NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CONSIDERATIONS FOR ADOPTING ZERO TRUST PRINCIPLES AND USER AND ENTITY BEHAVIOR ANALYTICS INTO DEVELOPMENT, SECURITY, AND OPERATIONS FOR PROTECTION AGAINST INSIDER THREATS

by

Laurie M. Lehman

December 2023

Thesis Advisor: Logan O. Mailloux
Second Reader: Kristin M. Giammarco

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
CONSIDERATIONS FOR ADOPTING ZERO TRUST PRINCIPLES AND USER AND ENTITY BEHAVIOR ANALYTICS INTO DEVELOPMENT, SECURITY, AND OPERATIONS FOR PROTECTION AGAINST INSIDER THREATS

5. FUNDING NUMBERS

6. AUTHOR(S) Laurie M. Lehman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

This thesis explores the incorporation of zero trust principles and user and entity behavior analytics (UEBA) into a single model to guide the design, development, integration, and deployment of information technology and specifically to the development, security, and operations (DevSecOps) of software applications to detect and protect against insider threats. The security benefits of fully implementing zero trust principles along with an integrated UEBA process in the enhanced DevSecOps methodology is studied along with a detailed analysis to explore emerging behaviors. The study serves to: (1) provide a seamless and coordinated path for integrating zero trust principles into DevSecOps execution; (2) offer useful recommendations to address cyber vulnerabilities; (3) enhance insider threat detection techniques in DevSecOps; and (4) validate whether the proposed model meets the desired outcomes for DOD components to achieve the required zero trust capabilities for data, assets, applications, and services (DAAS).

14. SUBJECT TERMS
cyber, cybersecurity, insider threat, user, behavior analytics, zero trust, development, security, operations, model, process

15. NUMBER OF PAGES
83

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

CONSIDERATIONS FOR ADOPTING ZERO TRUST PRINCIPLES AND USER AND ENTITY BEHAVIOR ANALYTICS INTO DEVELOPMENT, SECURITY, AND OPERATIONS FOR PROTECTION AGAINST INSIDER THREATS

Laurie M. Lehman
Civilian, Department of the Navy
BS, University of Maryland, Global Campus, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
December 2023

Approved by: Logan O. Mailloux
Advisor

Kristin M. Giammarco
Second Reader

Oleg A. Yakimenko
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis explores the incorporation of zero trust principles and user and entity behavior analytics (UEBA) into a single model to guide the design, development, integration, and deployment of information technology and specifically to the development, security, and operations (DevSecOps) of software applications to detect and protect against insider threats. The security benefits of fully implementing zero trust principles along with an integrated UEBA process in the enhanced DevSecOps methodology is studied along with a detailed analysis to explore emerging behaviors. The study serves to (1) provide a seamless and coordinated path for integrating zero trust principles into DevSecOps execution; (2) offer useful recommendations to address cyber vulnerabilities; (3) enhance insider threat detection techniques in DevSecOps; and (4) validate whether the proposed model meets the desired outcomes for DOD components to achieve the required zero trust capabilities for data, assets, applications, and services (DAAS).

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AI	artificial intelligence
ATO	authorization to operate
cATO	continuous ATO
CI	continuous improvement
CI/CD	continuous integration/continuous delivery
CISO	Chief Information Security Officer
DAAS	data, assets, applications and services
DAST	dynamic application security testing
Dev	development
DevSecOps	development, security, and operations
DOD	Department of Defense
IoT	internet of things
IP	internet protocol
IT	information technology
ML	machine learning
MP	Monterey Phoenix
NAVSEA	Naval Sea Systems Command
NIST	National Institute of Standards and Technology
NPS	Naval Postgraduate School
Ops	operations
SAST	static application security testing
SIEM	security information and event management
SP	Special Publication
SQL	structured query language
UEBA	user and entity behavior analytics
XSS	cross-site scripting
ZTA	zero trust architecture

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

As part of the Software Modernization Strategy, the Department of Defense (DOD) is focused on building resilient software and delivering capability to the warfighter faster. With ever-growing software complexity and increasing cyber-attacks, agile methodologies must be used to improve software development to build-in cybersecurity from the start. To meet these needs, this work considers the adoption of zero trust principles and user and entity behavior analytics (UEBA) into the development, security, and operations (DevSecOps) software development life cycle.

To determine the applicability and value zero trust and UEBA brings to the DevSecOps methodology, the following research questions are considered for this thesis:

1. What advantages does considering zero trust principles provide in protecting against malicious attacks?
2. How should zero trust security principles be integrated into the DevSecOps methodology?
3. How do user and entity behavior analytics (UEBA) solutions prove useful in detecting insider threats when incorporated with the zero trust principles?

To address these questions, a combination of qualitative research, modeling, and analysis was utilized. Initial documentation analysis resulted in the gathering of information and evidence-based insights. This knowledge gathering informed theoretical models to give context to the DevSecOps process, study the integration of security principles, techniques, and processes and as a resource to interpret the data results. The use of theoretical models also allows for the analysis of behavior(s) to be studied to fully understand the effects of emergent behaviors in the DevSecOps decision-making process.

Question 1 addresses the concept of incorporating "zero trust" into the DevSecOps methodology. The principle of zero trust highlights the importance of "never trust, always verify," which refers to having no trust in anyone or anything within or outside the network boundary. Zero trust principles are designed to protect against malicious attacks and focus

on minimizing the attack surface for attackers to exploit due to a reduction of trust given to users, devices, and applications. In addition to the classic multi-layered security approach called "defense in depth," zero trust principles create multiple barriers to prevent attackers from moving freely within the network. The robust implementation of micro-segmentation divides the network into smaller segments to enforce stricter access control and the ability to quarantine compromised segments without impacting the entire network. Micro-segmentation is a strict enforcement technique that offers a significant advantage by minimizing the impact of a breach anywhere in the network. It also prevents attackers from moving laterally throughout the network and compromising other network segments, computing resources, and data. This technique ensures that even if one segment of the network is comprised, the rest of the network remains secure.

Lastly, zero trust helps with aligning compliance standards so that they are consistently met. Compliance standards mandate robust data protection measures. This is accomplished by ensuring strict access control and only granting permission on a need-to-know basis. Should a breach occur, having encryption methods in place to protect sensitive data is also emphasized throughout the zero trust principles. Zero trust focuses on encryption, data loss prevention and secure data handling practices. According to George Finney's 2022 book *Project Zero Trust*, zero trust principles provide a solid foundation to create a security posture that aligns with compliance requirements, however, it is also a best practice to regularly conduct compliance assessments regularly to ensure full compliance with all applicable standards while minimizing information security risk (54–56).

For question 2, integrating zero trust security principles into the DevSecOps methodology requires aligning these principles with the different phases of software development. To achieve this, guiding principles must be considered during the various stages of software development:

- To apply the zero trust principle of "never trust, always verify" to all aspects of access control within the DevSecOps environment, strong authentication mechanisms, such as multi-factor authentication, for all users and devices should be implemented.

- To limit lateral movement for attackers, segment the network into smaller, isolated zones. Management of access requests and approvals ensures that access rights are being implemented correctly.

- To only grant necessary permissions to users, processes, and services through the principle of least privilege. Implementing these access rights can be accomplished using automated tools based on predefined policies.

Ensuring a comprehensive security posture starts with early integration into the software development life cycle phases of the DevSecOps process. For example, embedding security design considerations, such as threat modeling, the use of secure coding practices and the review of code for vulnerabilities before software is deployed. To ensure the security of sensitive information, end-to-end encryption must be implemented, along with encryption keys and key management systems to control access to data. Across all phases, automation must be built in wherever possible, especially for responding to incidents where zero trust architectures should be configured to automatically quarantine or isolate compromised entities to prevent further damage. Likewise, these systems must quickly analyze anomalies that could indicate insider threats or compromised accounts by leveraging UEBA tools and practices to detect potentially malicious behavior.

For question 3, combining UEBA solutions with zero trust is essential for detecting insider threats within an organization. UEBA proves useful by establishing baseline behavior patterns to identify deviations from normal behaviors, which could lead to suspicious activity. UEBA collects and analyzes information about users, such as roles, access history, and interactions which allows zero trust least access principles to be rigorously applied. For example, combining user history information with zero trust principles enables access requests to be rigorously evaluated with real-time information. Continuous monitoring using UEBA aligns well with the strict zero trust monitoring requirements where patterns indicative of insider threats can be quickly identified and isolated. Behaviors such as unauthorized data access, frequent access of sensitive information, and irregular login times can trigger zero trust alerts.

The benefits of integrating zero trust principles and UEBA practices into the DevSecOps methodology are plentiful. As with any process, but especially with the integration of zero trust and UEBA, encouraging open communication between development, security, and operations teams to ensure security concerns are addressed at every phase of development. To keep this integration successful, DevSecOps process owners should regularly assess and refine the zero trust and UEBA implementation based on evolving threats and from insights gained from incident analysis to make informed adjustments. While not easy to implement, a holistic approach should be taken as this integration will involve cultural changes, process changes, and technological implementation. Finally, to foster a shared understanding and commitment to these principles and practices, all members of the DevSecOps team(s) should be educated and the zero trust implementation must be accurately documented to include policies, access controls, monitoring mechanisms, and incident response procedures.

This thesis demonstrates that implementing zero trust principles and UEBA practices can enhance security and reduce the risks posed by modern cyber threats. The proposed approach provides a proactive strategy to combat malicious insider attacks while also meeting DOD's zero trust mandate. This research analysis results in a security focused software development environment that is not only accelerating software development, but enhancing the protection of users, systems, and valuable defense data.

References

Finney, George. 2022. *Project Zero Trust*. Indianapolis: John Wiley and Sons.

ACKNOWLEDGMENTS

Completing the PD21 Systems Engineering Management program at the Naval Postgraduate School (NPS) was a personal goal I set out to achieve at a time much later in my career. I want to thank my family, especially my husband, Eric, and my three sons, Tyler, Cody, and Travis. They allowed me to step back in my duties as wife and mom to complete this accomplishment. Many nights of no dinner on the table or laundry done, was well understood. My friends, I cannot thank them enough for giving me the support I needed to fulfill this goal.

As I started my thesis journey, my head was filled with so many ideas, so many topics, so much excitement to work on a piece of research that resonated passion within me, and so much worry about getting it done on time. My thesis advisor, Dr. Logan Mailloux, worked with me to help pull those ideas and topics together and scope out what I wanted to work on. This was challenging. My thoughts would go in several directions, but his expertise in thesis and article research and writing allowed me to focus and form a topic of interest in today's world of computer security. To him, I thank you for clearing my head and letting me see the path before me.

To my second reader and Monterey Phoenix (MP) expert, Dr. Kristin Giammarco, from whom I learned an enormous amount of information and technical skills to apply modeling to a process. As a new user of MP, she taught me how to use the tool to expose emergent behaviors and was an advocate for my models and ideas. I am forever grateful for the opportunity to use MP and learn from the outcomes it provides.

Finally, this 221 Cohort was the best group of friends one could ever ask for. Like I said, going back to college late in my career posed challenges for me and this group never lost patience with me along the way and was always lifting each other in support and comradery. Putting up with my questions, frantic calls/chats, allowing me to grow with them, and creating friendships means so much to me. Special thanks to Kevin, David, April, and Team Boomba for working with me and allowing me to learn from you and your talents.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

In this thesis, a proposal is presented to integrate zero trust principles and user and entity behavior analytics (UEBA) in the development, security, and operations (DevSecOps) of software applications. The main objective of this work is to enhance the security of software and software enabled systems by detecting and preventing malicious insider threats.

A. PROBLEM DEFINITION

The Deputy Secretary of Defense, Kathleen Hicks, issued a memorandum on February 1, 2022, approving the *DOD Software Modernization Strategy*. This strategy outlines the strategy to expedite the delivery of software that align with the Department's priorities (White House 2022a). As Ms. Hicks states,

> Delivering a more lethal force requires the ability to evolve faster and be more adaptable than our adversaries. The Department's adaptability increasingly relies on software and the ability to securely and rapidly deliver resilient software capability is a competitive advantage that will define future conflicts. Transforming software delivery times from years to minutes will require significant changes to our processes, policies, workforce, and technology. (White House 2022a)

Furthermore, the DOD's Chief Information Security Officer (CISO), David McKeown states,

> While DOD has historically been very good at perimeter defenses, it has a lot of tools that it has deployed. We're successful with 99.9% of all attack vectors. But there is this advanced capability that nation-state actors have [in which] they can get a foothold through a variety of means – phishing, brute force attacks on vulnerabilities that are on servers, web attacks, and hacking the code. And once they get a foothold, what we've found over time is we have to struggle to find them and then finally, eradicate them from an app on the network and have confidence that they're gone from the network. (Cronk 2021)

The federal government has been successful in implementing traditional security practices, often using a perimeter-based approach as stated above. However, with traditional security comes the notion of once someone or something is inside the network,

they are generally granted trusted privileges. Zero trust challenges this approach and promotes a "never trust, always verify" methodology. Complementing this strict security mindset, UEBA enables the detection of anomalous and dangerous occurrences or actions that deviate from "normal" behavior patterns within a network (Brook 2020). For example, UEBA can detect data breaches, sabotage, privilege abuse, compromised accounts, and changes in permissions by employing machine learning, advanced algorithms, and statistical analysis to examine the normal conduct of users on the system (Brook 2020). UEBA can determine when there is a deviation from established patterns, alerting to the potential of real threats (Brook 2020).

Cybercriminals, hackers, and attackers are innovative and traditional methods only offer minimal capabilities to detect these threats. Especially concerning, understanding insider user behavior, is crucial to promote the national security of the United States, safeguard the well-being of its citizens, and prevent information theft and harm from our allies and partners (White House 2022b). Thus, the DOD must consider advances in software development, utilization of software factories, and more secure DevSecOps methodologies to prevent and mitigate dangerous insider threats. This thesis uniquely examines the integration of zero trust principles and the employment of UEBA into the DevSecOps methodology.

B. THESIS OBJECTIVES

This thesis explores the incorporation of zero trust principles and UEBA into a single model to guide the design, development, integration, and deployment of information technology, and specifically, integration into the DevSecOps pipeline to detect and protect against insider threats. The unified model is used to study the security benefits of implementing zero trust principles with an integrated UEBA process to improve security and discover threatening behaviors. To examine how zero trust principles and UEBA practices can be integrated into the DevSecOps methodology, the following research questions are analyzed:

1. What advantages does considering zero trust principles provide in protecting against malicious attacks?

2. How should zero trust security principles be integrated into the DevSecOps methodology?

3. How do UEBA solutions prove useful in detecting insider threats when incorporated with the zero trust principles?

To address these questions, a combination of qualitative research, modeling, and analysis is utilized. Initial documentation analysis resulted in the gathering of information and evidence-based insights. This knowledge gathering informed theoretical models and provided context to the DevSecOps process while allowed for, the study and integration of security principles, techniques, and processes, as well as a key resource to interpret the data results. The use of theoretical models also allows for the analysis of behavior(s) to be studied to fully understand the effects of emergent behaviors in the DevSecOps decision-making process.

A model of the DevSecOps process was architected to clearly define the relationships between security and the software development life cycle phases. This involved analyzing literature on zero trust principles, UEBA concepts, and the DevSecOps approach to more fully understand and identify areas where these principles, practices, concepts, and process flows should be integrated into a single unified DevSecOps methodology. Model development also involved determining where and how UEBA practices should be integrated to monitor user and entity behavior in real-time. Each model was verified for consistency against knowledge gained from the literature review and formalized processes. The modeling effort was validated through a hypothetical scenario using MP to identify and understand emergent behaviors.

The outcome of this thesis provides a process model and analysis which combines zero trust principles and UEBA practices into the requirements, design, test, and security phases of the DevSecOps process. Furthermore, this model is tailorable, allowing systems and security engineers to shape the proposed approach into the DevOps Continuous Integration/Continuous Delivery (CI/CD) pipeline. Ultimately, this work provides the next generation of DevSecOps professionals with a secure and efficient means of engineering safe and secure complex software systems.

This study serves to: (1) verify whether the proposed model provides a seamless and coordinated zero trust execution; (2) offer useful recommendations to address cyber vulnerabilities; (3) enable insider threat detection techniques; and (4) validate whether the model establishes desired outcomes for DOD components to achieve the minimum required zero trust capabilities for data, assets, applications, and services (DAAS). It is expected that this work will result in recommendations beyond just DAAS, to include insights gained with respect to people, processes, and technology.

C. OVERVIEW OF THESIS

Chapter I introduces challenges the federal government and security professionals face when developing complex software applications and systems. The chapter also discusses and motivates the significance of addressing defense focused security concerns while considering the implementation of zero trust principles along with UEBA practices into the DevSecOps methodology. Finally, research questions are presented.

Chapter II provides a detailed background for the reader to gain a clear understanding of necessary terminology and definitions associated with this work. This chapter is framed with a series of questions to help facilitate the reader's interest and understanding, prior to referring to specific language and knowledge in later chapters.

Chapter III provides an overview of the research approach and methods used for developing and studying an integrated DevSecOps model with zero trust principles and UEBA practices. Appendices A and B provide details on the modelling effort as well as the verification and validation effort.

Chapter IV is a detailed discussion of the integrated DevSecOps process model. The research and the models developed and analyzed help to understand and provide insights towards incorporating zero trust and UEBA into the DevSecOps software development life cycle phases. The chapter concludes with a summary of the applicability of this research to the federal government, specifically in considering the adoption of zero trust into the strategic guidance for consistent implementation across the NAVSEA Warfare Centers.

Chapter V summarizes the work accomplished with a focus on the author's research contribution and recommendations. Concluding thoughts are also presented.

Appendix A provides a detailed discussion of model verification and validation activities. Ensuring the logical and correct implementation of the model in MP is achieved through verification. Execution of behavior models is an important step in the development and validation process of the model. Validation is accomplished by comparing the modeled process behavior to subject matter knowledge about actual process behavior to ensure its completeness and accuracy in representation.

Appendix B provides the source code developed to build these models in MP.

THIS PAGE INTENTIONALLY LEFT BLANK

II. COMMON PRINCIPLES, METHODS, AND RESOURCES

Before delving into the details of this research, it is essential to ensure that the reader has a clear understanding of DevSecOps, zero trust, and UEBA. Therefore, these critical aspects are discussed.

A. INTRODUCTION TO DEVELOPMENT OPERATIONS

The combination of software development (Dev) and information technology operations (Ops) forms a successful practice known as "DevOps" which is an agile methodology where the software development life cycle is accelerated to allow for continuous delivery with high quality. See Figure 1 for an illustration of the DevOps approach. DevOps offers speed and high-quality software through the practice of incremental delivery pipelines. However, like most things in today's technological world, there are many system-level challenges, including cybersecurity. DevOps can provide products released several times a day, but it is difficult for security testers to keep up with the frequent releases searching for software flaws.

DevOps focuses on responsive software capability for the warfighter, however, to take advantage of a fully integrated life cycle, security is a key role that must be considered. Forgoing security in DevOps results in more data breaches largely due to software vulnerabilities. Security mistakes have the potential to cause considerable harm and cost organizations millions of dollars to fix (CISA OCE 2020).

> In the past, the role of security was isolated to a specific team in the final stage of development. That wasn't as problematic when development cycles lasted months or even years, but those days are over. Effective DevOps ensures rapid and frequent development cycles (sometimes weeks or days), but outdated security practices can undo even the most efficient DevOps initiatives. (Red Hat 2023)

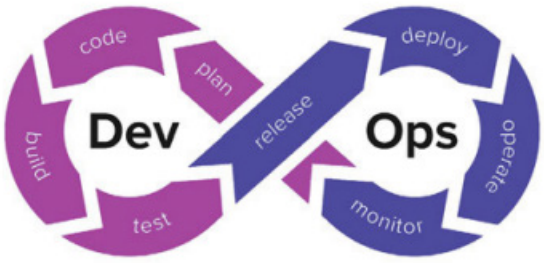


Figure 1. DevOps Model. Source: Miller, Giachetti, and Van Bossuyt (2022).

B. INTRODUCTION TO DEVELOPMENT, SECURITY, AND OPERATIONS

DevSecOps extends the principles of DevOps to include security considerations throughout each step in the software development life cycle (Red Hat 2023). Adopting the DevSecOps methodology into a culture where everyone takes part in ensuring cybersecurity is "baked in" and not "bolted on" to the system development life cycle is crucial. Carol Woody (2021) states in *A Cybersecurity Engineering Strategy for DevSecOps* presentation, that the methodology seamlessly integrates the Development-Security-Operations pipeline which is a complex design itself bridging the relationship between people and technology (15).

The DevSecOps pipeline as illustrated in Figure 2, fosters a culture of collaboration, automation, and continuous improvement. It ensures that security is not a separate phase but is integrated into every aspect of the software development and deployment process. This approach helps the federal government meet their objectives and commitments while maintaining a strong focus on security (Woody 2021, 15).

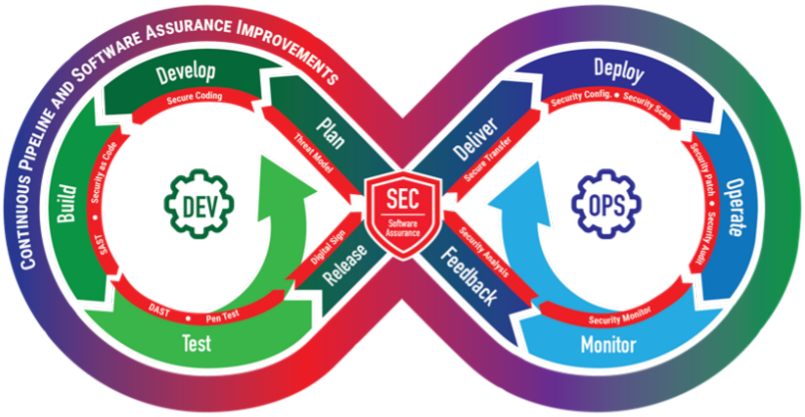


Figure 2. DevSecOps Pipeline. Source: Woody (2021).

The DevSecOps methodology is designed to align with and adapt to the iterative release process while simultaneously identifying security vulnerabilities that may be present in those releases. This approach also ensures that key cybersecurity requirements are met, allowing for the creation of a scalable DevSecOps platform that can support both internal and external software development within organizations. Additionally, the DevSecOps methodology can be used to aid the DOD's software development modernization efforts by addressing various complex areas such as training, culture, technology, tool support and configuration, policy, interoperability, and metrics.

Transitioning from DevOps to DevSecOps is a complex process that requires the cooperation and collaboration of all teams. The goal is to establish a shared culture where everyone takes responsibility for security while concurrently developing software; it is not solely the responsibility of developers to incorporate security measures. Security experts must join forces with the DevSecOps team to ensure that security practices are implemented throughout the entire software development life cycle with minimal disruption. Also, the security team should have the authority to identify, discuss, and reject unsafe choices while also providing alternative solutions.

C. INTRODUCTION TO ZERO TRUST

Zero trust recognizes that traditional network perimeters are no longer effective in a world where threats come from both external and internal sources. Zero trust promotes a

holistic security approach that integrates security into every aspect of an organization's infrastructure and operations, with the assumption that trust is never assumed, and continuous verification is essential to maintain security. Zero Trust is defined as

> a collection of concepts and ideas designed to minimize uncertainty in enforcing accurate, least privilege per-request access decisions in information systems and services in the face of a network viewed as compromised. (Rose et al., 2020, 4)

With most organizations standing up and implementing zero trust strategies within the United States government, the National Institute of Standards and Technology (NIST) was consulted to develop guidance and standards for implementation. NIST Special Publication (SP) 800–207 Zero Trust Architecture provides "general deployment models and use cases where zero trust could improve an enterprise's overall information technology security posture" (Rose et al., 2020, ii). While NIST SP 800-207 focuses on abstract reference architecture, Finney (2022) provides four practical design principles, detailed below, and five design methodology steps, shown later to assist the federal government in their zero trust implementation. These principles are essential for the success of zero trust for the DOD:

> 1. Define business outcomes: Ask the question, "What is the business trying to achieve?" This aligns zero trust with the grand strategic outcomes of the organization and makes cybersecurity a business enabler instead of the business inhibitor that it is often seen as today,
>
> 2. Design from the inside out: Start with the DAAS elements and the protect surfaces that need protection and design outward from there,
>
> 3. Determine who or what needs access: Determine who needs to have access to a resource to get their job done. It is very common to give too many users too much access to sensitive data for no business reason, and
>
> 4. Inspect and log all traffic: All traffic going to and from a protect surface must be inspected and logged for malicious content and unauthorized activity, up through Layer 7. (Finney, 2022, 165)

Integrating zero trust within the DevSecOps methodology marks a change in approach towards authentication and security mechanisms, indicating a shift in philosophy (DOD 2022, 1). Standing up a zero trust strategy may sound daunting or expensive,

however, John Kindervag states that, zero trust design principles and methodology can aid in reducing complexity (Finney 2022, 160). He also indicates implementing zero trust takes building a team and collaborating with leaders to understand the business functionality and where security plays a role in how the business uses technology as indicated in the principles above (Finney 2022, 160). Having a plan based on team objectives and goals is important for gauging value and for setting a timeframe. The federal government should look to focus efforts into six-to-nine-month initiatives (Finney 2022, 163). Another recommended tactic when determining the best time to engage in such initiatives is during fiscal year budget cycles. This allows funds to be allocated to cover any costs associated with implementing zero trust and provides a window of opportunity to show the value based on the budget allocation (Finney 2022). Keeping to a timeframe with manageable goals is important for not only achieving success but for team morale as well as milestone achievement no matter how small.

With zero trust it is essential to establish a baseline for the zero trust profile. By having a baseline, metrics captured from changes are visible as the zero trust strategy matures and refines over time. This is a significant cultural change that requires support and implementation from all stakeholders within the DOD, starting from FY2023 till FY2027 and beyond (DOD 2022, 1).

D. INTRODUCTION TO USER AND ENTITY BEHAVIOR ANALYTICS

UEBA is a cybersecurity approach that detects and prevents insider threats, compromised accounts, and other malicious activities within an organization's network (Gurucul 2022). It leverages advanced analytics, machine learning, and behavioral modeling techniques to monitor and analyze the behavior of users and entities (such as devices, applications, systems, and even data access) to identify anomalous or suspicious activities. The main idea behind UEBA is to establish a baseline of "normal" behavior for each user and entity on the network (Fortinet 2023). By continuously monitoring their actions and interactions with various resources and systems, UEBA can detect deviations from these established patterns. When significant deviations occur, the system triggers

alerts or notifications to security analysts, allowing them to investigate potential security incidents proactively.

The integration of zero trust and UEBA into the DevSecOps methodology has been studied, but there is limited literature on the subject. Most notably, NIST published the NIST SP 800-207 Zero Trust Architecture (Rose et al. 2020) document, which outlines the creation of a zero trust architecture for planning infrastructure and workflows. This publication explains how zero trust enhances cybersecurity paradigms and improves the overall protection of users, assets, and resources. It also provides deployment models and use cases showing where and how zero trust principles can be employed to improve security postures. However, NIST SP 800-207 does not discuss how zero trust can be integrated into the software development life cycle or how UEBA can enable behavioral analytics to monitor network activity.

In July 2021, the Software Engineering Institute located at Carnegie Mellon University (Sanders et al. 2021) produced a white paper discussing how zero trust can combat a high-level of intrusions. The paper identifies the stages of the DevSecOps process where zero trust can be implemented and addresses the question of how to reason about zero trust considerations in a hybrid computing environment. They also suggest the successful integration of zero trust and DevSecOps requires organizations to understand their current operational state well, how each part functions together, and the impacts that can occur between them. This paper also comments on the benefits of integrating zero trust and UEBA into DevSecOps to protect against malicious threats, particularly insider threats. Of particular importance are principles that Sanders et al. state are required for implementing zero trust:

1. Ensure all resources are accessed securely, regardless of location,
2. Inspect and log all traffic,
3. Adopt a least privilege strategy and strictly enforce access control,
4. Ensure all components support application programming interfaces (APIs) for event and data exchange,
5. Automate actions across environments and systems, driven by context and events, and
6. Deliver tactical and strategic value. (Sanders et al., 2021, 3)

E. MITIGATING GROWING CYBERSECURITY RISKS

Using classic systems software engineering models, whether it be the waterfall or V model, engineers gather requirements, develop a design and code, conduct testing, and deliver the complete product. Often this cycle can take months or years to accomplish for large defense systems. However, in today's evolving world, there is a need to pivot to an agile software development process to meet emerging needs. With the rapid pace of technology advancements, the focus becomes agility, flexibility, and adaptability to develop software incrementally and deliver capability sooner. With these shifts in technology comes the addition of cybersecurity risk as depicted in Table 1.

Table 1. Addition of Cybersecurity Risk. Source: Woody (2021, 5).

From	**To**	**Cybersecurity Risk**
Hardware-based solution	Software-intensive system	Compounded by new software vulnerabilities daily
Waterfall methodology	Agile at scale approach	Struggle to meet today's demands to get capabilities to the warfighter faster
Organization owned infrastructure	Shared infrastructure (e.g., Cloud)	Creates a larger attack surface where data can be compromised
Compliance verification upon completion before fielding (e.g., Authorization to Operate (ATO))	Continuous integrated monitoring (e.g., Continuous ATO (cATO))	Software remains fielded despite lack of adhering to compliance policies
Systems developed from requirements and architectural designs	Systems assembled primarily from reused (often 3rd party) components that map to requirements	Reused code contains unknown defects that make their way into the system
Development life cycle tailored to the system under development	DevSecOps Development Factory using 3rd party tools and automation	Reuse of tools and automation creates risk for injection of vulnerabilities

"Most of the software in use today is assembled from existing code connected with third-party services and products, which escalates the cybersecurity risk" (Woody 2021, 27). Developers can save time and money by extensively reusing existing code (Woody 2021, 27). The reused code contains unknown defects that propagate vulnerabilities (Woody 2021, 27). Thus, much of the code in the software supply chain is from third-party components, open source, and code libraries (Woody 2021, 27). The news from these sources is concerning in a high-consequence operational climate already bursting with cybersecurity risk. What if the code was used by a cyber-criminal to hack military operations? As Finney (2022) stated, "Trust is a vulnerability."

In the realm of cybersecurity, it is widely acknowledged in the literature that cyber threats are becoming increasingly sophisticated and frequent. These threats can be daunting and difficult for cybersecurity experts to manage. The federal government faces significant challenges in protecting its data and systems from sophisticated insider threats, as do other organizations. These challenges stem from the fact that malicious insiders often have legitimate access and insider knowledge of the federal government's operations, making it more difficult to detect and prevent their activities. Gurucul discusses challenges associated with "privileged access misuse, data exfiltration, account compromise, flight risk users, and remote access monitoring" (Gurucul, n.d.). Gurucul suggests that employing UEBA can effectively address insider threats. They provide a visual aid, shown in Figure 3, which illustrates the process of creating behavioral baselines, validating anomalous behaviors through identity and access analytics, alerts, and human resources data (Gurucul, n.d.).

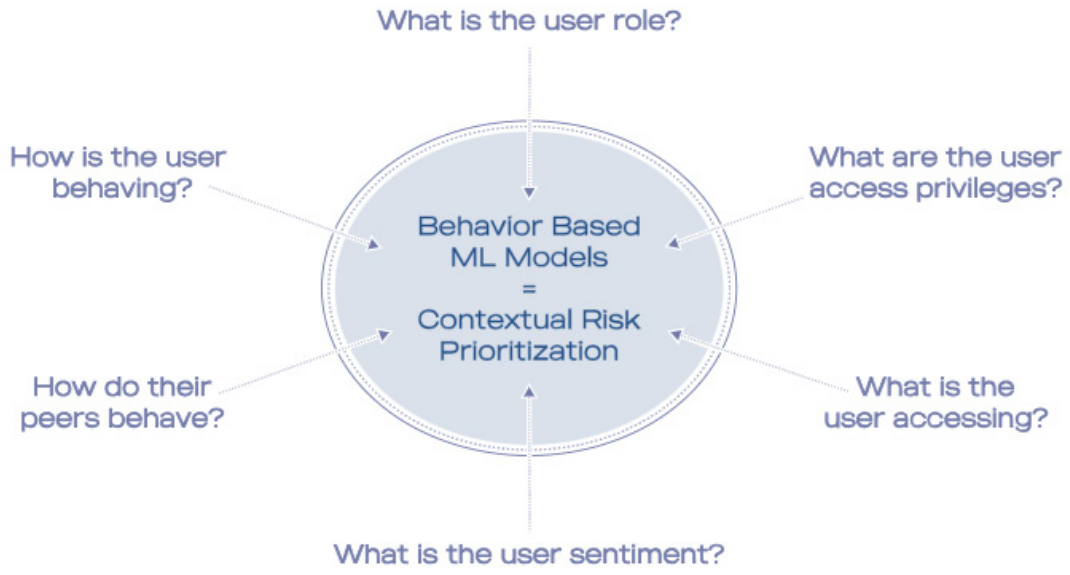


Figure 3. Gurucul's Insider Threat Solution. Source: (Gurucul, n.d.).

According to the *2023 Insider Threat Report*, "74% of organizations say insider attacks have become more frequent and they feel moderately to extremely vulnerable" (Cybersecurity Insiders 2023). Cybersecurity Insiders surveyed 326 professionals to understand the latest cybersecurity trends (Cybersecurity Insiders 2023). Upon investigation, the most significant impacts resulting from insider attacks were "loss of critical data, brand damage, and operational disruption or outage" (Cybersecurity Insiders 2023). According to the survey, it is more difficult to detect and prevent internal attacks than external attacks. This is because trusted insiders have authorized access, and it is challenging to differentiate between legitimate use and malicious attacks (Cybersecurity Insiders 2023). "At its core, insider risk is a data protection problem" (Cybersecurity Insiders 2023).

Incorporating zero trust principles alongside UEBA within a DevSecOps methodology can help the federal government proactively defend against all types of threats, from cybercriminals to nation-state-sponsored disrupters and malicious insiders (DOD 2022). Improved cybersecurity engineering strategies, built upon zero trust, must be integrated into DevSecOps to secure the software supply chain, and protect the cyber

operational attack surface (Woody 2021). For example, integrating security testing into the testing process as well as throughout the entire DevSecOps process, is a stated best practice. This integrated approach also has the added benefit of not adding delays into the development process (Finney 2022).

III. RESEARCH METHODOLOGY

This research effort began with a thorough literature review to understand the software development life cycle under the DevOps paradigm. This included assessing the current state of the DevOps process to identify strengths, weaknesses, and potential gaps. Furthermore, the researcher gathered and analyzed literature on zero trust principles, UEBA concepts, and the DevSecOps approach. This detailed understanding helped identify areas where zero trust and UEBA principles, practices, concepts, and process flows can be integrated into the DevSecOps methodology. Lastly, a literature review to understand cybersecurity trends and risks due to the use of third-party code and specifically the challenges faced by malicious insiders was conducted. This information highlights the importance of data protection and the use of zero trust and UEBA principles to prevent network attacks.

The second main research effort included the development of a unified DevSecOps process model to clearly define the relationship between the cybersecurity aspects of high-interest and the software development life cycle phases. The goal of this proposed model is to incorporate zero trust principles clearly and effectively into the DevSecOps process. For example, focusing on the principle of "never trust, always verify" results in segmentation and strict access controls throughout the DevSecOps approach. The proposed model also includes determining where and how UEBA practices can be integrated to monitor user and entity behavior in real-time.

The proposed model was developed in three rounds of increasing complexity which resulted in three models detailed in Chapter IV. Each round of development included detailed review and corrections by the advisor and student. Once the baseline model was established and validated, additional complexities were introduced into each cybersecurity activity to create a more in-depth model. The result is an integrated model that can be used to study and assess the feasibility of using zero trust and UEBA to create a powerful cybersecurity solution.

Finally, the model was verified for consistency against knowledge gained from the literature review, against existing process specifications, and detailed reviews by subject matter experts. Details of the verification are provided in Appendix A. Validation was accomplished through the analysis of the integrated zero trust and UEBA model to ensure efficiency and pace of development efforts are not hindered. Validation also occurred through a hypothetical scenario using MP to uncover emergent behaviors as detailed in Appendix B.

Based on this knowledge capture, insights gained, and analysis performed, it is expected that recommendations can be offered for improving cybersecurity and meeting zero trust implementation requirements. Additionally, the model is used to perform emergent behavior analysis seeking to identify undesirable behaviors that can occur within a system or process. By applying human reasoning and using an open-source language, approach and tool called Monterey Phoenix (MP), it is possible to identify both unexpected and unwanted behaviors. In particular, emergent behavior analysis helps the system designers identify hazardous operational scenarios that were not previously considered (Giammarco 2022, 9).

IV. CONSIDERATION FOR ADOPTING ZERO TRUST PRINCIPLES AND UEBA INTO DEVSECOPS FOR PROTECTION AGAINST INSIDER THREATS

This chapter looks at the incorporation of the zero trust principles into the DevSecOps methodology and how UEBA can enhance the goals and objectives of the zero trust principles. The development of a single, enhanced process demonstrates progress towards the adoption of zero trust into DevSecOps and for the entirety of the DOD. This approach highlights the importance of following a process while adjusting and reacting to both recognized and unrecognized malicious individuals to prevent data loss. The integrated model combines cybersecurity paradigms and UEBA with developers, operators, and security teams working in tandem to develop with new capabilities.

"Maintaining diligence in identifying, assessing, and mitigating cybersecurity risks for the federal government is imperative to guard against disruption, degradation, and unauthorized alteration of information and information systems" (Boyens et al. 2021). Improving DevSecOps processes also has great effects on the secure and safe development of DOD weapon systems, software, core processes, risk profile, and national security.

A. THE BASELINE MODEL: DEVSECOPS

Before an executable model was developed, an initial representation of the DevSecOps process was built to understand the relationships between the software development phases, the inclusion of existing cybersecurity activities (i.e., no zero trust principles), and the overall process flow.

Figure 4 depicts the DevSecOps software development life cycle including eight distinct phases, namely Plan, Develop, Build, Test, Release & Deliver, Deploy, Operate, and Monitor. These phases were studied to accurately understand and accurately align zero trust principles and UEBA practices. As shown in Figure 4, each phase is given due importance with a focus on specific security activities that cybersecurity applies. As currently designed, the Cybersecurity phase provides a blanket of protection over the whole

system along with the Monitor phase to iteratively and continuously provide ongoing awareness of information security, vulnerabilities, and threats (DOD 2021, 36).

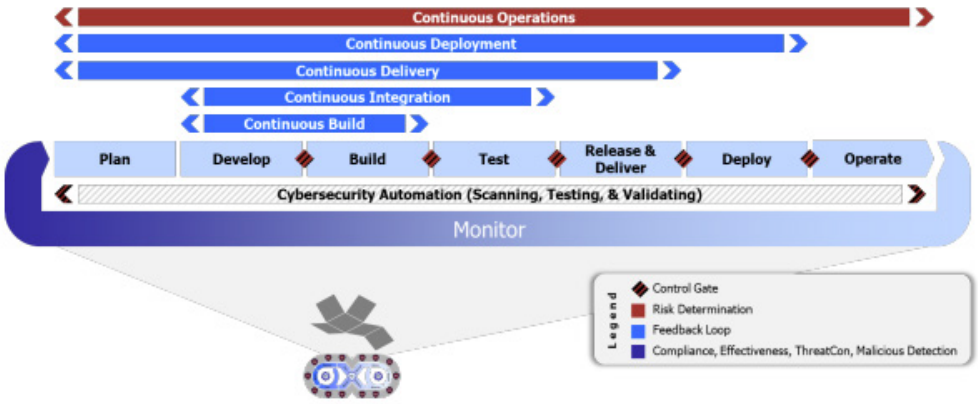


Figure 4. DevSecOps Life cycle Phases. Source: DOD (2021, 6).

This baseline model provides a fundamental representation of the process (or system of interest) and serves as a starting point to understand the essential elements and behaviors of the process before incorporating additional complexities. It is also used as a reference point for comparison with the proposed unified model.

Figure 5 depicts the initial DevSecOps model using a visual modeling tool called Seabird (https://openseabird.com). The DevSecOps process consists of eight phases illustrated across the top of the model. These phases start with the Plan phase, move to the Develop phase, and continue through to the Operate phase. As the process moves through each phase, appropriate security activities are conducted (color coded and shown below each phase). Each security activity enters and exits the Security phase as necessary as required for each phase in the DevSecOps life cycle. When the process reaches the Operate phase, it enters an iterative cycle where a next sprint in an agile software development process is initiated. In this way, the software under development makes its way through the process again and again until satisfactorily completed. While this iterative approach allows for the rapid delivery of software to the warfighter through small increments, it is currently not sufficiently secure to meet DOD security requirements.

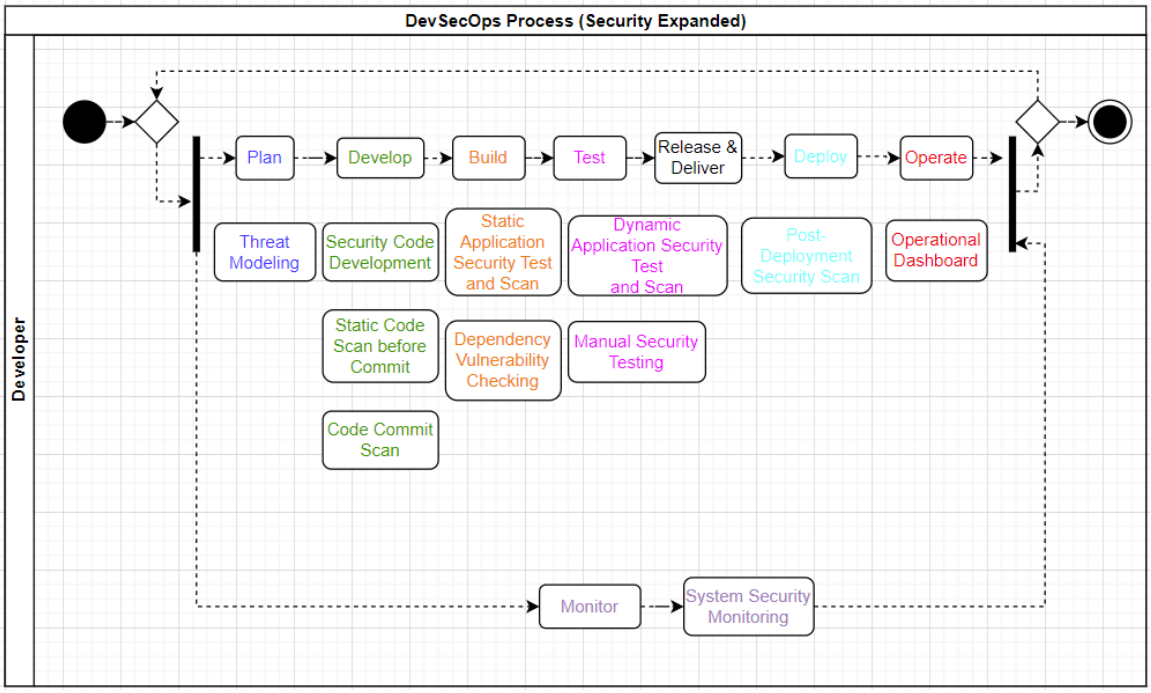


Figure 5. Initial Representation of the DevSecOps Model

After a detailed review of the Seabird model, an executable DevSecOps model was developed using MP as shown in Figure 6. This model focused on building the relationships from the various Security activities to the eight software development phases. During model development, the essential processes and relationships were introduced, reviewed, and further refined to accurately capture the flow and behaviors within the model.

Figure 6. Baseline Executable DevSecOps Model

This MP executable model serves as a benchmark for comparative analysis for the remainder of the analysis effort. By comparing this baseline model to later, more advanced models, the effectiveness of the recommended additions and/or process modifications can

be more fully considered and formally evaluated. The baseline executable DevSecOps model is built upon assumptions defining its scope and limitations. First, security is a critical element throughout the entire DevSecOps process, and it is enforced during each phase of the life cycle. Second, performing security activities as early as possible is advantageous for finding and fixing issues at low cost, while also remaining agile. Lastly, security testing should utilize automation as much as possible. The impact of these assumptions and additional insights gained through the literature review and analysis of the DevSecOps model are listed below:

4. During the Plan phase, the Threat Modeling activity allows for the software architects to utilize the system design to identify and mitigate potential security issues early. Building in the complexity of not identifying potential threats and weaknesses leaves the development effort with no mitigation plan defined to handle unforeseen vulnerabilities. (DOD 2021, 12)

- During the Develop phase, one of the key activities is Security Code Development. This involves incorporating security considerations into the system's initial design to establish a secure foundation from the outset. Secure coding guidelines and code reviews are implemented to prevent the introduction of vulnerabilities during code development. As is common practice during software development, integrating non-peer-reviewed open-source software often adds complexity and introduces risks from potentially insecure and unreliable code. Thus, Static Code Scans are conducted to identify any vulnerabilities in the code and to recommend mitigation solutions. A Code Commit Scan is then performed to ensure that no confidential information is present before uploading the code to the repository. This activity can be especially important because security breaches can jeopardize national security initiatives if sensitive data is leaked (DOD 2021, 17).

- The Build phase includes Static Application Test and Scan activities. Here code and components are tested for vulnerabilities as they are developed and integrated, thus providing a secure setup while simultaneously minimizing the risk of late vulnerability discovery (DOD 2021, 21). Static Application Security Testing (SAST) involves the use of tools to examine every line of code looking for coding patterns, constructs, and potential vulnerabilities. If not performed, it could leave the code vulnerable to Structured Query Language (SQL) injection attacks, cross-site scripting (XSS), and buffer overflows. It is also crucial to conduct Dependency Vulnerability Checking to help mitigate supply chain risks by detecting vulnerabilities in dependent components through public disclosure of open source vulnerabilities (DOD 2021, 21).

- During the Test phase the Dynamic Security Application Test and Scan cybersecurity activity is performed. This activity executes Dynamic Application Security Test (DAST) tools and treats the application like a black box, meaning it does not require access to the source code and instead tests the running application to observe its behavior from requests it receives. During this testing, activities are conducted from an attacker's perspective, simulating a real cybersecurity attack to observe how the application responds. This activity is depicted as Manual Security Test in the model and is commonly known as "penetration testing."

- The Deploy phase includes a Post-Deployment Security Scan, which ensures that the environment into which the software is being deployed is secure and that the deployment process itself does not introduce new vulnerabilities (DOD 2021, 33).

- During the Monitor phase, System Security Monitoring scans and assesses the security of all system components. This phase can be complex as it involves detecting vulnerabilities, discovering non-compliant findings, and

observing alerts and notifications on an operational dashboard to perform a health check of the entire application (DOD 2021, 41).

B. THE INTEGRATED MODEL: COMPILATION OF DEVSECOPS, ZERO TRUST, AND USER AND ENTITY BEHAVIOR ANALYTICS

Figure 7 is a visual representation of what it might look like to incorporate zero trust and UEBA into the DevSecOps methodology. This model has undergone several rounds of review and revision by the author and primary advisor to ensure correctness and clearly communicate a single integrated model. The model captures the essence of what organizations need to implement from zero trust and UEBA to protect the confidentiality, integrity, and availability of data and particularly to reduce gaps in merging the processes with the goal of preventing and mitigating insider threats. This representation was used to develop the executable MP model shown in Figure 8. The integrated model was validated by comparing existing development processes and environments to the integration of the zero trust principles into the DevSecOps approach. Detailed information on the model and Verification and Validation efforts can be found in Appendices A and B.

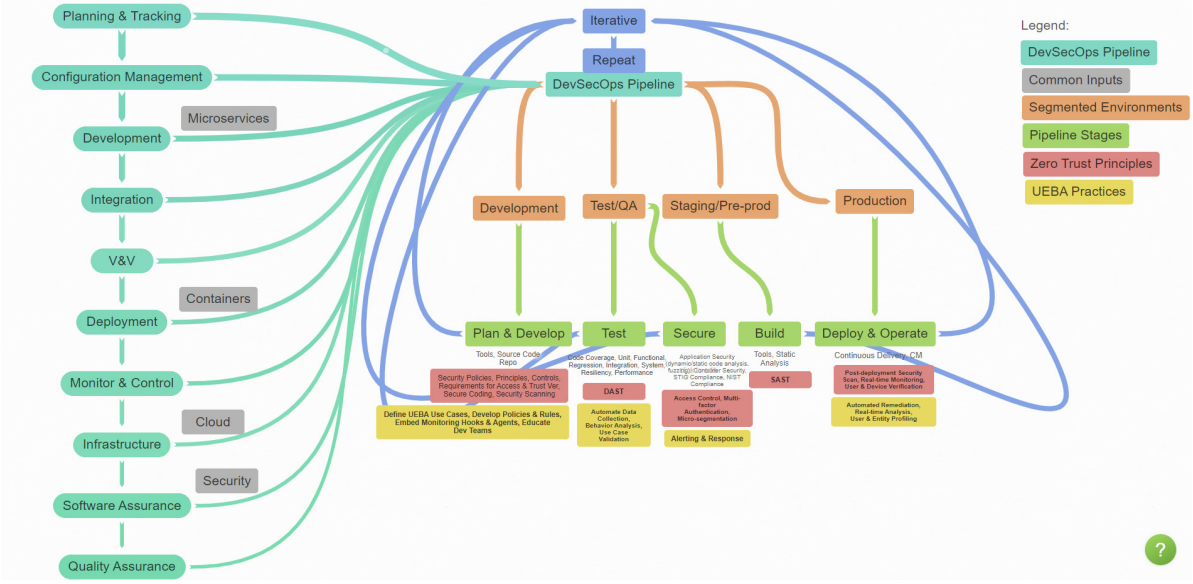


Figure 7. Initial Visual Representation of the DevSecOps, Zero Trust, and UEBA Model

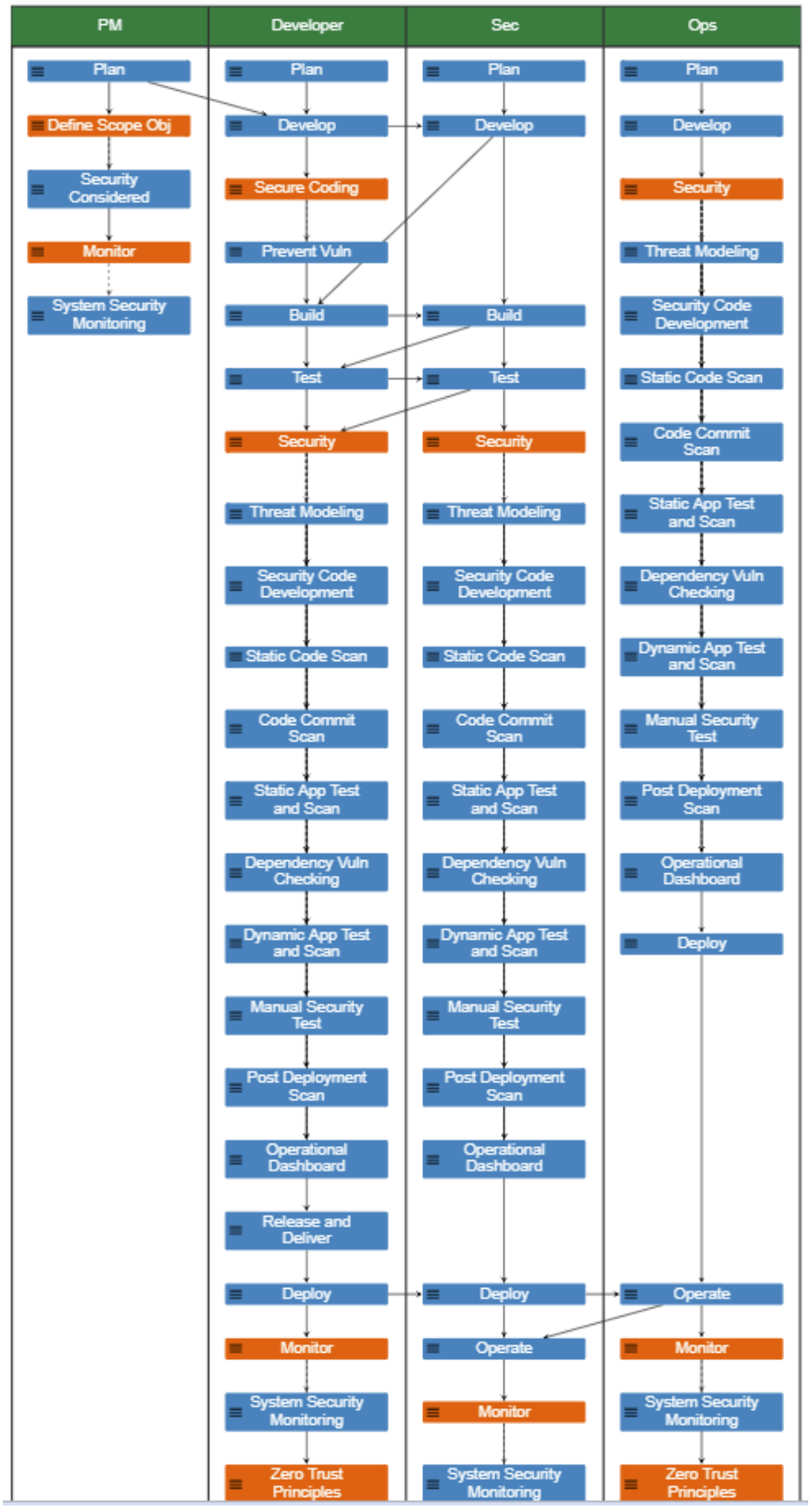
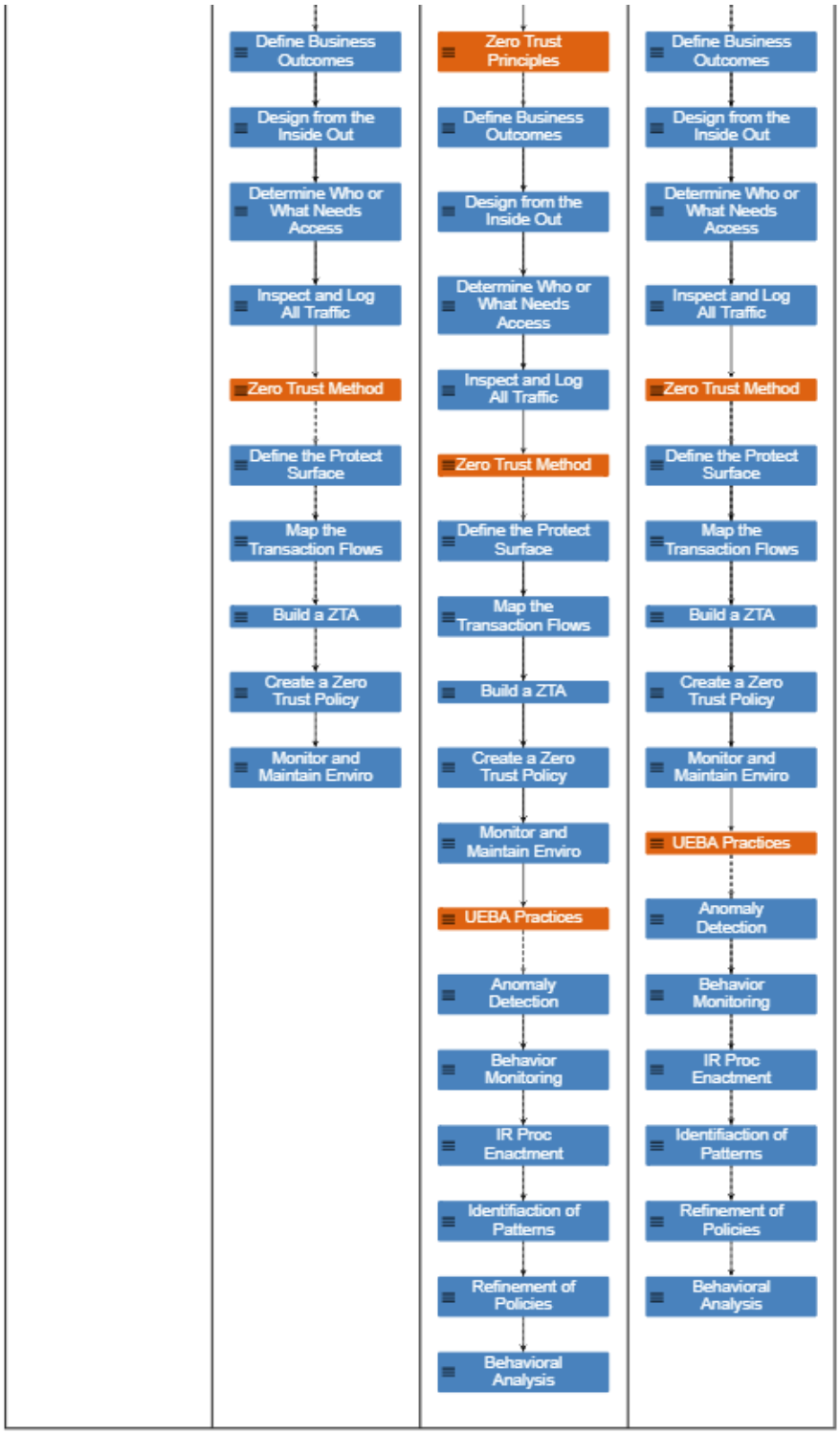


Figure 8. The Integrated Model



(Figure 8 continued)

The integrated model begins with a holistic picture of the team, including the Program Manager, Developer, Security, and Operations personnel. All these teams are responsible for planning the software development effort, and specifically defining the project's requirements, scope, and objectives. Security considerations are factored into the planning phase to identify potential risks and establish security goals. Those decisions are passed to the Develop phase where the actual coding of the software occurs. Security practices, such as secure coding best practices and code reviews, are implemented to ensure vulnerabilities are not introduced during the coding process.

The Build phase involves "integrating code changes from multiple developers into a shared repository" (Sanders et al. 2021). Automated security tests, such as static code analysis and vulnerability scanning can catch security issues early. The Test phase is where comprehensive testing is performed, including functional, performance, and security testing. Security testing encompasses activities such as penetration testing, security scanning, and vulnerability assessment. Once testing results are satisfactory, the Release and Delivery phase focuses on performing automated software releases. While security controls, like access controls, authentication mechanisms, and encryption are integrated throughout the process, they are primarily realized in this phase. The phase takes automation one step further and deploys the software application to the production environment. The Deploy phase ensures security measures are in place and confirms the integrity of the deployment.

Once the software application is deployed the Operate phase identifies and responds to security incidents and anomalies in real time. The Monitor phase is an iterative activity that provides the tools and techniques, such as intrusion detection systems (IDS) and log analysis to alert on unusual activity.

In Figure 9, the Five-Step Zero Trust Design Methodology as proposed by George Finney (2022) is further explored. Zero trust is integrated into the DevSecOps methodology through the overall security activity spanning each software development life cycle phase and specifically four principles. Additionally, a detailed depiction of UEBA is described in conjunction with Figure 9. These principles are broken down as follows:

1. Define the protect surface: Identify the DAAS elements to protect,
2. Map the transaction flows: Zero Trust is a system, and to secure the system, understanding how the network operates is imperative to a successful zero trust deployment,
3. Build a ZTA: The first two steps illuminate the best way to design the ZTA. Each zero trust environment is tailor-made for each protect surface,
4. Create a zero trust policy: Using the Kipling Method, create a zero trust policy effortlessly by answering the following question: Who should be allowed to access a resource? The validated "asserted identity" is defined in the 'Who' statement. This replaces the source internet protocol (IP) address in a traditional firewall rule, and
5. Monitor and maintain the environment: One of the design principles is to inspect and log all traffic, all the way through Layer 7. Telemetry from cloud, network, and endpoint controls can be analyzed using advances in behavioral analytics, machine learning, and artificial intelligence to stop attacks in real-time and improve security posture over the long term. (Finney, 2022, 166)

Lastly, UEBA works by gathering large amounts of information from multiple sources like logs, network traffic user activities, active directory, defense-in-depth systems, and account management systems (Fortinet 2023). Once the data is collected, the UEBA system creates a profile of normal behavior for each user and entity based on historical data. This profile is updated regularly as new information becomes available (Gurucul 2021). The system continuously monitors and analyzes ongoing user and entity activities in real-time, looking for unusual or abnormal patterns that deviate from the established norms. UEBA also considers additional contextual information when analyzing behaviors, such as the day and time of the activity. When the system detects significant anomalies or potential threats, it generates alerts for security analysts. These alerts help security teams quickly respond to and investigate potential security incidents, enabling them to take appropriate action before any harm is done. UEBA solutions often employ machine learning algorithms to adapt and improve over time, refining their behavioral models and becoming more accurate in identifying threats.

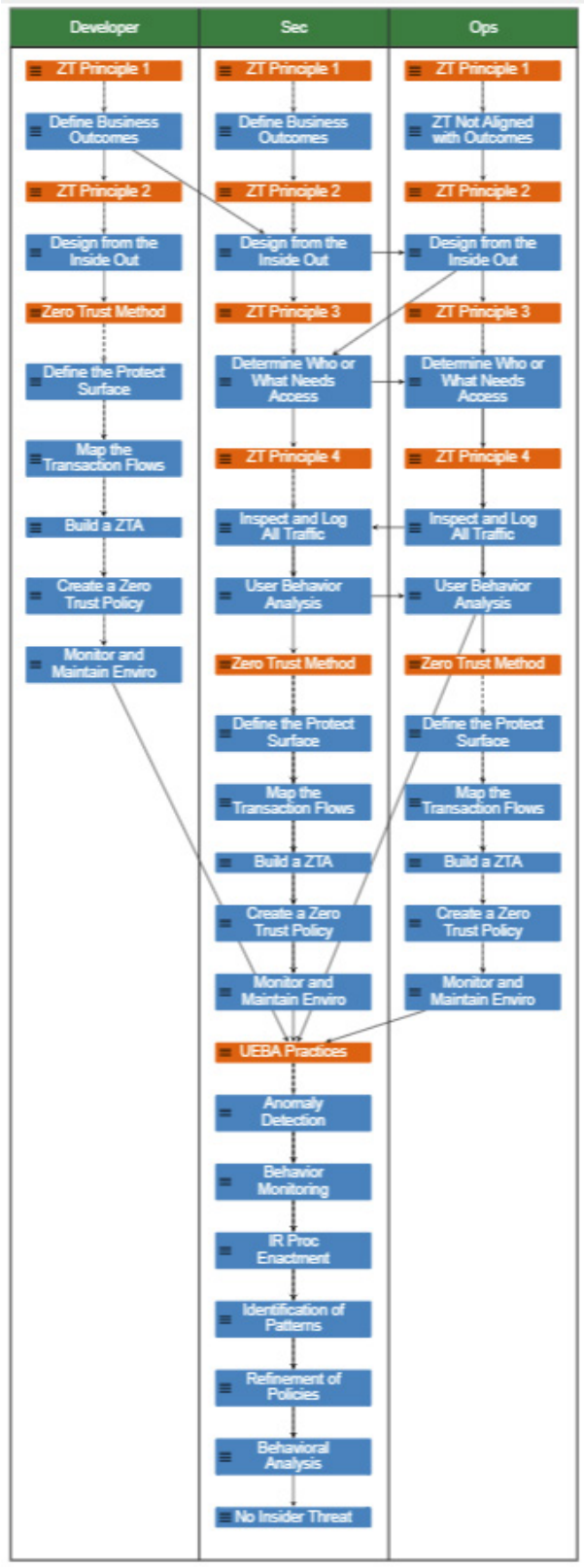


Figure 9. Zero Trust and UEBA Model

C. INTEGRATED PROCESS MODEL ANALYSIS

Combining DevSecOps, zero trust, and UEBA creates a robust and comprehensive cybersecurity approach that offers numerous benefits. Our model and analysis show that the federal government can significantly enhance their DevSecOps security posture and enhance organization's ability to detect and mitigate malicious insider threats within their network. When analyzing and comparing the proposed models of Figures 7, 8, and 9 to the prior baseline models of Figures 5 and 6, the findings of our research are:

1. Improved Security Design: The proposed integrated model provides a multi-layered security approach that addresses various aspects of cybersecurity. DevSecOps integrates security throughout the software development life cycle, zero trust enforces strict access controls, and UEBA monitors user behavior for anomalies. This layered approach enhances the overall security posture of the organization and instills rigorous "Defense in Depth" in the development process.

2. Proactive Threat Detection: Integrating UEBA practices into software development ensures DOD systems are continuously monitoring user and entity behaviors to detect suspicious activities. By analyzing behavioral patterns, potential insider threats, compromised accounts, and other unusual activities that traditional security measures may miss are identified. This proactive threat detection posture helps to mitigate security risks before they escalate. For instance, the April 2023 Massachusetts Air National Guard data leak (Perez 2023) could potentially have been avoided if these proactive measures had been in place. Instead, our national security was put at risk by a malicious insider that intentionally sought to cause harm for personal gain.

3. Improved Incident Response: The proposed approach allows for faster and more effective incident response. When a security incident is detected, DevSecOps practices facilitate rapid identification and mitigation. Additionally, incorporating zero trust principles limits lateral movement,

and UEBA provides valuable insights into the incident's scope and impact, enabling a coordinated and targeted response.

4. Reduced Attack Surface: Zero trust principles reduce the attack surface by verifying users and devices before granting access to resources thereby preventing unauthorized access to critical systems, applications, and data. This approach limits attackers' ability to identify and access valuable data while also limiting their ability to move laterally across the network.

5. Continuous Security Integration: DevSecOps promotes the seamless integration of security practices throughout the software development life cycle. By incorporating security measures from the early stages of development, vulnerabilities can be identified and addressed early on, reducing the likelihood of security flaws in the final product while also ensuring a large solution tradeoff design space and reducing cost.

6. Adaptive Security Policies: Zero trust and UEBA support adaptive security policies based on real-time risk assessments. Access controls can be dynamically adjusted based on user behavior, device health, and other contextual factors. This flexibility allows organizations to maintain good security practices without hindering productivity.

7. Compliance and Auditing: The combined approach aids in meeting various compliance requirements, where DevSecOps ensures security is built into software applications. Furthermore, zero trust aligns with data protection regulations and UEBA provides detailed user activity logs for auditing purposes.

8. Data Protection: Zero trust principles and UEBA contribute to improved data protection. With zero trust, sensitive data is compartmentalized and only accessible to authorized users. UEBA helps detect unauthorized access or unusual data traffic patterns, enhancing data loss prevention capabilities.

9. Holistic Security Monitoring: Integrating DevSecOps, zero trust, and UEBA creates a comprehensive security monitoring environment. These approaches generate valuable data for security analysts to identify potential threats and make more well-informed decisions to strengthen security defenses.

Next, Table 2 presents a detailed listing of steps taken to detect and mitigate malicious insiders based on our research findings. The intent of Table 2 is to help the reader understand our results and how these benefits might be realized within a protected network.

Table 2. Detection and Mitigation of Insider Threats with the Integrated Zero Trust and UEBA Model

Step	Description	Outcome
1	Zero Trust Architecture (ZTA) Implementation	All users and devices must authenticate and authorize before accessing any resources on the network.
2	Identity & Access Management	Enforces strict access controls based on user roles and responsibilities. Users are granted minimum level of access necessary to perform their tasks.
3	Strong Authentication	Multi-factor authentication (MFA) required for all users accessing the network. Adds an extra layer of security; ensures only authorized individuals can gain access.
4	Micro-Segmentation	Segment the network into smaller zones with their own access controls. Limits lateral movement within the network; prevents insider from accessing sensitive data.
5	UEBA	Monitor user behavior for deviations from normal behavior patterns (unusual data access or privilege escalation); trigger alerts.
6	Continuous Monitoring	Constantly monitor network traffic, user activities, and system behaviors. Identifies anomalies and potential threats (such as unusual login locations, access to sensitive data at odd hours, repeated failed login attempts, or sudden spikes in data transfer).

Step	Description	Outcome
7	Data Loss Prevention	Monitor data flows to prevent unauthorized transfer of sensitive data; detects insider attempting to exfiltrate data.
8	Just-in-Time Access	Provides temporary access for users only when needed. Minimized window of opportunity for malicious insider to perform unauthorized actions.
9	Privileged Access Management	Grant temporary, controlled access to privileged accounts only when required; monitor all activities.
10	Insider Threat Programs	Encourages reporting of suspicious activities; defines procedures for handling reports.
11	Security Information and Event Management (SIEM)	Aggregate and analyze security event data to a central repository; aids in real-time detection and incident response.
12	Incident Response Plan	Outline steps to take in case of an insider threat incident, such as isolating affected systems, gathering evidence, and notifying appropriate parties.
13	Red Teaming & Penetration Testing	Simulate insider threat scenarios and identify weaknesses in the ZTA.
14	Ongoing Review & Improvement	Refine ZTA based on lessons learned, emerging threats, and changes in the organization's infrastructure.

D. DETECTION OF EMERGENT BEHAVIORS THROUGH MODELING IN MONTEREY PHOENIX

In this section, emergent behavior analysis of the unified DevSecOps process is performed using MP. Emergent behavior refers to complex patterns or behaviors that arise from interactions between simpler activities or entities. These behaviors can be expected or unexpected, wanted or unwanted, helpful, or harmful making modeling and detection essential for understanding, controlling, or mitigating effects. Performing analysis and modeling of these often elusive behaviors identifies new risks, especially for cybersecurity of complex software systems.

Through modeling the DevSecOps process and building the follow-on DevSecOps, zero trust, and UEBA models, unexpected and unwanted emergent behaviors were detected. These emergent behaviors uncovered uncertainties and unstated assumptions that led to surprising and undesired behaviors. The MP model produces event traces that simulate the DevSecOps process behaviors. These event traces are visually analyzed to detect emergent patterns, anomalies, and/or unexpected outcomes. The goal of this analysis is to mitigate (or manage) emergent behaviors that are unexpected and unacceptable and to use the knowledge gained from modeling to implement well understood behaviors thereby reducing overall risk.

Shown in Figure 10 is the DevSecOps event trace which detected an unsafe emergent behavior. This analysis reveals a situation in which the Developer can transition to Release and Deliver prior to the completion of the Security activities. However, since it is essential to perform security activities, such as finding security vulnerabilities early in the software development life cycle process, this finding is undesirable and ultimately unacceptable when developing critical software applications. Allowing security activities to be skipped could result in unknown software vulnerabilities making their way into weapon systems and could cause great damage or harm to the overall strategic mission.

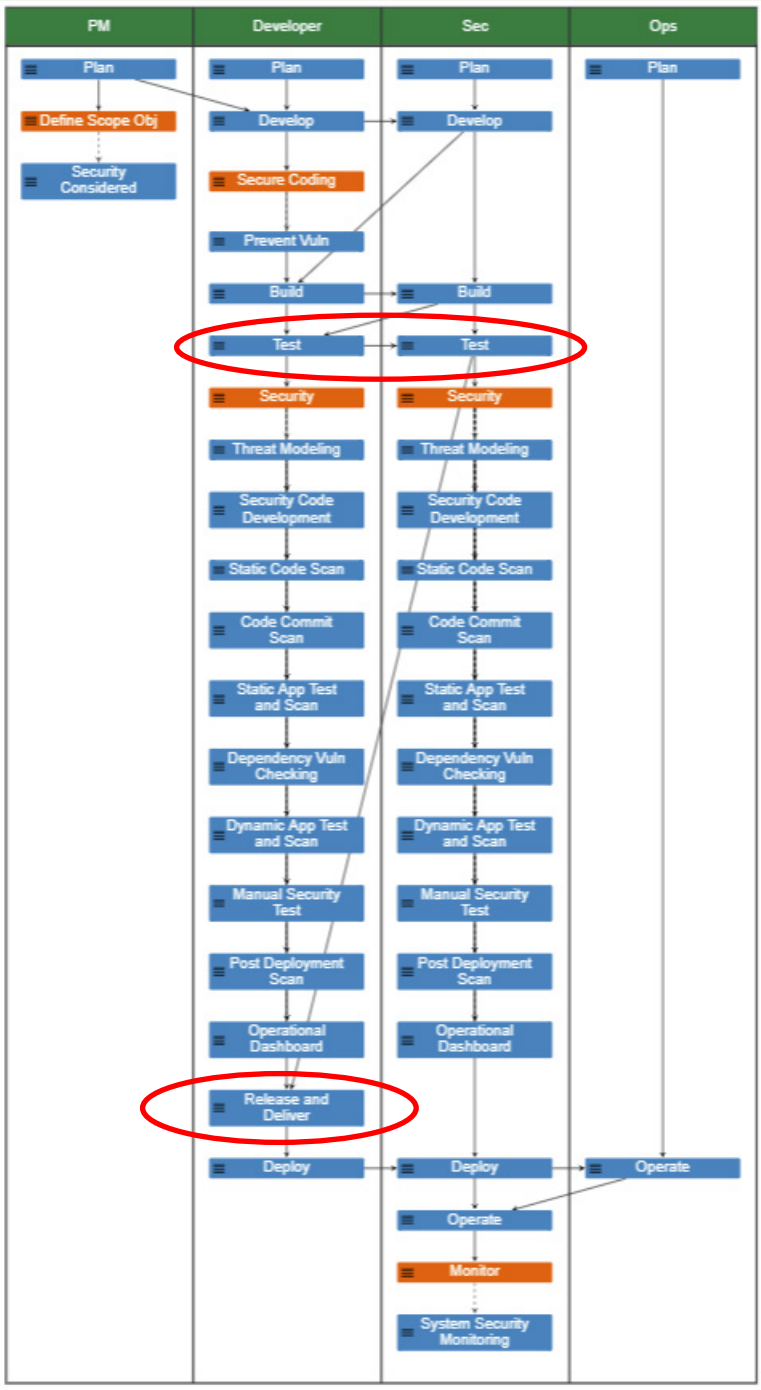


Figure 10. DevSecOps Model – Emergent Behavior

By composing behaviors from individual entities within the process, along with rules for their related interactions, emergent behaviors can be detected. Emergent behavior analysis is key to detecting when processes fail to follow the intended rules, relationships,

and dependencies applying to each of these interactions. See Appendix A for an in-depth explanation of this emergent behavior.

E. RESEARCH FINDINGS

The findings of this research suggest that incorporating zero trust principles and UEBA practices into the DevSecOps methodology results in significant enhancements throughout the software development life cycle. These improvements include:

- A stronger security posture by integrating security into the development process.
- Early detection of anomalies and threats reduces the time attackers can exploit vulnerabilities.
- Strict access controls and continuous monitoring help to reduce the attack surface.
- Automated responses to threats enable proactive incident response, leading to faster incident resolution and containment.
- Implementation of access controls and behavior analysis contributes to data protection and security compliance.
- UEBA provides valuable insights into user and entity behavior, which can be used to inform security measures and improvements.
- Continuous improvement is possible by refining security measures based on evolving threats and user behaviors.

This research concludes that adopting zero trust principles can improve the DOD's mission by modernizing technology, refining security processes, and enhancing operational performance. The outcomes of this research include increased collaboration between developers, security teams, and operators, as all are important contributors to rethinking how infrastructure can be utilized to implement security-by-design more effectively and efficiently. The proposed process model helps protect data and minimizes

threats to national security. To ensure comprehensive management and monitoring of identity-based risks and unknown threats, it is recommended that DOD Components incorporate zero trust principles and UEBA practices consistently.

This research also highlights the necessity of thoroughly understanding processes, precise modeling, and meticulous validation for effective detection of unwanted or unknown emergent behaviors. This is crucial when applying science and engineering to mission-critical defense systems, particularly DOD weapon systems fit with munitions.

V. CONCLUSIONS

In today's work of ubiquitous connectivity there is no clear cybersecurity perimeter, (Gurucul 2021). Adding to the complexity of security architectures is the abundance of various computing devices and emerging technologies, such as Internet of Things (IoT) devices, virtualized systems, automation, artificial intelligence (AI), and ML. These modern complexities make protecting modern networks exclusively challenging (Gurucul 2021). However, the implementation of zero trust and UEBA provide an effective means for architecting and implementing more secure systems, networks, and applications. "Zero trust is based on the idea that trust is not a right, it is a privilege, granted individually on a one-time basis, that provides certain services based on that trust" (Gurucul 2021). Complementing zero trust, UEBA is an essential tool to further strengthen network security. UEBA helps organizations detect insider threats, credential misuse, data breaches, and other unauthorized activities, ultimately reducing the risk of data breaches and security incidents. Thus, zero trust and UEBA work together to vastly improve an organization's security posture detecting threats that traditional security measures tend to overlook.

A. SUMMARY OF WORK

To integrate zero trust security principles into the DevSecOps methodology, requires security to be properly embedded throughout all the software development life cycle phases. During the Plan phase, identifying critical users, devices, and application roles and responsibilities enables strict access controls to be correctly implemented and identifies the location of critical assets and data. During the Develop phase, implementing security controls and authentication mechanisms for data protection in application code is essential. While in the Testing phase, traditional quality assurance testing is conducted and should be supplemented with more rigorous security testing such as static code analysis testing, software scanning, and penetration testing. In the Deploy phase, the use of just-in-time access and limited privilege access must be judiciously granted to ensure minimal changes take place and the deployment pipeline is continuously monitored to ensure only users essential to the deploy activities carry them out. Likewise, data must be encrypted both in transit and at rest to ensure sensitive data

remains confidential. During the Operate phase, micro-segmentation should be carefully applied to limit any lateral movement by malicious insiders. Finally, the Monitor phase continuously monitors all entities and ensures they are verified and authenticated throughout the DevSecOps software development life cycle. This ensures suspicious and compromised users and entities become quarantined to prevent further damage across the network.

In conjunction with zero trust, UEBA solutions prove useful in detecting insider threats by identifying deviations from normal behavior. Contextual analysis allows for understanding users and entities roles, responsibilities, access history, and interactions. UEBA detects anomalies such as unusual patterns that could indicate unauthorized data access. Through continuous monitoring, user and entity behavior is monitored and updated in real-time throughout interactions with the systems, related entities, data, and other contextual information. For example, UEBA compliments the zero trust principle of Access Control by looking at frequent access to sensitive resources and unusual login times.

While developing a model that includes DevSecOps, zero trust, and UEBA may require careful planning, collaboration between teams, and investment in appropriate tools, the advantages of this integrated approach far outweigh the benefits of using each approach in isolation. Combining UEBA with zero trust enhances the ability to identify anomalous activities and limit damage caused by security breaches from malicious insiders (or compromised accounts). The result is a robust and dynamic security ecosystem capable of defending against a wide range of cyber threats.

Finally, analyzing and comparing behavior models made it clear that zero trust principles and UEBA practices greatly enhance the DevSecOps methodology. By implementing strict access control, multi-factor authentication, micro-segmentation, and detection of unusual behaviors, it is possible to prevent malicious insiders from causing serious damage.

B. SUMMARY OF IMPROVEMENTS TO DEVSECOPS

Integrating zero trust principles and UEBA practices into the DevSecOps methodology results in several significant improvements across the software development life cycle. Zero trust principles provide many advantages in protecting against malicious attacks. First, zero trust enables an enhanced security posture by embedding security throughout the development

process. It reduces the attack surface by applying strict access controls and continuous monitoring where security measures are continually refined based on evolving threats. Lastly, zero trust provides a comprehensive approach that enhances the security and resilience of software development while promoting collaboration and efficiency.

Table 3 lists the zero trust improvements in blue with the corresponding UEBA improvements directly underneath.

Table 3. Integrated DevSecOps Model Improvements

Enhanced Security Posture
• Anomaly Detection
Early Threat Detection
• Anomalies in User Behavior, Access Patterns, System Interactions
Reduced Attack Surface
• Real-time Monitoring
Lateral Movement Mitigation
• Entity Behavior Monitoring
Proactive Incident Response
• Access Revocation • Isolation of Compromised Entities • Trigger Incident Response Procedures
Data Protection & Compliance
• Data Breach Detection • Unauthorized Access Attempts
Behavioral Insights
• Identification of Patterns
Collaboration & Transparency
• Shared Insights into Security Incidents
Continuous Monitoring & Improvement
• Refinement of Access Controls, • Policies, • Behavioral Analytics
Efficient Compliance Audits
• Behavior Analysis
Adaptive Access Controls
• Based on Real-time Behavioral Analysis

C. RECOMMENDATIONS FOR FUTURE WORK

It is recommended that a thorough strategy be developed to integrate zero trust principles and UEBA practices into the DevSecOps methodology to provide comprehensive security improvements that safeguard against a wide range of threats. Building this strategy provides a holistic approach to enhance early detection, incident response, and the overall system security posture while promoting collaboration and continuous enhancement of security measures across all phases of the software development life cycle. By proceeding with this effort, the federal government can create a security-focused development culture that continuously monitors for insider threats and responds to incidents promptly.

With a continuously growing threat landscape, work to improve predictive analytics must continue as a valuable tool to predict, prevent, and mitigate advanced cyber threats. Predictive analytics uses advanced technologies such as AI, ML, and big data to analyze historical and real-time data. Predictive analytics is extremely beneficial in forecasting potential cyber threats before they happen, which helps organizations take proactive measures to protect users, devices, and software (Frąckiewicz 2023).

Based on UEBA insights, researching how to implement advanced threat hunting capabilities that allow DevSecOps teams to search for indicators of compromise and potential insider threats is also extremely beneficial (Chakraborty and T N 2022).

Finally, given the United States government's zero trust mandate, a collaborative effort with industry partners and security organizations to share insights, best practices, and lessons learned in integrating zero trust and UEBA effectively into the DevSecOps methodology should be started. This would assist in refining zero trust strategies, architectures, implementations, and understanding costs associated with meeting DOD's zero trust requirements for cybersecurity.

APPENDIX A. VERIFICATION AND VALIDATION OF BEHAVIOR MODELS

A. VERIFICATION

Verification is an essential process for ensuring that the model accurately reflects the process and complies with the specified requirements. This process helps to detect and correct any errors, ensuring the model's reliability and instilling confidence in the simulation or analysis results.

1. Monterey Phoenix Modeling Language

One way this verification is accomplished is by describing the process behavior captured in the requirements in the MP modeling language. This involves using the process as defined in Figure 4 as the source for process requirements and translating the behaviors within those process requirements into models that represent the behavior as was shown in Figures 6, 8, and 9. By specifying the entities to be used (i.e., an event), their relationships (i.e., precedence and inclusion) and a behavior description (i.e., concurrent, alternate, optional, and loops of zero or more or one or more iterations) MP achieves a formal description that is executable and machine-readable (Giammarco 2022, 4–5). Figures 11 and 12 provide a conceptual description of behavior concepts possible to represent as an event, and basic relations among the events used in MP.

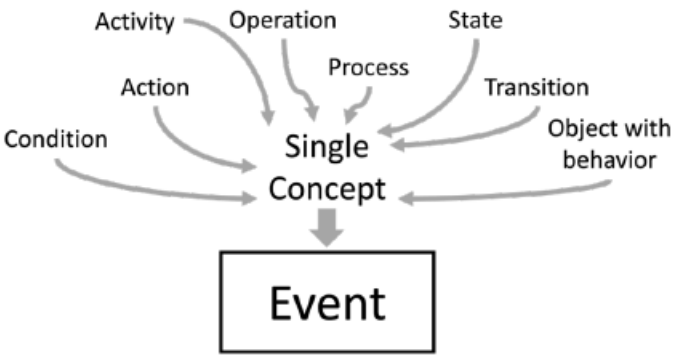


Figure 11. Definition of Event. Source: Giammarco (2022, 4).

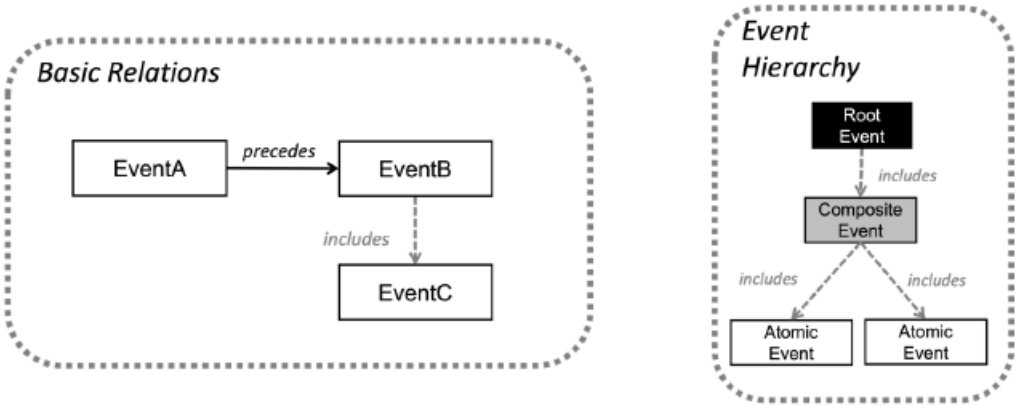


Figure 12. Definition of Relations – Precedence and Inclusion. Source: Giammarco (2022, 5)

2. Verification Steps

To ensure the accuracy and reliability of the models created for this research, I drew on my professional experience in software development and cybersecurity, which gave me a thorough understanding of the DevSecOps process and the various activities that take place during each phase. The knowledge I had acquired through training, reading, and studying DevSecOps models, as well as through practical experience on real-world projects, provided valuable insights that I was able to incorporate into this study. The models used in this research were developed and refined through multiple review cycles with the primary advisor to ensure an accurate representation of the process and behaviors. While analyzing the findings, it became clear that zero trust principles and UEBA practices greatly enhance the DevSecOps methodology. By implementing strict access control, MFA, micro-segmentation, and detection of unusual behaviors, it is possible to prevent malicious insiders from causing serious damage. During the Monitor phase, Figure 13 depicts the iterative process behavior and UEBA practices that feed into zero trust principles.

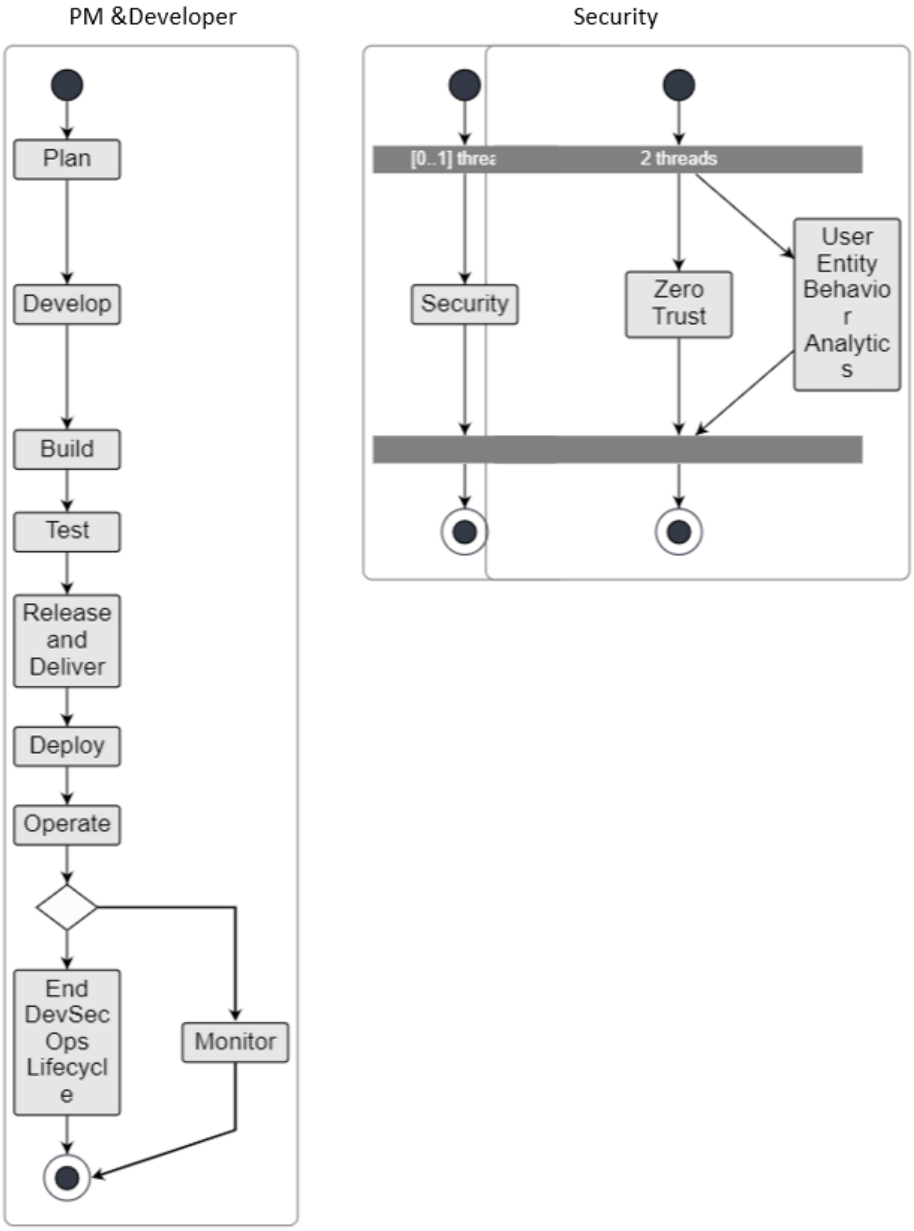


Figure 13. Models of Behavior Program Manager and Developer and Security

B. VALIDATION

Validation follows verification where the model was executed to provide a precise and unambiguous representation of process behavior so that real-world stories could be assessed to determine predictive capabilities. By executing the behavior models, a determination was made as to whether the model adhered to logical behavior, requirements,

and other expectations (undocumented requirements). This helped identify any discrepancies or unexpected emergent behaviors early in the development process.

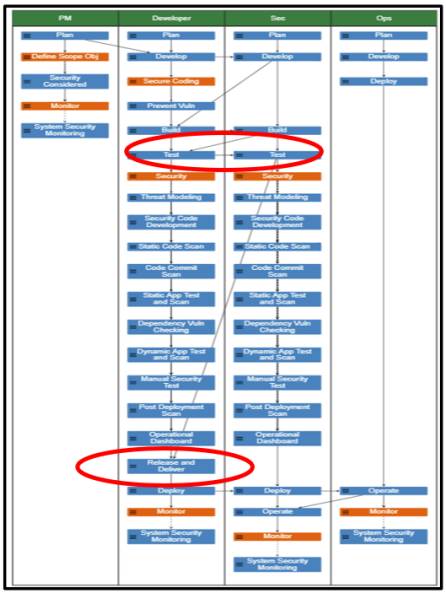
1. Unexpected Emergent Behavior

Once the behavior model was generated, the next step was human inspection of the set of event traces for possible mistakes that were accepted without being challenged in the model captured in Figure 10 (Giammarco 2022, 15). "The beliefs determined from this inspection process to be false were the point in which storytelling was used to assist in challenging each belief" (Giammarco 2022, 15–16).

Through the generation of stories, "the emergent behavior seen in the trace in Figure 14 was explained or validated to be a strong (unexpected) and negative (unacceptable) emergence" (Giammarco 2022, 17–18).

Story: The developer within the DevSecOps software development process has developed code, performed a build, and proceeded to test. However, the developer transitioned to Release and Deliver before completing the Security activities. This resulted in unknown software vulnerabilities making their way into an integrated weapon system, allowing adversaries to exploit the vulnerabilities through cyber attacks causing great damage to the overall strategic mission.

This situation resulted in additional requirements as follows:
1) Removal of the direct transition to Release and Deliver from Test.
2) Enable a process flow for the developer to remain in the Test phase until security testing is complete before transitioning to Release and Deliver.



The developer transitions directly from the Test phase, skipping essential security activities, to the Release and Deliver phase of the DevSecOps software development process.

Figure 14. Unexpected Emergent Behavior

APPENDIX B. MONTEREY PHOENIX MODELS CODE

A. BASELINE EXECUTABLE DEVSECOPS MODEL

```
SCHEMA DevSecOps

ROOT PM: Plan Define_Scope_Obj Monitor;

ROOT Developer: Plan Develop Secure_Coding
        Build Test Security Release_and_Deliver Deploy Monitor;

ROOT Sec: Plan Develop Build Test Security Deploy Operate Monitor;

ROOT Ops: Plan Develop Deploy Operate Monitor;

Define_Scope_Obj: (Security_Considered | Sw_Security_at_Risk);

Secure_Coding: (Prevent_Vuln | Vuln_Detected);

Vuln_Detected: (Patch | Risk_of_Exploit);

Monitor:
        System_Security_Monitoring;

Security:
        Threat_Modeling
        Security_Code_Development
        Static_Code_Scan
        Code_Commit_Scan
        Static_App_Test_and_Scan
        Dependency_Vuln_Checking
        Dynamic_App_Test_and_Scan
        Manual_Security_Test
        Post_Deployment_Scan
```

```
        Operational_Dashboard;
/* Interactions */
COORDINATE $a1: Plan FROM PM,
                $a2: Develop FROM Developer
                        DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Develop FROM Developer,
                $a2: Develop FROM Sec
                        DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Develop FROM Sec,
                $a2: Build FROM Developer
                        DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Build FROM Developer,
                $a2: Build FROM Sec
                        DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Build FROM Sec,
                $a2: Test FROM Developer
                        DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Test FROM Developer,
                $a2: Test FROM Sec
                        DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Test FROM Sec,
                $a2: Security FROM Developer
                        DO ADD $a1 PRECEDES $a2; OD ;
```

COORDINATE $a1: Security **FROM** Developer,

$a2: Release_and_Deliver **FROM** Developer

DO ADD $a1 **PRECEDES** $a2; **OD** ;

COORDINATE $a1: Release_and_Deliver **FROM** Developer,

$a2: Deploy **FROM** Developer

DO ADD $a1 **PRECEDES** $a2; **OD** ;

COORDINATE $a1: Deploy **FROM** Developer,

$a2: Deploy **FROM** Sec

DO ADD $a1 **PRECEDES** $a2; **OD** ;

COORDINATE $a1: Deploy **FROM** Sec,

$a2: Operate **FROM** Ops

DO ADD $a1 **PRECEDES** $a2; **OD** ;

COORDINATE $a1: Operate **FROM** Ops,

$a2: Operate **FROM** Sec

DO ADD $a1 **PRECEDES** $a2; **OD** ;

COORDINATE $a1: Operate **FROM** Sec,

$a2: Monitor **FROM** Sec

DO ADD $a1 **PRECEDES** $a2; **OD** ;

GLOBAL

SHOW ACTIVITY DIAGRAM PM, Developer, Sec, Ops;

B. THE INTEGRATED MODEL

SCHEMA DevSecOps_ZT_UEBA

ROOT PM: Plan Define_Scope_Obj Monitor;

ROOT Developer: Plan Develop Secure_Coding

Build Test Security Release_and_Deliver Deploy Monitor

Zero_Trust_Principles Zero_Trust_Method;

ROOT Sec: Plan Develop Build Test Security Deploy Operate Monitor

Zero_Trust_Principles Zero_Trust_Method UEBA_Practices;

ROOT Ops: Plan Develop Security Deploy Operate Monitor

Zero_Trust_Principles Zero_Trust_Method UEBA_Practices;

Define_Scope_Obj: (Security_Considered | Sw_Security_at_Risk);

Secure_Coding: (Prevent_Vuln | Vuln_Detected);

Vuln_Detected: (Patch | Risk_of_Exploit);

Monitor:

System_Security_Monitoring;

Security:

Threat_Modeling

Security_Code_Development

Static_Code_Scan

Code_Commit_Scan

Static_App_Test_and_Scan

Dependency_Vuln_Checking

Dynamic_App_Test_and_Scan

Manual_Security_Test

Post_Deployment_Scan

Operational_Dashboard;

Zero_Trust_Principles:

Define_Business_Outcomes

Design_from_the_Inside_Out

Determine_Who_or_What_Needs_Access

Inspect_and_Log_All_Traffic;

Zero_Trust_Method:

Define_the_Protect_Surface

Map_the_Transaction_Flows

Build_a_ZTA

Create_a_Zero_Trust_Policy

Monitor_and_Maintain_Enviro;

UEBA_Practices:

Anomaly_Detection

Behavior_Monitoring

IR_Proc_Enactment

Identifiaction_of_Patterns

Refinement_of_Policies

Behavioral_Analysis;

/* Interactions */

COORDINATE $a1: Plan **FROM** PM,

$a2: Develop **FROM** Developer

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: Develop **FROM** Developer,

$a2: Develop **FROM** Sec

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: Develop **FROM** Sec,

```
        $a2: Build FROM Developer
                DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Build FROM Developer,
        $a2: Build FROM Sec
                DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Build FROM Sec,
        $a2: Test FROM Developer
                DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Test FROM Developer,
        $a2: Test FROM Sec
                DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Test FROM Sec,
        $a2: Security FROM Developer
                DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Security FROM Developer,
        $a2: Release_and_Deliver FROM Developer
                DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Release_and_Deliver FROM Developer,
        $a2: Deploy FROM Developer
                DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Deploy FROM Developer,
        $a2: Deploy FROM Sec
                DO ADD $a1 PRECEDES $a2; OD ;
COORDINATE $a1: Deploy FROM Sec,
```

```
            $a2: Operate FROM Ops
                    DO ADD $a1 PRECEDES $a2; OD ;
        COORDINATE $a1: Operate FROM Ops,
            $a2: Operate FROM Sec
                    DO ADD $a1 PRECEDES $a2; OD ;
        COORDINATE $a1: Operate FROM Sec,
            $a2: Monitor FROM Sec
                    DO ADD $a1 PRECEDES $a2; OD ;
        GLOBAL
        SHOW ACTIVITY DIAGRAM PM, Developer, Sec, Ops;
```

C. ZERO TRUST AND UEBA MODEL

```
        SCHEMA Zero_Trust_UEBA
        ROOT Developer: ZT_Principle_1 ZT_Principle_2 Zero_Trust_Method;
        ROOT Sec: ZT_Principle_1 ZT_Principle_2 ZT_Principle_3
            ZT_Principle_4 Zero_Trust_Method
            (*      UEBA_Practices
                    Insider_Threat
            *)
            UEBA_Practices
            No_Insider_Threat;
        ROOT Ops: ZT_Principle_1 ZT_Principle_2 ZT_Principle_3
            ZT_Principle_4 Zero_Trust_Method;
        ZT_Principle_1:
            (Define_Business_Outcomes | ZT_Not_Aligned_with_Outcomes);
```

ZT_Principle_2:

(Design_from_the_Inside_Out | Protection_Only_at_Perimeter);

ZT_Principle_3:

(Determine_Who_or_What_Needs_Access

| Least_Privilege_Not_Implemented);

ZT_Principle_4:

(Inspect_and_Log_All_Traffic (User_Behavior_Analysis |

Anomalies_Undetected) | Logs_Not_Collected);

Zero_Trust_Method:

Define_the_Protect_Surface

Map_the_Transaction_Flows

Build_a_ZTA

Create_a_Zero_Trust_Policy

Monitor_and_Maintain_Enviro;

UEBA_Practices:

Anomaly_Detection

Behavior_Monitoring

IR_Proc_Enactment

Identification_of_Patterns

Refinement_of_Policies

Behavioral_Analysis;

/* Interactions */

COORDINATE $a1: Define_Business_Outcomes **FROM** Developer,

$a2: Design_from_the_Inside_Out **FROM** Sec

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: Design_from_the_Inside_Out **FROM** Sec,

$a2: Design_from_the_Inside_Out **FROM** Ops

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: Design_from_the_Inside_Out **FROM** Ops,

$a2: Determine_Who_or_What_Needs_Access **FROM** Sec

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: Determine_Who_or_What_Needs_Access **FROM** Sec,

$a2: Determine_Who_or_What_Needs_Access **FROM** Ops

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: Determine_Who_or_What_Needs_Access **FROM** Ops,

$a2: Inspect_and_Log_All_Traffic **FROM** Ops

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: Inspect_and_Log_All_Traffic **FROM** Ops,

$a2: Inspect_and_Log_All_Traffic **FROM** Sec

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: Inspect_and_Log_All_Traffic **FROM** Sec,

$a2: User_Behavior_Analysis **FROM** Sec

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: User_Behavior_Analysis **FROM** Sec,

$a2: User_Behavior_Analysis **FROM** Ops

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: User_Behavior_Analysis **FROM** Ops,

$a2: UEBA_Practices **FROM** Sec

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: Monitor_and_Maintain_Enviro **FROM** Developer,

$a2: UEBA_Practices **FROM** Sec

DO ADD $a1 **PRECEDES** $a2; **OD ;**

COORDINATE $a1: Monitor_and_Maintain_Enviro **FROM** Ops,

$a2: UEBA_Practices **FROM** Sec

DO ADD $a1 **PRECEDES** $a2; **OD ;**

GLOBAL

SHOW ACTIVITY DIAGRAM Developer, Sec, Ops;

LIST OF REFERENCES

Avi. 2020. "An Introduction to DevSecOps for Beginners." Geekflare. October 2, 2020. https://geekflare.com/devsecops-introduction/.

Boyens, Jon, Angela Smith, Nadya Bartol, Kris Winkler, Alex Holbrook, and Matthew Fallon. "Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations." National Institute of Standards and Technology, October 27, 2021. https://doi.org/10.6028/NIST.SP.800-161r1-draft2.

Brook, Chris. 2020. "What Is User and Entity Behavior Analytics? A Definition of UEBA, Benefits, How It Works, and More." Digital Guardian. May 31, 2020. https://digitalguardian.com/blog/what-user-and-entity-behavior-analytics-definition-ueba-benefits-how-it-works-and-more.

Chakraborty, Sankha, and Nisha T N. *Next Generation Proactive Cyber Threat Hunting – A Complete Framework*. *AIP Conference Proceedings*. Vol. 2519, 2022. https://doi.org/10.1063/5.0109674.

CISA OCE. "Cost of a Cyber Incident: Systematic Review and Cross-Validation," October 26, 2020. https://www.cisa.gov/sites/default/files/publications/CISA-OCE_Cost_of_Cyber_Incidents_Study-FINAL_508.pdf.

Cronk, Terri. "Summit Highlights DOD's Cybersecurity Initiatives, Challenges." *DOD News*, October 8, 2021. https://www.defense.gov/News/News-Stories/Article/Article/2806264/summit-highlights-dods-cybersecurity-initiatives-challenges/.

Department of Defense. "DevSecOps Fundamentals Guidebook: DevSecOps Tools & Activities." Department of Defense Office of Republication and Security Review, March 2021. https://dodcio.defense.gov/Portals/0/Documents/Library/DevSecOpsTools-ActivitiesGuidebook.pdf.

———. 2022. *DOD Zero Trust Strategy*. Washington, DC: Department of Defense. https://federalnewsnetwork.com/wp-content/uploads/2022/11/DOD_Zero_Trust_Strategy_v1.0_Final_508-date-cover_20221116.pdf.

Finney, George. 2022. *Project Zero Trust*. Indianapolis: John Wiley and Sons.

Fortinet. 2023. "What Is User Entity and Behavior Analytics (UEBA)?," https://www.fortinet.com/resources/cyberglossary/what-is-ueba.

Frąckiewicz, Marcin. 2023. "Combating Cyber Threats with the Power of Predictive Analytics." *TS2 SPACE* (blog). August 27, 2023. https://ts2.space/en/combating-cyber-threats-with-the-power-of-predictive-analytics

Giammarco, Kristin. 2022. "Exposing and Controlling Emergent Behaviors Using Models with Human Reasoning." In *Emergent Behavior in System of Systems Engineering*, pp. 23–61. CRC Press.

Gurucul. 2021. *EBook: ABCs of UEBA*. Los Angeles: Gurucul. Electronic book.

Gurucul. 2022. "Gurucul User Entity Behavior Analytics (UEBA)." July 7, 2022. https://go1.gurucul.com/UEBA-Datasheet?_gl=1*chqh1d*_ga*MTQwNzY0OTI4My4xNzAxNzM5MDk2*_ga_21HFD5BZPC*MTcwMTczOTA5NS4xLjEuMTcwMTczOTExMi40My4wLjA.*_ga_XK6L3BZR7J*MTcwMTczOTA5NS4xLjEuMTcwMTczOTExMi40My4wLjA.

Gurucul. n.d. "Find Insider Threats Before Data Exfiltration." Accessed September 30, 2023. https://gurucul.com/solutions/insider-threat.

Miller, Andrew, Ronald Giachetti, and Douglas Van Bossuyt. 2022. "Challenges of Adopting DevOps for Combat Systems Development Environment." *Defense Acquisition Research Journal* 29 (99): 22–48. https://doi.org/10.22594/dau.21-870.29.01.

Perez, Zamone. 2023. "Alleged Air Force Leaker Shared Intel with Foreign Nationals, FBI Says." August 10, 2023. https://www.armytimes.com/news/your-military/2023/08/10/alleged-air-force-leaker-shared-intel-with-foreign-nationals-fbi-says/.

Red Hat. 2023. "What Is DevSecOps?" March 10, 2023. https://www.redhat.com/en/topics/devops/what-is-devsecops.

Rose, Scott, Oliver Borchert, Stu Mitchell, and Sean Connelly. 2020. *Zero Trust Architecture*. NIST Special Publication 800-207. Gaithersburg, MD: National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207.

Salitin, Manya, and Ali Zolait. "The Role of User Entity Behavior Analytics to Detect Network Attacks in Real Time," 1–5, 2018. https://doi.org/10.1109/3ICT.2018.8855782.

Sanders, Geoffrey, Timothy Morrow, Nathaniel Richmond, and Carol Woody. "Integrating Zero Trust And Devsecops," July 2021. https://apps.dtic.mil/sti/pdfs/AD1145432.pdf.

White House. 2022a. *Department of Defense Software Modernization Strategy*. Washington, DC: White House. https://media.defense.gov/2022/Feb/03/2002932833/-1/-1/1/DEPARTMENT-OF-DEFENSE-SOFTWARE-MODERNIZATION-STRATEGY.PDF.

———. 2022b. “Executive Order On Enhancing Safeguards For United States Signals Intelligence Activities.” October 7, 2022. https://www.whitehouse.gov/briefing-room/presidential-actions/2022/10/07/executive-order-on-enhancing-safeguards-for-united-states-signals-intelligence-activities/.

Woody, Carol. 2021. “A Cybersecurity Engineering Strategy for DevSecOps.” Software Engineering Institute, Carnegie Mellon University. https://resources.sei.cmu.edu/asset_files/Webinar/2021_018_101_740883.pdf.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California