, "Scam baiter: Why I risk death treats to expose online cons," BBC News, 7 October 2017. [Online]. Available: <https://www.bbc.com/news/technology-41518352>. [Accessed 18 May 2020].
- [6] 419 Eater, "419 Eater," [Online]. Available: <https://www.419eater.com/>. [Accessed 18 May 2020].
- [7] T. Hofmann, "Unsupervised Learning by Probabilistic Latent Semantic Analysis," *Machine Learning*, vol. 42, no. 1-2, pp. 177-196, 2001.
- [8] T. J. Hazen, "Spoken Language Understanding: Systems for Extracting Semantic Information from Speech," *Topic Identification*, pp. 319-356, 2011.
- [9] T. J. Hazen, "Latent Topic Modeling for Audio Corpus Summarization," in *In Proceedings of the 12th Annual Conference of the International Speech Communication Association*, 2011.
- [10] J. Isacenkova, O. Thonnard, A. Costin, D. Balzarotti and A. Francillon, "Inside the SCAM Jungle: A Closer Look at 419 Scam Email Operations," in *IEEE Security and Privacy Workshops*, San Francisco, CA, 2013.
- [11] O. Thonnard, P.-A. Vervier and M. Dacier, "Spammers Operations: A Multifaceted Strategic Analysis," *Security and Communication Networks*, pp. 336-356, 2012.
- [12] Y. Shen and O. Thonnard, "MR-TRIAGE: Scalable Multi-Criteria Clustering for Big Data Security Intelligence Applications," in *IEEE International Conference on Big Data*, Washington DC, 2014.
- [13] D. Premack and G. Woodruff, "Does the Chimpanzee Have a Theory of Mind?," *Behavioral and*

Brain Science, vol. 1, no. 4, pp. 515-526, 1978.

- [14] C. P. Ferguson and T. S. Ferguson, "Models for the Game of Liar's Dice," in *Stochastic Games and Related Topics*, Springer, 1991, pp. 15-28.
- [15] J. R. Anderson, D. Bothell, M. D. Byrne, S. Doublas, C. Lebiere and Y. Qin, "An Integrated Theory of the Mind," *Psychological Review*, vol. 111, no. 4, pp. 1036-1060, 2004.
- [16] C. Lebiere, P. Pirolli, R. Thomson, J. Paik, M. Rutledge-Taylor, J. Staszewski and J. R. Anderson, "A Functional Model of Sensemaking in a Neurocognitive Architecture," *Computational Intelligence and Neuroscience*, 2013.
- [17] E. A. Cranford, C. Lebiere, C. Gonzales, S. Cooney, P. Vayanos and M. Tambe, "Learning about Cyber Deception through Simulations: Predictions of Human Decision Making with Deceptive Signals in Stackelberg Security Games," in *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, 2018.
- [18] E. A. Cranford, C. Gonzales, P. Aggarwal, S. Cooney, M. Tambe and C. Lebiere, "Towards Personalized Deceptive Signaling for Cyber Defense using Cognitive Models," in *17th Annual Meeting of the International Conference on Cognitive Modeling*, 2019.
- [19] E. A. Cranford, C. Gonzales, P. Aggarwal, M. Tambe and C. Lebiere, "Adaptive Cyber Deception: Cognitively-Informed Signaling for Cyber Defense," in *Hawaii International Conference on System Sciences*, 2020.
- [20] J. R. Anderson, K. Koedinger, R. Pelletier and A. T. Corbett, "Cognitive Tutors: Lessons Learned," *The Journal of Learning Sciences*, vol. 4, pp. 167-207, 1995.
- [21] J. R. Anderson, C. F. Boyle and G. Yost, "The Geometry Tutor," *Journal of Mathematical Behavior*, pp. 5-20, 1986.
- [22] C. Gonzales, J. F. Lerch and C. Lebiere, "Instance Based Learning in Dynamic Decision Making," *Cognitive Science*, vol. 27, no. 4, pp. 591-635, 2003.
- [23] F. Stajano and P. Wilson, "Understanding Scam Victims: Seven Principles for Systems Security," *Communications of the ACM*, vol. 54, no. 3, pp. 70-75, 2013.
- [24] L. Laribee, Development of Methodological Social Engineering Taxonomy Project, Naval Postgraduate School (Thesis), 2006.
- [25] J. R. Anderson and C. Lebiere, *The Atomic Components of Thought*, Mahwah, NJ: Lawrence Erlbaum Associates, 1998.

- [26] D. Reitter and C. Lebiere, "Accountable Modeling in ACT-UP, a Scalable, Rapid-Prototyping ACT-R Implementation," in *Proceedings of the 2010 International Conference on Cognitive Modeling*, Philadelphia, PA, 2010.
- [27] D. Wallach and C. Lebiere, "Conscious and Unconscious Knowledge: Mapping to the Symbolic and Subsymbolic Levels of a Hybrid Architecture," in *Attention and Implicit Learning*, Amsterdam, Netherlands, John Benjamins Publishing Company, 2003.
- [28] I. Erev, A. E. Roth, E. Haruvy, S. Herzog, R. Hau, R. Hertwig, T. Stewart, R. West and C. Lebiere, "A Choice Prediction Competition, for Choices from Experience and from Description," *Journal of Behavioral Decision Making*, vol. 23, no. 1, pp. 15-47, 2010.

11.0 LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

ACT-R	Adaptive Character of Thought - Rational
API	Application Programming Interface
ASED	Active Social Engineering Defense
CHESS	Continuously Habituating Elicitation Strategies for Social-engineering-attacks
CM	Cognitive Model
CRI	Choquet Fuzzy Integral
DARPA	Defense Advanced Research Project Agency
GT	Game Theory
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
HIT	Human Intelligence Task
IBL	Instance-Based Learning
ID	Identification
IP	Internet Protocol
JIRA	Java Mail, Rafael, Alanis
JSON	JavaScript Object Notation
LISP	List Processing
MLN	Multi-Layer Network
MTurk	Amazon Mechanical Turk
NLG	Natural Language Generation
NLP	Natural Language Processing
OS	Operating System
OWA	Ordered Weighted Average
PLSA	Probabilistic Latent Semantic Analysis
REST	Representational State Transfer
RT	Reaction Time
SMS	Short Message Services
STIX TM	Structured Threat Information Expression
TA	Technical Area
TF-IDF	Term Frequency-Inverse Document Frequency
ToM	Theory of Mind
TRSS	Thomson Reuters Special Servers
UMAP	Uniform Manifold Approximation and Projection
URL	Uniform Resource Locator

APPENDIX A – Opponent Strategy Specifications in Liar’s Dice

We describe the detailed opponent strategies that we implemented for the computer agents, and explain the reward structure modifiers for attacker and defender roles.

Aggressive/Risk-Averse Strategy: Roller. This is modeled after the Nash ideal form of play as follows. As a roller, if the number rolled is 6 or above, a truthful claim is put forward. On the other hand, if the roll is less than 6, higher truthful claims are preferred. We have arranged the aggressive strategy opponent to generally follow these principles but skewed to deceive more often (i.e., claim higher numbers). The opponent makes truthful claims ~70% of the time.

Although this approach may not be the best, it represents a challenge and distinctly contrasts the loss-averse strategy. Table A-1 is the summary of the probabilities that an aggressive roller would claim (column) given the rolled dice value (row).

Table A-1. Strategic Probabilities that an Aggressive Roller would Claim (column) given the Rolled Dice Value (row).

		Claim										
Roll		2	3	4	5	6	7	8	9	10	11	12
	2	70%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%
	3	35%	35%	4%	4%	4%	4%	4%	4%	4%	4%	4%
	4	23%	23%	23%	4%	4%	4%	4%	4%	4%	4%	4%
	5	18%	18%	18%	18%	5%	5%	5%	5%	5%	5%	5%
	6	0%	0%	0%	0%	70%	6%	6%	6%	6%	6%	6%
	7	0%	0%	0%	0%	6%	70%	6%	6%	6%	6%	6%
	8	0%	0%	0%	0%	6%	6%	70%	6%	6%	6%	6%
	9	0%	0%	0%	0%	6%	6%	6%	70%	6%	6%	6%
	10	0%	0%	0%	0%	6%	6%	6%	6%	70%	6%	6%
	11	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%
	12	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%

Aggressive/Risk-Averse Strategy: Caller. We have modeled the aggressive caller strategy after Nash ideal, in which lower claims are accepted and challenge probability increases along with the increased claims until 100% at the maximum roll. We ensure that the aggressive opponent will call more frequently for lower claims than the higher claims. This design may not be the ideal play, but it provides volunteers with distinctive gameplay behaviors when they face two opponent personas in tandem.

Table A-2. Strategic Probabilities that an Aggressive Caller would Claim the Roll.

		Roll										
Call %		2	3	4	5	6	7	8	9	10	11	12
		0%	1%	6%	10%	25%	33%	50%	64%	75%	100%	100%

Defensive/Loss-Averse Strategy: Roller. Overall, the loss-averse strategy ensures that the most claims are truthful to reduce risk, with only a 10% chance of issuing an untruthful claim, and all the truthful claims are considered to be equally valid. A truthful claim is always risk free, although being overly conservative can make the behavior predictable.

Table A-3. Strategic Probabilities that a Loss-Averse Roller would Claim (column) given the Rolled Dice Value (row).

		Claim										
Roll		2	3	4	5	6	7	8	9	10	11	12
	2	90%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%
	3	45%	45%	1%	1%	1%	1%	1%	1%	1%	1%	1%
	4	30%	30%	30%	1%	1%	1%	1%	1%	1%	1%	1%
	5	23%	23%	23%	23%	2%	2%	2%	2%	2%	2%	2%
	6	18%	18%	18%	18%	18%	2%	2%	2%	2%	2%	2%
	7	15%	15%	15%	15%	15%	15%	2%	2%	2%	2%	2%
	8	13%	13%	13%	13%	13%	13%	13%	3%	3%	3%	3%
	9	11%	11%	11%	11%	11%	11%	11%	11%	11%	5%	5%
	10	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
	11	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%	0%
	12	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%

Defensive/Loss-Averse Strategy: Caller. A loss-averse caller is skewed to only call higher claims than the opponent. Note that we implemented the blackjack-dealer-like cutoff at 11, which is modifiable as described below.

Table A-4. Strategic Probabilities that a Loss-Averse Caller would Claim the Roll.

		Roll										
Call %		2	3	4	5	6	7	8	9	10	11	12
		0%	0%	0%	0%	1%	10%	25%	33%	50%	100%	100%

Attacker Reward Structure Modifier. The modifier biases either Aggressive or Loss-Averse strategies to make larger claims than the original values according to the values shown in Table A-1-Table A-4, and increases the likelihood of challenges during the calls. The modifier can effectively shorten the match and provide countermeasures against human opponents that aim at maximizing their point totals by providing high claims early in the match. For Roller, if the roll is less than 6, the likelihood to make claims less than 6 is reduced by: $2.5\% * \text{number of rounds modifier}$ (ii. ee. , mmaamm : 4). The likelihood to make claims larger than 6 is increased by: $2.5\% * \text{number of rounds modifier}$ (ii. ee. , mmaamm : 4). A similar modifier is applied for the Caller role. For example, in round 2, claims < 6 have a $(-2.5\% * 2)$ chance while claims > 6 have a $(+2.5\% * 2)$ chance. Table A-5 is the aggressive opponent and attacker (caller) reward table.

Table A-5. Reward Table of the Aggressive Opponent and Attacker (caller).

		Roll										
Call %		2	3	4	5	6	7	8	9	10	11	12
		0%	0%	1%	5%	30%	38%	55%	69%	80%	100%	100%

Defender Reward Structure Modifier. This modifier biases in the opposite direction as now extended rounds are incentivized. Lower value claims have the effect of extending the match simply due to the game rules. In order to defend against human opponents that will attempt to minimize their claims for maximum point totals, our computer strategy is simply to increase the likelihood that low claims will be challenged. We flip the modifier such that there are higher chances of both claims and challenges for lower values. Values < 6 receive a $(+2.5\% * \text{number of rounds modifier})$, while while values > 6 receive a $(-2.5\% * \text{number of rounds modifier})$.

APPENDIX B – Liar’s Dice Interface and Survey Questions

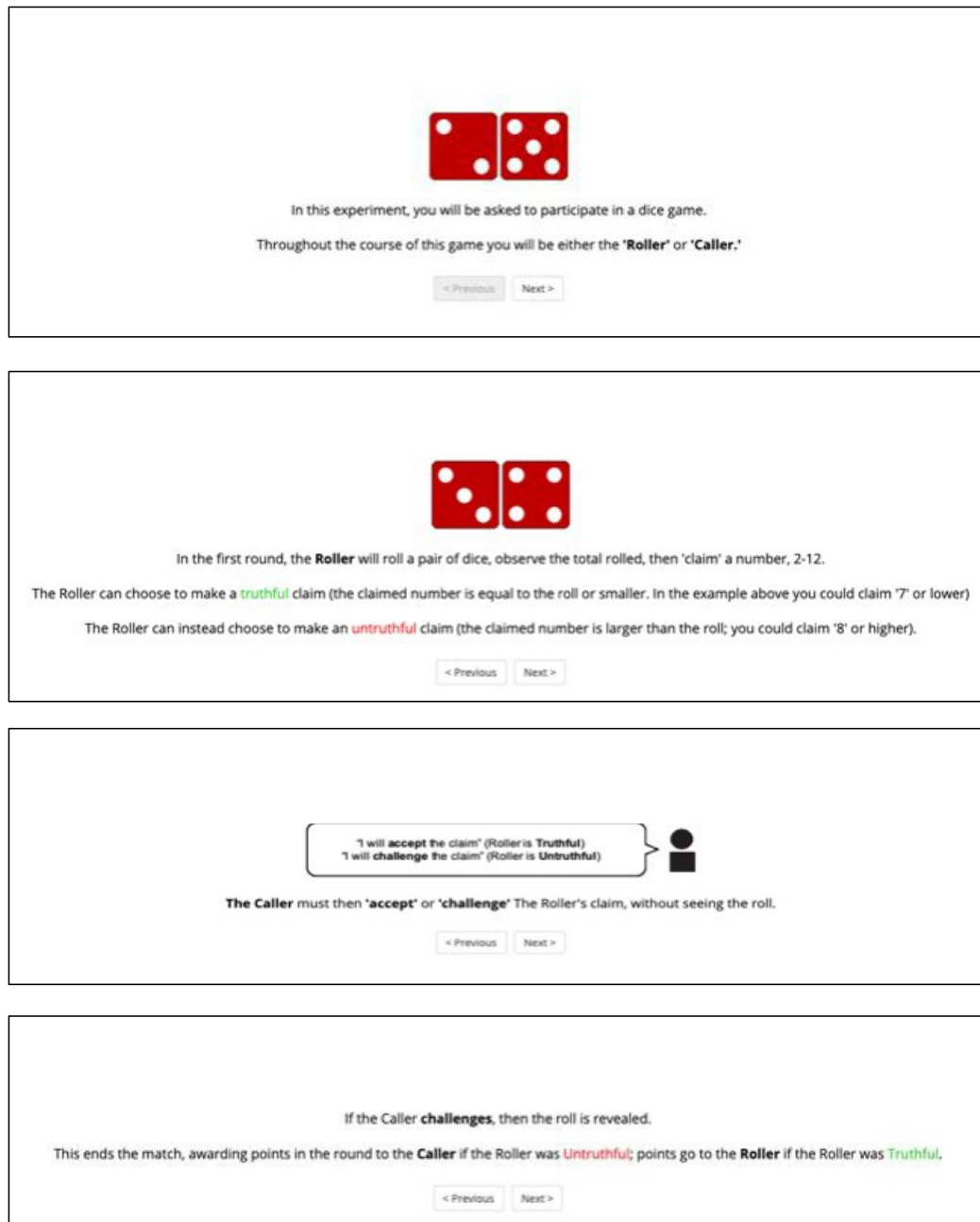


Figure B-1. Instruction Pages of the Experiment (1/3).

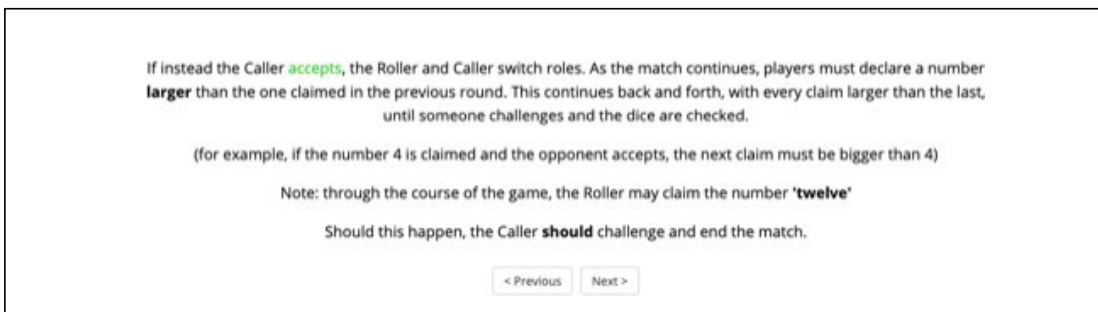


Figure B-2. Instruction Pages of the Experiment (2/3).

	Player 1	Player 2	Points
Round 1	 Claim "2"	 → 	(+1)
Round 2	 ← 	 Claim "5"	(+2)
Round 3	 Claim "6"	 → 	+3 to Player 2

The number of points that the Roller or Caller receives is based on how long you can draw out a match

In other words, it is better to try to challenge (or be challenged) late.

You will be awarded 1 points for successfully challenging in Round 1, and be awarded more points for each successive round (2 pts for win in Round 2, 3 pts for win in Round 3, to a maximum of 5).

However, in order to obtain points, the player must win the overall match

Point values will start at 1 pt.

< Previous Next >

After one player challenges, the match ends and a new match begins. A screen will notify you when a new match is starting.

< Previous Next >

You will be asked after each roll how you **think** the opponent will respond. After a block of 5 matches, you will also be asked general questions about your opponent. At the end of the experiment we will ask a longer set of questions.

These are not worth points, but please carefully consider the questions asked and answer as accurately as possible

< Previous Next >

As we want you to try your best at this game, you will be awarded a BONUS of 5 dollars, should you score over 35 points over the 20 matches you will be playing today.

< Previous Next >

Figure B-3. Instruction Pages of the Experiment (3/3).

You will now play 3 practice matches to help learn the rules.

Don't worry, this will not count towards your final score!

Press Any Key when you're ready to continue

Your Score
0

New Match!

You are starting a new match (**Match 1 of 3**)

Next

Your Score
0



23456789101112

You are the **Roller** this round

You have rolled the above. Please select a number to **claim**.

Remember you are claiming that the chosen number is **less than or equal to** your roll above.

Your Score
0

What do you think your opponent is going to do?

Strongly Believe ChallengeSomewhat Believe ChallengeI don't knowSomewhat Believe AcceptStrongly Believe Accept

Continue

Your Score
0

The opponent

accepted

your claim.

--Press any key to continue--

Figure B-4. Example Interfaces of Practice Matches and the Game Flow when the Participant becomes a Roller and the Opponent Accepts the Claim.

Your Score

10

I understand my opponent's strategy.

Strongly Disagree

Disagree

Neutral

Agree

Strongly Agree

My opponent has a tendency to make claims that are untruthful.

Strongly Disagree

Disagree

Neutral

Agree

Strongly Agree

I think that my opponent is skilled at this game

Strongly Disagree

Disagree

Neutral

Agree

Strongly Agree

I think that I am skilled at this game

Strongly Disagree

Disagree

Neutral

Agree

Strongly Agree

I feel like I know what my opponent will do next.

Strongly Disagree

Disagree

Neutral

Agree

Strongly Agree

Continue

Figure B-5. Survey Questions that we ask when Participants Complete every Five Matches.

Your Score
19



Figure B-6. Exit Survey that is Displayed when Participants Finish 20 Matches (1/3). *These questions are to measure participants' beliefs in trust and deception.*

The image shows a survey form with seven questions, each followed by a horizontal line with 'False' on the left and 'True' on the right, and a small circle in the middle for a response. The questions are:

- I have been a victim of identify theft.
- Someone I know has been a victim of identity theft.
- I have been the subject of an attempted scam over text messages (SMS), phone, or e-mail.
- I regularly use a phone, tablet, laptop, or other computing device in my personal life.
- I maintain an account on a social network service e.g., Facebook, Instagram, Twitter.
- I have enjoyed playing this game.
- Do you feel by the end of the study you understood the opponent's strategy?

Figure B-7. Exit Survey that is Displayed when Participants Finish 20 Matches (2/3).

These questions are to measure participants' beliefs in trust and feedback.

Please select your age range:

18-24 25-34 35-44 45-54 55-64 Above 65

My gender is:

Male Female

Please select your highest completed education level:

Less than Highschool Diploma Highschool Some College Bachelor's Degree Graduate/Professional Degree

Please select your household income:

\$0 - \$24,999 \$25,000 - \$49,999 \$50,000 - \$74,999 \$75,000 - \$99,999 \$100,000 - \$149,999 Greater than \$150,000

Please select your ethnic background:

Caucasian/White Hispanic/Latino African American/Black Native American/Indigenous Peoples Asian/Pacific Islander Other

Please select your marital status:

Single, never married Married or Domestic Partnership Widowed Divorced Separated

Please select your employment status:

Employed for wages Self-employed Out of work and looking for work Out of work and not looking for work Homemaker Student Military Retired Unable to work

Continue

Figure B-8. Exit Survey that is Displayed when Participants Finish 20 matches (3/3).
These questions are to collect participants' demographics.

APPENDIX C – Liar’s Dice Cognitive Model

The model is built using the ACT-UP variant of the ACT-R architecture [25] [15]. ACT-UP [26] is a toolkit implementation of ACT-R that greatly accelerates prototyping and expands scaling properties such as speed and size of models. The model is developed using the instance-based learning (IBL) approach [22] that assumes that decisions are made by dynamically generalizing past experiences to the current situation.

Those experiences are stored in the declarative memory of the cognitive architecture, whose mechanisms support adaptive storage and associative retrieval. Experiences are stored in memory as a combination of situation features, decision taken, and observed outcome. The availability of memories is controlled by their activation:

$$A_i = \log \sum_{j=1}^N t_j^{-d}$$

where i is the memory (out of N), t_j is the time since the j^{th} reference, and d is the decay parameter controlling the power law of recency while the summation over all references to that memory provides the power law of practice.

Given a particular situation, relevant memories are retrieved by computing their match score that combines their activation with their degree of relevance:

$$M_i = A_i + \text{MP} \sum_{j=1}^l \text{Sim}(d_j, v_{ij})$$

where j is a feature in the situation representation, d_j is the corresponding value in the current situation, v_{ij} is the corresponding value in memory i , and Sim is the similarity function reflecting the difference between those two values.

Rather than retrieving a single memory, a consensus outcome is generated using the memory blending mechanism satisfying this constraint:

$$V = \text{argmin} \sum_{i=1}^k P_i * \text{Sim}(V_j, v_{ij})^2$$

where V is the consensus value among the set V_j of possible values, and P_i is the probability weight of memory i , reflecting its match score P_i through a Boltzmann softmax distribution.

The representation of instances consists of memory chunks associating the situation context, the decision(s) made, and the outcome and/or utility of the decision(s). The instance representation for this task consists of a single chunk that captures both bidder and caller roles for a given round rather than separate chunks for bidder and caller decision and outcome. The context is represented by the round number, the identity of the player making the bid, the amount of the previous bid, and the roll. The latter is only represented if the model plays the role of the bidder. All those context features, other than player identity, are represented as numbers to enable generalization across similar situations. Two decisions are represented: the claim made by the bidder, and the challenge made by the caller in response to the bid.

Finally, the outcome of the round is represented in terms of which player won, or a tie if the decision made was not to challenge the bid, in which case the game goes on to another round.

Because of the range of bids that can be generated, the bidding model is conceptualized as a control problem [27]. Specifically, the model matches the current context specification, i.e., round, player, previous bid and roll, and the desired outcome, and extracts using the blending mechanism the bid most likely to achieve that outcome under the current situation. The desired outcome differs between attacker and defender roles. For the attacker, the desired outcome for the round is specified as one that results in a challenge by the other player, and a win for the player making the bid. For the defender, the desired outcome for the round is one that results in a decision not to challenge by the other player, and thus a tie for the round. The bid generated in this way is constrained to be within the rules, i.e., it will be changed to one over the previous bid is less or equal to the previous bid, and to a maximum of 12 if it is over that amount. If no bid can be generated, e.g., because no relevant instances yet exist, a random bid is generated by sampling from a uniform distribution in the range of legal bids.

Because the calling model only has to consider a binary action, whether to challenge the bid or not, it is instead conceptualized as a decision-making problem [28]. If the model is in the role of the attacker, it generates an expected outcome for challenging the bid in the current situation (i.e., round number, player identity, previous bid, and current bid). If that expected outcome is a win, then the model decides to challenge the bid. If the expected outcome cannot be generated (e.g., for lack of relevant instances) then the model also decides to call, as a default exploratory mechanism for an attacker. If the expected outcome of challenging the bid is a loss, then the model decides not to challenge. If the model is in the role of the defender, then the expectation outcome for the next round is generated. If that outcome is a loss, then the model challenges the bid. In all other cases, the model decides not to challenge.

Note that these models are not guaranteed to generate optimal behavior. Rather, they generate rational behavior given their cognitive limitations, in terms of available knowledge and cognitive mechanisms and time pressure of decisions. For instance, the models do not have time to consider all possible outcomes, including potential future combinations. Rather, they aim to generate satisficing behavior that adapts to its opponent as its experience increases. In the following results, the model was exercised against the various strategies defined as the opponents in the human subject experiment, in the same sequence experienced by the human participants. The model was reset between runs to simulate independent subjects.