IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 3, 2021, accepted April 23, 2021, date of publication April 30, 2021, date of current version May 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3076918

Security Testing for Naval Ship Combat System Software

CHEOL-GYU YI[1], (Student Member, IEEE), AND YOUNG-GAB KIM[1,2], (Member, IEEE)
[1]Department of Computer and Information Security, Sejong University, Seoul 05006, South Korea
[2]Department of Convergence Engineering for Intelligent Drone, Sejong University, Seoul 05006, South Korea

Corresponding author: Young-Gab Kim (alwaysgabi@sejong.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant 2021R1A2C2012635.

ABSTRACT Military weapon systems are considered as ''system of systems'' (SoS). They comprise various equipment based on computers and networks and have been developed using commercial computing technology for several decades. The state-of-the-art weapon systems are information technology (IT) systems, for example, the cyber-physical system. In particular, the naval combat system, which is a weapon system, is a representative system interconnecting a number of equipments by using commercial computing technology. It is a software-based complex system, which produces and shares information about naval tactical situations by interconnecting the various systems installed on ships or remote platforms. Moreover, it performs tactical combat functions automatically or manually for assigned missions. As the core function and performance of the combat system shift from being hardware-centric to software-centric, cybersecurity threats to software that can affect the combat systems may emerge as a novel issue. The failure of the combat system to perform normal combat functions in an actual naval combat situation owing to cybersecurity issues is a very serious risk to naval operations. However, software security testing is not conducted systematically during system development because the cybersecurity of the combat system is evaluated to be less important than its function and performance, resulting in the development of an insecure and vulnerable combat system against cybersecurity threats. To build a secure combat system against cyberattacks, it is important to derive systematic and practical security testing for the combat system software during system development. This paper analyzes the previous researches on a software security test, characteristics of the combat system software, and guidelines for the software security testing of the Korean military's weapon system development. In addition, it proposes improved software security testing to strengthen the cybersecurity of the combat system based on its characteristics and missions.

INDEX TERMS Weapon system, combat system, software security, security test.

I. INTRODUCTION

For several decades, military weapon systems have been developed using commercial computing technology. Further, their coverage and utilization are increasingly expanding. Recently, state-of-the-art weapon systems, which are information technology (IT) systems, have also been developed such as the cyber-physical system, which is considered to be more software-dependent [1]. The expansion of the integration of heterogeneous systems in weapon systems has increased software complexity and connectivity. Moreover, weapon systems are considered to be complex ''system of systems'' (SoS), considerably complicating the analysis of system security vulnerabilities against cyberattacks [2]. The ''SoS'' comprises a large complex system integrated by the interconnection of independent and preexisting individual systems [3]. According to the ''Weapon system cybersecurity'' reported by the Government Accountability Office (GAO), weapon systems have numerous cyber vulnerabilities similar to those of other types of automated information systems. Moreover, appropriate security controls for cyber vulnerability are insufficient, resulting in cyberattack threats [1], [4].

In particular, the naval combat system, which is one of the weapon systems, is a representative system interconnecting performance and functions of the diverse subsystem of the

The associate editor coordinating the review of this manuscript and approving it for publication was Francesco Mercaldo.

combat system. Furthermore, the usability and importance of software are increasing because the core functions and performance of the combat system transform from being hardware-centric to software-centric [5]. Therefore, a new issue emerges, i.e., security problems in the combat system software can compromise its critical combat performance and function and disrupt the completion of naval missions.

With the development of cyberattack technology, the US navy designed a system called Common Architecture System Assurance (CASA) to deal with cyberattacks on the naval combat system. CASA is specialized security information and event manager. It collects and analyzes cybersecurity-related events from the subsystems of the combat system. Moreover, it reports mission-critical impact or cyber incidents to the watch officer [6].

However, software security testing is not conducted systematically during the development of weapon systems, including the combat system, because the cybersecurity of the combat system is evaluated to be less important than its function and performance. This results in the development of a combat system vulnerable to cyberattacks. Furthermore, in the Korean military, software security testing during system development is not mandatory for the entire weapon system installed on the platform (e.g., naval ship), including the combat system, and is only conducted for the application software of the battle management information system (e.g., C4I system and command and control system) by referencing ''Guideline for software development security'' [7], [8]. This guideline can be a good standard for security testing of the application software of general IT systems. However, it is not suitable for the naval combat system software to perform real-time combat missions in a unique environment different from that associated with general IT systems. In addition, this guideline is mainly described from the developer's point of view and can not cover the entire software that comprises the combat system.

Furthermore, the major problem associated with the combat system software security testing is the lack of a defined systematic and specific security test criteria in system development. To fundamentally reinforce the cybersecurity of the combat system software, specific and practical security testing must be performed, which can be applied to the combat system software in the early stage of system development from the perspective of cybersecurity.

Thus far, various methodology studies have been conducted for software security testing, which contributes to the improvement of security testing. However, software security testing for weapon systems such as combat systems has not been studied due to the closed environments and access restrictions of the combat system itself. As a result, systematic and practical software security testing has not been performed and a robust combat system against cyberattacks was not developed. Thus, combat systems are expected to exhibit various vulnerabilities.

In this paper, in order to resolve this problem, we focused on a systematic approach that can perform a specific and practical security test rather than a theoretical and methodological approach. We propose a more effective and practically applicable software security test by improving the security test derived from a previous study, wherein the categories of software security testing and detailed subitems corresponding to the categories are defined [9].

The following are the contributions of this paper based on a previous study.

1. Establishment of a specific test framework for conducting software security testing based on the characteristics of the combat system software: We defined the categories of security testing that satisfy the security attributes in accordance with the characteristics of the combat system from the perspective of cybersecurity and not from the perspective of developers.

2. Derivation of detailed security test subitems that correspond to the software comprising the combat system software: We derived detailed security test subitems relevant to the categories of security testing and applied them to the software comprising the combat system software.

3. Reflection of security requirements for the design of the combat system software during the combat system development process: It is desirable that software security must be considered during the early stage of system development. We present a direction by which the detailed security test details derived from this paper can be reflected in the design of the combat system software.

4. Development of a combat system software robust against cyberattacks: Systematic security testing performed based on the characteristics of the combat system software removes vulnerabilities in the software in advance and develops a robust combat system against cyberattacks.

The remainder of this paper is organized as follows. Section 2 discusses previous studies related to software security testing and introduces the concept and characteristics of the naval combat system software. Moreover, it discusses the software security testing of the weapon system such as the naval combat system and analyzes its limitations. Section 3 proposes an improved software security testing approach for combat systems. Section 4 evaluates the proposed security testing approach through comparison with papers related to the software security testing discussed in Section 2, the existing software security testing for the Korean military's weapon system, and the Common Criteria (CC). Finally, Section 5 provides the conclusion and future work.

II. RELATED WORK

In this section, we analyze the existing studies on software security testing to understand how security testing is applied to software. In particular, we describe the security testing techniques classified into four categories as defined in the existing study and analyze the existing studies related to the security testing techniques classified above.

Then, we describe the concept and structure of the naval combat system software and the characteristics of each sub software that composes the combat system software.

TABLE 1. Security testing techniques in a secure software development lifecycle.

Security testing techniques	*Description*	*Stage of secure software development lifecycle*
Model-based security testing	Based on requirements and the design model created during analysis and design	Analysis / Design
Code-based testing and static analysis	Based on analysis and on the source code and byte code created	Development
Penetration testing and dynamic analysis	Executed on running systems or production environment from attacker's perspective	Deployment
Security regression testing	Conducted when changes occur in the existing code of software	Maintenance

In addition, we explain the concept and procedure of software security testing when developing weapon systems such as combat systems in the Korean military, and based on this, derive the limitations of software security testing.

A. SOFTWARE SECURITY TESTING

Software security testing is an important process to validate whether the security functions and features of the software are appropriately implemented in accordance with the requirements related to security properties (e.g., confidentiality, integrity, availability, and authentication) and to identify potential security vulnerabilities in the software during software development [10]–[12]. From the perspective of system development projects and businesses, security testing activities ensure the quality of the system and reduce development risk and cost [12].

Software security testing can be divided into two execution types: security functional testing and security vulnerability testing. Through security functional testing, it can be validated whether the security requirements are appropriately achieved in the software. Through security vulnerability testing, cyber vulnerabilities can be identified in the software [13]. Cyber vulnerability is a flaw in system design, implementation, and operation [10]. It may not be discovered at times and will continue to occur. Thus, a malicious attacker compromises the software system by exploiting the vulnerability of the system.

Felderer *et al.* [11] suggested that vulnerabilities can be effectively identified, and the security functionality of the software can be ensured through security testing. The author classified security testing into four categories according to its characteristics with respect to the system development lifecycle, which is presented in Table 1. The categories of security testing are as follows.

- Model-based security testing: Model-based testing (MBT) derives its test case from a system under test (SUT) and/or its environment for software testing [13]. Model-based security testing follows the MBT approach, verifies the software requirements related to security properties, and verifies whether the model has the specified security features [11].
- Code-based testing and static analysis: This technique can be used to analyze and review the source code of the program to manually or automatically identify vulnerabilities in the software during code development. An automated static analysis tool is used to analyze the software code and identify security bugs [14], [15].
- Penetration testing and dynamic analysis: The tester executes this test on a running system and compromises the system by sending a payload from the perspective of a cyber attacker [16]. This testing is similar to a real attack by a third party who possesses insufficient information about the target system. In general, the third-party attacker uses manual and automatic tools to identify vulnerabilities [17]. Fuzz testing is applied to identify potential security vulnerabilities by sending valid or invalid data to a system [18].
- Security regression testing: This technique ensures that any changes in the system after its development do not harm its security [19]. Security regression testing is an important process in the software development lifecycle because source code changes frequently occur due to fixes, patches, and enhancements. When modifying the existing code, the tester should perform security testing to validate whether new security bugs have been introduced [20].

Following, we discuss the previously studied software security testing.

Michele *et al.* [21] proposed a model-based security testing framework for web applications, which is based on the MBT approach combined with knowledge of penetration testing. The proposed framework uses model-checking techniques for conducting an automatic search for detecting possible vulnerabilities in web applications. Security analysts can perform security testing without missing important vulnerabilities checkpoints by using this method. It is also reused in different web applications.

Brucker *et al.* [15] introduced static application security testing (SAST). This is performed by the development team. As the development team integrated SAST into the steps of the software development lifecycle, SAST supported the identification and prevention of vulnerabilities in the program source code.

Valentine *et al.* [22] proposed an automatic penetration testing method to evaluate the security level of the cloud application when considering the cloud application features.

This method uses the knowledge of the application architecture and catalog information, including security-related data (e.g., threat, attack, weakness, and vulnerabilities), and comprises three phases (*preparation*, *scanning*, *and pen-testing*) for testing.

Longoria [23] suggested a prioritization of the security regression test case based on the threat model. The threat model is important for performing security analysis during software development. This model emphasizes the assets that must be protected and the threats to those assets as well as threat mitigation. The author explained how the threat model is applied to prioritize the security test case.

Josip *et al.* [24] introduced planning-based security testing of web applications. They proposed the application of artificial intelligence (AI) in the security testing of web applications, which use automated planning for test suites to test common cross-site scripting and structured query language injection vulnerabilities. In addition, they proposed an approach based on automated planning, which was obtained from classical AI.

Pekaric *et al.* [25] presented security testing techniques for automotive engineering by focusing on the application software and service layer of the Automotive Open System Architecture (AUTOSAR). AUTOSAR is an open and standardized software architecture for electronic control units in the automotive domain. They linked the security testing techniques (MBT, code-based testing, penetration testing and dynamic analysis, regression testing, and risk-based testing) to the AUTOSAR layers and identified and defined the relation between security testing techniques and vehicle lifecycle phases.

Seana *et al.* [26] proposed an approach for security testing of unmanned aerial vehicles (UAVs) based on the MBT method. They developed an approach to building a test suite using a behavioral model, attack model, and mitigation model based on the characteristics of the UAV. Thus, they generated a scalable four-step test suite to identify vulnerabilities in a UAV and prevent cyberattacks.

Olivero *et al.* [27] introduced a security testing approach named Testing for Security in System of Systems (TeSSoS), which satisfied the modeling security requirements and testing security properties in the SoS. They also proposed five sequential steps (SoS discovery, red security requirement specification, blue requirement specification, security implementation, and SoS evaluation and validation) to identify security threats and define security features of the SoS. Particularly, their assessment approach from the perspective of the attacker can facilitate in effectively discovering security vulnerabilities and developing a secure SoS.

As previously described, an approach for software security testing has been mainly studied. However, research on concrete and systematic software security testing that can be practically applied in combat system development is insufficient. Even though studies on security testing of UAVs, which has recently become an issue, have been conducted, research on security testing for complex weapon system software, such as the naval combat systems installed on platforms (e.g., naval ship), is still lacking.

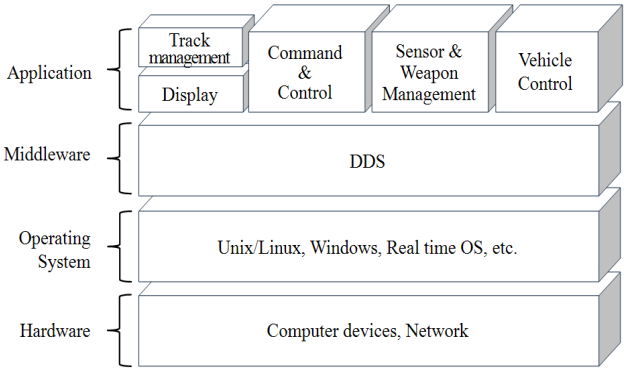

FIGURE 1. Combat system software structure.

B. NAVAL SHIP COMBAT SYSTEM SOFTWARE

The combat system is a complex weapon system based on computers and networks. It comprises various heterogeneous systems, including sensors, weapons, and navigation systems, in a naval ship. It produces and shares information on a tactical situation by networking different systems and automatically conducts target detection and tracking, threat analysis, weapon allocation, engagement, and kill evaluation [28]. The US navy applied the concept of open architecture in naval combat system development. The naval open architecture is a multifaced business and technical strategy for efficient development and maintenance of interoperable systems. Its core principles include modular design, reusable application software, and improved interoperability and performance through cutting-edge technologies [29]. The combat system software is developed according to this concept and includes application software, middleware, and operating system, as presented in Figure 1. The application software at the top performs the actual multiple mission-critical functions, such as track management, command and control, and sensor and weapon management and engagement, for naval combat missions. It is developed based on the modular concept. Therefore, it can easily deal with the possibility of changes according to the assigned mission and specifically required combat functions. In addition, it is installed and run on the computer server and operator console [30].

The middleware of the combat system is present between the application software and operating system to provide common service. Previously, the combat system used military-specific middleware; however, recently, the commercial middleware data distribution service (DDS) standardized by the object management group (OMG) is used. DDS, as a communication middleware, supports real-time communication between the nodes based on the publication–subscription model for real-time processing in a distributed environment [31], [32]. The security standards of the DDS provide data confidentiality and integrity, mutual authentication and access authority of the network participants, and non-repudiation [33], [34]. The operating systems used are

commercial operating systems, such as Linux, Unix, and VxWorks, which is a real-time operating system. The application software design may need to be changed if the configuration settings of the operating system are changed because the software comprising the combat system software are highly connected to each other.

C. WEAPON SYSTEM SOFTWARE SECURITY TESTING

The Korean military applied the ''Manual for the development and management of weapon system software'' [30]. This manual describes the concept and process for the development of weapon system software and provides a specific direction for software testing. The combat system follows this regulation because it is part of the weapon system.

1) CONCEPT AND PROCEDURE

Generally, software security testing is conducted along with the reliability test in the development of weapon system software. The software security and reliability test identifies errors and defects in the weapon system software and proactively eliminates the security weaknesses causing cyberattacks. Moreover, it verifies whether the software meets the specified requirements with technical accuracy and adequacy. As presented in Figure 2, the tests are conducted by the R&D supervisory institution in the ''software implementation'' and ''software integration and test'' of the system development stage. Thereafter, the development test and evaluation (DT&E), as well as operational test and evaluation (OT&E), are conducted by the R&D supervisory institution and military. Although security testing is performed through the above process, its application is limited from the perspective of system integration because these stages are performed in the late process of system development. The occurrence of critical security problems during these stages incurs additional time and money for resolving the issue.

The security test activities are specified through various procedures and documentation associated with weapon system development. The detailed plans, procedures, and results of security testing are described in the software test plan (STP), software test description (STD), and software test report (STR) for managing test traceability. The security test requirements are preferentially and roughly considered during the exploratory development stage. Further, they are reviewed during the ''software requirement analysis'' and ''software structure and critical design'' of the system development stage. Security testing is only applicable to developed software, open software, and autogenerated code and not to the operating system or middleware. The target languages used include C, C++, JAVA, and C# [30].

2) SOFTWARE SECURITY TESTING

Software security testing of the weapon system, which is conducted in accordance with the ''Guideline for software development security'' [7] of the Republic of Korea's Ministry of the Interior and Safety, is applied only to the battle management information system (e.g., C4I system and command and control system) and not to the entire weapon [8]. This guideline was formulated with reference to the common weakness enumeration (CWE), the security weakness provided by the MITRE, and various official announcements related to secure coding and only provides a criterion for verifying secure coding of application software. This guideline is implemented to ensure the secure software of general IT systems (e.g., web system and DBMS) that can deal with cyberattacks by minimizing the security vulnerabilities that can be attributed to developer's mistakes and logical errors in the software development process. It also focuses on technical control for the implementation of security functions to achieve software confidentiality, integrity, and availability and specifies the security design and implementation criteria based on the security requirements identified during software requirement analysis. The security design criteria of software design are defined as 20 detailed security requirements in four categories (input data verification and presentation, security function, error handling, and session control). As presented in Table 2, the security implementation criteria of the software implementation are related to the aforementioned security requirements of the software design. These criteria comprise 47 items in seven categories (input data verification and presentation, security function, time and status, error processing, code error, encapsulation, and application programming interface misuse) [7].

If security testing is to be conducted on the combat system software, the previously mentioned guideline can be applied to combat system software security testing. However, such guideline is only for the secure coding of general IT systems and is not suitable for a combat system in a mission-critical environment. Moreover, the combat system does not provide web and DBMS services.

3) LIMITATION OF SOFTWARE SECURITY TESTING

When a weapon system is developed, software security testing is perceived to be less important than performance or function. Furthermore, conducting security testing on the weapon system, including a combat system, is not mandatory. It is only conducted on the application software of the battle management information system (e.g., C4I system and command and control system). In this regard, security testing of the combat system software has the following limitations.

First, there is no defined systematic and specific strategy for software security testing that can be applied to the combat system software comprising the application software, middleware, and operating system in the system development stage based on the characteristics and environments of the combat system. The software comprising the combat system software is closely linked. Thus, systematic and specific security testing must be conducted during the early stage of system development. Moreover, there is a very close correlation between software security testing and analysis and design. Failure to systematically and comprehensively review the specific

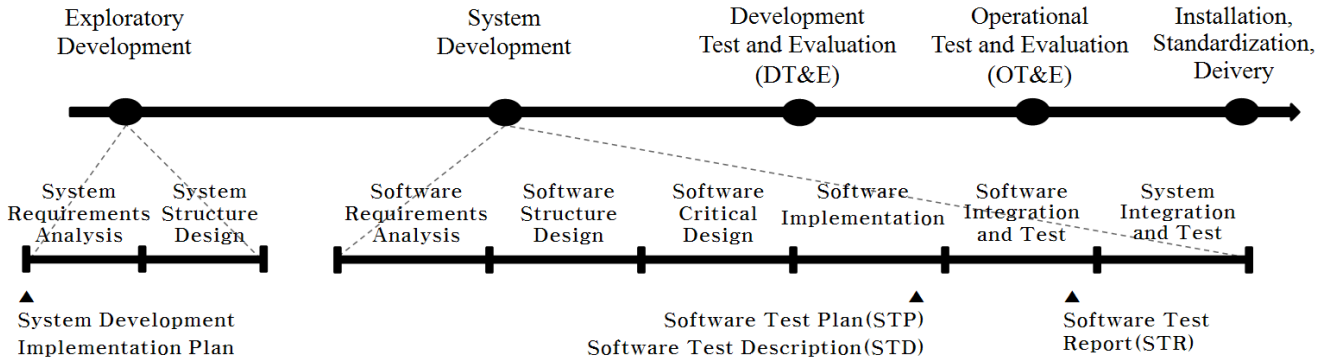

FIGURE 2. Weapon system software development process.

TABLE 2. Criteria for secure software implementation in the software implementation phase.

Category	*Subitems*	*Description*
Input data verification and presentation	• SQL injection • Cross-site scripting • Path traversal and resource injection • Operating system command injection • Unrestricted upload of files with dangerous form • URL redirection to untrusted site • XQuery injection • Xpath injection • LDAP injection • Cross-site request forgery • HTTP response splitting • Uncontrolled format string insertion • Memory buffer overflow • Integer overflow • Reliance on untrusted input in the security domain	This verifies the validity of the user (or program) input data and handles it appropriately in case of failure.
Security function	• Permission to access significant functions without proper authentication • Inappropriate authentication • Incorrect permission assignment for critical resource • Use of weak encryption algorithm • Use of inadequate encryption key length • Plaintext storage of significant information • Use of insufficient random numbers • Plaintext transmission of significant information • Hard-coded password • Hard-coded encryption key • Use of vulnerable passwords • Information disclosure due to cookies saved in user HDD • System information in source code comment • Code download of without integrity check • Use of one-way hash without salt on a saved password • Absence of repetitive authentication attempt limitation	This applies the policy of authentication, access control, authority management, password, and so on.
Time and status	• Time of check, time of use race condition • Loop with unreachable exit condition or recursive function	This prevents security weaknesses caused by improper management of time and state in parallel systems or environments where more than one process is running.
Error processing	• Information disclosure through error message • Detection of an error condition without action • Improper processing for unusual or exceptional conditions	This prevents important information from being leaked because of improper handling of errors.
Code error	• Null pointer deference • Improper resource shutdown or release • Use after freed resource • Use of an uninitialized variable	This prevents security weaknesses caused by coding errors that developers can make.
Encapsulation	• Disclosure of data due to wrong session • Leftover debug code • Exposure of system data to unauthorized control • Private array-typed field returned from a public method • Public data assigned to private array-typed field	This avoids information disclosure and authority issues that arise from insufficient encapsulation of sensitive data or functionality.
Application programming interface misuse	• Security decision defendant on DNS lookup • Use of vulnerable appication programming interace	This prevents usage of vulnerable application programming interface.

security requirements in software analysis and design results in developing a system vulnerable to cybersecurity threats.

Second, even if the security test guideline is applied to the previously mentioned battle management information system

(e.g., C4I system and command and control system), it is not suitable for the combat system software. This is because the guideline mainly focuses on general IT systems, including web and DBMS services. The combat system software requires optimized security testing when considering the performance and function of the combat system. In addition, this guideline is good for the software of general IT systems. However, it was formulated from the developer's point of view rather than security and is mainly applied to the application software. Thus, it must be improved to enable its application to the combat system software.

To fundamentally reinforce the cybersecurity of the combat system software, specific and practical security testing must be conducted, which can be applied to the combat system software during the system development stage when considering the security properties from the perspective of cybersecurity.

III. PROPOSED COMBAT SYSTEM SOFTWARE SECURITY TESTING

Security testing of the combat system software must be conducted not only on the application software but also on the middleware and operating system to enhance the cybersecurity of the combat system. In particular, security testing of the middleware and operating system must be integrally conducted together with the application security testing because the development direction of the application software is influenced by the characteristics and security configuration of the middleware and operating system. Furthermore, in the early stage of weapon system development, review and thorough management of the relation of security requirements between each software are mandatory. The basic directions for the improvement of security testing are as follows.

- Establishment of security test categories from the perspective of cybersecurity: To enhance the combat system software security, it is important to establish systematic security test categories that can satisfy the security properties from the perspective of security. Such categories are important to security testing and derive the detailed security test subitems accordingly. Even if the ''Guideline for software development security'' [7] is applied to the combat system software security testing, it is not suitable for the combat system software. Further, there is a limitation that the mandatory security control fields are systematically reflected in the security testing.
- Optimization of the detailed security test subitems on the application software: We derive the optimized detailed security test subitems to eliminate security vulnerabilities based on the aforementioned security categories and the ''Guideline for software development security'' [7] when considering the characteristics and environment of the combat system. In particular, we derive subitems for security testing of the operating system with reference to the ''Guideline for analysis and assessment of technical vulnerability in the main ICT infrastructures'' [35]. This guideline evaluates whether a system is vulnerable by verifying the security configuration of the system. The middleware security test subitems are derived from the DDS security standard proposed by OMG.

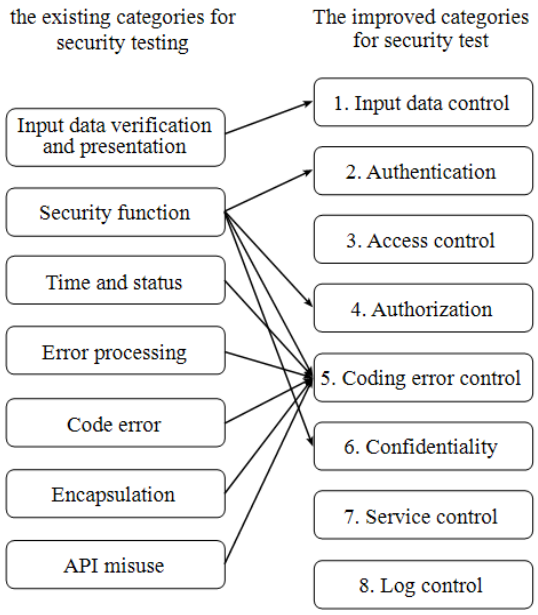

FIGURE 3. Establishment of the improved security test categories of the combat system software.

A. ESTABLISHMENT OF SECURITY TEST CATEGORIES

As presented in Figure 3, improved security test categories were established based on the ''Guideline for software development security'' [7]. The existing security test categories are inappropriate and unsystematic from the perspective of cybersecurity and have various mixed subitems of security testing complexly because they are classified from the perspective of source code development in spite of including measures for known vulnerabilities.

Therefore, the security test categories should be improved in consideration of the characteristics and environments of the combat system and security properties that must be possessed by the combat system. Well-defined security test categories are the cornerstone of systematic security testing. Moreover, system security architects or testers can effectively derive and categorize security test subitems based on them.

From the perspective of cybersecurity, we established the improved eight categories *(i.e., input data control, authentication, access control, authorization, coding error control, confidentiality, service control, log control)* from the existing categories. Several categories *(i.e., access control, service control, and log control)* are considered to be critical security properties.

Table 3 presents the description of each category and the corresponding software comprising the combat system software. Accordingly, the detailed security test subitems were systematically derived from the improved security test categories, and they can become the standard for conducting security testing on the combat system software.

TABLE 3. Security test categories of the combat system software.

Category	*Description*	*Combat system software*		
		Application	*Middleware*	*Operating system*
1. Input data control	• It should be verified whether inappropriate input data in source code are exploited. • It should be verified whether there are vulnerabilities such as overflow.	√		
2. Authentication	• It should be verified that appropriate authentication for significant functions or domain participants is performed • It should be verified that the account and password for login are not vulnerable.	√	√	√
3. Access control	• Appropriate access controls for critical resources or domain participants should be verified. • Network access based on IP or port, or use of operating system commands making an important impact on the system should be verified.	√	√	√
4. Authorization	• It should be verified that the appropriate permissions are granted for critical resources (e.g., files, directories).	√		√
5. Coding error control	• It should be verified whether a vulnerable information exposure or improper status are to be happened due to a developer's mistake or improper error handling.	√		
6. Confidentiality	• It should be verified that significant information is exposed due to non-encryption, improper encryption key usage or hard-coded critical information in source code.	√	√	
7. Service control	• It should be verified whether unnecessary services (e.g., SNMP, ftp, finger, r-command) and functions (e.g., shared folder and word processor in windows) that can be exploited are activated in the operating system. • It should be verified whether the mandatory security functions (e.g., console firewall in windows) are configured.			√
8. Log control	• It should be verified that important logging information is generated and managed for post-cyber accident analysis.	√		√

B. DERIVATION OF DETAILED SECURITY TEST SUBITEMS

We derived the detailed test subitems required by the category corresponding to each software based on the characteristics and functions of the software comprising the combat system software. As presented in Table 3, the test categories of each software can be identical; however, there are different detailed test subitems depending on the characteristics of each software.

1) APPLICATION SOFTWARE

Security testing of the application software includes eight categories and 37 subitems, excluding cyber control items related to web services and DBMS that are not correlated with the combat system among the security control items presented in the ''Guideline for software development security'' [7]. And we added access control and log control categories. This security testing determines whether there known vulnerabilities in the source code.

Table 4 presents the subitems of the application software security testing: 1. ''Input data control'' identifies inappropriate input data and command or buffer overflow; 2. ''Authentication'' verifies improper authentication for important functions and checks if the account's password is vulnerable; 3. ''Access control'' verify that access to the significant tactical function is properly controlled; 4. ''Authorization'' identifies incorrect permission assignment for the critical resource of the software; 5. ''Code error control'' verifies whether the mistake of the developer resulted in the usage of wrong function and unintended disclosure of information; 6. ''Confidentiality'' checks whether there is a hard-coded password or encryption key value on the source code and if weak cryptographic algorithms and keys can be used to verify the storage and transmission of sensitive information in the plaintext; 8. ''Log control'' verifies whether the audit log is generated and managed for important security data (e.g., log-in, access, tactical information processing).

TABLE 4. Subitems of the application software security testing.

Category	*Subitems*
1. Input data control (6 items)	• Path traversal and resource injection • Operation system command injection • Integer overflow • Uncontrolled format string insertion • Memory buffer overflow • Reliance on untrusted input a security decision
2. Authentication (4 items)	• Permission without proper authentication to significant function • Inappropriate authentication • Use of vulnerable password • Use of one-way hash without salt on the saved password
3. Access control (1 item)	• Access control to significant tactical function (weapon, track management and control, etc.)
4. Authorization (1 item)	• Incorrect permission assignment for critical resource (Executable file, configuration file, library, directory, etc.)
5. Code error control (16 items)	• Code download without integrity check • Time of check, time of use race condition • Loop with unreachable exit condition or recursive function • Information disclosure through error message • Detection of error condition without action • Improper processing for unusual or exceptional conditions • Null pointer dereference • Improper resource shutdown or release • Use after freed release • Use of an uninitialized variable • Disclosure of data due to wrong session • Leftover debug code • Exposure of system data to an unauthorized control • Private array-typed field returned from a public method • Public data assigned to private array-typed field • Use of vulnerable API
6. Confidentiality (8 items)	• Hard-coded password in source code • Hard-coded encryption key in source code • Significant system information (ID, password, etc.) disclosure through comments • Use of weak encryption algorithm • Use of inadequate encryption key length • Plaintext storage of significant information • Plaintext transmission of significant information • Use of insufficient random numbers
8. Log control (1 item)	• Generation and storage of audit log related to significant security data (log-in, access, tactical information processing, etc.)

2) MIDDLEWARE

The categories of middleware security testing are 2. ''Authentication,'' 3. ''Access control,'' and 6. ''Confidentiality.'' Security testing of middleware must be performed with optimized subitems of test because the security function should not excessively affect the critical combat mission performance. Thus, security testing for middleware should minimize the impact on the performance of the middleware.

Table 5 presents subitems of the middleware security testing. These subitems can be used to ensure that the appropriately authenticated participants in the domain can access information and verify access control for the domain participants. Moreover, they can be used to verify whether sensitive data are encrypted.

3) OPERATING SYSTEM

Security testing of the operating system includes five categories and 30 subitems. They can be used to verify whether the security configurations of the operating system are appropriately established. They are considered as very important factors for the combat system because the system can be easily accessed and exploited by cyber attackers if security configurations with known vulnerability are not appropriately established. Table 6 presents the subitems of operating system security testing.

2. ''Authentication'' verifies the improper use of account and password. 3. ''Access control'' checks if the remote access and accessible IP and port are properly controlled. 4. ''Authorization'' verifies the authority on critical directories and checks whether files can be exploited with inappropriate permissions. 7. ''Service control'' checks if the unnecessary services and programs to be exploited for hacking are running. 8. ''Log control'' verifies that the significant log saving function of the system is established for tracing, post-analysis, and audit following cyberattacks. [Windows] is the subitem corresponding to the Windows operating system.

C. SECURITY TESTING IN THE COMBAT SYSTEM DEVELOPMENT PROCESS

The proposed approach for software security testing mainly focuses on ''software implementation'' and ''software integration and test'' of the system development stage, as presented in Figure 2. However, we present the roles and advantages of the proposed security testing in the process of combat system development by applying the proposed security testing to the V model as presented in Figure 4. The V model was derived from the waterfall model and is widely used in system engineering. It facilitates the systematic development of weapon systems by matching the requirements, system analysis, and design process on the left side with the test

TABLE 5. Subitems of the middleware security testing.

Category	*Subitems*
2. Authentication	• Appropriate authentication for domain participants
3. Access control	• Access control for domain participants
6. Confidentiality	• Encryption for the transmitted message

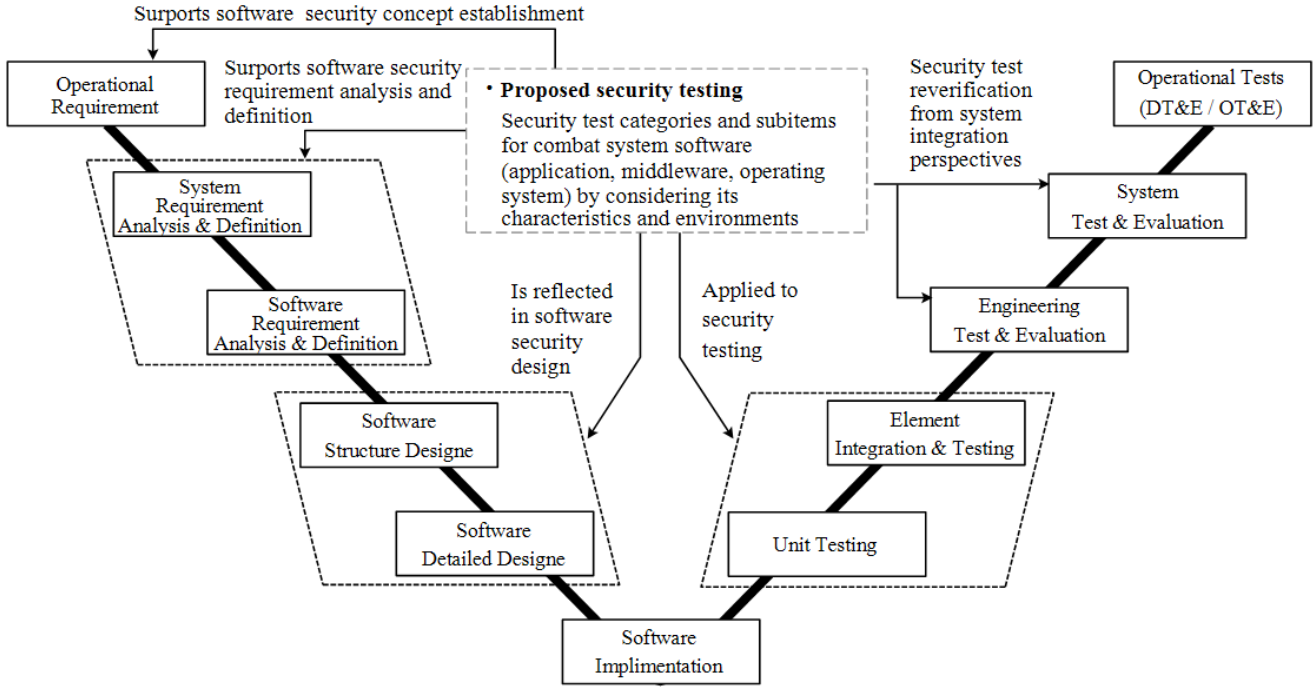

FIGURE 4. Roles of the proposed security testing in the V model of the combat system development.

and evaluation process on the right side. In particular, it has been utilized for developing the Aegis combat system in US navy [36]–[38].

The proposed categories and specific subitems of software security testing are basically applied in the unit testing phases and elementary integration & evaluation phase on the right side of V model. It can be verified whether the proposed security testing is appropriately reflected, as each modular application software is developed and connected, and the application software, middleware, and operating system, which comprise the combat system software, are closely integrated. In the process on the left side, the proposed security testing can facilitate the establishment of the required security concept of the software in the operational requirement phase, and support security requirement analysis in the software requirement analysis & definition phase, and reflect the previously defined security requirements in the design phases.

In addition, after the security testing is performed in the unit testing phase and elementary integration & testing phase, it is possible to reverify whether the tested critical security requirements have been properly developed and integrated from a system integration perspective in the engineering test & evaluation phases and system test & evaluation phase by using the subitems of the proposed security testing. The reason for enabling the above activities is that the test concept and detailed subitems of the security testing have already been defined in consideration of the characteristics and environment of the combat system software. As a result, the proposed security testing effectively reflects the security requirements in the early stages of combat system software development and supports security-related activities in all stages.

Therefore, the proposed security testing approach reinforces the security of the combat system through the overall process of system development and the software security test process. In addition, by systematically and comprehensively reviewing the software security from the initial process of system development, the security issues identified in the latter half of the development process can be prevented.

IV. EVALUATION AND DISCUSSION

In this paper, an improved new security testing approach for the combat system software is proposed. This provides a systematic and practical framework for software security testing, which has not yet been effectively achieved when developing combat systems. In this section, the proposed

TABLE 6. Subitems of the operating system security testing.

Category	*Subitems*
2. Authentication (6 items)	• Delete unnecessary and default account • Vulnerable password check (use mix of numbers, characters, special character) • Log-in security check [windows] • Administrator and guest account control [windows] • Anonymous account control in administrator group [windows] • Control of "root" account remote access
3. Access control (4 items)	• Control of "su" command use by regular users • Control of accessible IP and port • Security management of remote access[windows] • Remote access control to SAM (security account management) file [windows]
4. Authorization (8 items)	• Setting UMASK, PATH to files • Setting access privilege to user home and significant directories • Check if normal files are deleted in dev/ directory • Check if files and directories without owner are deleted • Setting owner and privilege to significant system files • Check if there exist files with unnecessary SUID, GUID setting • Do not allow anonymous enumeration of SAM accounts and shares [windows] • Directory security management
7. Service control (11 items)	• Check unnecessary services (SNMP, ftp, etc.) • Setting owner and privilege to "cron" files • NFS/RPC services stop or security setting • Delete the shared folder [windows] • Setting stop to vulnerable daemon (finger, r-commands, exec, etc.) in inetd.d file • Stop unnecessary schedule jobs • Delete unnecessary applications (MS-Office, word processor, etc.) [windows] • Setting console firewall [windows] • Setting registry to defend DoS attack [windows] • Restrict remote support function [windows] • Check if console security programs (USB control, NAC, etc.) are installed [windows]
8. Log control (1 item)	• Setting significant system log storage

security testing is analyzed by comparing it with the previously studied software security testing discussed in related work, the existing security testing applied to the battle management information system of the Korean military, and CC part 2, which is the international criteria for the security evaluation of general IT systems. CC part 2 defines the security function requirements, which include 11 security function classes (e.g., security audit, communication, cryptographic support, user data protection, and security management) for IT system security. These requirements describe the desired security behavior expected from the evaluation target to meet the security objectives [39].

As presented in Table 7, evaluation fields related to the combat system software security, such as applicability to the combat system software, key function factors (secure coding, security configuration, and vulnerability verification) for security testing, and consideration of the characteristics of the combat system software, are analyzed. The previously studied security testings focused on a theoretical methodology rather than a specific and practical approach. Moreover, they were mainly applied to the application software and not to the middleware and operating system. They were not suitable for security testing of the combat system software, considering its characteristics and environments.

Thus, although the existing security testings can ensure secure coding for the application software, they are insufficient for verifying security configuration and identifying vulnerabilities in the middleware and operating system.

CC part 2 is a good standard for security evaluation; however, it does not include secure coding and security configuration setting of the operating system. In addition, it does not match the characteristics of the combat system software.

However, security testing of the overall combat system software is systematically and holistically performed because the proposed security testing is applied to the middleware, operating system, and application software, which constitute the combat system software, and the characteristics of the combat system are satisfied.

Nevertheless, the proposed security testing exhibits a limitation in that it cannot prevent all cyberattacks. This paper focuses on the prevention of cyberattacks to software during the development of a combat system and the implementation of robust cybersecurity software using the proposed security testing. Security testing for installation or configuration of an additional security system for the combat system has not been considered here.

The combat system is a mission-critical system that processes tactical data and engages enemies in real-time; thus, security testing cannot take precedence over performance and function tests. It is very important to balance security, performance, and function tests by considering the characteristics and environment of the combat system. Therefore, we propose practical and optimized security testing by fully considering the above factors.

Security vulnerabilities of software can be caused by faults or errors generated by mistakes of software designers and developers or other reasons. They provide a potential security threat to software, as well as have a significant impact on the quality and reliability of the software. Recently,

TABLE 7. Comparison of the proposed security test.

	Applicability to combat system software			*Key function factors for security test*			*Consideration of the combat system software's characteristics*		
	Application	*Middleware*	*Operating system*	*Secure coding*	*Security configuration*	*Vulnerability verification*	*Environment*	*Performance*	*Function*
Michele et al. [21]	n/a	n/a	n/a	✓	✓	✓	constraint	constraint	constraint
Brucker et al. [15]	✓	n/a	n/a	constraint	constraint	constraint	constraint	constraint	constraint
Valentine et al. [22]	✓	n/a	n/a	✓	✓	✓	✓	constraint	constraint
Longoria [23]	✓	n/a	n/a	constraint	constraint	constraint	constraint	constraint	constraint
Josip et al. [24]	n/a	n/a	n/a	✓	✓	✓	constraint	constraint	constraint
Pekaric et al. [25]	✓	n/a	n/a	constraint	constraint	constraint	constraint	constraint	constraint
Seane et al. [26]	✓	n/a	n/a	constraint	✓	✓	constraint	constraint	constraint
Olivero et al. [27]	✓	n/a	n/a	constraint	✓	✓	constraint	constraint	constraint
Existing security test [7]	✓	n/a	n/a	✓	constraint	constraint	constraint	constraint	constraint
CC [39]	✓	✓	constraint	n/a	✓	✓	constraint	constraint	constraint
Proposed	✓	✓	✓	✓	✓	✓	✓	✓	✓

* ✓: applicable, n/a : not applicable, constraint : limitation or insufficiency to the evaluation factor.

various studies on software fault predictions that can be performed before software tests are in progress to improve the quality and reliability of software [40].

In the future, new cyber vulnerabilities in the software will be discovered. Accordingly, detailed subitems of the proposed security testing must be continuously modified and improved. Moreover, the cooperation among software developers, security experts, and designers from the start of the combat system development process is very important. The combat system software security must also be reviewed in the early stage of system development and reflected systematically in software analysis and design.

In particular, subitems, which can verify such attacks, must be developed and applied to the security testing if a new zero-day attack to the combat system software (application software, middleware, and operating system) is discovered.

V. CONCLUSION

The naval combat system is a software-centric complex weapon system integrated with the network and computer software. The core performance and function of this system are influenced by software. Recently, the use of commercial software has significantly increased. However, systematic software security testing has not been applied to the real testing process in the system development stage during the combat system development, restricting the development of a robust combat system against cyberattacks. To solve this problem, an effective and practical software security testing approach is proposed in this paper to fundamentally deal with cyberattacks when considering the characteristics and environment of the combat system.

First, we present a concrete and practical framework for software security testing by establishing eight security test categories for the combat system software, which comprise the application software, middleware, and operating system, from the viewpoint of cybersecurity. These categories are mandatory security controls that must be adhered to by the combat system software. Also, optimized detailed security subitems are derived based on these categories when considering the characteristics and environment of the combat system. Moreover, a combat system software that is fundamentally robust against cyberattacks can be developed because the proposed security testing helps in indicating the security requirements in the early stage of combat system development by reviewing and defining them during software analysis and design, as presented in Figure 4.

In the future, cyberattacks to the military weapon system called "SoS" (e.g., naval combat system, missile systems, and airplanes) will emerge. Therefore, this situation must be analyzed and studied by weapon system and security experts, and appropriate countermeasures must be established based on the mission characteristics associated with the weapon system. It is also important to develop a security framework that can be practically applied in weapon system development and guidelines for the security of weapon system software. In future work, we will develop additional subitems of the security test in consideration of future combat system trends. It is necessary to extend the proposed approach to a framework that can be applied to combat systems and other weapon systems.

REFERENCES

[1] C. Chaplain, "Weapon systems cyber security: DOD just beginning to grapple with scale of vulnerabilities," Government Accountability Office, Washington, DC, USA, Tech. Rep. GAO-19-128, Oct. 2018.

[2] R. Koch and M. Golling, "Weapons systems and cyber security—A challenging union," in *Proc. 8th Int. Conf. Cyber Conflict (CyCon)*, May 2016, pp. 191–203.

[3] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of systems engineering: Basic concepts, model-based techniques, and research directions," *ACM Comput. Surveys*, vol. 48, no. 2, pp. 1–41, 2015.

[4] J. K. Lee, "Trend of weapon system cyber security policy," *J. Korea Inst. Inf. Secur. Cryptol.*, vol. 28, no. 6, p. 84, Dec. 2018.

[5] K. Y. Heo, K. Y. Kwon, and T. S. Kim, "A study on the promotion of reliability test for imbedded software of weapon system," *J. Korean Soc. Syst. Eng.*, vol. 11, no. 1, p. 66, Jun. 2015.

[6] W. William, H. Brian, S. Adam, and S. Owen, "Common architecture system assurance: Information assurance for the next generation of combat systems," *Lead. Edge, Combat Syst. Eng. Int.*, pp. 137–139, Feb. 2013.

[7] *Guideline For Software Development Security*, Korea Internet Secur. Agency, Seoul, South Korea, Jan. 2017.

[8] *Manual for Development and Management of Weapon System Software*, Defense Acquisition Program Admin., Seoul, South Korea, Aug. 2017.

[9] C.-G. Yi and Y.-G. Kim, "A study on software security test of naval ship combat system," *J. Korean Inst. Commun. Inf. Sci.*, vol. 45, no. 3, pp. 628–637, Mar. 2020.

[10] T. Y. Gu, Y. S. Shi, and Y. Y. Fang, "Research on software security testing," *J. Comput. Inf. Eng.*, vol. 4, no. 9, p. 1446, Jun. 2010.

[11] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, and A. Pretschner, "Security testing: A survey," in *Advances in Computers*, vol. 101. Innsbruck, Austria: Innsbruck Univ., 2016, pp. 1–5.

[12] *Risk-Based and Functional Security Testing*. Accessed: Jul. 2013. [Online]. Available: https://www.us-cert.gov/bsi/articles/best-practices/security-testing/risk-based-and-functional-security-testing

[13] M. Felderer, P. Zech, R. Breu, M. Büchler, and A. Pretschner, "Model-based security testing: A taxonomy and systematic classification," *Softw. Test., Verification Rel.*, vol. 26, no. 2, pp. 119–148, Mar. 2016.

[14] B. Chess, and J. West, *Secure Programming With Static Analysis*. London, U.K.: Pearson, 2007.

[15] A. Brucker and U. Sodan, "Deploying static application security testing on a large scale," Sicherheit 2014-Sicherheit, Schutz und Zuverlässigkeit, Tech. Rep. 228, 2014.

[16] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, "NIST technical guide to information security testing and assessment," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NIST SP 800-115, 2008, pp. 2–25.

[17] S. N. Matheu-García, J. L. Hernández-Ramos, A. F. Skarmeta, and G. Baldini, "Risk-based automated assessment and testing for the cybersecurity certification and labelling of IoT devices," *Comput. Standards Interfaces*, vol. 62, pp. 64–83, Feb. 2019.

[18] P. Godefroid, M. Y. Levin, and D. Molnar, "SAGE: Whitebox fuzzing for security testing," *Queue*, vol. 10, no. 1, pp. 20–27, Jan. 2012.

[19] M. Felderer and E. Fourneret, "A systematic classification of security regression testing approaches," *Int. J. Softw. Tools Technol. Transf.*, vol. 17, no. 3, pp. 305–319, Jun. 2015.

[20] U. Waheed, "Security regression testing framework for Web application development," M.S thesis, Dept. Info., Oslo Univ., Oslo, Norway, 2014.

[21] M. Peroli, F. De Meo, L. Viganò, and D. Guardini, "MobSTer: A model-based security testing framework for Web applications," *Softw. Test., Verification Rel.*, vol. 28, no. 8, p. e1685, Dec. 2018.

[22] V. Casola, A. De Benedictis, M. Rak, and U. Villano, "Towards automated penetration testing for cloud applications," in *Proc. IEEE 27th Int. Conf. Enabling Technol., Infrastruct. Collaborative Enterprises (WETICE)*, Jun. 2018, pp. 24–29.

[23] J. A. Longoria, "Prioritizing security regression test cases using threat models," Ph.D. dissertation, Dept. Elect. Comp. Eng., Texas Univ., Austin, TX, USA, 2016.

[24] J. Bozic and F. Wotawa, "Planning-based security testing of Web applications with attack grammars," *Softw. Qual. J.*, vol. 28, pp. 1–28, Mar. 2020.

[25] I. Pekaric, C. Sauerwein, and M. Felderer, "Applying security testing techniques to automotive engineering," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, Aug. 2019, pp. 1–10.

[26] S. Hagerman, A. Andrews, and S. Oakes, "Security testing of an unmanned aerial vehicle (UAV)," in *Proc. Cybersecurity Symp. (CYBERSEC)*, Apr. 2016, pp. 26–31.

[27] M. A. Olivero, A. Bertolino, F. J. Dominguez-Mayo, M. J. Escalona, and I. Matteucci, "Security assessment of systems of systems," in *Proc. IEEE/ACM 7th Int. Workshop Softw. Eng. Syst. Syst. (SESoS), 13th Workshop Distrib. Softw. Develop., Softw. Ecosyst. Syst. Syst. (WDES)*, May 2019, pp. 62–65.

[28] *Defense Agency for Technology and Quality*, Dictionary Military Tech. Terms, Seoul, South Korea, 2018.

[29] *Combat System Principle and Understanding*, ROK Nav. Educ. Training Command, Jinhae-gu, South Korea, Sep. 2016.

[30] *Surface Navy Combat System Engineering Strategy*, U.S. Navy PEO IWG, Washington, DC, USA, Mar. 2010.

[31] J. H. Han, "Message encryption methods for DDS security performance improvement," *J. Korea Inst. Inf. Commun. Eng.*, vol. 22, no. 11, p. 1555, Nov. 2018.

[32] *DDS Portal*. Accessed: Mar. 2015. [Online]. Available: http://www.portals.omg.org/dds

[33] Y. K. Go and C. S. Kim, "Cryptographic overhead of DDS security for naval combat system security," in *Proc. KIISE Korea Comput. Congr.*, Jun. 2017, p. 1217.

[34] *DDS Security Standard Document*. Accessed: Sep. 2016. [Online]. Available: http://www.omg.org/spec/DDS-SECURITY/1.0

[35] *Guideline for Analysis and Assessment on Technical Vulnerability of Main ICT Infrastructures*, Korea Internet Secur. Agency, Seoul, South Korea, Dec. 2017.

[36] S. Balaji and M. S. Murugaiyan, "Waterfall vs. V-model vs. agile: A comparative study on SDLC," *Int. J. Inf. Technol. Bus. Manage.*, vol. 2, no. 1, pp. 26–30, 2012.

[37] S. P. Gregg, D. M. Albert, and P. Clements, "Product line engineering on the right side of the 'V,'" in *Proc. 21st Int. Syst. Softw. Product Line Conf.*, Sep. 2017, pp. 165–174.

[38] A. Al-Momani, F. Kargl, R. Schmidt, A. Kung, and C. Bösch, "A privacy-aware V-model for software development," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2019, pp. 100–104.

[39] *Common Criteria Part 2: Security Functional Components, Ver. 3.1 Rev. 5*, Apr. 2017, pp. 13–14.

[40] T. B. Alakus, R. Das, and I. Turkoglu, "An overview of quality metrics used in estimating software faults," in *Proc. Int. Artif. Intell. Data Process. Symp. (IDAP)*, Sep. 2019, pp. 1–6.

CHEOL-GYU YI (Student Member, IEEE) received the B.S. degree in naval architecture from Naval Academy, South Korea, in 1995, and the M.S. degree in computer science and industrial systems engineering from Yonsei University, Seoul, South Korea, in 2002. He is currently pursuing the Ph.D. degree with the Department of Computer and Information Security, Sejong University. His current research interests include weapon system security, naval combat systems, software security testing, threat hunting, and security operation center.

YOUNG-GAB KIM (Member, IEEE) received the B.S. degree in biotechnology and genetic engineering and minored in computer science and engineering and the M.S. and Ph.D. degrees in computer science and engineering from Korea University, Seoul, South Korea, in 2001, 2003, and 2006 respectively. He was an Assistant Professor with the School of Information Technology, Catholic University of Daegu. He is currently an Associate Professor with the Department of Computer and Information Security, and the Department of Convergence Engineering for Intelligent Drone, Sejong University. He has published over 180 research articles in the field of computer science and information security. His current research interests include the Internet of Things (IoT) security, big data security, network security, home networks, security risk analysis, and security engineering. As a Korean ISO/IEC JTC 1 Member, he is contributing in developing data exchange standards.