Systems Theory and Information Security: Foundations for a New Educational Approach

Joseph R. Laracy, Thomas Marlowe
Department of Mathematics and Computer Science
Seton Hall University, 400 South Orange Avenue
South Orange, NJ 07079, USA
joseph.laracy@shu.edu
thomas.marlowe@shu.edu

ABSTRACT: *Information security education has traditionally been approached with a variety of tools. Models such as Bell-LaPadula and Clark-Wilson, cryptography, and formal methods seek to design systems without certain classes of vulnerabilities. Red teaming seeks to find vulnerabilities that were missed and security software often removes the vulnerabilities. To a lesser extent, probabilistic risk assessment and game theory have also been applied to assess threats. However, on their own, in isolation, these approaches have not "solved" the information security crisis. Internet security in particular is an area of great concern given the plethora of vulnerabilities that enable threats to confidentiality, integrity, availability, non-repudiation, authorization, authentication, and auditability. A new approach to information security engineering education is necessary that views the Internet as a complex, socio-technical system. A systems perspective acknowledges that security can only be achieved through a holistic model that addresses technological architecture and software processes, organizational behavior, and human factors. This paper suggests a novel method for information security education to identify and characterize current deficiencies in a network security control structure, elucidate the relationship between software/systems engineering and security risks, and inform an architectural description of a secure information system architecture.*

Keywords: Systems Theory, Security Models, Socio-Technical Systems, Information Security Education

Received: 10 December 2017, Revised 19 January 2018, Accepted 27 January 2018

DOI: 10.6025/isej/2018/5/2/35-48

1. Introduction

Given the equivocal use of many terms in the information security field, the following definitions are provided: *Security* is a system property that implies protection of the informational, operational, and physical elements from malicious intent. A *vulnerability* is a weakness in a system that can be exploited to violate the system's intended behavior relative to security. Finally, a *threat* is an intentional action aimed at exploiting a vulnerability.

The global Internet is in a state of crisis. The proliferation of viruses, worms, Trojan horses, SPAM, phishing attacks, and other malicious activity presents a variety of security risks. Virus writers continue to develop Internet worms that self-propagate and infect victim computers, enabling the writers to achieve objectives ranging from remote control, to "back door" access, to data destruction. A market for botnets further exacerbates this problem. Additional threats must be addressed with regard to secure communications. In particular, threats to confidentiality, integrity, availability, non-repudiation, authorization, authentication, and auditability are present for even the most advanced Internet users. Products have been developed as point solutions to some of the aforementioned problems, but a comprehensive solution seems elusive.

The vulnerabilities that enable the current security crisis are deeper than any particular software or hardware application. In fact, they are embedded in the architecture of the Internet itself. From the 1960s to the 1980s, much of network research focused on what one can broadly call "performance." Very successful research was pursued to support heterogeneity of networks, increased throughput, superior robustness, and other performance related qualities. When the popularity of the Internet exploded in the early 1990s, threats began to emerge and security became a significant issue. Thankfully, greater attention is being paid to security issues in software engineering [1].

In universities today, there are several places for security-oriented courses. Departments of Mathematics tend to focus on algorithms and protocols [2]. Computer Science courses may cover those topics as well address issues in preventative software engineering, program analyses, and software implementation [3-5]. The emerging field of Data Science gives emphasis, on the one hand, to consequences for intellectual property, privacy issues, and threat pattern identification [6], and on the other, to the use of data science techniques for detecting and dynamically addressing security and privacy issues [7]. Business / IT courses introduce students to trade-off and economics analysis [8], as well as business processes and best practices for maintaining security [9]. Computer Engineering focuses on infrastructure vulnerabilities, including hardware, devices, and networks, and other engineering disciplines dealing with the physical architecture of buildings, connections, and services, and the interaction with management of physical threats (e.g., fire, flood, or outages) [10]. The full spectrum of threats, preventative techniques, and approaches for the mitigation of consequences is rarely addressed, and even if so, mostly as a collection of topics rather than as an integrated view. While we clearly cannot, either in this paper or in a course, cover the field completely, and while we prioritize threats arising from or realized through the Internet, we feel the need for a systemic synthesis of issues and approaches.

We therefore propose a new approach to information security engineering education, which views the Internet as a complex, socio-technical system. A systems perspective acknowledges that security can only be achieved through a holistic model that addresses technological architecture, organizational behavior, and human factors. This paper suggests a novel method for information security education to identify and characterize current deficiencies in a network security control structure, elucidate the relationship between software engineering and security risks, and inform an architectural description of a secure information system architecture.

2. Toward a Systems-Theoretic Paradigm

The problem of information security is very amenable to a systems approach. According to security expert, Peter Neumann, "holistic approaches are those that consider systems in their entirety rather than just focusing on specific properties or specific components" [11]. It is observed that such complex systems have emergent properties. Systems theorist, Peter Checkland, describes emergence as "the principle that whole entities exhibit properties which are meaningful only when attributed to the whole, not to its parts–e.g., the smell of ammonia. Every model of *a human activity system* exhibits properties as a whole entity which derive from its component activities and their structure, but cannot be reduced to them" [12].

Students must come to appreciate that security is an emergent system property. It is impossible to completely evaluate the security of an individual device, e.g., an individual computer (or even, in many cases, a software application), in isolation. The security of a personal computer or smart phone can only be determined by its relationship within a broader context, i.e., a socio-technical system. For example, a PC might be considered "secure" when it is sitting isolated at home. However, once that computer is connected to a wireless LAN, and therefore the Internet, a whole new class of vulnerabilities emerges. An individual PC might be bolted to a desk, require a boot-up password, and have an encrypted file system. However, a security expert would never classify such a system as "absolutely secure." This is because security is a *system* property.

Exhaustively classifying the technical properties of an individual computer will not imply any level of security for a network system. Students should realize that organizational behaviors and human factors must also be considered. For example, does the organization have such restrictive policies on passwords that users need to write them down on post-it notes and stick them to their monitors? Conversely, are the policies so lax that a user may make the password the same text string as the user name? And does the organization have adequate policies and procedures for access control and permissions?

The necessity of students considering security within a socio-technical space is perhaps best illustrated through the use of covert channels. According to Bruce Schneier, covert channels "are a way to mess with the minds of people working in security-model research" [13]. Covert channels bypass controls such as the "no read up, no write down" rule in the Bell-LaPadula model. Suppose for instance that a malicious insider (e.g., a disaffected employee) wants to write Top Secret information to a database that is only classified as Confidential. The Bell-LaPadula model would prevent the software from executing the instruction. However, what if the insider can manipulate network traffic such as sending two packets in quick succession to mean a logical "1" and two packets spaced out as a "0"? Such an attack would not necessarily be restricted to just the network itself. It could include CPU usage, memory allocation, hard-drive access, print queuing, and other aspects of the system. Even the whitespace in a Top Secret document could be used to encode information [13].

Neumann points out that vulnerabilities usually arise in three ways [14]:

1. A technological gap exists between what a computer system is actually capable of enforcing and what it is expected to enforce.

2. A socio-technical gap exists between computer system policies and social policies such as computer-related crime laws, privacy laws, and codes of ethics, or between organizational policies and their technological enforcement.

3. A social gap exists between social policies and actual human behavior.

A further vulnerability arises unless the software suite (and to some degree its remote collaborators) are considered as part of the system. Security for the underlying architecture does not guarantee security for an application running in that architecture, which could be vulnerable to, for example, injection faults [1].

Students should come to know that efforts pursuing isolated solutions are bound to be overcome by innovative hackers. The human factor in security cannot be ignored. Through the use of "social engineering" techniques, malicious actors often create disruptions to system confidentiality, integrity, and availability. In particular, this quote from famous hacker, Kevin Mitnik, and his co-author, William Simon, is quite striking: "As developers invent continually better security technologies, making it increasingly more difficult to exploit technical vulnerabilities, attackers will turn more and more to exploiting the human element. Cracking the human firewall is often easy, requires no investment beyond the cost of a phone call, and involves minimal risk" [15].

In an article in *Science*, Ross Anderson and Tyler More offer a compelling analysis on the importance of proper economic incentives in security engineering and education [8]. Simson Garfinkel calls the current phenomenon "perverse market incentives." The market rewards companies that publish special-purpose products for the security problem. This creates conditions where a plethora of products may be installed on a particular computer, some conflicting with each other [16]. David Clark, a Senior Research Scientist at the MIT Computer Science and Artificial Intelligence Laboratory, recounted to one of the authors an amusing story in which a friend discovered that her email was no longer filtered for viruses after she enabled the "POP over SSL" feature in her email client. Her antivirus system had been screening her email through the use of a POP proxy. Once SSL was enabled, the POP proxy couldn't examine the contents of the mail stream because it was cryptographically protected against such man-in-the-middle attacks!

Meanwhile, students should realize that non-security-focused companies that produce software, PCs, and networks often have little economic incentive to remove vulnerabilities from their products. First, time-to-market is a major driver in the software industry, and building in robust security adds both time and cost. Second, most user bases accept the practice of releasing a product with little-to-no security, waiting for hackers to identify the most obvious vulnerabilities, and then waiting for the patches on the company's website. Students must be firmly convinced that looking at technology, organizations, and people in isolation does not provide much useful information about the security of a system, although an assurance of security is impossible without these analyses. It is also necessary to explore information security at the interface between technical architecture, organizational policies, and human behavior.

3. Security Models

Many students who take a course in information security are introduced to the Bell-LaPadula model. It is commonly referred to as “no read up, no write down” and is a formal state transition model that defines access control rules. Users can only create content at or above their own security level and only view content at or below their own security level [17]. The model was the foundation of the DoD Trusted Computer System Evaluation Criteria (TCSEC) commonly referred to as the “Orange Book.”Although the authors explicitly mention the use of system theory in the development of their model, they still view security largely as a mathematical property. Weaknesses, such as neglecting the problem of the clandestine exchange of information, motivated other researchers to develop competing models.

Nonetheless, the Bell-LaPadula model is certainly not inherently inconsistent with a more systems-theoretic approach, especially when combined with “need to know/share” and “right to create, delete, and/or modify” access restrictions. (Such restrictions can also include time and place of access, current user role, and interactions with version control/software configuration management.) The emphasis on control ties it very closely to the underlying foundations of Systems-Theoretic Accident Model and Processes for Security which will be described in greater detail shortly. Bell-LaPadula focuses only on disclosure control, i.e., confidentiality. STAMP-Sec is designed to address issues with confidentiality, availability, integrity, non-repudiation, authentication, authorization, and auditability. It is perfectly reasonable for a STAMP-Sec solution to include elements or the entirety of the Bell-LaPadula model as part of an approach to maintain confidentiality in an environment that is multi-level secure, e.g., the Pentagon.

In addition to the emphasis on control, one can characterize Bell-LaPadula as using another STAMP-Sec concept, systems theory. Both STAMP and Bell-LaPadula cite the work of Ludwig von Bertalanffy [18]. However, STAMP-Sec’s use of systems theory is less mathematical and more design/operations based because it also incorporates the work of Ashby, Forrester, Wiener, Sterman, and Senge [19-23]. Bell-LaPadula rely more on the mathematics of relations and set theory as well as the thought of Hammer, Klir, Zadeh, and Polak [24-26].

Students should be aware that historically, following the Bell-LaPadula model, research interests shifted from military/intelligence computer systems to commercial systems. During this time period, other models focusing on data integrity rather than confidentiality arose such as the Biba model (1977), which is a read up/write down model, i.e., the mathematical dual of Bell-LaPadula [27]. In a fundamental departure from earlier models, David Clark and David Wilson defined a model that focuses on user transactions. Constrained Data Items (CDI) are subject to integrity constraints with integrity verification procedures to determine if the system is in a valid state. The model also permits transformation procedures to move the system from one valid state to another [28].

Like Bell-LaPadula, Clark-Wilson is also not inconsistent with the central idea of control as it is expressed in STAMP. In fact, it shares a particular similarity with regard to achieving control through the imposition of constraints. According to Checkland, “control is always associated with the imposition of constraints, and an account of a control process necessarily requires our taking into account at least two hierarchical levels. At a given level it is often possible to describe the level by writing dynamics equations… But any description of a control process entails an upper level imposing constraints on the lower…” [12].The fundamental concept in STAMP-Sec is a *constraint*, rather than an event. Similarly, a fundamental concept in Clark-Wilson is the CDI. In this respect, the Clark-Wilson model could be understood as a set of constraints, control structure components, and control actions, for the general threat of an attack on data integrity.

4. Security Methodologies, Tools, and Approaches

Security courses introduce students to a variety of methodologies, tools, and approaches. In a computer science class for example, students may learn about utilizing formal methods. The application of formal methods to secure systems involves the use of rigorous mathematical techniques to prove properties about a specification as well as verify that a particular implementation successfully achieves the same properties. Research in this space involves developing model checkers, theorem provers, static analyzers, and software specification languages [29]. Software and hardware developed with robust formal methods are shown to have very few vulnerabilities. One of the most successful application of these methods was the Provably Secure Operating System project, PSOS [30, 31]. A serious drawback associated with formal methods is cost. A formal specification is often longer than the code itself. Additionally, a graduate level education in discrete mathematics is required to use the technique for real systems. On a less mathematically rigorous level, there are also guidelines for writing secure code and for avoiding common pitfalls [1-3].

There are also at least four practical objections to relying entirely on formal methods, even just to deal with software issues.

1. There are serious scalability issues with formal methods, especially theorem proving, if the full power of an object-oriented or higher-order functional language is to be supported. Outside of very high security environments/applications, these more expressive languages may be needed as development using less expressive approaches may not be as efficient, effective, or robust.

2. A security risk based on a software flaw cannot be eliminated from a system until it has been identified, or lawful behavior has been fully described and constrained.

3. Checking an application with formal methods tends to be a lengthy and time-consuming process, which would be impractical to apply after every minor change in the application—and a system would have to be rechecked after any change to any hosted application. But security should be designed into the system early in development, and problems should not be allowed to accumulate between validations, as cost to rework grows with intervening changes or development.

4. Modern software development typically relies on third-party development environments, compilers, operating systems, libraries, tool suites, etc. No formal guarantees of a source code program can be trusted if the supports that translate it into target code, or for that matter the formal methods tools themselves, have not themselves been rigorously verified—a verification that will have to be repeated each time any one of these third-party components has been updated.

Hardware reliability methods with origins in mechanical and nuclear engineering, such as probabilistic risk assessment (PRA), have also been modified for security analysis. They usually combine attack trees (fault trees) and elementary game theory to quantify a security scenario [32]. Unfortunately, these models have highly limiting assumptions thereby making their applicability limited, and outside of its intended domain, questionable. For example, George Apostolakis points out that PRA does not handle software, human factors, culture, or design errors well [33]. Game theory also assumes the rationality of the attacker, e.g., the attacker performs a cost-benefit analysis and only attacks when he perceives it maximizes a utility function such as profitability. Clearly, many malicious actors do not behave in this way (or, in an alternative view, are motivated by very different reward functions). Donn Parker points out some of the problems with quantitative risk methods: "Security risk is not measurable, because the frequencies and impacts of future incidents are mutually dependent variables with unknown mutual dependency under control of unknown and often irrational enemies with unknown skills, knowledge, resources, authority, motives, and objectives–operating from unknown locations at unknown future times with the possible intent of attacking known by untreated vulnerabilities that are known to the attackers but unknown to the defenders" [34]. Hardware methods also cannot fully analyze software with loops, recursion, or non-determinism due to distribution, without significant modification or loss of completeness.

Red teaming is an admirable activity to complement other security analyses as well as reduce the complacency that often sets in after extended periods without incidents. The goal of any red team is to confront the plans, programs, and assumptions of the client organization. Teams may challenge organizations at strategic, operational, or tactical levels depending on the area that needs the most attention. The words of William Schneider, Jr., former Chairman of the Defense Science Board, best capture the state of red teaming, "Red teams can be a powerful tool to understand risks and increase options. However, the record of use of red teams in DoD is mixed at best" [35]. Related activities may fall under the categories of "white-hat" and "gray-hat" hacking, or "certified ethical hacking".

The development of modern cryptographic techniques is one of the most significant contributions to the field of information security. Students with some mathematical background in number theory learn algorithms such as AES, Diffie-Hellman, RSA, and Elliptic Curve Cryptography. These algorithms have provided the foundation for secure computing and communications by enabling confidentiality and data integrity [36]. Technologies that make use of these algorithms, such as digital signatures and certificates, when embedded in larger solutions, have solved important problems related to authentication and non-repudiation. However, cryptography is not a "silver bullet" for the security problem. David Clark astutely remarked to one of the authors in a conversation that "cryptography is 'perfect,' no one cracks codes, they just steal the key." Therefore, the larger system and environment within which a cryptographic algorithm is embedded must be part of a security solution.

Virus scanners, SPAM filters, firewalls, and other products make up the typical toolbox, and part of the computing environment, for the Internet user. Business students studying information systems learn that these tools are vital to survival in cyberspace. However, their capabilities are often overestimated, and more pro-active, systemic solutions have been avoided. While it is unlikely that the threats which necessitate these tools will ever completely vanish, it is necessary to consider the deeper issues that enable virus propagation, junk email proliferation, and hacker attacks.

System Dynamics (SD) is an approach to understanding the non-linear behavior of complex systems developed at the MIT Sloan School of Management. The use of SD modeling to understand information security properties is a very recent development. Some work done by these researchers has focused on insider-threats [37-39], work processes and organizational learning in the transition to Integrated Operations [40-42], and understanding the software vulnerability black market [43, 44]. The results of these projects indicate that SD is a promising modeling technique to understand security in complex systems. In a systems-theoretic approach to security, SD can assist with understanding the relationship between software project risks and security risks, modelling malicious actors, and capturing the behavioral dynamics of security control structures.

5. A New Approach

Systems-Theoretic Accident Models and Process (STAMP) is an approach to understanding accident causation. It was developed by Nancy Leveson at MIT as part of her research into software system safety. Although originally developed for safety, many of the theory's constructs are applicable to security. David Zipkin has shown the applicability of the model on policies for managing malicious software [45] while Joseph R. Laracy has explored its use for bio-defense planning [46] and air transportation [47]. The model with security extensions that will be defined in this paper is referred to as STAMP-Sec.

STAMP acknowledges two types of complexity in engineering systems, behavioral and structural. In contrast to a traditional scientific method that relies on analytic reduction, systems theory states that complex systems must be considered holistically. The theory was well developed by Bertalanffy, Ashby, and Wiener in the mid-twentieth century in response to challenges encountered in biology, communication, and control. During this time, scientists and engineers began to recognize a new type of structural complexity. Organized simplicity is exhibited in traditional, deterministic systems that easily can be decomposed into subsystems and components such as in structural mechanics. The re-synthesis of the subsystems does not yield any unexpected properties because the component interactions are well defined and often linear. Conversely, it is not straightforward or useful to decompose systems that exhibit unorganized complexity. However, statistical techniques are applicable because of the regularity and randomness that characterize the network structure. The Law of Large Numbers becomes applicable and average values can be computed such as in statistical mechanics, e.g. ideal gases in chemistry. The "new" structural complexity theory, organized complexity, describes systems with a sufficiently complex structure to make it impractical or impossible for them to be modeled with analytic reduction, and not random enough to be modeled using statistics [48].

When considering behavior, systems scientists acknowledge that the failure of linear, cause-and-effect reasoning is the result of dynamic complexity [22]. Most people associated the word "complexity" with combinatorial complexity, i.e., detail complexity. The needle-in-a-haystack problem is an example of detail complexity. Conversely, John Sterman points out that dynamic complexity arises in systems that are [22]:

1. Tightly coupled
2. Governed by feedback
3. Non-linear
4. History (path) dependent
5. Self-organizing
6. Adaptive
7. Counterintuitive
8. Policy resistant
9. Characterized by trade-offs

It is important for students to appreciate that even systems with low detail complexity can exhibit dynamic complexity. STAMP explicitly acknowledges the presence of dynamic complexity in engineering systems. In fact, one of the motivations for its development was the failure of traditional, event-driven models to handle dynamic complexity [49]. A STAMP analysis manages the dynamic complexity of engineering systems through the incorporation of non-linear feedback in its models. Both static control structures and System Dynamics models include feedback. In a real system, many processes and activities are occurring in parallel. These processes interact with each other in ways that are difficult to predict. It is therefore a gross oversimplification to draw a linear chain of events that does not acknowledge concurrency and feedback.

Systems characterized by organized and dynamic complexity exhibit strong, non-linear interactions and coupling between subsystems and components. Therefore, a top-down approach needs to be applied to such systems. According to Checkland, two underlying concepts provide insight into these complex systems: emergence and hierarchy as well as communication and control [12]. Abstractions for complex systems often involve layers. In the case where the abstraction is hierarchical, the level of organization increases as one moves toward higher layers. Additionally, the step from level n to $n + 1$ yields new properties that are not discernable at level n. This phenomenon is referred to as emergence, or emergent properties. Control actions in such systems are the result of constraints imposed by level $n + 1$ onto level n (or lower). In order for the right control action to be applied, level n must communicate with level $n + 1$. Dynamic equilibrium in such systems is achieved through the presence of feedback loops. According to Leveson, "systems are not treated as a static design, but as a dynamic process that is continually adapting to achieve its ends and to react to changes in itself and its environment" [48]. Therefore, security engineering must affect both the design and operation of complex systems.

In order to better understand security incidents, the STAMP-Sec model specifies the following concepts:

1. Constraints (security requirements)
2. Controls
3. Context
    - Roles and Responsibilities
    - Environmental and Behavior Shaping Factors
4. Flaws in the Controlled Process
5. Dysfunctional Interactions, Failures, Flawed Decisions, and Erroneous Control Actions
6. Reasons for Flawed Control Actions and Dysfunctional Interactions
    - Control Algorithm Flaws
    - Incorrect Process, Interface, or Mental Models
    - Inadequate Coordination among Multiple Controllers
    - Reference Channel Flaws
    - Feedback Flaws

Security constraints are requirements that are written to prevent the instantiation of particular threats. Control actions to implement the constraints are then defined for particular components in the socio-technical system. It is also necessary to understand the context that supports incidents such as the responsibilities of people and technology and the environmental influences. One must then look at the flaws in a controlled process that would create inadequate control. Such flaws are caused by dysfunctional interactions, failures, flawed decisions, and erroneous control actions. Finally, the reasons for these flaws and dysfunction must be identified.

In general, a controller can provide four types of inadequate control [48]:

1. A required control action is not provided.
2. An incorrect or unsafe control action is provided.
3. A potentially correct or adequate control action is provided too late (at the wrong time).
4. Correct control action is stopped too soon.

There are many ways that these inadequate controls can lead to a security system being compromised. They fall into one of three categories: inadequate enforcement of constraints, inadequate execution of control actions, or inappropriate or missing feedback [48]. The introduction of a malicious agent does not violate the assumption of the taxonomy originally developed for safety. In a safety scenario, poor engineering or management may offer inadequate enforcement of constraints, execution of control actions, or feedback such that a hazard that is "exploited" inadvertently in system operations. In a security scenario, poor engineering or management may offer inadequate enforcement of constraints, execution of control actions, or feedback such that a vulnerability is created which may be intentionally exploited in system operation. Whether one is concerned with safety or security, the problem is inadequate control. STAMP-Sec extends the safety list to capture security issues:

1. Inadequate Enforcement of Constraints (Control Actions)

    1.1. Unidentified threats

    1.2. Inappropriate, ineffective, or missing control actions for identified threats

        1.2.1. Design of control process does not enforce constraints

            1.2.1.1. Flaws in creation process

            1.2.1.2. Process changes without appropriate change in control (asynchronous evolution)

            1.2.1.3. Incorrect modification or adaptation

        1.2.2. Process models inconsistent, incomplete, or incorrect

            1.2.2.1. Flaws in creation process

            1.2.2.2. Flaws in updating process (asynchronous evolution)

            1.2.2.3. Time lags and measurement inaccuracies not accounted for

            1.2.2.4. Inadequate coordination among controllers and decision makers (boundary and overlap areas)

2. Inadequate Execution of Control Action

    2.1. Communication flaw

    2.2. Inadequate actuator operation

    2.3. Time lag

3. Inadequate or Missing Feedback

    3.1. Not provided in system/organizational design

    3.2. Communication flaw

    3.3. Time lag

    3.4. Inadequate detection mechanisms

Students should take note that many of these inadequacies are not associated with simply an event-based risk. Rather, flaws in communication and control as well as time lags and flaws in the design process contribute to threats. Therefore, security engineering with the STAMP-Sec framework takes the following approach:

1. Identify the system-level threats

2. Write security requirements, or constraints, that would prevent the instantiation of the threats

3. Create a socio-technical control structure that enforces the constraints

    a. Components within the control structure are assigned responsibility to execute particular constraints.

    b. Control actions for each component are defined to implement the constraints.

4. Identify inadequate control actions that could lead to an insecure state.

5. Determine ways that constraints could be violated and attempt to eliminate them. In particular, use System Dynamics to consider how and why the security control structure might change over time, potentially leading to ineffective controls.

Initially, we can define at least seven system-threats, attacks on:

1. Confidentiality
2. Availability
3. Integrity
4. Non-repudiation
5. Authentication
6. Authorization
7. Auditability

Note that threats can be unintentional, due to a catastrophe such as a fire, or due to the interaction of subsystems, such as destruction of storage media by a sprinkler system during a fire emergency, showing the need for security to interact with traditional risk analysis and risk management activities.

The final step of a STAMP based analysis is System Dynamics (SD) modeling. System Dynamics is used to understand how the static control structure (designed in the earlier stages), and the attackers themselves could evolve. In particular, one is interested in evolution to insecure states such that security constraints would no longer be enforced by components in the socio-technical system. Unlike system safety models in which hazards are often "generated" endogenously within the socio-technical system, security threats may develop exogenously. While the "insider-threat" risk must be addressed, malicious actors outside of the system under discussion must be also modeled [47].

System Dynamics was created at MIT in the 1950s by Jay Forrester. Its theoretical basis comes from control theory and non-linear dynamics. Complex systems, whether they are technical, organizational, or some combination, often exhibit highly non-linear behavior where the relationship between cause and effect is not intuitively obvious. Martinez-Moyano et al. describe system dynamics as "a computer-aided approach to policy analysis and design that applies to dynamic problems arising in complex social, managerial, economic, or ecological systems. Dynamic systems are characterized by interdependence, mutual interaction, information feedback, and circular causality" [37].

System Dynamics models are constructed by a combination of positive (reinforcing) and negative (balancing) feedback loops in addition to state and rate variables [22]. At its most fundamental level, a SD model is a system of coupled, first order, non-linear, ordinary differential or difference equations. It is graphically presented in an easy to understand format and accessible to people with non-technical backgrounds. These models can be simulated to obtain numerical results. It is helpful to consider SD models at various levels of abstraction. Nicolas Dulac and Brandon Owens have developed a five-layer hierarchy of abstractions [50]. At the highest level of a model are the major loops. These are captured in what is usually referred to as a causal loop diagram. Large models may be broken into components, or modules, to achieve intellectual manageability goals. Within these modules, state, rate, and auxiliary variables are defined. Like any modeling activity, these variables carry a number of assumptions that are made explicit with data and equations.

Students should learn that a good way to capture specification and design information is through the use of "Intent Specifications" [51, 52]. Based on principles from cognitive engineering, their goal is to "provide specifications that support human problem solving and the tasks that humans must perform in software development and evolution" [51]. An Intent Specification can be understood along three dimensions. The first dimension is the intent view. These views begin at Level 0, and express the goals of Program Management and are further developed through Level 1: System Purpose (Customer View), Level 2: System Design Principles (Systems Engineering View), Level 3: System Architecture (Interface between Systems and Component Engineers), Level 4: Design Representation (Component Designer View), Level 5: Physical Representation (Component Implementer View), Level 6: System Operations (Operations View). The second dimension is concerned with the relationship of parts to the whole. These include environmental factors, operator procedures, endogenous system artifacts, and verification &

validation activities. The third dimension of an intent specification is refinement over time.

While these principles provide a solid basis for security education, it would be foolish to think that any approach will capture every problem, be properly comprehended or properly implemented, or have every identified problem correctly understood or resolved immediately, nor that any large application will remain unchanged in design, implementation, or use, nor, importantly, that the suite of applications running on a given system will remain unchanged. In addition, it needs to be understood that advances in the theory of cryptography, or changes in computing technology, environments, practices, or protocols, or at the extreme, a transition to practical quantum computing, may introduce security challenges and/or countervailing tools that could not have been anticipated. For these reasons, students must also learn the need to apply good software development and risk management principles, to employ up-to-date configuration management, validation, testing, and monitoring methods and tools, and to update security and risk analyses incrementally as for regression testing, and also to understand the need for comparable approaches and analyses for organizational and human factors. Students should be able to identify major themes and approaches in each of these dimensions. Each new application should be developed (or modified) to be as secure as possible in isolation, and validated in the system model before it is deployed, and then on the deployment platform before being released.

6. Course Themes and Structure

As discussed above, the STAMP-Sec approach views computer security as a multidimensional problem, extending well beyond the classical boundaries of the computing environment. In this light, a course (or program) inspired by STAMP-Sec should emphasize the following themes.

- System security is a not just a software issue or a management issue, it is a systems issue.

- An individual software system lives in a socio-technical environment. This environment comprises development and deployment structures, application artifacts and data, the hardware infrastructure, people within and outside the enterprise, physical structures, and societal stakeholders with interests in its products and services, in its effects on society, and in privacy, confidentiality, standards, and/or regulation.

- An enterprise likewise lives in such a socio-economic environment, further complicated by corporate policies and practices, institutional relationships and rivalries, business considerations including management, marketing, and planning, and legal and regulatory constraints.

- Technical protocols (for cryptography, access control, threat detection, etc.) must consider not only mathematical foundations but also software implementations and deployment platforms.

- The social aspect needs to address not just malicious attacks and social engineering, but must also be aware of consequences of carelessness and indolence, e.g., in password management. It also interacts with physical and management structures as well as the computer system itself.

- Application security should leverage current security-aware practices for development, coding, tools, analyses, deployment, storage and communication, access control, and change management.

- Security must be integrated with requirements and risk analysis, with change and configuration management, and with both technical and management processes, procedures, and policies.

- All facets of security must be continually updated to address new threats and defenses, to incorporate theoretical and practical developments, and to respond to changes in the computing, physical, social, and legal/regulatory environment.

- Managers, both at the corporate and technical levels, need to be aware of security concerns, and to work with security professionals, and must create and enforce reasonable security policies and procedures.

- A security professional needs to be aware of all of these facets and appropriate resources, and should aim to develop a sense as to when any aspect needs to be reexamined, and, further, serve as an enterprise resource and assessor for security relevant artifacts and activities.

- All enterprise personnel must receive appropriate security orientation and training; compliance and awareness must be an ongoing activity for all enterprise personnel.

These points, and the STAMP-Sec approach as a whole, subsume within a wider context the computing-centric view of security concerns as Hardware, Software, Data, People & Organizations, and Procedures & Processes, and inform the course sequence structure. Depending on the program, on the intended audience, and on space in the curriculum, this approach should be complemented by discipline-specific courses, possibly from multiple disciplinary perspectives.

STAMP-Sec might serve best as an organizational principle for a security program or concentration, and as a one- or two-semester upper-level integrative course in such a program. It could also inspire an (again, one- or two-semester upper-level) elective in a major in a differently-structured security program. Finally, it could be utilized for a course or sequence in one of its component disciplines, showing how the questions, concepts, tools, and approaches of that discipline are complemented by those of other disciplines in realizing the complete security landscape.

The rest of this section considers a two-course integrative sequence for a security concentration structured on STAMP-Sec, with students coming from multiple disciplines. Students work in (fixed or changing) teams. Areas not covered by individual courses may need a more substantial overview, while the emphasis for more familiar areas can be their roles and interactions. As necessary or desired, this approach can be interwoven with a discipline-specific, practice-oriented thread, such as those discussed in Section 1. Such a sequence could proceed as follows.

- The computer security universe: dimensions of threats and examples. A high-level overview of defenses and responses.
- Socio-technical environments. Computer security viewed as part of such an environment—the software system perspective and the enterprise perspective. Systems thinking, cybernetic systems, and system dynamics.
- Security models. The STAMP-Sec approach. Scenarios and case studies.
- The security design process. Requirements, design, review. Sandboxing and testing. Interaction with management policy and procedures, with privacy and related concerns, and with other facets. An extended example.
- Facets of security: cryptography, protocols, and applications; secure computing environments, development, analyses, and tools; secure deployment, detection, and response; social engineering; management aspects and access control; physical aspects; privacy and related factors. Constraints on security structures: need to provide service, tension between information and privacy, user behavior (as with password selection), cost/time/effort required. Tradeoffs between security and other risks. Selected examples depending on student background and level.
- The security review process in detail. Examples and case studies.
- Multiple guarantees, views, and interfaces: users, operators, clients of users, management, development team, requirements, risk management, and security team. Information for collaborators, services and clients—enough but not too much.
- The security professional: role, responsibilities, products, and interactions with others. Specific responsibilities: training, monitoring, updating, advising and reviewing, and certifying. Possible guest lecture(s).
- The dynamics of a security-focused socio-technical system. The tension between security assurance and the need for agility and updating. Security, maintenance, and evolution: configurations, sandboxing, and transitions revisited.
- Overview of the security package. The role of an overview structure such as STAMP-Sec and its interactions. Responsibilities for security.

If this sequence is to be used as a capstone, a seminar format may be desirable, with students developing an annotated bibliography and presenting survey papers, white papers, case studies, or book chapters across the disciplines. Students from a given discipline would be encouraged to combine their discipline with others in the bibliography and presentations. The assessments suggested above would then be reduced. Teams would be presented with an extended case study/scenario and asked to prepare a security assessment with recommendations. The instructor and perhaps an outside collaborator would be available for consultation.

7. Conclusion

Current approaches to a course or two-course sequence in information security tend to be discipline-centered, in mathematics, computer science, information technology or information management, data science, or other areas. Such courses often focus on one or two aspects: theory, the computing and communications environment, organizational behavior and procedures, or human factors, or at best pay attention to multiple areas in isolation; they may also stress either inadvertent flaws or hostile action. Moreover, courses or even programs tend to look at (often a specific subset of) individual problem areas, and techniques, approaches, and remedies for those problems, rather than a global view. This presents a substantial deficiency in security education.

However, there are signs that information security education is moving in a positive direction. A recent call for "systems thinking" education is a move in the right direction [53]. In this paper, we propose an interdisciplinary approach that, while acknowledging the need to consider each of the discipline-specific areas, takes a global systems theory perspective on security and the sub-problems of confidentiality, availability, integrity, non-repudiation, authentication, authorization, and auditability. Students will see how the different aspects integrate within a unified overview and governing process, and how the different aspects of the system—hardware, software, communications, organizations, and people—have to be analyzed both separately and together to provide reasonable security guarantees, and how that analysis must be updated given changes in any of those components or their relationships.

No single approach will offer a *completely* comprehensive solution to securing the turbulent and turbid Internet. However, we believe that STAMP-Sec's methodology for incorporating socio-technical factors offers a broad-spectrum solution that is beneficial for students. By addressing the essential themes of constraints (i.e., security requirements), controls, context (e.g. roles and responsibilities), flaws in the controlled process, dysfunctional interactions, and the reasons for those dysfunctional interactions, students will gain a integrative view of security issues. An implication of adopting STAMP-Sec for security education includes forming students to consider the highly dynamic, non-linear behavior of contemporary information systems and what feedback and control actions are necessary to maintain key security properties.

Acknowledgments

The authors would like to thank Cyril Ku of William Paterson University, Scott Thompson of Novartis International AG, and the anonymous reviewers for their comments on earlier drafts of this article.

References

[1] Howard, M., LeBlanc, D., Viega, J. (2010). 24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them. New York: McGraw-Hill.

[2] Forouzan, B. (2008).Cryptography and Network Security. New York: McGraw Hill.

[3] Viega, J., McGraw, G. (2002). Building Secure Software: How to Avoid Security Problems the Right Way New York: Addison-Wesley.

[4] Chess, B., West, J. (2007). Secure Programming with Static Analysis. Boston: Pearson Education.

[5] Podjarny, G. (2018). The Three Faces of DevSecOps (InfoQ). Accessed 6/11/18 at https://www.infoq.com/articles/three-faces-DevSecOps

[6] Cloud Security Alliance (2012). Top Ten Big Data Security and Privacy Issues. Accessed 02/19/18 at https://www.isaca.org/Groups/Professional-English/big-data/GroupDocuments/Big_Data_Top_Ten_v1.pdf

[7] Dua, S., Du, X. (2011). Data Mining and Machine Learning in Cybersecurity. Boca Raton: CRC Press.

[8] Anderson, R., Moore, T. (2006) The Economics of Information Security, *Science* 314 (5799) 610-613.

[9] Neubauer, T., Klemen, M., Biffl, S. (2006). Secure Business Process Management: A Roadmap, *In:* Proceedings of the First International Conference on Availability, Reliability and Security (ARES'06) 457-464. Washington: IEEE Press.

[10] Security in a Box (2010), Protect Your Information from Physical Threats. Accessed 6/11/18 at https://securityinabox.org/en/guide/physical/

[11] Neumann, P. G. (2006). Holistic Systems, *ACM SIGSOFT Software Engineering Notes*, 31 (6) 4-5.

[12] Checkland, P. (1999). Systems Thinking, Systems Practice. New York: John Wiley & Sons. 314.

[13] Schneier, B. (2015). Secrets and Lies: Digital Security in a Networked World. New York: John Wiley & Sons. 130.

[14] Neumann, P. G. (1994). Computer Related Risks. Reading: Addison-Wesley 3.1.

[15] Mitnick, K. D., Simon, W. L (2003). The Art of Deception: Controlling the Human Element of Security. New York: John Wiley and Sons 4.

[16] Garfinkel, S. L. (2005). Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable, Ph.D. Thesis, Computer Science Department. Cambridge: MIT.

[17] Bell, D., LaPadula, L. (1973). Secure Computer Systems: Mathematical Foundations, MITRE Corporation Technical Report.

[18] vonBertalanffy, L.(1969). General System Theory. New York: George Braziller, Inc.

[19] Ashby, W. R. (1956). An Introduction to Cybernetics. London: Chapman and Hall.

[20] Forrester, J. (1961). Industrial Dynamics. Cambridge: Productivity Press.

[21] Wiener, N. (1965). Cybernetics*:* or the Control and Communication in the Animal and the Machine*.* Cambridge: MIT Press.

[22] Sterman, J.(2000). Business Dynamics: Systems Thinking and Modeling for a Complex World. Boston: Irwin McGraw-Hill.

[23] Senge, P. (2006). The Fifth Discipline. New York: DoubleDay.

[24] Hammer, P. C. ed. (1969). Advances in Mathematical Systems Theory. University Park, Pennsylvania: Pennsylvania State University Press.

[25] Klir, G. J. (1969). An Approach to General Systems Theory. New York: van Nostrand Reinhold Company.

[26] Zadeh, L. A., Polak, E., (1969). System Theory. New York: McGraw-Hill Book Company.

[27] Bishop, M. (2005). *Introduction to Computer Security.* Boston: Pearson Education.

[28] Clark, D., Wilson, D. (1987) A Comparison of Commercial and Military Security Policies, *In:* IEEE Symposium on Security and Privacy. Oakland, CA.

[29] Wing, J.M. (1998). A Symbiotic Relationship Between Formal Methods and Security, *In:* The Workshops on Computer Security, Fault Tolerance, and Software Assurance: From Needs to Solutions. Pittsburg, PA.

[30] Feiertag, R. J., Levitt, R. J., Robinson, L. (1977). Proving Multilevel Security of a System Design, *In:* Proceedings of the Sixth ACM Symposium on Operating System Principles. West Lafayette, IN.

[31] Neumann, P. G., Feiertag, R. J. (2003). PSOS Revisited, *In:*Proceedings of the 19th Computer Security Applications Conference. Las Vegas.

[32] Buldas, A., Laud, P., Priisalu, J., Saarepera, M., Willemson, J. (2006). Rational Choice of Security Measures via Multi-Parameter Attack Trees, *In:* Critical Information Infrastructures Security First International Workshop (Samos Island, Greece).

[33] Apostolakis, G. E. (2004). How Useful is Quantitative Risk Assessment?, *Risk Analysis,* 24 (3) 515-520.

[34] Parker, D. B. (2007). Risks of Risk-Based Security, *Communications of the ACM,* 50 (3) 120.

[35] Schneider, W. (2003) The Role and Status of DoD Red Teaming Activities,*The Defense Science Board*, September.

[36] Menezes, A. J., Van Oorshot, P. C., Vanstone, S. A. (2001). *Handbook of Applied Cryptography.* New York: CRC Press.

[37] Martinez-Moyano, I. J., Rich, E., Conrad, S., Anderson, D. F., Stewart, T. R. (2008). A Behavioral Theory of Insider-Threat Risks: A System Dynamics Approach, *ACM Transactions on Modeling and Computer Simulation* 18 (2) art. 7.

[38] Martinez-Moyano, I. J., Rich, E., Conrad, S., Anderson, D. F. (2006). Modeling the Emergence of Insider Threat Vulnerabilities, *In:* Proceedings of the 2006 Winter Simulation Conference. 562-568.

[39] Rich, E., et. al. (2005). Simulating Insider Cyber-Threat Risks: A Model-Based Case and a Case-Based Model, *In:* Proceedings of the 23rd International Conference of the System Dynamics Society. Boston, MA. 1-28.

[40] Rich, E., Gonzalez, J. J. (2006). Maintaining Security and Safety in High-threat E-operations Transitions, *In:* Proceedings of the 39th Hawaii International Conference on Systems Science*.* Kauai, Hawaii, 1-10.

[41] Rich, E., et al. (2007). Emergent Vulnerability in Integrated Operations: A Proactive Simulation Study of Risk and Organiza tional Learning, *In:* Proceedings of the 40th International Conference on System Sciences. Waikoloa, Hawaii.

[42] Gonzalez, J. J., et al. (2005). Helping Prevent Information Security Risks in the Transition to Integrated Operations, *Telektronikk* 101 29-37.

[43] Radianti, J., Gonzalez, J. J. (2006). Toward a Dynamic Modeling of the Vulnerability Black Market, *In:* The Workshop on the Economics of Securing the Information Infrastructure. Arlington, VA.

[44] Radianti, J., Gonzalez, J. J. (2007). Understanding Hidden Information Security Threats: The Vulnerability Black Market, *In:* Proceedings of the 40th International Conference on System Sciences. Waikoloa, Hawaii.

[45] Zipkin, D. (2005). Using STAMP to Understand Recent Increases in Malicious Software Activity, S.M. Thesis, Technology and Policy Program, MIT. Cambridge, MA.

[46] Laracy, J. R. (2006). A Systems Theoretic Accident Model Applied to Biodefense, *Defense and Security Analysis*, 22 (3) 301-310.

[47] Laracy, J. R. (2017). A System-Theoretic Security Model for Large Scale, Complex Systems Applied to the Next Generation Air Transportation System, *International Journal of Communications, Network, and System Sciences* 10 (5) 75-105.

[48] Leveson, N. (2002). System Safety Engineering: Back to the Future. Cambridge, MA.

[49] Leveson, N. (2004). A New Accident Model for Engineering Safer Systems, *Safety Science*, 42 (4) 237-270.

[50] Dulac, N., Owens, B. D., Leveson, N. (2007). Modelling Risk Management in the Development of Space Exploration Systems, *In:* Proceedings of the 2nd Annual International Association for the Advancement of Space Safety (IAASS) Conference. Chicago, IL, 14-16.

[51] Leveson, N. (2000). Intent Specifications: An Approach to Building Human-Centered Specifications, *IEEE Transactions on Software Engineering* 26 (1) 5-35.

[52] Leveson, N., Weiss, K. (2004). Making Embedded Software Reuse Practical and Safe, *In:* Twelth International Symposium on the Foundations of Software Engineering. Newport Beach, CA, 171-178.

[53] Raj, R., Parrish, A. (2018). Toward Standards in Undergraduate Cybersecurity Education in 2018, *IEEEComputer*, 51 (2) 73.