Towards a Framework to Measure Security Expertise in Requirements Analysis

Hanan Hibshi[1,2], Travis Breaux[1]
Institute for Software Research, Carnegie Mellon Univeristy[1]
Pittsburgh, Pennsylvania, USA
College of Computing, King Abdul-Aziz University[2]
Jeddah, Saudi Arabia
{hhibshi,breaux}@cs.cmu.edu

Maria Riaz, Laurie Williams
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
{mriaz,lawilli3}@ncsu.edu

***Abstract*—Research shows that commonly accepted security requirements are not generally applied in practice. Instead of relying on requirements checklists, security experts rely on their expertise and background knowledge to identify security vulnerabilities. To understand the gap between available checklists and practice, we conducted a series of interviews to encode the decision-making process of security experts and novices during security requirements analysis. Participants were asked to analyze two types of artifacts: source code, and network diagrams for vulnerabilities and to apply a requirements checklist to mitigate some of those vulnerabilities. We framed our study using Situation Awareness—a cognitive theory from psychology—to elicit responses that we later analyzed using coding theory and grounded analysis. We report our preliminary results of analyzing two interviews that reveal possible decision-making patterns that could characterize how analysts perceive, comprehend and project future threats which leads them to decide upon requirements and their specifications, in addition, to how experts use assumptions to overcome ambiguity in specifications. Our goal is to build a model that researchers can use to evaluate their security requirements methods against how experts transition through different situation awareness levels in their decision-making process.**

***Index Terms*—Security; requirements analysis; patterns; decision-making; situation awareness**

I. Introduction

Each year, new security breaches exploit well-known vulnerabilities that have obvious, well-documented solutions. Hewlett-Packard's top cyber security risks report in 2011 presents many popular attacks against web applications, such as SQL injection attacks [14]. In addition, the OWASP Top 10[1] web application security vulnerabilities and the SANS Top 20 Critical Security Controls[2] aim to reduce the most common vulnerabilities. Finally, high profile standards bodies publish security control catalogues, including the ISO/IEC 27000 Series standards and the U.S. National Institute of Standards and Technology (NIST) Special Publication 800 Series that contain best practice security requirements. Despite these broadly disseminated, diverse and in-depth sources of security knowledge, information systems continue to be susceptible to known vulnerabilities. These systems operate with poor security practices, such as unencrypted wireless networks, the same administrative password across multiple systems, and unexpired, outdated passwords [2].

The lack of information system security is unlikely due to an absence of security requirements analysis methods. Research in requirements engineering has sought to address security, including abuse and misuse cases [6, 20], anti-goals [16], extended models that use anti-goals to surface vulnerabilities [16], and the use of trust assumptions to construct assurance arguments [12, 13]. Combined with the wealth of available security knowledge, we hypothesize insecure information systems persist because security analysts experience two challenges: a) difficulty in perceiving relevant risks in the context of their information system designs to select appropriate security requirements; and b) difficulty in deciding which requirements are appropriate to minimize risk. We propose that requirements analysis methods evaluation should address these two difficulties.

The contributions of this paper include:

- A novel coding methodology to apply Situation Awareness, which we applied to a new domain (security);
- Security decision making patterns based on Situation Awareness that distinguish novices from experts;

The remainder of this paper is organized as follows: in Section II, we present background on Situation Awareness; in Section III, we present our research method; in Section IV, we present the research results, including an example decision-making pattern from our study; in Section V we conclude with our discussion and future work.

II. Situation Awareness and Security Risk

We investigated security expert decision-making using *Situation Awareness* (SA), which is a framework introduced by Mica R. Endsley in 1988 [9]. In SA, we distinguish a user's "*perception* of the elements in the environment within a volume of time and space, the *comprehension* of their meaning, and the *projection* of their status in the near future" during their engagement with a system. Perception, comprehension and projection are called the *levels* of SA. To illustrate, consider SQL injection, in which an attacker inserts an SQL statement fragment into an input variable (often via a web form) to gain unauthorized database access. When an expert conducts a source code vulnerability assessment, they look for cues in the code to place input sanitization, which is a mitigating security

[1] https://www.owasp.org/index.php/Top_10_2013
[2] http://www.sans.org/critical-security-contro

requirement. Upon finding such cues (perception), analysts proceed to reason about whether the requirement has or has not been implemented (comprehension). Once understood, they can informally predict the likelihood of an SQL injection attack and the consequences on the system (projection).

We believe SA can be used to explain how analysts perform risk assessments. The NIST Special Publication 800-30 defines risk as the product of the *likelihood* that a system vulnerability can be exploited and the *impact* that this exploit will have on the system. The ability to predict likelihood and impact depend on the analyst's ability to project prospective events based on what they have perceived and comprehended about the system's specification. If the expert succeeds in all three SA levels, then they have "good" SA and they should be able to make more informed decisions about security risks. Failure in any level results in "poor" SA that leads to incorrect decisions or no decisions at all. In section III, we describe our method to detect the SA-levels in security expert interviews.

The SA framework is flexible and could be customized according to the needs of a system. Examples of fields in which SA has been applied include military operations [7], command and control [11], cyber security [3, 15] and many others [8, 19]. Researchers have modeled SA in intelligent and adaptive systems [7, 11, 19]. Feng et al. [11] proposed a context-aware decision support system that models situation awareness in a command-control system. Their focus was to have entity agents based on a "rule-based inference engines" that provide decision support for users. They applied Endsley's concepts and focused on "Shared Situation Awareness" along with a computational model that they applied to a case study of a command and control application. Chen et al. extended a cyber intrusion detection system using a formalization of SA concepts; the logic formalization is derived from experts' experiences [3]. Jakobson proposed a framework of situation aware multi-agent systems that could be cyber-attack tolerant [15]. To our knowledge, SA has not been widely adopted in requirements engineering.

III. Research Method

We chose an exploratory, qualitative research method that aims to understand the symbolic and cognitive processes of specific security analysts, as opposed to testing hypotheses against specific variables [5]. The purpose of this approach is to develop a theory of security analysis from a rich dataset that we can later test in a controlled experiment. Consequently, this theory is grounded in the domain and findings from this study are only valid for this dataset. Our method consists of three main phases:

- The *preparation phase*, in which we developed the research protocol, including customizing SA for security analysis, selecting the system artifacts to use in the security analysis, and the security analysts to interview;
- The *interview phase* where we elicited responses from the selected analysts; and
- The *qualitative data analysis phase* in which we coded the interview transcripts and systematically drew inferences from the data.

We employed coding theory [18] to link SA concepts to the dataset and validate whether our observations are consistent and complete with respect to that dataset. In the first cycle, we applied the *hypothesis coding* method to our dataset [18] using a predefined code list derived from Endsley's SA levels; this method tests the validity of the initial code list. In the second cycle, we applied theoretical coding to discover decision-making patterns from the dataset.

We now discuss the three phases.

A. The Preparation Phase

The SA framework can be tailored to the field of interest by mapping SA levels to statements made by domain analysts. We tailored the framework by having the interviewer link the levels to software artifact features by verbally probing the analyst during the interview process. Thus, we expected the dataset to show how analysts build situation awareness and to help us further discover how perceptions of security risk evolve as the analyst's awareness increases. The inability to perceive risk may be due to limitations in analyst' knowledge or ambiguities in the artifacts. We define the SA levels as follows:

- *Level 1: Perception:* the participant acknowledges perceiving security cues in the given artifact. Examples include: "there is a picture of a firewall here" or "there are SQL commands in the code snippet." Each observation excludes any deeper interpretation into the meaning of the perception.
- *Level 2: Comprehension:* the participant explains the meaning of cues that they perceived in Level 1. They provide synthesis of perceived cues, analysis of their interpretations, and comparisons to past experiences or situations. Examples of comprehension include: "the firewall will help control inbound and outbound traffic..." and "the SQL commands are used to access the database which might contain private information, so we need to check the input to those commands, but this is not done in the code..."
- *Level 3: Projection:* the participant has comprehended sufficient information in Level 2, so they can project future events or consequences. In security, projections include potential, foreseeable attacks or failures that result from poor security. Examples include: "this port allows all public traffic, which makes the network prone to attacks... ", or "unchecked input opens the door to SQL injection…"

At Level 3, we expect participants can make security related-decisions. Decisions include steps to modify the system to mitigate, reduce or remove vulnerability. Continuing with the SQL injection example, one decision could be: "this port should be closed" or "a function should be added here that checks the input before passing it to the SQL statement." Closing the port prevents an attacker from exploiting the open port in an attack, whereas checking the input can remove malicious SQL in an SQL-injection attack.

1) Security Artifacts: We presented each participant with two security-related artifacts. We chose the artifacts to reflect a stratified cross-section of a system and its environment, noting that security requirements should be mapped to each artifact in different ways and analysts require different skills to do this mapping:

Source Code: we present participants with JavaScript code snippets, corresponding SQL statements, and a user interface related to the snippet. The code contains two vulnerabilities, an SQL injection attack and unencrypted user information. JavaScript is a subset of a general purpose programming language, i.e., no templates, pointers, or memory management, thus, we expect analysts with general programming language proficiency and knowledge of SQL injection to be able to spot these vulnerabilities. We also list a high level security goal and we ask participants if the goal had been met.

Network Diagrams: we present participants two network diagrams in sequence: ND1 followed by ND2. Diagram ND1 shows an insecure network, and diagram ND2 show a network with security measures that address weaknesses in ND1. We ask participants evaluate ND2 and decide whether it is an improvement over ND1. Finally, we present 15 security requirements to participants, which we explain are part of a security improvement process, and we ask participants to assess whether the network in ND2 satisfies the requirements.

The two selected artifacts are typical examples comparable to what is taught in college-level security courses.

2) Choosing Experts for the Study: In this study, we aimed to observe how security expertise affects requirements analysis. However, security experts are not all equal: some people have more expertise than others in particular areas, and training in academia is different than hands-on practice. To cover a broad range of expertise, we invited former practitioners and Ph.D. students at different stages of matriculation, all working in security.

B. The Interview Phase

We designed the interviews to study the *process used by the expert to* reach a solution or security-related decision, and not to study the correctness of the decision or degree of security improvement. We only ask the following kinds of questions:

- What cues did the participant look at? *(Perception)*
- How were the cues interpreted? *(Comprehension)*
- Why did they interpret a cue that way? *(Comprehension)*
- Do they recognize any other possible interpretations, if so, why? *(Comprehension)*
- What are the future consequences of each interpretation? *(Projection)*
- Based on those projected consequences, what is the best practice? *(Decision)*

Our approach differs from how SA is traditionally studied in human operator environments (e.g., airplane cockpits and nuclear power plants) using the Situational Awareness Global Assessment Technique (SAGAT), in that our participants are not immersed in a simulation per se. Rather, we present artifacts (source code, network diagrams) to participants with prompts to evaluate artifacts for vulnerabilities. We observe their ability to conduct requirements analysis in their proposed modifications (decisions) and evaluation of requirements satisfaction in ND2.

In addition, we ask participants to share information about their decision-making, such as unstated assumptions and what artifacts cues led participant to reach a decision. We were careful not to guide participants in a certain direction by keeping our questions general. We avoided questions such as: what do you *perceive*, *comprehend*, or *project*. For example, if the participant identified an attack scenario, we would follow with "why would you think such an attack would occur", or "could you describe how it could happen?" In their responses, we found participants returning to the artifact to identify cues, and explain their interpretation.

Given our interest in distinguishing novice from expert analysts, we asked participants to provide a brief description of their relevant background. Questions to elicit background information were asked twice: at the interview start, we ask participants about their security background, their education, industry experience, and security topics of interest; at the end, we ask the participant about analysis process they used during the interview and how it relates to their background while the participant is describing their analysis process. Finally, we recorded the interviews for transcription and analysis.

C. The Qualitative Data Analysis Phase

Grounded analysis is used to discover new theory and to apply existing theory in a new context [5]. We apply grounded analysis [5] in three steps: (1) we transcribe the interviews; (2) beginning with our initial coding frame (see Table I), we code the transcripts, while discovering new codes to further explain the data; and (3), we review previously coded datasets to ensure the newly discovered codes were consistently applied across all transcripts. Table I shows the complete coding frame: the first eight codes (P, C, J, D, including the variants that account for uncertainty U*) constitute the initial coding frame and were inspired by Ensley's terminology for the Situation Awareness [8]; the remaining four codes were discovered during our analysis to account for the interview mechanics. Two researchers (the first and third authors) separately coded the transcripts and then met to resolve disagreements. To efficiently identify disagreements, we used the *Fuzzy Lookup Excel add-in* that is based on fuzzy string matching developed at Microsoft Research [1]. Each coder recorded their start and stop times.

To ensure all statements are coded, we applied the null code {NA} to any statements that did not satisfy the coding criteria, such as when participants request a scrap of paper to draw a figure, or when they ask how much time is remaining for the interview, and so on. We code statements, such as: "I took a course in security..." or "I saw on the news a security breach related to this artifact" as background {BG}, including their personal experience and knowledge. If the participant compares and contrasts comprehended information from the artifact to their experience or knowledge, then that information is coded as comprehension {C}. To improve construct validity, the two raters resolved borderline cases by discussing and refining the definitions and heuristics.

After the first cycle coding, we conducted a second cycle or axial coding to identify decision-making patterns. We defined cut offs between coded sequences by sequentially numbering each statement and then assigning group numbers to statements that address the same idea. The groups serve to delineate transitions between units of analysis. We programmatically

extracted SA-level sequences that we later associated with separate, named patterns, and we searched the dataset without the paragraph cut offs to assess pattern validity (i.e., does the SA-level sequence always correspond to the pattern name that we assigned). We recorded false positives from results, which we report as pattern accuracy. We now discuss our results.

TABLE I. SITUATION AWARENESS ANNOTATION CODES

Code Name and Acronym	**Definition and Coding Criteria Used to Determine Applicability of the Code**
Perception (P)	Participant is acknowledging that they can see certain cue(s)
Comprehension (C)	Participant are explaining the meaning of cue(s) and conducting some analysis on the data perceived
Projection (J)	Participant is predicting possible future consequence(s) or risk(s) involved
Decision (D)	Participant is stating their decision.
Uncertain Perception (UP)	Uncertainty at perception level: participant is missing certain data that would help they need to analyze the artifact.
Uncertain Comprehension (UC)	Uncertainty at comprehension level: participant is not missing data but they can't interpret their meaning confidently
Uncertain Projection (UJ)	Uncertainty at projection level: participant cannot predict possible future consequences confidently
Uncertain Decision (UD)	Uncertainty in decision: participant is not confident about the decision that should be made
Assumption (A)	Participant is stating assumption(s)
Ask Question (Q)	Participant is asking the interviewer questions
Probe (Pro)	Interviewer is triggering the participant's thinking with questions or guidance information
Background (BG)	Participant is providing information regarding their personal background
Null code (NA)	Statement is not applicable to code criteria above

D. Pilot Study

We piloted the study on two experts: P1 is an expert with extensive hands-on and academic expertise in networks and systems security; and P2 is a novice who has only academic security experience. The purpose of the pilot is to test our interview protocol and apply any needed modifications to the questions or protocol before conducting additional interviews.

Both participants analyzed the network diagram artifact, but P2 refused to think deeper about certain details and reported more uncertainties. One insight that we observed in the pilot was the ability of the more experienced participant to make assumptions when faced with uncertainty. When the novice participant was faced with uncertainty, their solution was to ask the interviewer clarifying questions. The following excerpt is an example of an assumption that participant P1 made when they analyzed a requirement to implement time synchronization for logging and auditing capabilities (the codes in curly brackets are defined in Table I, above):

```
{UP}I don't see an NTP server on this network{/UP}
{C}but I know that Windows Domain Controller can act
as NTP{/C}, {A}so I am going to assume that when they
install it they'll probably leave that box checked
because it's a default option{/A}.
{D}I think that is probably happening here {/D}
```

When P2 was faced with uncertainty, however, they turned to the interviewer and asked: "`{Q} What kind of software does this thing has? {/Q}`"

An observation during our pilot is that, although we asked participants to verify security requirements, they actually performed requirements validation. An explanation may be that security experts rely on background knowledge and apply known security requirements. We found experts often adding missing requirements, explaining how to apply a requirement, evaluating whether a requirement was feasible, and prioritizing requirements, as P1 does with ND2 when analyzing a requirement from the list pointing out that this specific requirement is less critical than another requirement that they had analyzed earlier in the interview: `{C} but I don't think its as critical as say the DMZ one, but I think its sort of whatever is the next tier of criticality{/C}.`

IV. ANALYSIS RESULTS

In this section, we report preliminary results from analyzing two additional interviews: advanced expertise (P3) and novice level expertise (P5). In total, we conducted 9 interviews; due to space limitations, however, we only present two contrasting examples. We computed Cohen's Kappa to measure inter-rater reliability for agreement between two raters [4], which was 0.65 for participants P3 and P5. Table II shows the number of coded statements for the two participants broken down by code; the two interviews yielded 526 statements covering both participants and all artifacts (code and network diagrams).

Participant P3 has more hands-on experience (+15 years) compared to P5 (almost 10 years). Alternatively, P5 has an M.S. degree in software engineering compared to P3, who has a B.S. degree. Notably, P3 with less formal education, produced more perceptions, comprehensions, projections, and decisions compared to P5 who reported more uncertainties and made less assumptions.

TABLE II. FREQUENCIES OF CODED STATEMENTS FOR PARTICIPANTS P3, P5

Code	**P3**	**P5**
Perception	50	19
Comprehension	69	5
Projection	32	3
Decision	41	10
Uncertain Perception	4	24
Uncertain Comprehension	15	20
Uncertain Projection	2	1
Uncertain Decision	2	1
Probe	93	17
Question	18	5
Assumption	11	2
Background	3	3
NA	64	12
Total	**404**	**122**

Similar to the pilot, participants verified and validate the requirements by providing deeper analysis based on their background knowledge. Participant P3 did this more often, perhaps due to their extensive hands-on experience. The following example shows how P3 analyzes the first requirement and links the requirement to diagram ND2:

```
{P}your firewall.{/P} {C}which is your first point
of entry to both DMZ traffic and intranet site
```

traffic and also to your users, has all of these on separate subnets{/C} {D}So the first rule here about stuff being unavailable all comes down to whether or not this firewall is properly configured. {/D}

Participant P5, however, asked to see more specific information about the underlying software and configuration of the servers and firewalls. P5 also said that without detailed specifications, the requirements analysis could not be performed.

V. Decision-Making Patterns

Frequencies of SA-levels are insufficient to distinguish experts from novices, and conceal the participants' decision-making process and transitions between SA levels. To address this limitation, we extracted decision-making patterns that ground the SA framework in the data. We now present these patterns, using the acronyms from Table I to express patterns.

The classic SA pattern follows Endsley's SA framework: P→C→J→D, wherein perception statements (P) precede comprehension statements (C), and the "→" means the coded statement on the left-hand side appeared before the coded statement on the right-hand side in the transcript. While this pattern did not appear in our dataset, variations on this original pattern do exist, including two occurrences of P→C→J for participant P3 with 100% accuracy (no false positives). Table III summarizes the identified patterns for the two participants, including the pattern sequence (Pattern Seq.), the number of occurrences (Freq.), the accuracy, which is the ratio of verified pattern instances among the total number of occurrences, and the participants who exhibited each pattern. The table presents patterns from the analysis of both participants reviewing all three artifacts (code and network diagrams).

TABLE III. Variations of SA Patterns

Pattern Seq.	Freq.	Accuracy	Participants
P→C→D	2	100%	P3
C→D	9	77%	P3
C→J	6	83%	P3, P5
UP→UC	2	100%	P3, P5
UC→A	2	100%	P3
UC→Q	1	100%	P5

The pattern P→C→D in Table III shows how participant P3 jumps from comprehension to decision without reporting any projections to the interviewer. The patterns C→D and C→J show how a participant appears to skip perception and directly begin comprehension, which in turn, may lead to either projection or decision. These patterns do not assume that participants are not experiencing the missing SA levels. However, it may be that more experienced participants transition between SA levels more quickly in their mind than they can verbally articulate in the interviews.

The remaining three patterns in Table III concern uncertainty. Uncertainty arises when the participants are not certain about a perception (UP), comprehension (UC), and so on. Based on the few observations in Table I, uncertainty may propagate from one level to another (UP→UC), it may lead to an assumption (UC→A), or it may lead to a question (UC→Q). For example, we see participant P5's uncertain perception from looking at the source code that led to uncertain comprehension. Participant P5 expresses what would be ideal is for them to view the complete code and not only the snippet, which could provide missing cues to support their analysis:

{UP}You know, certainly it would be better if I actually had real code in front of me instead of having a printout. But I don't know the schema.{/UP} {UC}So I'm gonna have guesses, right, and then whether or not it's actually any good. I have no way to evaluate whether it's doing the right thing or not without knowing the schema. I'm not sure.{/UC}

The more experienced participant P3 perceives that a firewall is connected to three subnets in ND2, but was not sure what to comprehend about the firewall rules that were not shown. Hence, P3 assumes the following (UC→A):

{A} Presumably it's gonna have routing rules about which subnet can talk to which subnet and which cannot talk to which subnet. {/A}

This assumption improves network security, because it limits potentially malicious communications across the subnets. The less experienced participant P5, however, asked questions when faced with uncertainty (UC→Q):

{P}So looking at this one, first of all, its' nicely colorized lines,{/P} {UC}but I'm not sure if it's to discriminate the different networks or actually that they have different semantics such as like one network is encrypted, one network is non-encrypted.{/UC} {Q} Do people have to have tunnels on here?{/Q} {Q}Are they all administrated by the same domain controller?{/Q}

Less experience could lead to lower confidence, which may explain the difference between participants who produce assumptions instead of questions in the face of uncertainty. Based on how participants described their background, P3 had more hands-on experience with systems compared to P5, who had less hands-on experience and more training in research. This difference in background may explain why we observe more assumptions made by P3 as opposed to more questions asked by P5. Another difference we noticed is that P5 exhibits a stronger ability to trace requirements to network diagram elements and decides whether a requirement was satisfiable based on their comprehension. Participant P5, however, could not trace the requirements and commented that those requirements are too abstract and that they would need to see more detail to make a decision. As we discuss in Section VI, the SA levels provide insight into how security analysts make decisions to improve security.

VI. Discussion

In our study, we observe how the Situation Awareness (SA) framework can be used to measure how security analysts with varying expertise analyze requirements and specifications. The SA framework summarizes human cognition in three levels (perception, comprehension and projection) that were previously applied to user interface design [8, 10]. Compared to prior work, security requirements analysis is a natural fit for SA: analysts inspect visual notations (specifications) to evaluate requirements satisfiability; the failure to satisfy

requirements leads to security failures, which appear as projections in SA. The advantage of SA in this context is the ability to quantify and connect perceptions and comprehensions to failures, noting that novices and experts may command this ability differently with different notations. Recall from Section V that expert and novice participant react to uncertain comprehensions, differently. While the expert was likely more confident to make assumptions about the system's configuration and behavior, the novice instead asked questions in an attempt to obtain more cues that enable deeper analysis.

We believe the SA method described herein can be used to motivate new approaches to requirements presentation and evaluation. Presently, requirements can be characterized and presented using different methods, such as scenarios, use cases, goal models, and so on. What is missing, however, is an empirically valid framework for comparing these methods based on a measurable impact from security analysts. Our early findings suggest that SA can be used to measure how experts react differently to the same artifact and that requirements can be presented in ways to link perceivable cues to desirable projections while accounting for varying levels of expertise. Abstract presentation, such as high-level goals, offer flexible interpretations that may depend on expert abilities to reason through uncertainty, whereas detailed presentations such as textual use cases may be easier for novices who depend on more prescriptive descriptions. Mead and Christian claim that security requirements can be either too vague (high-level) or too constrained (low-level), which makes them unusable [17]; we believe our results further support this claim. As part of future work, we propose to investigate how SA can be used to analyze the effect of ambiguity in requirements and specification and its impact on reaching sound security decisions. The result of future work would be a framework to empirically evaluate new security requirements methods and to measure the role of expertise in those methods based on observable analyst behavior.

VII. Conclusions

In this paper, we present a new approach to measure security requirements expertise and to understand expert decision-making processes. Our contribution is a systematic method to apply the Situation Awareness (SA) framework to trace how analysts move from perception and comprehension to projection and decision-making. We summarize preliminary results to show traces across the SA levels in the form of patterns that can be used to distinguish experts from novices. While the original SA framework aims to model the decision-making process with respect to user interfaces, we are interested in discovering how security analysts comprehend problem descriptions and requirements notations, and how this comprehension leads to changes in requirements or design decisions. In future work, we plan to report results of a larger study and to evaluate these results in a controlled experiment.

Acknowledgment

We thank our study participants and Dr. Jennifer Cowley at SEI-CERT who consulted on SA. This research was funded by Army Research Office (Award #W911NF-09-1-0273) and National Security Agency.

References

[1] A. Arasu, S. Chaudhuri, K. Ganjam, R. Kaushik, "Incorporating String Transformations in Record Matching," *2008 ACM SIGMOD Int'l Conf. Mgmt. Data*, 2008, pp. 1231–1234.

[2] Travis D. Breaux, David L. Baumer. Legally "Reasonable" Security Requirements: A 10-year FTC Retrospective. *Computers and Security*, 30(4):178-193, 2011

[3] P.-C. Chen, P. Liu, J. Yen, and T. Mullen, "Experience-based cyber situation recognition using relaxable logic patterns," *IEEE Int. Multi-Disciplinary Conf. on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2012, pp. 243–250

[4] J. Cohen, "Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit.," *Psych. B.*, 70(4): 213, 1968.

[5] J. Corbin, A. Strauss, "Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory," SAGE Pubs. 2007.

[6] J. McDermott, C. Fox, "Using abuse case models for security requirements analysis," *15th Annu. Comp. Sec. Apps. Conf.*, 1999., pp. 55–64.

[7] G. Digioia and S. Panzieri, "INFUSION: A system for situation and threat assessment in current and foreseen scenarios," in *2012 IEEE Int'l Multi-Disciplinary Conf. on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2012, pp. 316–323.

[8] M. R. Endsley, D. G. Jones, Designing for situation awareness: An approach to user-centered design. Taylor & Francis, 2003.

[9] M. R. Endsley, "Design and evaluation for situation awareness enhancement," *HFES Annual Mtg*., 1988, 32: 97–101.

[10] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *J. Hum. Factors & Erg. Soc.*, 37(1): 32–64, 1995.

[11] Y.-H. Feng, T.-H. Teng, A.-H. Tan, "Modelling situation awareness for Context-aware Decision Support," *Expert Sys. with Apps.*, 36(1): 455–463, Jan. 2009

[12] C. B. Haley, R. C. Laney, J. D. Moffett, B. Nuseibeh, "The effect of trust assumptions on the elaboration of security requirements," *12th IEEE Int'l Req'ts. Engr. Conf.,* 2004, pp. 102–111

[13] C. B. Haley, R. C. Laney, J. D. Moffett, B. Nuseibeh, "Using trust assumptions with security requirements," *Req'ts. Engr.*, 11(2): 138–151, 2006.

[14] *HP Top cyber Security Risks Report*, Hewlett-Packard Development Company, L.P.. Tech. report

[15] G. Jakobson, "Using federated adaptable multi-agent systems in achieving cyber attack tolerant missions," *IEEE Int. Multi-Disciplinary Conf. on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2012, pp. 96–102

[16] A. Van Lamsweerde, S. Brohez, R. De Landtsheer, D. Janssens, "From system goals to intruder anti-goals: attack generation and resolution for security requirements engineering," *RHAS*, 3: 49–56, 2003.

[17] N. R. Mead and T. Christian, "Security Quality Requirements Engineering (SQUARE) Methodology," in *Proc. of the 2005 Workshop on Soft. Engr. for Secure Systems & Building Trustworthy Applications*, New York, NY, USA, 2005, pp. 1–7.

[18] J. Saldana, The Coding Manual for Qualitative Researchers, SAGE Pubs, 2012.

[19] K. E. Schaefer, D. R. Billings, P. A. Hancock, "Robots vs. machines: Identifying user perceptions and classifications," *IEEE Int. Multi-Disciplinary Conf. on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2012, pp. 138–141

[20] G. Sindre, A. L. Opdahl, "Capturing security requirements through misuse cases," *NIK 2001, Norsk Informatikkonferanse 2001, http://www.nik. no/2001*, 2001.