Hierarchy-Driven Approach for Attack Patterns in Software Security Education

Joshua J. Pauli  
*College of Business and Information Systems*  
*Dakota State University*  
*Madison, SD, 57042, USA*  
josh.pauli@dsu.edu

Patrick H. Engebretson  
*College of Business and Information Systems*  
*Dakota State University*  
*Madison, SD, 57042, USA*  
pat.engebretson@dsu.edu

Abstract

We propose a hierarchy-driven approach to facilitate student learning and foster a deeper understanding of the importance of attack patterns in computer, network, and software security. This is a fundamental point in computer and software security education because the “patch and pray” mentality of software security is insufficient. This is especially important to make known to higher education students before entering the workforce. The importance and significance of our approach is justified by accentuating the deficiencies in previous ad-hoc approaches to teaching attack patterns. Because of the vast amount of information in attack pattern repositories, it is unrealistic to expect students to fully comprehend attack pattern fundamentals and its place in computer, network, and software security. The properties of our approach that fill these deficiencies are covered as well as the proven methodologies that our work is based upon. The concept of hierarchy and refinement are well understood and accepted in the software engineering community. We leverage both of these theories to construct a hierarchy-based approach to learning and applying attack patterns for software security. A summary of an active learning session is used to illustrate how our model is used in the classroom before taking the approach outside of the classroom as part of ongoing out-of-class assignments. The proposed approach is occurring at Dakota State University as part of an upper-level Software Security course during the fall 2007 semester.

Keywords: Attack Trees, Attack Patterns, Refinement, Hierarchy.

1. Introduction

Within the past five years, the fields of network, computer, and software security has begun to shift its focus away from perimeter defensive models, such as border routers, firewalls, and intrusion detection systems, to more proactive defensive models [10]. Until recently many companies have simply relied on a patch-when-exploited methodology to writing secure software [7]. This patch and penetrate methodology does nothing to address the underlying security issues. In order to better instantiate a proactive defense model, one must begin with software security and make sure that these priorities are carried throughout every phase of the software development lifecycle.

The goal of teaching attack patterns to students is to provide them with a semi-formalized representation of the attacker’s perspective. Equally as important is providing each student with a security-focused, expert level understanding of software development and the various ways in which software is currently exploited. Good security and the ability to combat malicious code is the byproduct of understanding said code’s mechanics as well as its motivations [3]. By applying this approach, students are able to see how the elements of each attack pattern are related to each other and how each element is related to elements of other attack patterns.

However, there are problems when trying to teach attack patterns to students. Attack Patterns are relatively new, having been introduced within the past decade [13]. Therefore, the breadth and depth of quality related material is limited [13]. It is the goal of this paper to specify a hierarchy-driven approach to attack tree education that emphasizes the importance of attacks at different levels of abstraction, and how they relate to the phases of the system development lifecycle.

Section 2 covers related work that our approach is derived from. Section 3 introduces the approach and how it is leveraged during a class meeting. Section 4 introduces future work to further strengthen the approach and the implementation as part of the course. We conclude in section 5.
2. Related Work

Fostering a deep and solicitous understanding of attack patterns can lead to the permeation of security throughout the software development life cycle, as well as heighten awareness of known exploits, vulnerabilities and weaknesses [9]. Integrating security throughout the software development lifecycle has long been understood as being very important [10], but practical teaching methods have been difficult to specify and implement. Integrating attack pattern knowledge into the SDLC can result in more secure software by creating less exposure to identified bugs and known flaws [10]. Attack patterns can be used to create a security checklist for developers, which in turn can lead to more secure software [4].

The origins of attack patterns can be traced back to concepts outlined by Gamma, et. al. when the foundation for today’s attack patterns where established as the concept of a general, repeatable solution to identified system development problems [8]. More recently the concept of presenting from an attacker’s perspective was done on an individual, or attack by attack basis, with no agreed upon formula, structure, or common language for consistently presenting such a viewpoint [10].

The lack of a common or united vocabulary makes it difficult to gather, analyze, and share pertinent information in meaningful ways which could be used to advance the discipline of software security. Moore began to formalize a concept of combining various types of malicious attacks (i.e. the attacker’s perspective) with the pattern framework [8]. Hoglund and McGraw built upon the foundation of Moore’s paper by more formally defining attack patterns and identifying 48 distinct attack patterns [10].

The final piece of the attack pattern puzzle was realized when the National Cyber Security Division of the Department of Homeland Security in conjunction with Cigital and MITRE Corporation agreed to sponsor CAPEC. The final result of this collective effort was published in March of 2007 and included a formalized attack driven perspective of software security with 101 different attack patterns outlined [6].

The Common Attack Pattern Enumeration and Classification (CAPEC) list provides an official schema and formal representation for defining attack patterns [6]. Release 1 of CAPEC included roughly 100 attack patterns, which tends to be too many attack patterns to effectively use in the classroom setting. CAPEC formally organizes and presents each attack pattern by gathering and displaying both primary and supporting data elements [5].

Primary elements include:

- Attack Pattern ID
- Attack Pattern Name
- Description
- Related Weaknesses
- Related Vulnerabilities
- Methods of Attack
- Examples-Instances
- References
- Solutions and Mitigations
- Typical Severity
- Typical Likelihood of Exploit
- Attack Prerequisites
- Attacker Skill or Knowledge Requirements
- Resources Required
- Attack Motivation-Consequences
- Context Description

Supporting elements include:

- Injection Vector
- Payload
- Activation Zone
- Payload Activation Impact
- Probing Techniques
- Indicators/Warnings of Attack
- Obfuscation Techniques
- Related Attack Patterns
- Relevant Security Requirements
- Relevant Design Patterns
- Relevant Security Principles
- Related Guidelines

Numerous approaches in the requirements phase have addressed software security including scenario-driven requirements analysis [2], misuse case-driven elicitation of non-functional requirements [1][15] goal-driven approaches to user requirements [11], abuse-frame oriented security requirements analysis [12], anti-goals [17], abuse frames, and reuse-based approaches to determining security requirements [16]. The anti-goal method in the goal-oriented requirements analysis can describe vulnerabilities or generate attacks that violate security goals [17]. The “actor, intention, goal” approach in the i* framework [19] models security and trust relationships as “software goals”: goals depending on another actor to satisfy them.
3. Hierarchy-Driven Model

The CAPEC list creates a forum for researchers, developers, security experts, and students to collaborate and share information through a common dialect. However, because of the vast quantities of information within CAPEC, an instructor has a daunting task to introduce the concept of attack patterns and attempt to make use of the CAPEC resource. With over 100 attack patterns and the corresponding Primary and Supporting elements for each, students can be easily overwhelmed by the size, scope and nature of learning such a system.

Given the significance of the subject matter and importance of such concepts, we developed our approach for teaching attack patterns. Our model will rely on utilizing a hierarchy to present attack pattern information logically. One of the major categories for learning strategies is the creation or use of a hierarchy [18]. Furthermore, the process of learning can be facilitated through the implementation of a teaching strategy and good teaching can be defined as one which encourages students to think and remember concepts for themselves [18]. Utilization of a hierarchy will help to bring order as well as facilitate a deeper understanding of relevant attack pattern elements. In order to help solidify the new model, students will first be introduced to the concept of a hierarchy model outside the realm of attack patterns. Upon completion of the abstracted model, students will be asked to apply the hierarchical model to the CAPEC Release 1 attack pattern list.

Specifically, we will focus on a slim element set to incorporate into our hierarchy model [10]. We utilize a top-down approach with the highest level being the most general element. Subsequent hierarchy levels will become more specific in nature and scope. Students are required to follow a minimum 1:1 ratio for each level of the hierarchy as refinement continues from level to the next. While higher ratios of refinement may be possible at each level, the 1:1 ratio between each level is sufficient to introduce attack patterns in an educational setting.

Our hierarchy model is introduced in Figure 1 where more details are realized about the attack pattern as refinement continues to subsequent levels of abstract. Vulnerabilities are at the highest level of abstraction to effectively group the attack patterns in understandable contexts for students.

The purpose is to give students a solid introduction to attack patterns, as well as introduce the CAPEC Release 1 list of attack patterns. As the hierarchy is populated, more details are known about each attack pattern. This information is more useful in a hierarchical format than a textual description with no clear connection between the elements because it is now known what elements influence other elements in the same attack pattern. Students can clearly see these connections between each level of refinement for each attack pattern; this knowledge can then be leveraged when analyzing and designing secure software.

![Figure 1. Hierarchy Framework](image)

In order to foster a deeper understanding for the students, a significant portion of time was spent specifying and documenting consistent definitions for our approach before populating the hierarchy. We use existing and accepted definitions as summarized below [10].

Level 1: Vulnerability is defined as a large or general classification used to group a collection of related errors that an attacker can exploit. Every attack pattern outlined by CAPEC is a child of one of the following vulnerabilities: Abuse of Functionality, Spoofing, Probabilistic Techniques, Exploitation of Authentication, Resource Depletion, Exploitation of Privilege/Trust, Injection, Data Structure Attacks, Data Leakage Attacks, Resource Manipulation, Protocol Manipulation, and Time State Attacks.

Level 2: Attack Pattern is a high level blueprint that describes various types of software attacks.

Level 3: Exploit describes a specific instance of an attack pattern. Level 3.1 Bug / Flaw is used to explain difference between a logical or design issue (flaw) and an implementation or coding issue (bug). CAPEC does not specify specific attack patterns as either a bug or flaw. Level 3.1 has been added to force the student to think deeply about the exploit being explored.

Level 4: Activation Zone is the area in a software package which is capable of activating or executing a payload or exploit.
Level 5: Injection Vector refers to the actual format of the input used in an attack. Level 5.1: Payload references any input given to the software in order to carry out an exploit. Note, the hierarchy allows for the fact that not every Attack Pattern makes use of a payload.

Level 6: Reward is the output event or desired outcome of a successful exploit.

After deciding on a set of definitions to be used throughout the course by all groups, students propose and decide on a consistent mapping of each level of abstraction. Each level must be mapped to a specific category in the CAPEC dictionary as introduced in figure 2.

In order to facilitate this step of our approach, students are broken into groups of two during the class period. Each group is assigned specific attack patterns (Level 2) from the CAPEC dictionary. Students are then required to populate their models by searching through the CAPEC dictionary and populating the elements missing from their hierarchy. Upon completion of several attack patterns, the model is integrated by having groups collaborate on the implementation of multiple attack patterns. Figure 3 introduces one “Server Side Include (SSI) Injection” attack pattern and Figure 4 introduces one “HTTP Request Smuggling” attack pattern after being populated and verified by the student groups. Upon completion of this exercise, the class had created approximately 80 attack pattern hierarchies covering 10 vulnerabilities.
4. Future Work

Because this is our first work dedicated to building this approach, there remains much work to be done. During future usages of this approach in the classroom setting, the documentation from previous classes can be used during future classes for revision and improvement of the proposed process. These historical artifacts can be reviewed and critiqued as an educational tool to spur the discussions on attack patterns, refinement, and hierarchies in secure software engineering coursework.

The goal of future revisions to our approach would be to increase the effectiveness of teaching attack patterns as an integral part of software security by incorporating lessons learned and student feedback. This could include the addition or subtraction of various levels from the proposed hierarchy. While the first semester will build the initial model, subsequent semesters of this course will refine and calibrate the model, therefore building an even more effective teaching approach.

One area of special importance is the ratio of elements as refinement continues and more details are known about each attack pattern. By using this in several different courses, a consensus may be reached on the appropriate number of elements from one level of abstraction to the next.

As the complexity of the approach continues to increase, the natural progression is to develop a software package to support the model. Such a model would eliminate human errors and greatly increase the processing time in populating the hierarchy. An appropriate software tool would have to be driven by a library of attack trees and refined elements as agreed upon by the course members and instructor(s).

We recognize that not every attack pattern as defined in CAPEC (Release 1) fits neatly into the proposed hierarchy. The most common missing CAPEC elements include the following: Activation Zone, injection vector, and Payload. These correspond to hierarchy levels 4, 5, and 5.1.

In these cases, students are asked to closely examine the Attack Pattern and fill in as much information as possible. This is still a valuable academic experience to gain knowledge related to attack patterns.

The definitions and differences between a Software Bug and a Software Flaw initially appeared to be adequate as students could clearly and correctly repeat the given definition of each category. However, because the categorization of Bug versus Flaw is not explicitly detailed by CAPEC, many students found it difficult to abstract this concept. As a result, students often debated openly with each other for several minutes before reaching a consensus on whether to classify an Attack Pattern as a Bug or a Flaw.

5. Conclusions

CAPEC presents, organizes, and displays attack patterns via a set of common criteria or schema referred to as primary and supporting elements. The large number of attack patterns presented by CAPEC (101) along with the capacious quantity of individual elements associated with each attack pattern can hinder the learning process for many students. Students who are presented with vast amounts of information need assistance and a proper plan to avoid feeling overwhelmed, lost, or even frustrated [14].

Because of the significance of the attack patterns in relationship to software security it is important that a student’s overture to the topic be positive. In order to avoid a negative experience for the students, our introduction to attack patterns and CAPEC focuses on a pared down number of elements and applying them to a hierarchy to aid in retention of information.

References


