DIGS – A Framework for Discovering Goals for Security Requirements Engineering

Maria Riaz, Jonathan Stallings[Γ], Munindar P. Singh, John Slankas, Laurie Williams
Department of Computer Science
[Γ]Department of Statistics
North Carolina State University
Raleigh, North Carolina, USA
[mriaz, jwstalli, singh, john.slankas, laurie_williams]@ncsu.edu

ABSTRACT

Context: The security goals of a software system provide a foundation for security requirements engineering. Identifying security goals is a process of iteration and refinement, leveraging the knowledge and expertise of the analyst to secure not only the core functionality but the security mechanisms as well. Moreover, a comprehensive security plan should include goals for not only preventing a breach, but also for detecting and appropriately responding in case a breach does occur. *Goal*: The objective of this research is to support analysts in security requirements engineering by providing a framework that supports a systematic and comprehensive discovery of security goals for a software system. *Method*: We develop a framework, Discovering Goals for Security (DIGS), that models the key entities in information security, including assets and security goals. We systematically develop a set of security goal patterns that capture multiple dimensions of security for assets. DIGS explicitly captures the relations and assumptions that underlie security goals to elicit implied goals. We map the goal patterns to NIST controls to help in operationalizing the goals. We evaluate DIGS via a controlled experiment where 28 participants analyzed systems from mobile banking and human resource management domains. *Results*: Participants considered security goals commensurate to the knowledge available to them. Although the overall recall was low given the empirical constraints, participants using DIGS identified more implied goals and felt more confident in completing the task. *Conclusion*: Explicitly providing the additional knowledge for the identification of implied security goals significantly increased the chances of discovering such goals, thereby improving coverage of stakeholder security requirements, even if they are unstated.

CCS Concepts

• **Security and privacy→Software and application security**

DOI: http://dx.doi.org/10.1145/2961111.2962599

Keywords

Security goals; security requirements; controlled experiment.

1. INTRODUCTION

Whereas the majority of software is built for functions other than providing security solutions, almost all software systems require security-related functionality built into the system [13]. Errors in

ESEM '16, September 08-09, 2016, Ciudad Real, Spain
© 2016 ACM. ISBN 978-1-4503-4427-2/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2961111.2962599

the identification of security requirements can lead to serious security concerns for the software system that impact core functionality, leading to loss of reputation, financial penalties, and even legal prosecution. According to Walia and Carver's classification of requirements errors [24], lack of adequate knowledge or expertise is one of the most commonly identified reasons for requirements errors. Moreover, omissions in requirements is the most commonly occurring requirements error [1]. Given that security expertise is limited and minimal resources are available for eliciting security requirements, security requirements are even more likely than other types of requirements to be left unspecified or inadequately specified [19].

Security goals provide a frame of reference for security requirements, not only capturing the rationale for the requirements, but also helping us assess the completeness of requirements [2]. Identifying security goals for a system is one of the initial steps during security requirements engineering [14]. The challenge is not only to identify an initial set of security goals for the system but also to identify any additional security goals that are implied based on the initially identified set of goals. For instance, to prevent a confidentiality breach of assets such as personal information, we need to ensure confidentiality and integrity of an authentication mechanism. Similarly, we might create new assets, such as audit records to keep track of authentication attempts, and should protect the integrity of the newly created audit log asset itself. These implied goals ought to be considered during security analysis to secure not only the core functionality but also the security mechanisms themselves [3]. Moreover, a comprehensive security plan should include goals not only for preventing a security breach, but also for detecting and appropriately responding in case a breach does occur. Systemizing the discovery of security goals is important for a system's security and for minimizing omitted requirements and unstated assumptions.

The objective of this research is to support analysts in security requirements engineering by providing a framework that supports a systematic and comprehensive discovery of security goals for a software system.

We have developed DIGS, a framework for systematically Discovering Goals for Security. The functional requirements and a list of the assets of a software system are input to the DIGS framework, and security goals associated with the initial, or any additionally identified, assets are its output. DIGS helps an analyst in systematic discovery of a system's security goals, such as goals related to confidentiality or accountability of assets, for different security actions (i.e., preventing, detecting, or responding to a breach) by codifying the pertinent knowledge as a set of security goal patterns. We explicitly capture relationships among goals and assumptions associated with goals to elicit implied security goals that should be considered based on the initially identified goals. When using DIGS, an analyst makes a conscious choice to select or not select a security goal pattern for an asset, which can be reasoned about and documented, minimizing errors of omission. Analysts may also revisit and identify goals not considered in an earlier analysis. DIGS supports discovery of security goals and helps organize security goals related to the assets. This organization helps in quickly identifying areas where goals have not been specified and that may need additional security fortification. Moreover, we map the security goal patterns to candidate security mechanisms that can also help in operationalizing the goals.

We evaluate DIGS in identifying implied security goals via a controlled experiment involving the analysis of two real-world systems. Our research contributes the following:

- The DIGS framework for systematically discovering security goals;
- An empirical evaluation of DIGS in a controlled setting; and
- A mapping of security goal patterns to applicable NIST Special Publication 800-53[1] security controls for operationalizing the goals.

This paper is organized as follows: Section 2 presents background and related work. Section 3 discusses the elements of DIGS framework, security goals patterns and implied security goals. Section 4 outlines the evaluation methodology for DIGS followed by results in Section 5. Section 6 provides discussion and lessons learned based on our findings. Section 7 concludes the paper.

2. BACKGROUND AND RELATED WORK

Existing approaches for security requirements engineering (SRE) largely focus on either modeling the security requirements or documenting the process of eliciting security requirements. For modeling security requirements, researchers have proposed goal-based approaches [4] that model an attacker's perspective, for instance, by construction of anti-models [11] and misuse cases [21]. In terms of processes, methodologies such as SQUARE [14] document various steps involved in security requirements engineering and provide guidelines for each step. The success of existing SRE approaches varies based on the skill, knowledge, and experience of the analysts. A comprehensive analysis of security requirements, starting from scratch, is time and resource consuming. A recent case study, documenting the use of SQUARE methodology in SRE, reported an effort of around 12 person-weeks for applying the methodology, with 3 person-days for identifying security goals [23].

Recent efforts have focused on reusing the knowledge of security requirements and automating parts of the SRE process. Organizational learning [10] automatically identifies security requirements in existing requirements artifacts for reuse in similar projects. In addition to explicitly stated requirements, Security Discoverer [17], identifies implied security requirements using supervised machine learning and automatically suggests applicable security requirements templates.

Security patterns capture security knowledge and provide reusable solutions to recurring security problems. Schumacher et al. [20] have documented a number of security patterns including patterns for secure design and architecture as well as security requirements. Yurina Ito et al. [8] have identified a need to investigate the use of security patterns in early phases of software development including analysis and requirements. Slavin et al. have developed an approach using inquiry-cycle model to select appropriate security requirements patterns [22]. Security requirement patterns available in literature cover only a small subset of security requirements, specifically related to access control, audit and some aspects of privacy [19]. The role of various security patterns in problem solving has not been empirically evaluated [16]. Moreover, the patterns cover only prevention and detection related requirements.

We provide a framework to systematically identify a comprehensive set of security goal patterns and implied goals during the initial phases of SRE to strengthen the consequent security requirements. We also organize existing knowledge sources related to security requirements in terms of our goal patterns to support an analyst in navigating through the available information for operationalizing the goals. Moreover, we use empirical software engineering guidelines and practices to evaluate our framework.

3. DIGS FRAMEWORK

A conceptual framework for systematically identifying security goals can assist analysts in considering a comprehensive set of applicable goals and making conscious tradeoffs. We present the DIGS framework in this section.

3.1 Elements of DIGS

We now describe the elements of DIGS in detail.

3.1.1 Assets

DIGS accepts as input a list of assets used and controlled by the software system, along with the functional requirements. The assets of the software system are its sensitive resources and services, such as patient's health record, that need to be protected. Assets can be mutually related, for instance, one asset can be composed of other assets. Assets can be ranked in terms of their security risks. The security risk of an asset is related both to how valuable an asset is and how easily the asset can be attacked. Relative security risks for assets can be computed using existing techniques, such as protection poker [25]. In certain domains, such as military and healthcare, assets are assigned to predefined classes, mostly concerning with different levels of confidentiality and privacy associated with the asset. We do not assume any specific classification for the assets, and ranking assets by security risks is optional. However, if the ranking is available, it can guide the selection of appropriate security mechanisms for operationalizing the goals.

3.1.2 Actions

For each information asset, we have identified a list of action categories that can be permitted or prohibited. We can either explicitly specify prohibited actions, or use the closed-world assumption where any action that is not explicitly permitted on an asset is prohibited. In addition to the standard CRUD (***create, read, update, delete***) actions, we add two additional action categories: (1) storage; and (2) transfer of information, based on our previous analysis of over 11,000 requirements sentences [17]:

- ***store***: actions related to storage and backup of the assets at rest, e.g., backing up log files.
- ***transfer***: actions related to transfer or sharing of the assets, e.g., sending patient health record from one service to another.

[1]http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf

These actions help us consider security of assets starting from the creation of an asset, through usage, storage, or transfer of assets, till the asset expires.

3.1.3 Security Properties

We model the security goals of a system in terms of the assets that need to be protected and the security properties we want to have for the assets. For instance, a security goal can be to ensure 'confidentiality of a patient record' where 'confidentiality' is the property and 'patient record' is the asset. We have identified the following six core categories of security properties [17]. Each security property counters a specific threat in the Microsoft STRIDE[2] threat model.

- Confidentiality (C)
  - counters threat of Information Disclosure
- Integrity (I)
  - counters threat of Tampering
  - counters threat of Elevation of Privileges
- Availability (A)
  - counters threat of Denial of Service
- Identification & Authentication (ID)
  - counters threat of Spoofing
  - counters threat of Elevation of Privileges
- Accountability (AY)
  - counters threat of Repudiation
- Privacy (PR)
  - counters threat of Information Disclosure

The actions on the various assets suggest applicable security properties.

- Confidentiality is important when performing 'read', 'store' and 'transfer' actions.
- Integrity is important when performing 'create, update, delete' and 'transfer' actions.
- Availability is important when performing all six action types. For 'Availability', the asset will be a service or a system functionality.
- Identification & Authentication is important when a system is accessed, prior to performing any of the six action types. If some actions are allowed without authentication, they should be explicitly specified.
- Accountability is important when performing any of the six action types.
- Privacy is important if the owner of information can exercise control over who can access the information during the actions 'read', 'store', and 'transfer'.

3.1.4 Security Actions

Proactively preventing a security breach is the ideal scenario. However, security breaches do occur. In case of a breach, we can know that a breach has occurred and take remedial actions only if the goals related to detecting and responding to a breach have been incorporated in the system [12]. Consider the security goal to ensure the confidentiality of a patient's health record. In case of a security breach, we should detect and respond to the breach as well. The goal of confidentiality of patient's health record can thus be refined into three goals: prevent breach of confidentiality; detect any breach of confidentiality; and respond to each breach of confidentiality. We define the following three main security actions:

- *Prevent* (*p*): proactively prevent a security breach [20].
- *Detect* (*d*): in case of a security breach, detect the breach [20].
- *Respond* (*r*): in case of a security breach, respond to the detected breach [20][5].

Considering each of these security actions during requirements engineering can help in discovering a more comprehensive set of security goals. Moreover, in certain exceptional situations, we may allow access to assets, such as 'patient health record', and later reason about and detect if the access was in accordance with the privacy guidelines or whether a breach has occurred.

3.1.5 Security Mechanisms

A security mechanism is a method, tool, procedure, or control put in place for operationalizing one or more security goals. Different mechanisms may be selected to support a security goal depending on the security action and asset's risk assessment. For instance, access control and encryption are two security mechanisms that support preventing a breach of confidentiality, whereas auditing mechanisms support the detection of a security breach. Similarly, for high-risk assets, we may employ additional mechanisms than for low-risk assets. NIST Special Publication 800-53, specifies a list of security and privacy controls for information systems. NIST categorizes controls in terms of different families such as access control (AC), audit and accountability (AU), identification and authentication (IA), media protection (MP). NIST also provides information about priority and usage of control based on the impact of a security breach. We have additionally mapped NIST controls to the security goals (i.e., the security properties and security actions that the control supports). For instance, in the IA-family of NIST controls, IA-2, IA-3, and IA-9 map to preventing a breach of identification and authentication of actors (users, devices and services respectively). Controls IA-5 and IA-6 map to the implied goals (see Section 3.3) of preventing breaches of confidentiality and integrity of the authentication mechanism itself. Controls IA-10 and IA-11 map to both preventing as well as responding to a breach of authentication mechanism. The complete mapping is available on our project website[3] and provides guidance toward applicable controls based on the identified security goals. We have also developed security requirements patterns based on NIST controls by abstracting and grouping related controls that support the same security goals[3].

3.2 Security Goal Patterns

To support the analysis of security of assets across multiple dimension, we have identified 18 patterns of security goals that cover all combinations of the 6 security properties and 3 security actions discussed earlier. In Table 1, we summarize the goal patterns and list the actions that indicate when different security properties should be considered for specifying security goals. For example, <read | store | transfer> type actions indicate a need for Confidentiality. To abbreviate, each pattern is assigned an identifier as: `<p|d|r>-<C|I|A|ID|AY|PR>`,
e.g., `d-PR` means 'detect a breach of Privacy'.

We can specify security goals for key system assets using the security goal patterns. The actions performed on the assets guide the choice of the applicable security properties. For instance, while reading the asset 'patient health record', we consider security properties of confidentiality, accountability, and privacy of health records as well as availability of system functionality to allow the read action. For each security property, we also consider goals related to all three security actions. We provide a subset of security goals that are identified for a patient's health record in Figure 1.

[2] https://msdn.microsoft.com/en-us/magazine/cc163519.aspx

[3] https://sites.google.com/site/digsstudy/

Table 1. Security goal patterns

Security Action	Security Property	Asset	Actor	Action type
<prevent (p) | detect (d) | respond to (r)> *a breach of*	Confidentiality **(C)**	of **<asset>**	when **<actor>** performs	<read | store | transfer>
	Integrity **(I)**			<create | update | delete | transfer>
	Availability* **(A)**			<create | read | update | delete | store | transfer>
	Id & Authentication ** **(ID)**			
	Accountability **(AY)**			
	Privacy **(PR)**			<read | store | transfer>

* Asset will be a service or system functionality.

** Applicable prior to any system access by default. Explicitly indicate the actions or assets that do not need authentication.

Goal A corresponds with preventing a breach of confidentiality. We can select appropriate security mechanisms, such as NIST control AC-3 for access enforcement, to operationalize this goal. Goal B is related to detecting a breach of privacy. We can use control AU-12 related to audit generation, to operationalize the goal. Goal C is related to responding to a breach of accountability for all types of actions listed in Section 3.1.2. Security goals shown in Figure 1 are generated using the security goal patterns as follows:

- *Goal A:* `prevent` a breach of <u>Confidentiality</u> of <u>patient health record</u> when <u>user</u> <u>reads the data</u> (i.e., `p-C`)
- *Goal B:* `detect` a breach of <u>Privacy</u> of <u>patient health record</u> when <u>user</u> <u>reads the data</u> (i.e., `d-PR`)
- *Goal C:* `respond to` a breach of <u>Accountability</u> of <u>patient health record</u> (i.e., `r-AY`)

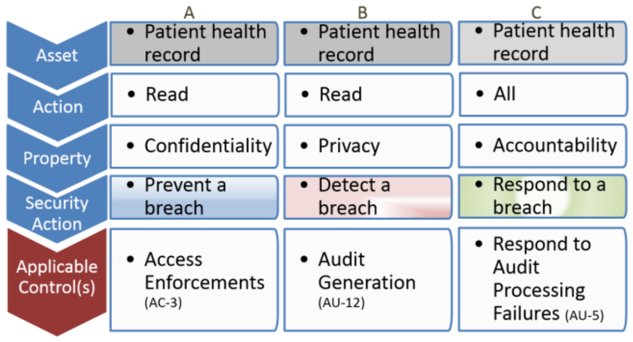

Figure 1. Security goals for 'patient health record'.

Information about the relative security risk of assets may also guide the selection of security mechanisms. To reduce complexity during analysis, we can group the assets that have the same security goals, risks, and mechanisms.

3.3 Implied Security Goals

Based on the initial set of security goals that are identified, other security goals might be applicable. For instance, we might create new assets (e.g., audit records in Figure 1) or incorporate new functionality in the system (e.g., access enforcement mechanisms in Figure 1) to meet the initially identified goals. Security of these new assets or functionality is implied for the overall security of the system. As an example, two security goals that are implied for the security of audit records are to prevent a breach of confidentiality and integrity of the audit records (Goals D and E), as shown in Figure 2.

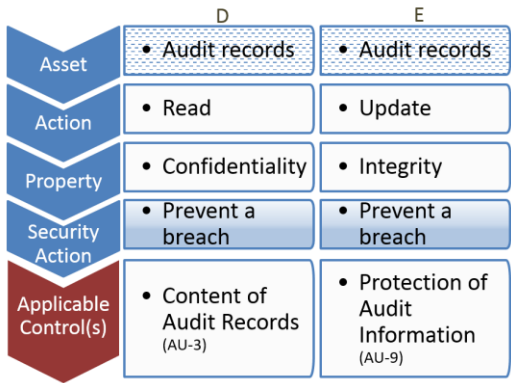

Figure 2. Implied security goals for 'audit records'.

For each of the initial goal patterns, we explicitly capture the implied goals to consider. For instance, to prevent a breach of confidentiality of assets, an implied goal is the integrity of access enforcement mechanisms. Similarly, to detect a breach of confidentiality of assets, an implied goal is the integrity of audit records. To respond to a breach of confidentiality of assets, an implied goal is to have mechanisms in place to temporarily limit system availability. A list of implied goals related to preventing a breach is given in Table 2. A complete list is available online. Moreover, we specify when an implied goal indicates the need for new assets or security-related functionality to be added to the system. For instance, access enforcement mechanisms may employ login credentials, result in the creation of encrypted assets, or in the creation of security-related metadata (e.g., access control lists, security attributes). We can iteratively apply the 18 goal patterns for any newly created assets to have a comprehensive analysis of the security of system's assets.

3.4 Steps for Applying DIGS

The functional requirements and assets of a software system are input to the DIGS framework and security goals associated with the initial, or additionally identified, assets are the output.

To apply DIGS for identifying security goals, repeat until all (initial and additional) assets are considered:

Step 1: Use security goal patterns to identify an initial (or added) set of security goals.

a. Identify goals related to various security properties based on the actions that are performed on the assets.
b. Identify goals related to different security actions, factoring in the asset's risk information, if available.

Table 2. Implied security goals for goal patterns related to preventing a breach

Security Action	Security Property	Additional Security Actions to Consider
prevent a breach of	**<Confidentiality | Integrity>**	**Id & Authentication** of actors
		Availability of *access enforcement mechanisms***
		Privacy of assets
	Availability	**Availability** of *backup functionality***
	ID & Authentication	**Confidentiality** of *identifiers and authenticators* *
		Integrity of *identifiers and authenticators* *
	Accountability	**Id & Authentication** of actors
		Availability of *logging or monitoring services***
		Integrity of *audit records* *
	Privacy	**Confidentiality** of assets
		Integrity of assets, including *consent forms* *

* Consider adding this new asset and related functionality to the system if not already available.

** Consider adding this new functionality to the system if not already available.

Step 2: Identify implied security goals based on the goals.

a. Add implied goals for each selected goal pattern to the set of goals, if applicable.
b. Identify any new functionality that might be added to the system based on 2-a.
c. Identify any new assets that might be created in the system based on 2-a and 2-b.

Consider the asset 'health record'. In Step 1, we identify all applicable security goal patterns for 'health record' and move to Step 2. In Step 2-a, we will look at the rows corresponding to all goal patterns selected in Step 1. For instance, for p-C (prevent a breach of confidentiality), we identify goal for availability of access enforcement mechanism in Step 2-a. Here, 'access enforcement mechanism' is the new functionality identified in Step 2-b. Similarly, 'login credentials' might be a potentially new type of asset related to access enforcement mechanism identified in Step 2-c. After identifying all the implied goals in this way, we go back to Step1 to identify additional goals for the new assets. For 'access enforcement mechanism', we already identified goal for integrity and now consider the remaining patterns to have a comprehensive analysis. The process will come to an end when no new functionality or asset is identified in Step 2-b and Step 2-c. In general, we expect the analyst to iterate through the steps no more than 2-3 times before saturation. Selecting all goal patterns or implied goals may not be feasible. Our objective is that an analyst be able to consider these goals and make conscious tradeoffs about including the respective security goals.

In addition to the discovery of security goals, DIGS supports the organization of the security goals by assets, security properties, and security actions. This organization helps in quickly assessing areas where goals have not been specified and that may need additional security fortification. Analysts may revisit and identify goals not considered in an earlier analysis. In this regard, DIGS may also be used for identifying potentially missing security requirements by mapping existing requirements to DIGS security goal patterns.

4. EVALUATION METHODOLOGY

We conducted a controlled user experiment to evaluate the DIGS framework. We now report our methodology for conducting the experiment, as adopted from Jedlitschka et al [9]. The artifacts used during the study include training material given prior to the study, reference material available during the study, and the forms to submit the task responses. The experiment artifacts and task details are available online[4].

4.1 Goals, Hypotheses, and Metrics

We conducted this experiment to evaluate whether DIGS supports the systematic and thorough discovery of security goals for a system. We analyze the initial set of security goals that participants identify using the 18 security goal patterns as well as any implied security goals identified based on the initial security goals. We did not control for knowledge of security goal patterns (Section 3.2) as both control and treatment groups were already familiarized with these security goal patterns.

The factors of interest that we controlled for are:

- Support of a systematic process for identifying the security goals using the DIGS framework (Section 3.4).
- Explicit knowledge of implied security goals (Section 3.3).

We explore the following null hypotheses:

H_{01}: Support of a systematic process does not impact a participant's ability to identify ***different types of security goals using security goal patterns***.

H_{02}: Explicit knowledge about ***implied security goals*** does not impact a participant's ability to identify such goals.

In Table 3, we list the metrics used for testing each hypothesis.

Hypothesis H_{01} considers differences due to the systematic process used by treatment group given the same knowledge about security goal patterns as control. We test H_{01} using the metrics of precision and recall of the identified security goals. Additionally, we consider recall of security goals related to each security action to see if differences were consistent across the security actions.

Hypothesis H_{02} is based on the differences due to the knowledge of implied goals available to treatment group. We test H_{02} by evaluating the recall of security goals identified initially versus the recall of implied goals identified from the discovered initial goals.

[4] https://sites.google.com/site/digsstudy/

Table 3. Metrics used for evaluation

H_{01}	Precision of security goals identified by individual participants. [TP / (TP + FP)]
	Recall of security goals identified by individual participants. [TP / (TP + FN)]
	Recall of security goals identified by individual participants, grouped by *security actions (i.e., prevention, detection, response).*
H_{02}	Recall of security goals identified by individual participants, grouped by *discovery phases* (*i.e., initial* or *implied*).

We compute the metrics for each participant's response based on an oracle of security goals (see Section 4.5) developed a priori to the evaluation. For each response, we count goals as follows:

- True Positive (TP): A security goal identified by participant that is in the oracle.
- False Positive (FP): A security goal identified by participant that is not in the oracle.
- True Negative (TN): A security goal not identified by participant that is not in the oracle.
- False Negative (FN): A security goal not identified by the participant that is in the oracle.

4.2 Participants

Our study participants were graduate students enrolled in a 16-week Computer and Network Security graduate course (CSC 574, Spring 2016) offered at NCSU. All the students received coursework credit for completing the task, similar to other class exercises. However, students could opt out of participating in the study[5], which would preclude the inclusion of their work in the study results. Of the 29 students present for the lecture, 28 gave consent to participate in the study. We assigned participants to treatment and control groups based on a pre-task quiz (Section 4.3). Each group had 14 participants.

Each participant was assigned a unique random access code to use throughout the tasks so we can link participants' responses across all the tasks they performed. However, we recorded no personally identifiable information about the participants. At the end of the task, participants filled out a post-task questionnaire to document their academic and work experience in computer science and security. Participants in both groups had an average of five years' experience in computer science. Participants had around one year of academic experience related to computer security, on average.

4.3 Experimental Design

The study consisted of three parts: a pre-task quiz, the main task, and a post-task questionnaire.

Pre-task quiz contained 15 multiple choice questions to assess the background knowledge of participants related to security goals based on the provided training material. Participants had 10 minutes to complete the quiz. Once the participants submitted the pre-task quiz (via a Google form), we automatically evaluated the responses. We assigned the responses into three terciles. We randomly assigned half of the participants from each tercile to the treatment and half to the control groups. The average pretask quiz scores for the control and treatment groups were 12.2 and 11.6, respectively, out of 15 points. As a result, we assume that neither group was inherently better at identifying security goals prior to the experiment.

Main task consisted of identifying security goals for two software systems given the system description and key assets, analyzing both systems in randomized order. For the main task, we provided a high-level description of a subset of features for the following two systems to each participant for analysis:

- iHRIS Manage[6] supports the Ministry of Health and other service delivery organizations to track, manage, deploy, and map the health workforce.
- Cyclos, SMS banking module[7], part of a secure and scalable payment software.

We selected these software systems as they manage diverse sets of assets. Operations on these assets cover all action types and require consideration for different security goals, thus allowing for a detailed analysis of the DIGS framework. Moreover, a description of key system features is also available online.

Each participant analyzed both systems, in random order, to identify security goals for each system. Both groups already had knowledge of security goal patterns, as presented in Table 1. Additionally, participants in the treatment group had knowledge of implied security goals based on the initial goals. All of the participants were asked to identify the security goals for the system. Participants in both the treatment and control groups were encouraged that once they have identified an *initial* set of security goals, they should try to identify any *implied* goals based on the discovered initial goals. Differences between the groups, in terms of available knowledge, are summarized in Table 4.

Participants were given 50 minutes to complete both tasks and allocated as much of the time as they wanted for each system. Treatment group participants additionally had to allocate time to understand the systematic process and implied goals given as part of reference material during that time. One participant, in the control group, only provided responses for Cyclos. All other participants analyzed both systems.

Table 4. Differences between Control and Treatment

	Control	Treatment
Knowledge	Security goal patterns (Section 3.2)	Security goal patterns (Section 3.2) + Implied security goals (Section 3.3)
Process	No specific methodology	Methodology outlined for applying DIGS (Section 3.4)

Post-task questionnaire, given at the conclusion of the main task, consisted of questions related to background experience of participants, as well as self-assessment on whether the tasks and methodology to complete the tasks were clear. The purpose of the latter is to determine whether students in the treatment group were able to understand the instructions given to them. This information could potentially explain the evidence or lack of statistical differences between the two task groups. For example, results may indicate no differences between the task groups because the treatment group did not understand the instructions.

[5] The study was approved by the NCSU IRB (6548) and HRPO.

[6] http://www.ihris.org/ihris-suite/health-workforce-software/ihris-manage/

[7] http://www.cyclos.org/mobilebanking/

4.4 Experimental Analysis Methodology

The experimental design followed that of an analysis of variance (ANOVA) of a split-plot design [15] where the whole-plot factor was the task group (control or treatment) and the split-plot factor was the system (iHRIS and Cyclos). We used the restricted maximum likelihood (REML) procedure [6] to fit the statistical model:

$$y_{ijk} = \mu + \alpha_i + e(W)_{ij} + \beta_k + (\alpha\beta)_{ik} + e(S)_{ijk}$$

where y_{ijk} is the response of interest, μ represents the grand mean, α_i represents the effect of the i-th task group, β_k represents the effect of the k-th system, $(\alpha\beta)_{ik}$ represents the interaction effect between the task and system group, $e(W)_{ij}$ represents the whole-plot error, and $e(S)_{ijk}$ represents the split-plot error. Both error terms are independent, normally distributed random variables with zero mean and variances of σ_W^2 and σ_S^2, respectively, and are mutually independent. Testing for task group differences had fewer residual degrees-of-freedom and so had less power than testing for system differences or task-system interactions. The effect size of any significant differences involving the task groups was further investigated using pairwise differences. All analyses were performed in JMP Pro 12[8].

4.5 Oracle of Security Goals

Prior to the evaluation, two of the authors created an oracle of all identifiable security goals for each asset in both the systems used in the study (iHRIS and Cyclos). The researchers involved in the creation of the oracle have 5 and 15 years of relevant experience. As a first step, both researchers individually voted 'YES' or 'NO' for each security goal that can be assigned to an asset based on the 18 patterns. The researchers had substantial agreement at the end of the individual voting based on the Cohen's Kappa score (0.754 for iHRIS; 0.87 for Cyclos; 0.814 overall). The 22 disagreements, out of 252 possible votes, were resolved with discussion. The end result was a consolidated oracle, where any goal voted 'YES' by either of the researchers was included in the oracle. The iHRIS and Cyclos systems had 144 and 108 total possible security goals, respectively, of which 108 and 61 were applicable to the system (voted 'YES'). Given the large number of goals in the oracle for the allocated time (almost 3-4 goals to identify per minute), we do not expect to see high recall values. Each goal in the oracle was categorized as follows:

- Security action: a) prevention; b) detection; or c) response.
- Discovery phase: a) identified based on the initial analysis of system assets using goal patterns (Initial goals); or b) identified based on initial goals (Implied goals).

When analyzing participants' responses, we examined whether different categories of goals were identified by the participants, blind to the group each participant belonged to. During the analysis, we did not find any goal not in the oracle already.

5. RESULTS

We evaluated the participants' responses in terms of the metrics listed in Section 4.1 and present the results in this section.

5.1 H$_{01}$: Security Goal Patterns

As shown in Table 4, both the control and the treatment groups had knowledge of the security goal patterns. We wanted to evaluate if both groups can identify security goals based on the provided goal patterns with or without the DIGS process.

We begin by investigating any significant differences based on the metrics of precision and recall of security goals between control and treatment using the ANOVA test (Section 4.4). All tests have a numerator degree-of-freedom of 1 and denominator degrees-of-freedom of approximately 26, based on the Kenward-Roger method, which is the default for JMP. Figure 3 shows plots of the means and standard errors of the responses by task group and system. An increasing slope in the lines indicates superior performance by the treatment group. In nearly all cases, we see the treatment means are higher than the control means. However, the standard errors are large.

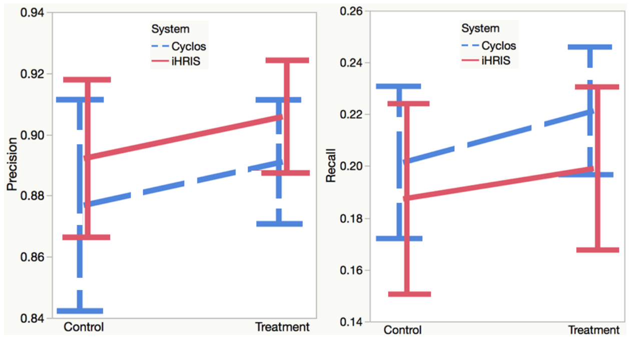

Figure 3. Means and standard errors of Precision and Recall by Task Group and System.

The differences between the performance of the treatment and control groups are not statistically significant, however, as shown in Table 5. On average, both groups had high precision (control: 0.88; treatment: 0.9) but low recall (control: 0.18; treatment; 0.21) since participants are more likely to have false negatives than false positives. Low recall can be explained by a number of factors including the limited time and resources available to participants, and the fact that missing a security goal would have no real consequences for the participants in an empirical setting. Participants spent between 15 and 19 minutes on each system on average identifying around 1 security goal per minute on average (control: 0.89; treatment: 1.24). The time spent on identifying security goals would likely be significantly greater in real life where the stakes are much higher. It is also likely that in practice a team of analysts and stakeholders would identify the system's security goals rather than an individual (see Section 5.3).

Table 5. Analysis of variance for Precision and Recall.

	Precision		Recall	
Effect	**F-value**	**P-value**	**F-value**	**P-value**
Task	0.314	0.580	0.168	0.685
System	0.329	0.571	0.579	0.454
Task* System	0.000	0.989	0.045	0.834

Next, we analyzed the recall of security goals for each of the three security actions to see if participants considered different goal patterns. The analysis of standardized responses for the correctly identified prevention, detection, and response security goals revealed no significant differences between either the task groups or systems. Moreover, we did not find a significant difference in the recall values for goals related to each security action in either group. Figure 4 and Table 6 show the results.

[8] http://www.jmp.com/en_us/software/jmp-pro.html

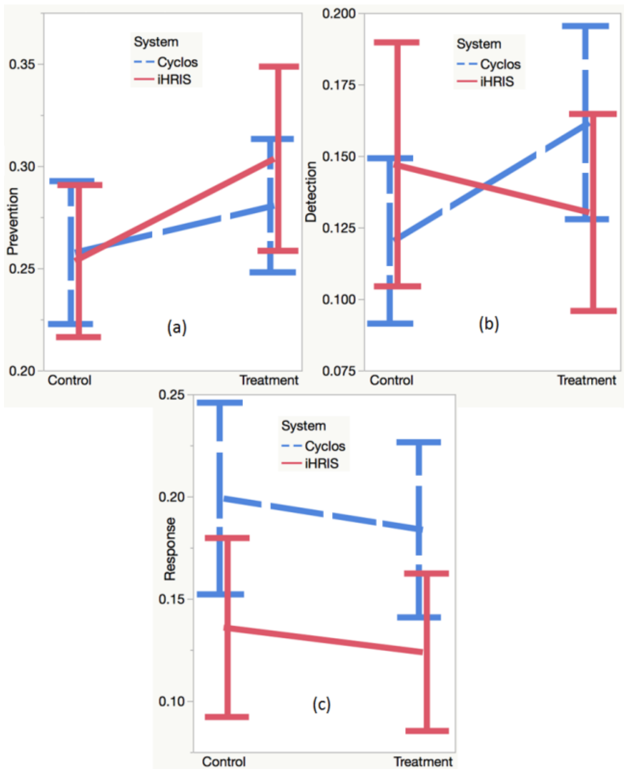

Figure 4. Means and standard errors of standardized responses for the correctly identified (a) Prevention, (b) Detection, and (c) Response security goals.

We fail to reject the null hypothesis H_{01}. Our data indicate that the knowledge captured in the form of security goal patterns is equally accessible to participants in both groups, with or without a systematic process.

We did not manipulate the knowledge of security goal patterns between the treatment and control groups in the current experiment. However, we would expect to see more pronounced differences in recall between the groups if control group did not have the knowledge of security goals patterns [18]. Moreover, considering the security actions for which requirements have been identified in previous experiments [18], we conjecture that knowledge of security goal patterns helped all participants to consider goals related to different security actions. Specifically, current participants in both groups considered goals related to responding to security breaches, which was not considered by participants in previous experiments.

Table 6. Analysis of variance for standardized responses for the correctly identified Prevention, Detection, and Response security goals.

	Prevention		Detection		Response	
Effect	**F-val**	**P-val**	**F-val**	**P-val**	**F-val**	**P-val**
Task	0.659	0.424	0.094	0.762	0.100	0.754
System	0.097	0.759	0.002	0.963	2.848	0.104
Task* System	0.389	0.538	0.850	0.365	0.000	0.990

We analyzed qualitative feedback from participants to see if the treatment and control groups had different perceptions about the task. In the treatment group, 11 of the 14 participants had a positive sentiment about performing the task while 3 participants felt unsure. In contrast, only 4 participants in the control group had a positive sentiment about performing the task, 6 felt unsure and 4 did not respond. Our results indicate that the treatment group was more confident in how the task should be completed, although this increased confidence did not significantly improve their performance in the given time and resource constraints.

Participants in both groups identified goals related to different security actions albeit the overall recall was low. Although providing a systematic process on top of the knowledge codified as security goal patterns does not seem to significantly improve the discovery of security goals, the systematic process can lead to a more positive experience while performing the task.

5.2 H_{02}: Implied Security Goals

We evaluate if the differences in knowledge of implied security goals had an effect on the quantity of implied security goals identified by both groups. We investigated the differences between the groups based on the proportion of correct goals that were identified for the initial set of assets (initial goals) and the proportion of correctly identified implied goals. Figure 5 and Table 7 show significant differences between the groups in identifying the implied goals. Thus we reject the null hypothesis H_{02} that explicit knowledge of implied security goals does not impact participants' ability to elicit these implied goals.

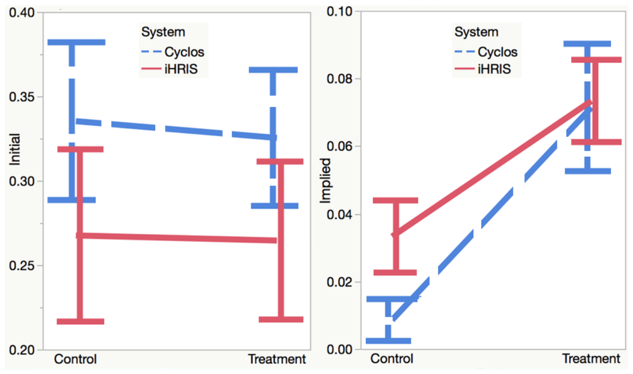

Figure 5. Means and standard errors of standardized Initial and Implied goals correctly identified.

Overall, a small proportion of implied security goals were identified for both the treatment and control groups. This was due to the proportions being calculated with respect to the total number of implied security goals in the oracle, while the participants were only able to identify such goals from their set of discovered initial goals. Differences in the proportions would be even more pronounced if we standardized the proportions with respect to the possible implied goals they could identify.

Table 7. Analysis of variance for standardized Initial and Implied goals.

	Initial		Implied	
Effect	**F-value**	**P-value**	**F-value**	**P-value**
Task	0.020	0.888	10.60	**0.003**
System	3.253	0.083	2.251	0.146
Task* System	0.004	0.953	1.638	0.212

Explicitly coding the knowledge about implied security goals supports the discovery of these goals. However, only a small proportion of implied security goals were identified for both the treatment and control groups overall.

5.3 Combined Security Goals by Group

Given that the recall by individual participants was low overall (Section 5.1), we wanted to compare the control and treatment groups in terms of the recall of the combined set of security goals identified by each group. We considered each group as if they were a team of analysts or stakeholders individually working to identify a set of security goals for the system.

For the combined analysis of recall for each group, any goal in the oracle that was identified by any of the participants in a group was counted as identified by that group. We list the recall values for initial and implied goal, as well as total recall for both the systems for treatment and control groups in Table 8. The control group, as a team, identified 61-62% of applicable security goals, whereas the treatment group identified 74-79% of all goals as a team. The difference in recall between treatment and control groups is more pronounced for the implied goals (12-24% for control vs. 40-54% for treatment). Of the 131 (101 initial; 30 implied) distinct security goals correctly identified by participants overall, the treatment group identified 27 goals (9 initial; 18 implied) that control did not. Whereas the control group identified only one goal that the treatment group did not. We cannot make any claims about the statistical significance of the observed differences based on these recall values since we did not replicate the groups in our experiment (i.e., we had only one control and one treatment group). However, given that neither group was inherently better at identifying security goals to begin with, these results indicate that participants in the treatment group identified a larger pool of security goals based on the additional knowledge and systematic process available to them. These results additionally indicate that different individuals may prioritize different security goals and working as a team, they might be able to carry out a more comprehensive analysis of the system's security.

Table 8. Recall of combined security goals
[C: Control; T: Treatment]

	iHRIS		**Cyclos**	
Goals	**C**	**T**	**C**	**T**
Initial	0.82	0.92	0.94	0.97
Implied	0.24	0.54	0.12	0.4
Total	**0.62**	**0.79**	**0.61**	**0.74**

Participants using DIGS, when considered as a team, identified a larger pool of security goals as compared to the control group.

5.4 Threats to Validity

We considered following threats to internal validity:

Selection: We used results of a pre-task quiz to create groups such that no group was inherently better at identifying security goals than the other. As part of the post-task questionnaire, we further asked participants about relevant expertise. Consequently, groups were evenly balanced in terms of security expertise (approximately one year of academic security experience) except for one participant in the control group who had over eight years of security experience. That participant had the highest total recall in the control group.

Interactions with selection: We do not have knowledge about how motivated or security-aware each participant was. This knowledge might have further helped in assessing why some participants are more inclined to identify security goals as compared to others.

Training: In the time given to perform the task, participants in the treatment group had to additionally understand DIGS whereas control group did not have to understand any new methodology. Although participants using DIGS understood the framework based on the feedback, allowing participants in the treatment group to understand DIGS prior to the task might have levelled the teams in terms of time available to solely focus on the task.

We considered following threats to external validity:

Representativeness of sample population: Participants had around one year of security related academic experience on average and a few months of security related work experience on average. In this respect, they can be considered equivalent to entry-level, non-expert security practitioners.

Task representativeness: We provided a high-level description of two real world systems from different domains for the task. The task was fairly representative. However, the time and resources to carry out the task were limited.

Experimental constraints that limit realism: Security analysts are usually familiar with the problem domain and work as part of a team to identify security goals over a period of weeks or months. The experimental constraints could have led to overall low recall of security goals.

We considered following threats to construct validity:

Measures used: We used standard measures of precision and recall computed against a-priori established oracle to assess participants' performance in identifying security goals.

6. DISCUSSION AND LESSONS LEARNED

We have presented the DIGS framework and results evaluating the framework using a split-plot design, allowing us to extract more information about the split-plot effects (iHRIS vs Cyclos) and their interactions with the task group without having to add more participants. Two key components of DIGS are the security goal patterns and explicitly documented implied security goals. Participants in the control group were partially familiar with DIGS, specifically, security goal patterns. Both groups identified security goals corresponding to different patterns, covering multiple security properties and actions captured in the patterns. Consequently, we could not evaluate how the control group might have performed in the absence of the knowledge of security goal patterns. However, based on the findings related to implied goals in the current experiment and findings from our previous experiments [18], we would expect to see more pronounced differences between control and treatment in terms of the different types of security goals identified and recall of security goals.

Participants using DIGS reported a more positive experience performing the task as compared to the control group and seemed robust to the effects of fatigue in the current experimental setup. The availability of a systematic process might have lessened the cognitive load on participants thus leading to a more positive experience. This effect is worth exploring in future studies.

Participants having similar understanding of the background concepts based on pre-quiz, performed differently when identifying the security goals. One of the reasons may be that participants who are similar in capability may be different in terms of security-awareness [7]. Future experiments should factor in that additional information when creating groups. Assigning the tasks to groups rather than individuals may additionally affect the discovery of security goals, potentially improving recall.

7. CONCLUSION

We have presented the DIGS framework for systematically discovering security goals for the assets in a system. DIGS supports organizing the security goals related to a particular asset, security property, or security action. This organization is intended to help in quickly identifying areas where goals have not been specified and that may need additional security fortification. By providing a set of 18 security goals patterns and corresponding implied goals, we assist an analyst to consider security of assets from multiple dimensions. We are integrating DIGS in a tool to automate parts of the analysis. We have conducted a controlled experiment to evaluate DIGS in identifying security goals. Our results indicate that participants are able to consider security goals commensurate to the knowledge available to them. Although the overall recall was low, participants using DIGS reported a more positive experience while performing the given tasks. Moreover, when considered as a team, participants using DIGS identified a larger pool of security goals as compared to the control group. Our research contributes towards systematic identification of security goals and helps in considering security requirements to meet those goals that may have been missed otherwise.

8. ACKNOWLEDGMENTS

This work is supported by NSA Science of Security lablet. We are thankful to the lablet and Realsearch group for their collaboration. Special thanks to Dr. William Enck for his help with the study.

9. REFERENCES

[1] Alshazly, A.A., Elfatatry, A.M. and Abougabal, M.S. 2014. Detecting defects in software requirements specification. *Alexandria Engineering Journal*. 53, 3 (2014), 513–527.

[2] Anton, A. and Potts, C. 1998. The Use of Goals to Surface Requirements for Evolving Systems. *Proceedings of the 20th International Conference on Software Engineering* (Washington, DC, 1998), 157–166.

[3] Bishop, M. 2002. An Overview of Computer Security. *Computer Security: Art and Science*. Addison-Wesley.

[4] Elahi, G. and Yu, E. 2007. A Goal Oriented Approach for Modeling and Analyzing Security Trade-Offs. *26th International Conference on Conceptual Modeling* (2007), 375–390.

[5] Firesmith, D. 2004. Specifying Reusable Security Requirements. *Jornal of Object Technology*. 3, 1 (2004), 15.

[6] Harville, D.A. 1977. Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association*. 72, (1977), 320–340.

[7] Hibshi, H., Breaux, T., Riaz, M. and Williams, L. 2014. Towards a framework to measure security expertise in requirements analysis. *1st International Workshop on Evolving Security and Privacy Requirements Engineering (ESPRE)* (Sweden, 2014).

[8] Ito, Y., Washizaki, H., Yoshizawa, M., Fukazawa, Y., Okubo, T., Kaiya, H., Hazeyama, A., Yoshioka, N. and Fernandez, E.B. 2015. Systematic Mapping of Security Patterns Research. *Plop*. (2015).

[9] Jedlitschka, A., Ciolkowski, M. and Pfahl, D. 2008. Reporting experiments in software engineering. *Guide to Advanced Empirical Software Engineering*. Springer London. 201–228.

[10] Kurt Schneider, Eric Knauss, Siv Houmb and Shareeful Islam 2012. Enhancing security requirements engineering by organizational learning. *Requirements Engineering*. 17, 1 (2012), 35–56.

[11] Lamsweerde, A. v. 2004. Elaborating Security Requirements by Construction of Intentional Anti-Models. *International Conference on Software Engineering (ICSE 2004)* (Edinburgh, Scotland, 2004), 148–157.

[12] LaPiedra, J. 2002. *The Information Security Process: Prevention, Detection and Response*.

[13] McGraw, G. 2006. *Software Security: Building Security In*. Addison-Wesley Professional.

[14] Mead, N.R., Houg, E.D. and Stehney, T.R. 2005. *Security Quality Requirements Engineering (SQUARE) Methodology*. Software Engineering Institute.

[15] Montgomery, D. 2012. Nested and Split-Plot Designs. *Design and Analysis of Experiments*. 604–641.

[16] Riaz, M., Breaux, T. and Williams, L. 2015. How Have We Evaluated Software Pattern Application? A Systematic Mapping Study of Research Design Practices. *Information and Software Technology*. 65, (2015), 14–38.

[17] Riaz, M., King, J., Slankas, J. and Williams, L. 2014. Hidden in plain sight: Automatically identifying security requirements from natural language artifacts. *22nd International Requirements Engineering Conference (RE)* (2014), 183–192.

[18] Riaz, M., Slankas, J., King, J. and Williams, L. 2014. Using templates to elicit implied security requirements from functional requirements - A controlled experiment. *8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (2014), 1–10.

[19] Riaz, M. and Williams, L. 2012. Security requirements patterns: Understanding the science behind the art of pattern writing. *Proceedings of the 2nd IEEE International Workshop on Requirements Patterns (RePa)*. (2012), 29–34.

[20] Schumacher, M., Fernandez-Buglioni, E., Hyberston, D., Buschmann, F. and Sommerlad, P. 2006. *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, Ltd.

[21] Sindre, G. and Opdahl, A.L. 2005. Eliciting Security Requirements with Misuse Cases. *Requirements Engineering*. 10, 1 (2005), 34–44.

[22] Slavin, R., Lehker, J.M., Niu, J. and Breaux, T.D. 2014. Managing security requirements patterns using feature diagram hierarchies. *Proceedings of the 22nd IEEE International Requirements Engineering Conference*. (2014), 193–202.

[23] Suleiman, H. and Svetinovic, D. 2013. Evaluating the effectiveness of the security quality requirements engineering (SQUARE) method: A case study using smart grid advanced metering infrastructure. *Requirements Engineering*. 18, 3 (2013), 251–279.

[24] Walia, G.S. and Carver, J.C. 2009. A systematic literature review to identify and classify software requirement errors. *Information and Software Technology*. 51, 7 (2009), 1087.

[25] Williams, L., Meneely, A. and Shipley, G. 2010. Protection Poker : The New Software Security “Game.” *IEEE Security and Privacy*. June (2010), 14–20.