2019-01-01

A framework for teaching security design analysis using case studies and the hybrid flipped classroom

Luburić Nikola, Sladić Goran, Slivka Jelena, Milosavljević Branko

A Framework for Teaching Security Design Analysis Using Case Studies and the Hybrid Flipped Classroom

NIKOLA LUBURIĆ, GORAN SLADIĆ, JELENA SLIVKA, and BRANKO MILOSAVLJEVIĆ,
University of Novi Sad, Faculty of Technical Sciences, Serbia

With ever-greater reliance of the developed world on information and communication technologies, constructing secure software has become a top priority. To produce secure software, security activities need to be integrated throughout the software development lifecycle. One such activity is security design analysis (SDA), which identifies security requirements as early as the software design phase. While considered an important step in software development, the general opinion of information security subject matter experts and researchers is that SDA is challenging to learn and teach. Experimental evidence provided in literature confirms this claim.

To help solve this, we have developed a framework for teaching SDA by utilizing case study analysis and the hybrid flipped classroom approach. We evaluate our framework by performing a comparative analysis between a group of students who attended labs generated using our framework and a group that participated in traditional labs. Our results show that labs created using our framework achieve better learning outcomes for SDA, as opposed to the traditional labs. Secondary contributions of our article include teaching materials, such as lab descriptions and a case study of a hospital information system to be used for SDA.

We outline instructions for using our framework in different contexts, including university courses and corporate training programs. By using our proposed teaching framework, with our or any other case study, we believe that both students and employees can learn the craft of SDA more effectively.

CCS Concepts: • **Social and professional topics** → **Computer science education**; **CS1**; • **Security and privacy** → *Software security engineering*;

Additional Key Words and Phrases: Security-oriented curriculum, cybersecurity education, threat modeling, risk assessment, security development lifecycle, architectural risk analysis, security requirements

ACM Reference format:
Nikola Luburić, Goran Sladić, Jelena Slivka, and Branko Milosavljević. 2019. A Framework for Teaching Security Design Analysis Using Case Studies and the Hybrid Flipped Classroom. *ACM Trans. Comput. Educ.* 19, 3, Article 21 (January 2019), 19 pages.
https://doi.org/10.1145/3289238

1 INTRODUCTION

There is a growing business need for organizations to both utilize computer systems and make them more accessible and connected with their environment. As a result, organizations are becoming more exposed to the threat of cyberattackers. Recent years have seen various

Authors' address: N. Luburić, G. Sladić, J. Slivka, and B. Milosavljević, University of Novi Sad, Faculty of Technical Sciences, Trg Dositeja Obradovića 6, Novi Sad 21101, Serbia; emails: nikola.luburic@uns.ac.rs, sladicg@uns.ac.rs, slivkaje@uns.ac.rs, mbranko@uns.ac.rs.

1946-6226/2019/01-ART21 $15.00
https://doi.org/10.1145/3289238

cybersecurity-related incidents, including ransomware campaigns [10], power grid attacks [17], and discoveries of 20-year-old vulnerabilities in widespread CPUs [8].

To combat the increasing threat of cyberattackers, organizations have started investing in the security of their computer systems. Two approaches to securing computer systems have been distinguished in literature and have been nicknamed "bolting security on" and "building security in" [22]. The former is characterized by a set of tools and practices that can be utilized to increase the security posture of an already-constructed computer system. The latter focuses on activities applied during the production of a computer system with the goal of making the implementation secure. Vendors who create and sell software and computer systems are tasked with providing secure solutions, with security being both built in and bolted on. To achieve this, vendors require a workforce trained in the craft of secure system design and construction, and the supply for such professionals is limited.

At the start of the decade, information security experts Bill Whyte and John Harrison asked the question "Why is secure software so seldom taught?" [32]. In their report, the authors examined the current state of practice in the field of secure software development (SSD) in the broader sense, encompassing security design analysis and other security-related software development lifecycle activities. They concluded that the main obstacle to building more secure software is the lack of a skill base. The authors discovered that most software companies lacked a structured approach to developing secure software and instead relied on "good people" in their workforce.

This lack of interest and incentive from the industry, coupled with the scarcity of funding for SSD research, resulted in a lack of people willing and able to research and teach SSD. As a result, it is hard to find software engineers who are knowledgeable in SSD practices. Indeed, as Chess and Arkin point out [5], finding and recruiting educated security professionals is a challenge, and most organizations are forced to create security personnel from non-security technical staff through expensive training programs.

In the past couple of years, however, cybersecurity has attracted a lot of attention, probably inspired by global events such as Stuxnet [16] and the attack on the Ukrainian power grid [17]. In 2013, for example, "Information Assurance and Security" was officially added as a knowledge area in computing curricula by ACM and IEEE Computer Society [12]. During the 2016 USENIX Workshop on Advances in Security Education, Yue presented a paper that discusses the current state of education in the field of cybersecurity [34]. He points out that in recent years, the importance of cybersecurity research and education has been increasingly recognized by major organizations, including NIST, NSF, NSA, and others. While Yue praises the cybersecurity educational initiatives provided by several institutions, he indicates that many universities still lack adequate offering in their curricula. He reveals that none of the top 50 computer science programs in the US include cybersecurity in their core, while many institutions do not even offer elective courses on the subject.

Since research in the field of SSD has had little support over the years, methodologies in this area are still in their infancy. While there are a number of ideas and initiatives found in the industry and literature, there is a surprising lack of actual experiments and evaluations. For example, a recent systematic mapping study done by researchers from Chile [29] explores methods for system threat identification and mitigation. Of approximately 30 identified methods, only three had some experimental evidence, while half a dozen showed vague signs of application in an industrial setting.

At our institution, we aim to tackle the problems listed above and contribute to the development of a security-aware workforce of software engineers. As part of the introductory course in cybersecurity, we teach a range of cybersecurity concepts, including applied cryptography,

authentication, access control, and SSD. In this article, we present a framework for teaching security design analysis (SDA), an activity that represents one of the core pillars of SSD.

Using our framework, we have developed laboratory exercises, which students completed during the 2016/2017 instance of our course. At the end of the semester, students conduct a complete SDA of a case study that is new to them. We evaluate our approach by comparing the quality of the SDA performed by students from the 2016/2017 course with the quality of the SDA achieved by students from the 2015/2016, who attended traditional laboratory exercises. We determine the quality of the SDA through a set of metrics and evaluation techniques proposed in Reference [24]. We also perform analysis to show that the obtained improvements are statistically significant.

Our comparative analysis determined that laboratory exercises generated using our framework yield better learning outcomes for SDA when compared to the traditional labs. Based on these results, we outline instructions for constructing a course or a workshop using our framework, which can be used to train students or employees on the subject of SDA.

Secondary contributions of this article consist of teaching materials, which include lab descriptions and a case study of a hospital information system.

The rest of the article is structured as follows: In Section 2, we examine SDA methodologies, as well as existing efforts in the field of cybersecurity education. Section 3 presents our framework for teaching SDA. Here we show the parts of the framework and discuss the needed inputs and ways in which to process them to develop laboratory exercises. We conclude this section by illustrating an application of our framework, which results in a set of labs that we used for the 2016/2017 instance of the course. Section 4 describes the framework evaluation. Here we examine the differences between the two instances of our course and explain our experimental design. We further discuss the student participants and offer a detailed analysis of the evaluation metrics used to assess the quality of our method. In Section 5, we discuss our findings and list the limitations of our study. Section 6 concludes our research and analyzes directions for further work.

2 RELATED WORK

When designing our course, we had to find answers to two important questions: what to teach and how to teach it. The first part of this section examines security design analysis techniques described in the literature. The second part of this section focuses on methods used to teach secure software development.

2.1 Security Design Analysis

Before we begin examining the various SDA methods, it is important to define a common understanding of what SDA is, as some ambiguity and inconsistency exist in the literature. SDA, sometimes referred to as threat modeling, entails the assessment of a software or system design regarding its ability to withstand attacks from adversaries [31]. What follows is a list of characteristics, which further describe SDA. This is not an exhaustive list but rather serves as an overview of the primary features of SDA:

- It represents a security-focused software development activity.
- We call targets of SDA *modules*. A module can be a minor component that is part of a software program; a software program; an enterprise information system, which consists of many software programs; or anything in between. The term *module* is used here as an umbrella term to encompass different levels of granularity of software components.
- The input for SDA is a set of design artifacts (i.e. architectural, dataflow, deployment diagrams) and security requirements for the target module.

- The purpose of SDA is to use the input information to determine threats to the target module, vulnerabilities introduced by the module design, as well as mitigation strategies that help achieve a higher security posture for the module.
- The output of SDA is a list of recommendations, which assist decision makers in managing risk, and help guide development staff in constructing a more secure solution. This is often referred to as the threat model document.
- SDA is often conducted by the team (or some subset of it) developing the target module, but it can also be performed by third-party auditors, the internal security team, or some other party.

In recent years, a number of SDA techniques have been proposed, both by the industry and the scientific community. One of the earliest proponents of secure software development was Microsoft, and SDA is a key activity of their security development lifecycle (SDL). During his time in Microsoft, Shostack wrote the book on SDA [28], focusing on different SDA strategies, threat management, supporting tools, and so on. Another prominent author on the subject is Schoenfield. In his book [26], Schoenfield covers attacker-centric SDA and describes his technique, which he illustratively applies to a set of case studies. He points out that SDA is both difficult to teach and difficult to learn and states that, based on his experience, only apprenticeship programs seem to work. Other notable organizations that promote and integrate SDA into their software development lifecycle include the SANS Institute [3], the Open Web Application Security Project (OWASP) [21], Adobe Systems Incorporated [11], and others. While each technique has unique aspects, all share the same common steps. We use these core steps as a baseline for our SDA technique, which we describe in detail in Section 3.3, where we talk about the concrete application of our framework.

Despite the growing interest in SDA, there have been few formal studies that test the effectiveness of these techniques. Scandariato et al. conducted a descriptive study, through which they evaluated Microsoft's STRIDE technique [24], which presents a method for threat identification, one of the main steps of SDA activity. The researchers performed their study in laboratory conditions with the help of 57 students and concluded that STRIDE was not difficult to learn or execute, but it did produce many false negatives and was time-consuming. Researchers from Norway performed a different kind of study, where their goal was to compare the effectiveness of attack trees, as compared to misuse cases [20]. Two experiments were conducted, where the first had 28 participants, while the second had 35. The results showed that attack trees were more effective when it came to identifying new ways in which a threat might be realized. We integrate both STRIDE and attack trees into the SDA technique that we teach our students.

2.2 Teaching Approaches in Secure Software Development

When it comes to teaching methods in the field of secure software development, there are only a handful of initiatives described in the literature, and almost all of them lack any form of evaluation.

Research was conducted under the NSF project "Developing case studies for information assurance education" [7]. As a result of the project, 12 case studies were created to be used in information assurance and risk management courses. Several publications were released that present the use of these case studies in different courses, where the impact on student learning was assessed. In general, the results showed that the use of the case study method was effective and that it enhanced learning.

Researchers from the Rochester Institute of Technology experimented with several teaching activities as part of their Engineering of Secure Software course [15, 19]. In Reference [19], Meneely and Lucidi introduce the Vulnerability of the Day, an activity whose purpose is to increase awareness of relevant vulnerabilities. During the first 10 minutes of each class, the teaching staff

demonstrate a vulnerability through live code examples and defines appropriate mitigations. While no formal evaluation of the effectiveness of the teaching method was performed, the authors note the overwhelming interest of the students for this activity, measured through questionnaires.

The same team of researchers created another classroom activity [15]. Students had to design a secure system, where one student acts as a malicious insider whose goal is to produce a flawed design, which can allow an attacker to infiltrate the system once it is created. Seen as a form of gamification, this initiative pins the students against each other without knowing who the insider is, stimulating a fun environment in which the students learn about threat analysis through attack and countermeasure identification. No experiment was conducted to test the effectiveness of this method, but student satisfaction was rated using a questionnaire based on a Likert scale.

Another group of researchers from the same institution created a multidisciplinary course that focuses on applied cryptography and examines the algorithmic, engineering, and practical aspects of security [18]. The article describes the various challenges of a security-focused multidisciplinary course and offers a thorough description of their course plan and lab learning goals and procedures. Evaluation of the course is conducted using questionnaires presented to the students to assess whether the learning objectives have been achieved.

Gamification has gained some traction when it comes to teaching cybersecurity. Over the years, several card games were produced to facilitate cybersecurity education, with many focusing on SDA. Denning et al. [6] constructed a card game called *Control-Alt-Hack*, which acts as a lightweight learning tool that raises awareness about cybersecurity and offers teaching staff a lighthearted way to talk about threats, attacks, and countermeasures. They evaluate their game by distributing copies of the game to a dozen information security courses, along with a survey to be filled by the teaching staff. The results of the questionnaire show that the game is well received and accomplishes the goal of facilitating interactive cybersecurity education.

Taking a similar approach, Shostack produced the *Elevation of Privilege* card game with the goal of teaching developers at Microsoft the craft of SDA [27]. Through his research, Shostack identified that the primary challenge to efficient SDA is the lack of intuition when it comes to determining threats, attack vectors. and security controls on a real system. Shostack notes that implementing security features is usually only slightly more challenging than implementing any sort of software feature. However, understanding where an attacker might strike or how an asset might be compromised is something that alludes many software developers. *Elevation of Privilege* was explicitly designed to teach cybersecurity in an enticing, supportive, and non-threatening way. No experimental evaluation is performed to test the efficiency of *Elevation of Privilege*.

One novel approach to developing the intuition that Shostack mentions was proposed by Kohno and Johnson [14]. In their work, the authors concur that the students need to attain a mindset focused on the broader societal and contextual issues surrounding information security. They use science fiction prototyping to stimulate such thinking, where students are asked to research about cutting-edge technologies, extrapolate their development to the near future, and imagine threats, vulnerabilities, attacks, and controls related to these future systems. While the authors note the usefulness of science fiction prototyping, no experiment nor evaluation is presented.

Recently, Carranza and DeCusatis critiqued the conventional approaches employed in cybersecurity education, both at universities and in industry-certified programs [4]. The authors recognized a tendency to emphasize memorization of facts over a more in-depth cognitive understanding of the subject. They propose the use of the flipped classroom model to teach cybersecurity, where students are expected to complete weekly reading assignments, after which they discuss the subject matter with the teaching staff through consultations. Furthermore, the authors examine a variant of the flipped classroom, called the hybrid flipped classroom. Here, students additionally attend group lectures, so as to gain a different view from the textbook on complex topics like

encryption and public key infrastructure. Once again, no formal evaluation was conducted to test the efficiency of the hybrid flipped classroom.

Finally, researchers from the Anderson School of Management examined cybersecurity training initiatives and awareness campaigns held in corporations [13]. The article examines different types of cybersecurity training and awareness methodologies and tools. The authors conclude that the current cybersecurity training and awareness programs are limited in their efficacy and list several ways in which this can be improved, including the use of the flipped classroom.

During our previous iterations of the course, we found that even students who did not attend laboratory exercises had little trouble learning how to use and implement security controls once they knew which specific controls to implement. However, the question of when a security control is needed and when to use a specific security control had alluded many students, even the ones that attended every class, signifying the lack of in-depth understanding of the subject. These findings fall in line with the conclusions made by Carranza and DeCusatis [4], as well as Shostack [27]. Based on this, we decided that the hybrid flipped classroom would be suitable for our context. We gave reading materials for students to learn on their own what specific security controls exist and how to use them, while using the laboratory exercise to put more emphasis on recognizing when to use them.

The most notable research that inspired our new course design is in References [4] and [13], though other examined literature did affect our overall design, especially articles related to the use of case studies in the classroom, developed under the NSF project [7].

3 THE TEACHING FRAMEWORK

In this section, we examine our teaching framework. Section 3.1 outlines the main components, inputs, and outputs of the framework. Section 3.2 describes the process of using our framework. Finally, Section 3.3 demonstrates the use of the framework that we utilized when we created our new laboratory exercises.

3.1 Framework Components

Our proposed framework consists of four parts: (1) the applied security design analysis method, (2) one or more case studies for which the SDA will be conducted, (3) the preparatory materials containing enough information for the lab participants to actively participate in the analysis, and (4) the laboratory exercises constructed by using our framework.

The input for the framework is the SDA method that is the main learning goal. The selected SDA method guides the design of the laboratory exercises. A single workshop might cover a specialized SDA, while SDAs targeting a broad domain may span several courses.

The preparatory materials are the materials that lab participants need to examine before the lab. This can be anything from reading materials (i.e., book chapters, scientific articles, blog posts) to videos (i.e. conference presentations, online course segments). In the context of security design analysis, these materials should detail vulnerabilities, attacks, and/or security controls, grouped around the distinct SDA subactivity.

The case study is the target of the SDA application and represents a module (as described in Section 2.1. The size and complexity of the case study dictates the time required for a complete analysis to be conducted. A single case study might be sufficient to cover all aspects of an SDA, though multiple case studies may be examined to cover the SDA in-depth and/or reinforce learning.

Finally, the laboratory exercises are the main output of the framework that contain preparatory materials for the lab participants and guidelines for the teacher on how to apply the SDA on the case study, using the information described in the preparatory materials.

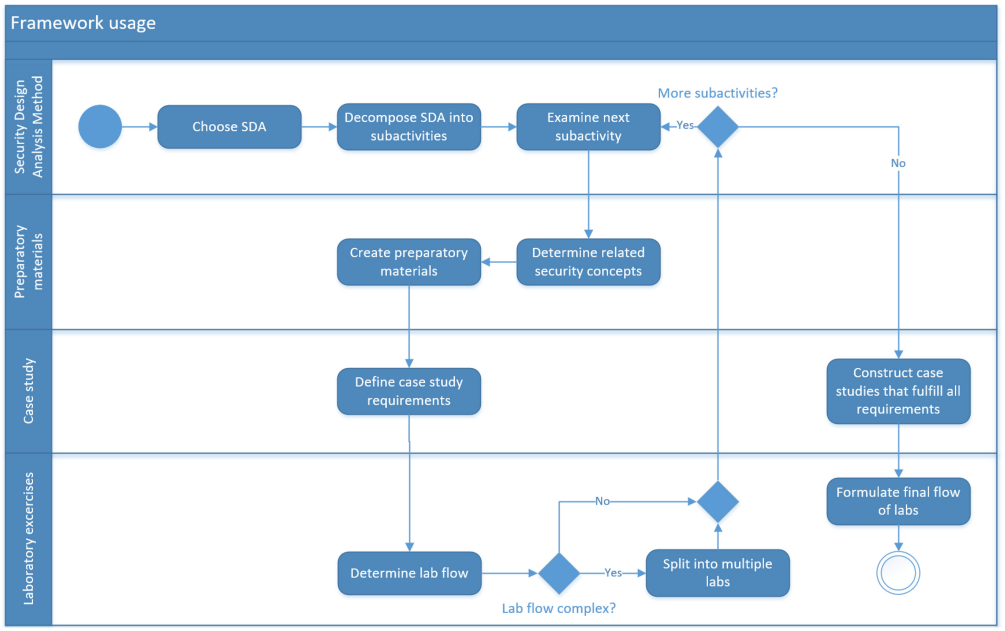

Fig. 1. Usage of the teaching framework to construct laboratory exercises.

3.2 Framework Usage

Our teaching framework helps formulate the laboratory exercises for teaching SDA. The usage of our framework is a process that takes as input the components listed in the previous section, as well as the constraints regarding the number of labs available and the length of each lab and outputs a set of laboratory exercises. The process is illustrated in Figure 1.

The first step of this process is to choose the concrete SDA method and set it as the learning goal of the labs. The next step entails dividing the SDA activity into aspects or subactivities. The decomposition of the SDA method should take into account the complexity of the method and the time limitations (number of labs, duration of each lab).

Each subactivity is analyzed to determine the related security concepts (i.e., attacks, vulnerabilities, security controls). Preparatory materials are created that teach the identified security concepts. These should contain information that is easy to consume, while being relevant for the discussions during the lab. It should be noted that the construction of preparatory materials can be a vast undertaking if the subject matter is complicated.

To construct suitable case studies, case study requirements for the particular SDA subactivity and identified security concepts need to be collected. For example, if a subactivity of an SDA examines threats of loss of confidential data, then the case study needs to work with sensitive data. Requirements for such a case study might be that it handles sensitive data, transports it over internal and external networks, and stores it inside a database. The breadth of the case study requirements is limited only by the time limit of the lab.

With a general idea of what the case study will look like, the set of security concepts that will be examined during the lab, and the targeted subactivity of the SDA, the lab constructors determine the general flow of the lab. If the lab appears to be too complex for the given time frame, then the lab can be split into multiple labs (provided that there is room to accommodate this).

This loop repeats until preparatory materials and case study requirements are created for each subactivity of the SDA method. Once all case study requirements are defined, the appropriate case studies should be created. The number of case studies can range from one to many, where one case study can be examined during one or multiple labs, while one or multiple case studies can be examined during a single lab. Each case study needs to be documented, so that lab participants can familiarize themselves with it, either through preparatory materials or during the lab exercise.

Finally, once the case study set is defined, the laboratory constructors need to formulate the final flow for all labs before the laboratory exercises are fully created. During this step, labs can be merged, further divided, or omitted based on the constraints of the working environment (equipment, time, etc.).

Following the method of the hybrid-flipped classroom, participants should go over the preparatory materials before the lab exercise. During the exercise, the teacher should guide the participants to complete the subactivity on the relevant segments of the case study by utilizing what they have learned from the preparatory materials and expanding upon that base.

3.3 Framework Application

We utilized our framework in the 2016/2017 instance of our course to create the laboratory exercises. To illustrate the usage of our framework we define our SDA method, our case study and the content of the preparatory materials.

3.3.1 Security Design Analysis Method. Following the process depicted in Figure 1, we first choose the SDA method. From the examined methods in Section 2.1, we extracted the common steps into an SDA method suitable for web-based enterprise information systems, as our students were familiar with such systems. Our method most resembles the one proposed by OWASP [21] and consists of the following three high level steps:

(1) Decomposing the module;
(2) Threat analysis;
(3) Risk analysis.

The first step takes as input the functional and security requirements of the system, as well as all documents and diagrams produced during the design phase (i.e. use case diagrams, deployment diagrams, etc.). Assets that require protection are identified. Actors that interact with the system are listed, including the entry and exit points through which their interaction takes place. Trust boundaries are determined and dataflow diagrams are produced to visualize both the data flowing through the system and the attack surface.

Once a proper understanding of the system has been achieved, threats are identified during the threat analysis step. We define threats as events that harm security objectives of assets. If a data asset is confidential, then a threat is the loss of that confidentiality. This is in line with Microsoft's STRIDE technique [24] for threat classification and discovery. We integrate STRIDE into our SDA method and use it to divide our SDA into subactivities. To examine identified threats in more detail, and determine the vulnerabilities of the system and attacks that realize the threat, we decompose threats using attack trees [25]. The final part of the threat analysis step is determining countermeasures at the design, implementation, deployment, and/or operations level.

Finally, during risk analysis, the identified threats are assessed based on the impact and the likelihood of it occurring. Risk mitigation strategies are selected, and final security controls are determined. This might include changes to the system design, additional work for the development or deployment team, or even additional activities required by management (i.e., the installation of annual social engineering educational courses for employees).

Table 1. Defined Subactivities of the Chosen SDA Method

Subactivity	Description
Decomposing the module	The case study is introduced. Subsystems, actors, assets, entry points, and dataflows are identified. The output of this step is a set of dataflow diagrams of the system and a set of threat agents.
Threat analysis: Information disclosure, Tampering	Threats related to information disclosure and tampering are identified and decomposed for the target module. Focus is placed on attacks and vulnerabilities mitigated by cryptographic controls.
Threat analysis: Denial of service	Threats related to denial of service are identified and decomposed for the target module. Focus is placed on attacks and vulnerabilities mitigated by network segmentation, high availability design, and DDoS protection controls.
Threat analysis: Spoofing, Repudiation	Threats related to spoofing and repudiation are identified and decomposed for the target module. Focus is placed on attacks and vulnerabilities mitigated by authentication and logging controls.
Threat analysis: Elevation of privilege	Threats related to elevation of privilege are identified and decomposed for the target module. Focus is placed on attacks and vulnerabilities mitigated by access control and input validation controls.
Risk analysis	Security requirements for the particular case study are examined to determine the impact of each threat. Basic risk calculation is performed to determine a prioritized list of security controls.

Table 2. Preparatory Materials Related to Chosen SDA Subactivities Listed in Table 1

Subactivity	Preparatory material content
Decomposing the module	Case study description
Threat analysis: Information disclosure, Tampering	Symmetric ciphers, asymmetric ciphers, hash functions PKI, digital signatures, TLS, password attacks
Threat analysis: Denial of service	Distributed denial of service attacks, High availability design
Threat analysis: Spoofing, Repudiation	Authentication controls, multi-factor authentication, session management, logging, spoofing attacks
Threat analysis: Elevation of privilege	Role-based access control, access control lists, input validation controls, injection attacks
Risk analysis	Risk management process description

We divide the selected SDA into six subactivities, which are listed in Table 1. Each subactivity is examined during one laboratory session. When appropriate, we analyze previous subactivities to introduce new attacks, vulnerabilities, and/or controls relevant to previous threats.

3.3.2 Preparatory Materials. Our preparatory materials contain cybersecurity concepts, attacks, vulnerabilities, and countermeasures concerning web-based information systems. The concepts covered in the materials are grouped based on the related subactivity, as demonstrated in Table 2.

3.3.3 Case Study. Based on the subacitivites listed in Table 1, and the security concepts identified in Table 2, we constructed a set of requirements for the case study, which are presented in Table 3.

Table 3. Case Study Requirements Related to Chosen SDA Subactivities Listed in Table 1

Subactivity	Case study requirements
Decomposing the module	System needs to be web-based, consist of multiple applications, and familiar to the context of the participants.
Threat analysis: Information disclosure, Tampering	System should process, transmit, and store sensitive data. System should work with password-based authentication.
Threat analysis: Denial of service	Some part of the system should require high availability. Some part of the system should communicate over the Internet.
Threat analysis: Spoofing, Repudiation	System should have password-based authentication for one group of users and multi-factor authentication for another group.
	System should contain service-to-service communication that requires authentication.
	System should contain sensitive functions that demand accountability.
Threat analysis: Elevation of privilege	System needs to have some form of shared user interface to demonstrate access control.
	System needs to have OS-level assets (i.e. files) that require access control.
	System needs to have some forms of command interpreters (i.e. SQL database, XML parser).
Risk analysis	System needs to have some form of security requirements derived from laws and regulations.

For our case study, we formed a description of a hospital information system (HIS) by examining scientific articles and grey literature related to applications of technology to the healthcare domain. The HIS was chosen as a suitable and highly relevant case study for the security assessment. First, it can be quite complex, servicing a wide array of different actors, which makes choosing a representative subset of functionality that much easier. There have been a number of papers and articles that call for, examine, and present technological innovation in the field of healthcare [9, 30]. As people live longer, more attention is directed at Smart Health systems that optimize the healthcare industry and reduce costs. The second reason the security analysis of such a system is suitable for our needs is the fact that hospitals deal with sensitive data. In particular, health records have been a major target of cybercriminals, but there is also a wide array of sensitive data common to most business systems, like personally identifiable information, financial records, and system user credentials. There are a number of articles in the literature calling for and proposing different security measures to protect these systems [2].

Our HIS case study is imagined to be deployed on a set of machines inside the hospital. The system interacts with a number of different actors, each chosen to present a particular set of attacks. This includes:

- *Hospital management*, which consists of a number of staff members who handle human resources, finances, hospital equipment, operating room schedules, and so on. This part of the system falls in the domain of business informatics. Managers interacts with the system through a web application, from their workstations that are located inside the hospital;
- *Medical staff*, including physicians, technicians and other relevant subjects concerned with patient management. Physicians use the system to examine their schedule, follow their patient's treatments and health records, communicate with their patients as well as with the management, and so on. Like hospital management, the medical staff interacts with the system inside the corporate network, using a web application;

- *Patients*, which use the system to follow their hospital appointments, recommended diet and therapy, treatment history, and so on. They interact with the system over the internet, using a mobile or web application. Following the current trends in technological development, a patient can have a number of wearable or implanted devices, which monitor the patient's physiological parameters and send them to the HIS;
- *The government*, which periodically contacts the hospital to get statistical data. The government service sends a request over the internet and uses the retrieved data to monitor the health of its citizens, detect early signs of epidemics or high volumes of a particular type of illness.

The resulting flow of the labs generated by our framework are described in detail in Section 4.2. It should be noted that we divide the second subactivity (threat analysis: information disclosure and tampering) into two labs, as the topics covered by this subactivity (cryptographic primitives and applied cryptography) require more time to go through. Furthermore, we do not construct a separate lab for the third subactivity (threat analysis: denial of service), as we consider it, for the most part, out of scope. The parts relevant for our students we integrate into other labs, where appropriate.

4 FRAMEWORK EVALUATION

In this section, we describe the evaluation of our proposed framework. In Section 4.1, we describe the context in which our course resides and the knowledge students gain before attending our course. Section 4.2 describes the structure of the two instances of our course, which we compare as part of our evaluation. Section 4.3 details our experiment design.

4.1 Course Context

Our course is an elective course for fourth-year undergraduate students of computer science studies. Around 50 students enroll in this course each year. Prior to attending our course, the students complete several mandatory courses covering the topics of data modeling, software engineering, network-based systems, distributed software systems, and information systems. They have no prior knowledge related to the domain of information security or any of its subdomains.

Based on student's prior knowledge, we focus our course on topics related to application and system security. As most of our students find employment as software engineers, we focus on topics that offer the most value to future software designers and developers. Topics include applied cryptography, data security (protecting data in transit, storage, and in use), authentication and access control mechanisms, and secure software development, with a focus on security design analysis.

4.2 Course Structure

Two instances of the course are relevant for this research. The 2015/2016 instance (referred to as the *old course*) uses the traditional classroom approach, where the professor holds lectures on a weekly basis for all students, while the teaching assistant (TA) runs laboratory exercises for groups of 10 to 16 students. The laboratory exercises focus on the topic from the previous week's lecture, where students complete assignments that require them to implement or use a specific security control or stop a common attack.

The 2016/2017 instance (referred to as the *new course*) contains one major difference, where the laboratory exercises utilize the hybrid flipped classroom [4] instead of the traditional approach. Preparatory materials containing security controls, attacks, and vulnerabilities are provided to

students beforehand, while the lab exercise is executed as a group discussion, in which students apply the learned content as part of an SDA on the case study.

The lecture materials used in both instances of the course are mostly identical, where the materials for the new course contain minor improvements in content and wording. However, to accommodate the hybrid flipped classroom, the materials for the laboratory exercises are restructured for the new course. In both instances of the course, there are six laboratory exercises, each lasting 2 hours.

The outline of the lab structure for the old course is as follows:

(1) The first lab presents an introduction to information security, with a focus on cybersecurity. A discussion of the digitalization of society takes place, focusing on security aspects. Various motivations of attackers are examined and the basic dictionary related to information security is examined (i.e., confidentiality, integrity, availability, threats, attacks, countermeasures).
(2) The second lab inspects cryptographic primitives, including symmetric ciphers, asymmetric ciphers, and hash functions. Lab participants run code examples and modify them to achieve confidentiality and integrity of message exchange between two actors.
(3) The third lab examines the public key infrastructure (PKI) and key management solutions. Students learn about digital signatures, examine X.509 certificates, and run applications for certificate issuing, revocation, and secure storage. Finally, transport layer security (TLS) is discussed.
(4) The fourth lab focuses on authentication, where students examine different authentication controls (passwords, smart card, biometrics), session management and logging controls. Spoofing and session attacks are discussed, such as session hijacking and cross-site request forgery.
(5) The fifth lab investigates access control mechanisms and ways to circumvent them. Role-based access control and access control lists are discussed. Injection attacks are analyzed that bypass access control checks and present a variety of threats, from information disclosure to denial of service. This includes SQL injection, cross-site scripting, XML injection, and buffer overflows. Input validation controls, such as whitelists and prepared statements, are examined.
(6) The sixth lab examines the hospital information system case study and applies everything that was learned during the past five labs. Participants, guided by the teacher, perform a security design analysis where they try to identify sensitive assets, threats, vulnerabilities, and attacks and define security controls that act as countermeasures.

The new course covers the same topics and uses the same materials but is almost completely restructured. Using the lab materials from the old course, preparatory materials are constructed and given to the students to examine before the lab, while the lab is dedicated to the application of that knowledge through the security design analysis of the HIS. While performing the SDA, the lab participants discuss the content of the preparatory materials, and the teacher helps clarify any misconceptions that arise. The outline of the lab structure for the new course is as follows:

(1) The HIS is introduced and examined as a real-world system that supports healthcare institutions. The system's purpose is discussed, as well as its actors, functions, dataflows, subsystems, and entry points. Initial security requirements are discussed and attackers are determined.
(2) Threats related to the loss of confidentiality and integrity of the data processed by the HIS are examined. Sensitive data assets, such as financial information, patient

information, and user credentials, are discovered, and protective mechanisms are proposed using cryptographic primitives. Cryptographic keys are identified as sensitive assets that require further protection.

(3) Controls proposed in the previous lab are enhanced using mechanisms such as PKI and TLS. Certificate verification is examined and attacks related to these technologies, such as certificate pinning and attacks against TLS, are discussed.

(4) Authentication and logging controls and attacks are examined in the context of the HIS. User interfaces and services that require authentication are determined and appropriate controls are selected. Sensitive actions of the HIS are determined and logging controls that achieve non-repudiation are designed.

(5) Access control and injection attacks are examined in the context of the HIS. Entry points that accept external input are identified and the optimal input validation controls are selected. An access control matrix that defines all user groups, roles, and permissions for the HIS is constructed.

(6) The final lab is spent discussing security requirements and risk analysis. Regulations such as HIPAA [1] are examined in the context of the HIS, and a basic risk analysis method is presented and performed to determine the critical vulnerabilities that need to be patched and the security controls that need to be applied to the HIS.

The main difference between our old and new course design, relevant to our experiment, is in the way we structure the teaching materials. In the new course, the lab time is dedicated to discussion and learning the difficult craft of security design analysis. In our experience, security concepts, such as specific controls, attacks, and vulnerabilities, have generally shown to be easy to learn, which is why we leave this to the students, as part of their preparation for the lab.

4.3 Experiment Design

To successfully complete the 2015/2016 instance of the course, students had to secure a partial banking system implemented as part of another course. They did this by producing a security design analysis of the system before implementing it, after which they implemented the identified security controls in their code and configurations.

To successfully complete the 2016/2017 instance of the course, students had to complete the same project as their predecessors, the 2015/2016 group. They performed a security design analysis of the banking system and then implemented the identified security controls during system development. It should be noted that a full banking system can have hundreds of critical assets, each of which can have several accompanying threats. To reduce the student workload, a subset of the system is examined.

Another thing to note is that during project development of both instances of the course, teams of students had a series of checkpoints for which they had to produce certain deliverables. We did this to track the progress of students more easily, as well as to reduce the chance of cheating.

By providing both groups of students with an identical project and roughly the same amount of time to complete it, we were able to do a comparative study of the effectiveness of our teaching technique, as opposed to the conventional teaching approach.

Students are put into teams of three or four members. By comparing the quality of threat models produced by teams from both groups of students, we evaluate our new design. We have adopted the approach from Reference [24] to define the quality of a threat model as a set of metrics, which include the following:

- The quality of the produced dataflow diagrams (DFDs);
- The quality of the threat identification step.

The quality of the DFDs is measured by examining:

- The number of produced DFDs;
- The average number of elements on each diagram.

We compare DFD sets produced by teams from both groups to a baseline DFD set produced by the professor and the TA.

The quality of the threat identification step is evaluated by looking at the following metrics:

- The number of correctly identified threats (TP, true positives);
- The number of incorrectly identified threats (FP, false positives);
- The number of unidentified relevant threats (FN, false negatives).

The maximum number of relevant threats a student can identify ($T_{rel} = TP + FN$) is closely tied to assets, as relevant threats are defined as losses of security objectives. We define relevant assets as assets for which true positive threats have been identified. This excludes abstract assets such as company reputation and user satisfaction. We also aggregate similar assets into a single asset. For example, all log files located on the bank corporate network are considered a single asset. The list of relevant assets includes but is not limited to:

- Data assets, including user credentials, credit card information, invoices, reports, and so on;
- System assets, including important subsystems and services, log and configuration files, the website, and so on;
- Infrastructure assets, such as workstations, the network, ATMs, and so on.

The maximum number of relevant threats that each student team can identify is directly linked to the concrete assets identified by each team N_{assets}. For each individual team, the professor and TA have analyzed the assets obtained by that team to determine their T_{rel}.

We summarize the obtained results by calculating the average correctness (precision) and average completeness (recall) of the individual teams of both class instances. The correctness of an individual team is calculated by dividing the number of correctly identified threats to the total number of threats identified by a particular team:

$$correctness = \frac{TP}{TP + FP}. \tag{1}$$

The completeness of an individual team is calculated by dividing the number of correctly identified threats with the estimated maximum number of threats (for that individual team):

$$completeness = \frac{TP}{T_{rel}}. \tag{2}$$

Our aim is to determine whether the student teams of 2016/2017 instance have on average achieved significantly higher correctness and completeness compared to 2015/2016 instance student teams. Thus, we analyze the obtained results by applying an unpaired test as we had different subjects in the two test groups. We use the Mann–Whitney test, a non-parametric analog of the unpaired t-test that does not require the assumption of normal distributions. We validate the null hypothesis H_0: "There is no difference between student teams of 2015/2016 instance and student teams of 2016/2017 instance in terms of achieved correctness/completeness." For statistical tests, we set the significance level of $\alpha = 0.05$.

We did not measure the quality of the threat decomposition step, nor the risk analysis step. Regarding threat decomposition, we could not find a suitable way to measure the quality of this step, as no evaluation metrics are proposed in literature. We decided to avoid comparing the quality

Table 4. Quality of the DFDs

Group	2015/2016 students	2016/2017 students	Teaching staff
Avg. # of level 1 DFD elements	9.64	10.71	10
# of level 2 DFDs	2.29	2.24	3
Avg. # of level 2 DFD elements per diagram	6.09	8.56	9.33
# of level 3 DFDs	6.36	0.24	0

of the risk analysis, as the 2016/2017 student group had one whole laboratory dedicated to this activity, while the 2015/2016 group only briefly examined this step.

To measure the effectiveness of our new approach, we only take into account threat model documents produced by teams where the document owner was present during the six laboratory exercises.

Twenty-eight students were present during the six lab sessions of the 2015/2016 instance, while 36 students attended the six lab sessions during the 2016/2017 instance. Not all of these students were threat model document owners as some of them belonged to the same team. Overall, 14 threat models were examined from the 2015/2016 instance of the course, while 17 threat models were looked at from the 2016/2017 instance.

5 RESULTS AND DISCUSSION

In this section, we discuss the results of our experiments and list the limitations of our study.

5.1 Experiment Results

We first analyze the quality of the DFDs produced by the two groups of students and compare them with the quality of the DFDs produced by teaching staff. Each group produced 1 level 1 DFD and multiple lower-level DFDs. We measure the average number of elements on the level 1 DFD, the number of level 2 DFDs, the average number of elements on level 2 DFDs, and the number of level 3 DFDs. When counting the number of elements, we take into account data stores, process nodes, and external entities. The results are presented in Table 4.

While both groups showed similar results when analyzing the high level view of the system through a level 1 DFD, the 2016/2017 student group showed significant improvement when it came to more detailed analysis. In general, the new generation of students showed better understanding of their system from the perspective of dataflows.

The main improvement was related to the significantly lower amount of level 3 DFDs. In general, level 3 DFDs signal a too-detailed model, which rarely adds value to the threat modeling activity. Indeed, of all level 3 diagrams produced by the 2015/2016 student group, not one was produced that introduced a new threat to the system, making the diagram useless.

It is the belief of the authors that this improvement is a direct consequence of previous experience, where we warned the students about excessive level 3 diagrams. Therefore, it is the opinion of the authors that the new teaching approach had little impact on the positive results.

Next, we analyze the number of identified relevant assets N_{assets}, which determines the maximum number of threats that each student team can identify. The average number of identified relevant assets for each course instance is presented in Table 5. The results presented in Table 5 show that the 2016/2017 student group identified more assets. We attribute this to our own increased experience, as we recognized that infrastructure assets were completely missed by the 2015/2016 students, and we put more emphasis on this topic in the next instance of the course.

Table 5. Number of Identified Relevant Assets

Group	2015/2016 students	2016/2017 students	Teaching staff
# of identified assets	9.57	12.18	16

Table 6. Number of Identified Threats

Group	2015/2016 students	2016/2017 students
Avg. # of correctly identified threats (TP)	11.36	22.82
Avg. # of incorrectly identified threats (TN)	5.36	3.18
Avg. # of missed threats (FN)	9.64	4.82
Correctness (precision)	68%	87%
Completeness (recall)	54%	81%

Finally, the results of the threat identification step are presented in Table 6. The results presented in Table 6 show a significant increase in the quality of the threat identification step between the two class instances. The differences in correctness/completeness between the instances of 2015/2016 and 2016/2017 were found to be statistically significant (the null hypothesis H_0 was rejected for completeness with the p-value of $1.78 \times 10^{-5} < 0.05$; for correctness, the null hypothesis was rejected with the p-value of $2.76 \times 10^{-5} < 0.05$). Thus, we may conclude that the student teams of the instance 2016/2017 have achieved significantly better correctness/completeness compared to the student teams of 2015/2016 instance.

Moreover, the authors of Reference [24] measure the correctness and completeness of the threat identification performed by their students when compared to their own threat identification. They choose the 80% threshold for both precision and recall as a good reference point for student success. According to this, we conclude that the 2016/2017 student group has achieved good precision and recall when it comes to threat identification.

It is the opinion of the authors that these results are a direct consequence of our hybrid flipped classroom approach. By supplying students with attacks and security controls beforehand, students were able to discuss and reason about threats and focus on the cognitive aspect of security design analysis during the lab exercises.

5.2 Limitations

Several limitations influence the results of the study. First and foremost, both the professor and the TA have gained certain experience in both teaching and the domain of information security. We cannot accurately determine the quality of the gained knowledge and skill between the old and new course. However, we can safely say that our understanding of both information security and teaching information security has increased in general. As Schoenfield points out [26], threat modeling is an art form, where experience plays a key role in the quality of the produced models. Put simply, if we were to apply our old class design to the next generation, then we are confident that we would get better overall results compared to the 2015/2016 instance of the course, if only marginally better.

The next limitation to consider is the fact that we have not properly measured the change in the quality of courses the students attended prior to our course. This is especially important for the software engineering and network-based systems courses, as we rely on the knowledge students receive here to reason about security. According to the official study program document, no major changes have occurred in the curriculum of these courses. To the best of our knowledge, no minor

changes in the teaching technique or the subject matter of the relevant courses have taken place. However, it is difficult to assess if this is actually the case.

A limitation of our framework is its inherent complexity, which requires additional effort to prepare the course. For the traditional classroom, the teacher prepares course materials that can then be used in the classroom. Our approach involves the construction of preparatory materials, in addition to the case studies that are examined in the classroom. Furthermore, an effort is required to align the preparatory materials, the case studies, and the security analysis method. As with regular course materials, the preparatory materials need to be periodically updated to stay relevant.

The final limitation is the lack of a larger dataset to analyze. Fourteen threat models were examined from the 2015/2016 instance of the course, and 17 from the 2016/2017 instance, which is relatively small scale.

6 CONCLUSIONS

Much effort, time, and money are put into the development and enhancement of products, while the quality of education is something that is often overlooked. As education can be seen as a branch of industry that produces people, it is an industry that needs far more attention from the scientific community, the other branches of industry, and the government.

Secure software development, in the broader sense, encompasses security-related activities that are practiced throughout the software development lifecycle. When it comes to teaching methods in the field of secure software development, there are only a handful of initiatives to be found in the literature, and almost all of them lack any form of evaluation.

We aimed to contribute to the development of a security-aware workforce of software engineers. As part of our course, we designed a novel framework based on the hybrid flipped classroom and case study analysis to teach students the practice of security design analysis. We evaluated our new approach by comparing the quality of the threat model documents produced by the 2015/2016 student group (who attended traditional labs) with the threat model documents produced by the 2016/2017 student group (who attended labs generated using our proposed framework). Our results show that the student teams of instance 2016/2017 have achieved better overall correctness and completeness on the threat identification task compared to the student teams of instance 2015/2016. The applied statistical test shows that the obtained differences are statistically significant. Therefore, our results show that a hybrid flipped classroom approach is the preferred alternative, as opposed to the traditional classroom, when it comes to teaching SDA.

The framework presented in this article can be used to generate a one-day workshop for a conference, a set of training exercises for employees in the corporate environment, or, as we have demonstrated, a set of labs for a university course. While the framework is designed for teaching security design analysis, the target of the SDA can be anything from a web-based system to a hardware chipset.

We will explore several directions as part of future research. First, we will continue to monitor the progress of our future students to provide higher confidence in our current results. While minor improvements are planned for the 2017/2018 instance of the course, we will try to keep the change to a minimum to reevaluate the teaching approach presented here.

Our main goal is to continue refining our teaching methods and our course design by continuously consulting with information security subject matter experts and researchers of the teaching method domain. We aim to offer students an information security course that can appeal to future secure software developers, architects, and security subject matter experts.

Next, we plan to do an in-depth study of recent information security teaching, training, and awareness initiatives and courses described in several articles [13, 23, 29, 33, 34]. Yuan et al. performed a review of the current efforts and available resources for teaching secure software

engineering [33]. They survey a number of secure software development processes, methods, and tools developed by industry and open source community. The authors create a guidebook that might prove useful for our research, where they organize the surveyed resources in an attempt to help educators integrate them into their courses. In Reference [23], a framework for the information security analysis of case studies is presented. While the approach is primarily aimed at project managers performing risk analysis, this form of systematic analysis might prove a valuable addition to our framework, if only to expand the risk analysis step that we cover at a basic level.

REFERENCES

[1] Accountability Act. 1996. Health insurance portability and accountability act of 1996. *Public Law* 104, 191.

[2] Ajit Appari and M. Eric Johnson. 2010. Information security and privacy in healthcare: Current state of research. *Int. J. Internet Enterprise Manage.* 6, 4 (2010), 279–314.

[3] Steven F. Burns. 2005. Threat modeling: A process to ensure application security. *GIAC Security Essentials Certification (GSEC) Practical Assignment* (2005).

[4] Aparicio Carranza and Casimer DeCusatis. 2015. Hybrid implementation of flipped classroom approach to cybersecurity education. *Natl. Cybersecur. Inst. J.* 2, 3 (2015), 45–54.

[5] Brian Chess and Brad Arkin. 2011. Software security in practice. *IEEE Secur. Priv.* 9, 2 (2011), 89–92.

[6] Tamara Denning, Adam Lerner, Adam Shostack, and Tadayoshi Kohno. 2013. Control-alt-hack: The design and evaluation of a card game for computer security awareness and education. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM, 915–928.

[7] National Science Foundation. 2008. Developing case studies for information security education. Retrieved January 14, 2018 from https://www.nsf.gov/awardsearch/showAward?AWD_ID=0737304.

[8] S. Gibbs. 2018. Meltdown and Spectre: 'worst ever' CPU bugs affect virtually all computers. The Guardian. Retrieved January 12, 2018 from https://www.theguardian.com/technology/2018/jan/04/meltdown-spectre-worst-cpu-bugs-ever-found-affect-computers-intel-processors-security-flaw.

[9] Saee Hamine, Emily Gerth-Guyette, Dunia Faulx, Beverly B. Green, and Amy Sarah Ginsburg. 2015. Impact of mHealth chronic disease management on treatment adherence and patient outcomes: A systematic review. *J. Med. Internet Res.* 17, 2 (2015).

[10] A. Hern. 2017. WannaCry, Petya, NotPetya: how ransomware hit the big time in 2017. The Guardian.Retrieved January 12, 2018 from https://www.theguardian.com/technology/2017/dec/30/wannacry-petya-notpetya-ransomwar.

[11] Adobe Systems Incorporated. 2010. Adobe Secure Product Lifecycle. Retrieved August 5, 2017 from http://www.ten-inc.com/presentations/Adobe_privacysecurity.pdf.

[12] Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* ACM, New York, NY.

[13] Sul Kassicieh, Valerie Lipinski, and Alessandro F. Seazzu. 2015. Human centric cyber security: What are the new trends in data protection? In *Proceedings of the 2015 Portland International Conference on Management of Engineering and Technology (PICMET'15)*. IEEE, 1321–1338.

[14] Tadayoshi Kohno and Brian D. Johnson. 2011. Science fiction prototyping and security education: Cultivating contextual and societal thinking in computer security education and beyond. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*. ACM, 9–14.

[15] Daniel E. Krutz, Andrew Meneely, and Samuel A. Malachowsky. 2015. An insider threat activity in a software security course. In *Proceedings of the 2015 IEEE Frontiers in Education Conference (FIE'15)*. IEEE, 1–6.

[16] Ralph Langner. 2011. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Secur. Priv.* 9, 3 (2011), 49–51.

[17] Robert M. Lee, Michael J. Assante, and Tim Conway. 2016. Analysis of the cyber attack on the Ukrainian power grid. *SANS Industrial Control Systems* (2016). Retrieved on January 13, 2018 from https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf.

[18] Marcin Lukowiak, Stanisław Radziszowski, James Vallino, and Christopher Wood. 2014. Cybersecurity education: Bridging the gap between hardware and software domains. *ACM Trans. Comput. Educ.* 14, 1 (2014), 2.

[19] Andrew Meneely and Samuel Lucidi. 2013. Vulnerability of the day: Concrete demonstrations for software engineering undergraduates. In *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 1154–1157.

[20] Andreas L. Opdahl and Guttorm Sindre. 2009. Experimental comparison of attack trees and misuse cases for security threat identification. *Inf. Softw. Technol.* 51, 5 (2009), 916–932.

[21] The Open Web Application Security Project. 2017. Application Threat Modeling. Retrieved January 13, 2018 from https://www.owasp.org/index.php/Application_Threat_Modeling.

[22] James Ransome and Anmol Misra. 2013. *Core Software Security: Security at the Source*. CRC Press, Boca Raton, FL.
[23] Alexandra Savelieva and Sergey Avdoshin. 2016. Integrating case studies into information security education. In *Emerging Trends in Information Systems*. Springer, 99–115.
[24] Riccardo Scandariato, Kim Wuyts, and Wouter Joosen. 2015. A descriptive study of microsoft's threat modeling technique. *Require. Eng.* 20, 2 (2015), 163–180.
[25] Bruce Schneier. 1999. Attack trees. *Dr. Dobb's J.* 24, 12 (1999), 21–29.
[26] Brook S. E. Schoenfield. 2015. *Securing Systems: Applied Security Architecture and Threat Models*. CRC Press, Boca Raton, FL.
[27] Adam Shostack. 2014. Elevation of privilege: Drawing developers into threat modeling. In *3GSE*.
[28] Adam Shostack. 2014. *Threat Modeling: Designing for Security*. John Wiley & Sons.
[29] Paulina Silva, René Noël, Santiago Matalonga, Hernán Astudillo, Diego Gatica, and Gastón Marquez. 2016. Software development initiatives to identify and mitigate security threats-two systematic mapping studies. *CLEI Electron. J.* 19, 3 (2016), 5.
[30] Emmanouil G. Spanakis, Silvina Santana, Manolis Tsiknakis, Kostas Marias, Vangelis Sakkalis, António Teixeira, Joris H. Janssen, Henri de Jong, and Chariklia Tziraki. 2016. Technology-based innovations to foster personalized healthy lifestyles and well-being: A targeted review. *J. Med. Internet Res.* 18, 6 (2016).
[31] Sven Türpe. 2017. The trouble with security requirements. In *Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE'17)*. IEEE, 122–133.
[32] Bill Whyte and John Harrison. 2010. State of practice in secure software: Experts views on best ways ahead. *Software Engineering for Secure Systems: Industrial and Research Perspectives*. IGI Global.
[33] Xiaohong Yuan, Li Yang, Bilan Jones, Huiming Yu, and Bei-Tseng Chu. 2016. Secure software engineering education: Knowledge area, curriculum and resources. *J. Cybersecur. Educ. Res. Prac.* 2016, 1 (2016), 3.
[34] Chuan Yue. 2016. Teaching computer science with cybersecurity education built-in. In *2016 USENIX Workshop on Advances in Security Education (ASE'16)*. USENIX Association, Austin, TX.

Received February 2018; revised October 2018; accepted October 2018