The Balance of Secure Development and Secure Operations in the Software Security Equation

Sean Barnum
The MITRE Corporation

Software security is about reducing the risk that software poses to those who use it or are affected by it. This requires thought and action more than simply at the point of development or use. It requires a more holistic approach, balancing secure development and secure operations. The bad news is that these two capable domains typically do not interact much or understand each other. The good news is that there are active ongoing efforts focused on addressing this gap.

Effectively addressing software security requires adequately balancing the secure development and the secure operations domains (see the mechanisms listed in Table 1).

The objective of security in development is to prevent security issues in the software causing vulnerability. In the best case, this means preventing such security issues from ever entering the software to begin with. This best-case approach is driven by activities such as effective security training, security policy definition, security requirements specification and review, secure architecture and design, and architectural risk analysis. In the worst case, this means at least preventing such security issues from ever being fielded into live systems. This later life-cycle approach is typically driven by activities such as secure code analysis, security testing, and penetration testing.

The objective of security in operations is to prevent security issues in deployed systems by securing their infrastructure, configuration, and use. So, the ultimate goal would be to have all operating software totally free from vulnerability and fully secure. Given the complexities involved in today's software and the ever-changing threat landscape, the reality is that no software can ever be presumed as *fully secure* and will typically be under ongoing and consistent attack. Beyond the initial security engineering of software operational deployment, the bulk of secure software operations is about continuous situational awareness and incident response. Recognizing real-world practicalities, it is focused on answering the foundational, ongoing secure operations questions:

- Are we being attacked? (Were we attacked?)
- How are we being attacked?
- What is the objective of the attack?
- What is our exposure?
- Who is attacking us?
- What should we do to protect against these attacks in the future?

The commonality between the secure development and secure operations domains is the central role of understanding how adversaries attack software. While both domains have a need to understand how software is attacked, the specific needs of each domain differ in level of abstraction and in purpose—but in a synergistic fashion. The secure development domain needs to understand the attacker's perspective in abstract terms in order to improve security across a wide range of contexts, rather than individual instances. The secure operations domain needs to understand the attacker's specific variations of behavior in gory detail in order to recognize it, understand it, estimate its effect, and plan its mitigation. Due to the reciprocal balance between the top-down perspective of secure development and the bottom-up perspective of secure operations, there is an opportunity for each domain to address its own requirements in such a way that also provides value to the other's primary focus (see Figure 1, next page).

Given the differing requirements between the two domains (to characterize attacks and potentially exchange this information), a flexible mechanism is required to capture, describe, and share knowledge about common patterns of attack. One such mechanism is the attack pattern object as specified and leveraged by the Common Attack Pattern Enumeration and Classification (CAPEC), as outlined at <http://capec.mitre.org>. CAPEC is a publicly available catalog of attack patterns along with a comprehensive schema and classification taxonomy intended to form a standard mechanism for identifying, collecting, refining, and sharing attack patterns among the software community. Established in 2000, the attack pattern concept represents a description of common attack approaches abstracted from a set of known real-world exploits. While this source of raw data comes primarily from the secure operations domain, attack patterns today are primarily a construct used by the secure development community to aid software developers in improving the assurance profile of their software.

In this role, attack patterns offer the secure development community unique value in several areas such as:

- Representing abuse cases (how an attacker would intentionally abuse a software system) during requirements elicitation, specification, and review.
- Mapping identified threats to the software's modeled attack surface as part of threat modeling activities during architecture and design.
- Guiding and prioritizing secure code analysis during implementation. This

Table 1: *Mechanisms for Secure Development and Operations*

Mechanisms of Secure Development	Mechanisms of Secure Operations
• Effective Security Training	• Secure Configurations
• Security Policy	• Firewalls
• Security Requirements	• Proxies
• Secure Architecture and Design	• Intrusion Detection Systems
• Secure Coding	• Intrusion Prevention Systems
• Security Testing	• Real-Time Data Monitoring
• Penetration Testing	• Operational Monitoring and Control
• Risk Management	• Incident Response
	• Forensics
	• Anti-Tamper Mechanisms

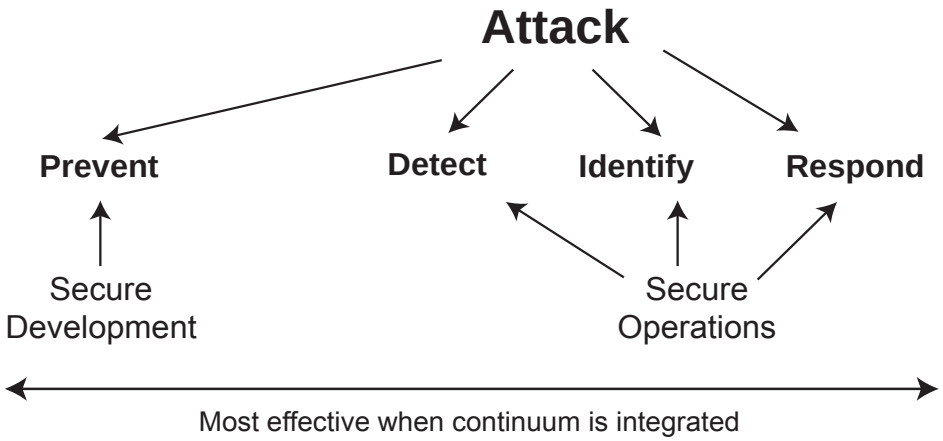


Figure 1: *How Secure Development and Operations Can Work Together*

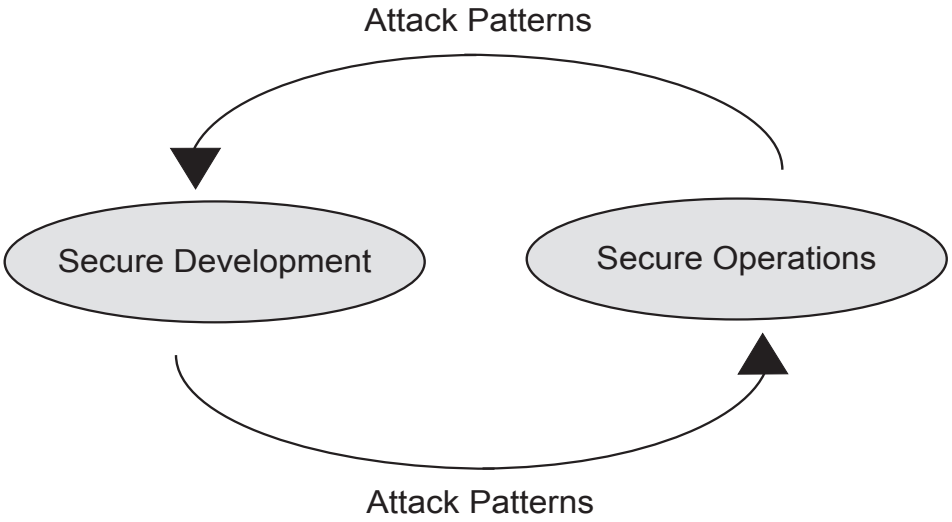


Figure 2: *Attack Patterns Bridge Secure Development and Operations*

Question	Role of Attack Patterns
Are we being attacked? (Were we attacked?)	Attack patterns offer structured descriptions of common attacker behaviors to help interpret observed operational data and determine its innocent or malicious intent.
How are we being attacked?	Attack patterns offer detailed structured descriptions of common attacker behavior to help interpret observed operational data and determine exactly what sort of attack is occurring.
What is the objective of the attack?	Elements of attack patterns outlining attacker motivation and potential attack effects can be leveraged to help map observed attack behaviors to potential attacker intent.
What is our exposure?	The structure detail and weakness mapping of attack patterns can provide guidance in where to look and what to look for when certain attack pattern behaviors are observed.
Who is attacking us?	Attack pattern threat characterization and detailed attack execution flow can provide a framework for organizing real-world attack data to assist in attribution.
What should we do to prevent against attacks in the future?	Attack patterns offer prescriptive guidance on solutions and mitigation approaches that can be effective in improving the resistance tolerance and/or resilience to instances of a given pattern of attack.

Table 2: *Attack Patterns Help Answer Questions Regarding Secure Operations*

includes identifying specific high-risk areas requiring greater analysis rigor as well as the most relevant weaknesses to look for.
- Identifying, specifying, and prioritizing security test cases.
- Serving as attack templates for penetration testing and objective persona descriptors for red team penetration testing.

The future potential for CAPEC attack patterns lies beyond their evolving and continued use within the secure development community. The secure operations community can utilize CAPEC to assist in situational awareness of deployed systems under attack and aid in response and mitigation. Several characteristics of attack patterns make them relevant for the secure operations community:
- Attack patterns provide high-level rather than simply low-level detailed patterns of attacks against software.
- Much of secure operations is about analyzing low-level activity for patterns and composing them into higher levels of abstraction to detect, identify, and respond to attacks.
- Software assurance attack patterns provide a top-down, high-level context for both the method and the intent of attacks.
- Efforts are currently under way to formalize the CAPEC attack pattern schema in order to provide adequate detail of attacks for aligning and integrating their context with bottom-up incident analysis characterizations.

Attack patterns offer a unique and practical bridge between the two domains, as shown in Figure 2.

Using attack patterns makes it possible for the secure development domain to leverage significant value from secure operations knowledge, enabling them to:
- Understand the real-world frequency and success of various types of attacks.
- Identify and prioritize relevant attack patterns.
- Identify and prioritize the most critical weaknesses to avoid.
- Identify new patterns and variations of attack.

Through the use of attack patterns, it is also possible for the secure operations domain to leverage significant value from secure development knowledge. This enables those in the secure operations domain to provide appropriate context to help answer the foundational secure operations questions (see Table 2).

One of the maturation paths currently under way for CAPEC involves integrat-

ing and refining lower-level attack attributes and characteristics to better support automatable integration of both domains. So far, this effort has been focused on enhancing attack pattern descriptions with greater levels of attack execution flow detail and on the addition of two new constructs: **Target_Attack_Surfaces** and **Observables**.

The **Target_Attack_Surfaces** construct is intended to give a structured characterization of the relevant portions of the targeted software that an attack is attempting to exploit. This sort of detail can be valuable within an operational context, assisting in attack detection, identification, and characterization through mapping of observed effects on target software assets and resources. The current draft schema (see Figure 3) focuses on characterizing functional services, protocols, command structures, etc. Future schema revisions should extend this conceptual construct to address a broader set of attack surface characteristics.

The **Observables** construct is intended to capture and characterize events or properties that are observable in the operational domain. These observable events or properties can be used to adorn the appropriate portions of the attack patterns in order to tie the logical pattern constructs to real-world evidence of their occurrence or presence. This construct has the potential for being the most important bridge between the two domains, as it enables the alignment of the low-level aggregate mapping of observables that occurs in the operations

Figure 3: *CAPEC - High-Level Attack Surface Draft Schema* (Figures 3 and 4 were created with Altova XMLSpy)

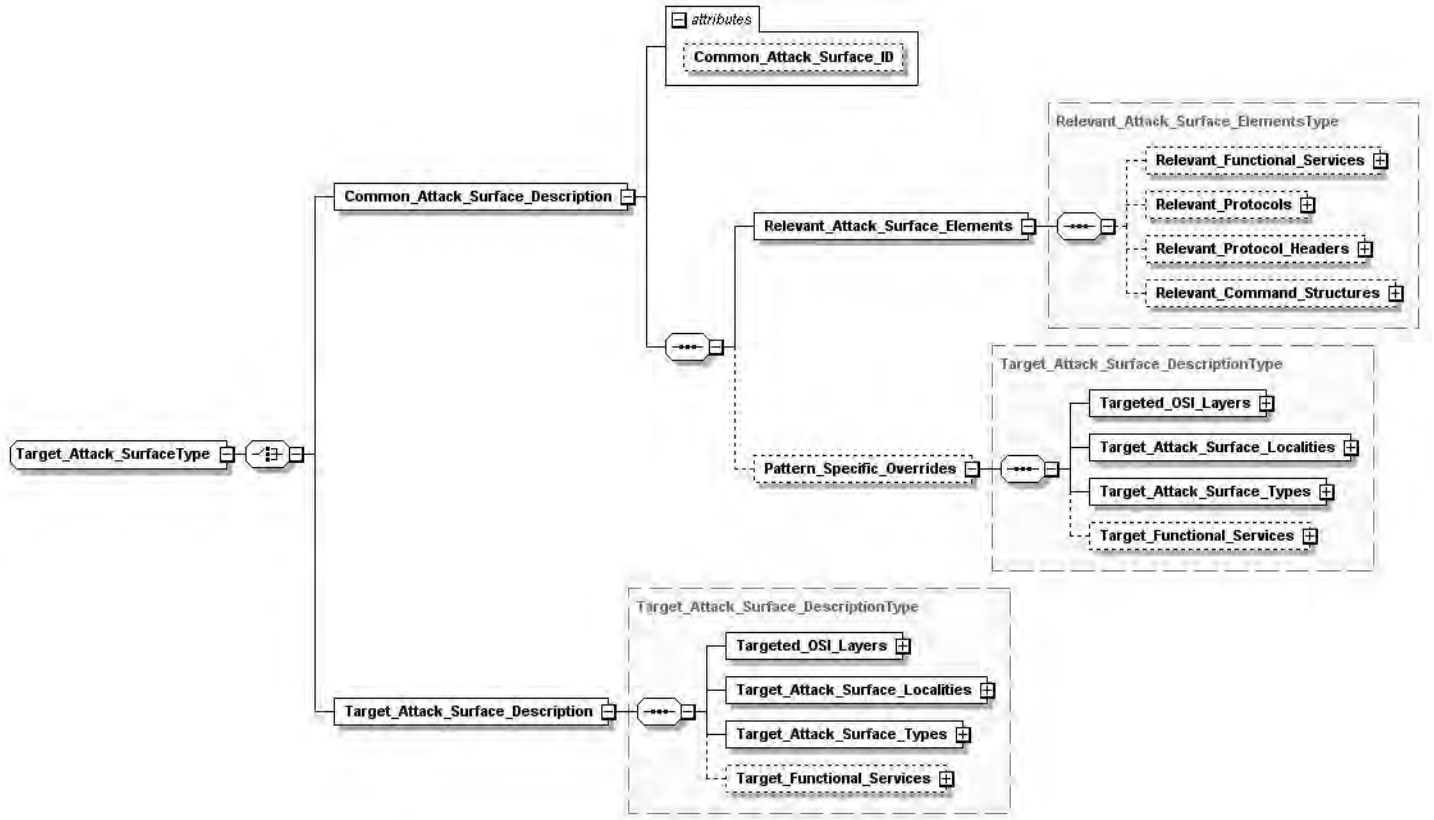
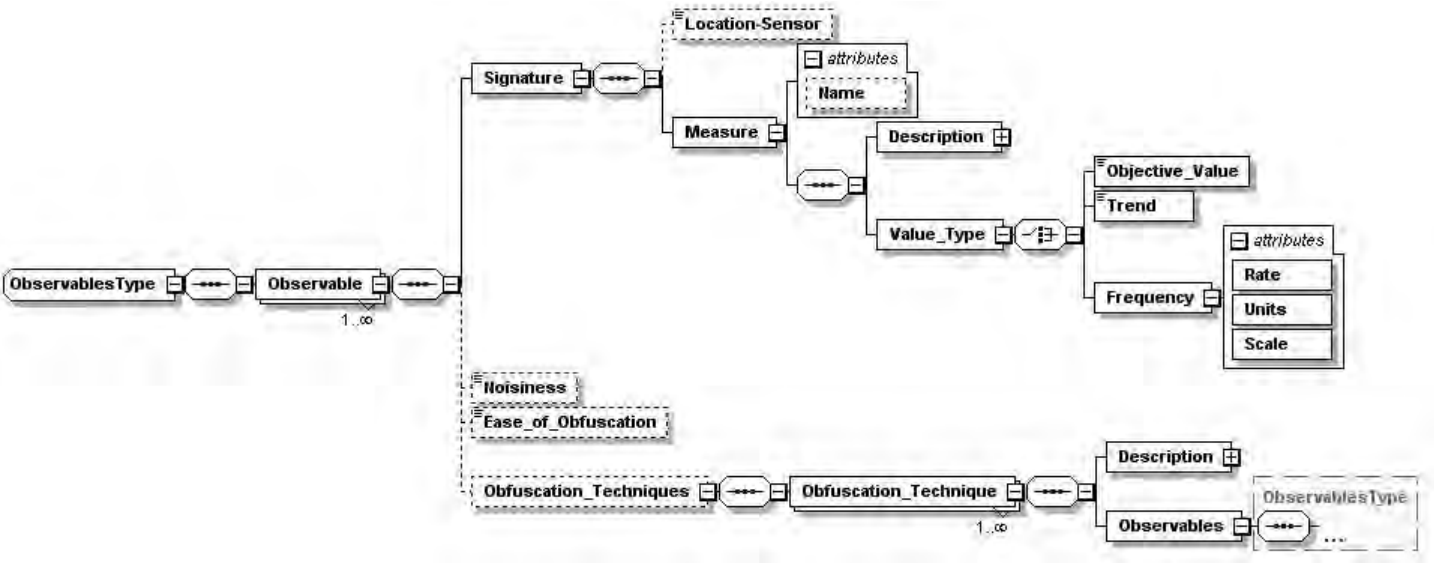


Figure 4: *CAPEC - Observables Draft Schema*



Software Defense Application

The DoD—along with its supporting defense industry—has identified cybersecurity as one of its top priorities today and going forward. The software security portion of this battle is currently being fought on two fronts, secure development and secure operations, with little coordination between the two. This article discusses the objectives and activities unique to each of these areas as well as some of their shared commonality in the relevance of understanding the attacker's perspective. Most importantly, it introduces attack patterns as a resource that characterizes this commonality and offers a practical and actionable bridge for coordination and collaboration between the secure development and secure operations communities. An understanding of attack patterns and their relevance to a unified approach to software security should be requisite knowledge for all those working in DoD software.

domain to the higher-level abstractions of attacker methodology, motivation, and capability that exist in the development domain. By capturing them in a structured fashion, the intent is to enable future potential for detailed automatable mapping and analysis heuristics.

The current **Observables** draft schema (see Figure 4 on the previous page) adorns the **Attack_Step**, **Attack_Step_Technique**, **Attack_Step Outcome**, and **Attack_Step Security_Control** elements of the attack pattern schema. It focuses on characterizing specific observable measures, their value, their sensor context, and how accurate or easy to obfuscate they are. Future schema revisions should flesh out the construct to cover other relevant dimensions. Changes will be based on input and collaboration from the operations community and other aligned knowledge standardization efforts needing this construct (e.g., Common Event Enumeration [CEE] and Malware Attribute Enumeration and Characterization [MAEC]).

People interested in learning more about CAPEC, CEE, MAEC, and other related knowledge standardization efforts can gain better insight and join in the community collaboration efforts by going to the Making Security Measureable Web site <http://msm.mitre.org> and the Software Assurance Community Resources and Information Clearinghouse <https://buildsecurityin.uscert.gov/swa>.

Summary

Effective software security requires a balanced approach between secure development and secure operations. The commonality between these two domains is the central role of understanding how adversaries attack software. CAPEC attack patterns offer a mechanism for structured characterization of common attacks that enable a useful exchange of information relevant to both domains, also aligning low-level observations to high-level contexts for mutual benefit.

CAPEC is currently a resource leveraged primarily by the secure development community, but there is an opportunity and a strong need for increased collaboration from the secure operations community. It will help shape and refine CAPEC to more effectively serve both communities, potentially acting as an integrating bridge to eventually yield a more holistic software security capability.

We encourage readers within both communities to become actively involved and lend their knowledge and voices to our unifying efforts.◆

About the Author

Sean Barnum is a software assurance principal at The MITRE Corporation, serving as a thought leader and senior advisor on software assurance and cybersecurity projects. He has more than 20 years of experience in the software industry in the areas of development, software quality assurance, quality management, process architecture and improvement, knowledge management, and security. He is involved in numerous knowledge standards-defining efforts, including Common Weakness Enumeration, CAPEC, and other elements of software assurance programs for the DHS, DoD, and the National Institute of Standards and Technology. He is co-author of the book "Software Security Engineering: A Guide for Project Managers."

Phone: (703) 473-8262
E-mail: sbarnum@mitre.org