Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.046871

ARTICLE

Systematic Security Guideline Framework through Intelligently Automated Vulnerability Analysis

Dahyeon Kim[1], Namgi Kim[2] and Junho Ahn[2,*]

[1]Computer Information Technology, Korea National University of Transportation, Chungju, 27469, Korea

[2]Department of AI Computer Science and Engineering, Kyonggi University, Suwon, 16227, Korea

*Corresponding Author: Junho Ahn. Email: jha@kyonggi.ac.kr

Received: 17 October 2023 Accepted: 25 January 2024 Published: 26 March 2024

ABSTRACT

This research aims to propose a practical framework designed for the automatic analysis of a product's comprehensive functionality and security vulnerabilities, generating applicable guidelines based on real-world software. The existing analysis of software security vulnerabilities often focuses on specific features or modules. This partial and arbitrary analysis of the security vulnerabilities makes it challenging to comprehend the overall security vulnerabilities of the software. The key novelty lies in overcoming the constraints of partial approaches. The proposed framework utilizes data from various sources to create a comprehensive functionality profile, facilitating the derivation of real-world security guidelines. Security guidelines are dynamically generated by associating functional security vulnerabilities with the latest Common Vulnerabilities and Exposure (CVE) and Common Vulnerability Scoring System (CVSS) scores, resulting in automated guidelines tailored to each product. These guidelines are not only practical but also applicable in real-world software, allowing for prioritized security responses. The proposed framework is applied to virtual private network (VPN) software, wherein a validated Level 2 data flow diagram is generated using the Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of privilege (STRIDE) technique with references to various papers and examples from related software. The analysis resulted in the identification of a total of 121 vulnerabilities. The successful implementation and validation demonstrate the framework's efficacy in generating customized guidelines for entire systems, subsystems, and selected modules.

KEYWORDS

Framework; automation; vulnerability analysis; security; guidelines

1 Introduction

Electronic devices are connected by networks and digitalized owing to the development of internet networks and the spread of various electronic devices such as mobile phones, TVs, cameras, game consoles, and computers [1–3]. In 2022, the number of internet users per day was 4.95 billion. At present, the internet penetration rate is 62.5% of the total global population [4]. In addition, it is estimated that the smartphone penetration in 2022 included 6.6 billion individuals. This represents an annual increase of 4.9% [5]. The market for software that can be used on various devices is increasing

in conjunction with the spread of computers and smartphones. According to the Software Products Global Market Report 2023 by ReportLinker [6], the global software product market increased at an average annual rate of 12.5% from USD 1333.48 billion in 2022 to USD 1500.2 billion in 2023.

As the number of various available software increases, there is an increase in the vulnerabilities identified in the software. Accordingly, 17,344, 18,325, 20,171, and 25,227 vulnerabilities were identified in 2019, 2020, 2021 and 2022, respectively. This indicates that the number of vulnerabilities identified each year is increasing [7]. Recently, the operating system of a petroleum extraction system was hacked. This shut down a U.S. oil pipeline [8]. In addition, the names, contact information, birthdays, and product registration information of U.S. customers were leaked owing to a vulnerability in Samsung Electronics' systems [9]. Vulnerabilities must be prevented before users utilize the products. Moreover, it should be feasible to handle 1) vulnerabilities that are identified after products are released and 2) new threats in response to the development of cyberattack technologies such as ransomware, viruses, and hijacking. Thus, methods are required for analyzing product vulnerabilities and deriving security guidelines to prevent and prepare for potential threats.

Existing software security vulnerability analyses [10–13] often focus on specific functionalities or modules, and in many cases, they are performed without sufficient validation. Such partial and arbitrary security vulnerability analyses make it challenging to comprehend the overall security vulnerabilities of the software [14–17]. Particularly, as software systems become increasingly complex, this arbitrary and partial approach proves insufficient in addressing real-world security issues. Due to restricted access to internal confidential information, understanding this logic can be challenging. Therefore, understanding the logic of complex software is crucial for comprehending software security vulnerabilities.

To address these challenges, the research developed a framework capable of automatically and rapidly analyzing a product's overall functionality and security vulnerabilities. The framework encompasses several stages, including product functional analysis, threat modeling using automated tools, linking vulnerabilities to functionalities, thorough functional analysis verification based on threat scenarios, and the automated generation of guidelines and security assessments for each functionality. Understanding software security vulnerabilities necessitated information about the software's comprehensive functionality. Due to restricted access to detailed software functionality information, data were gathered from functional specifications, operating environments, software architectures, reports, and thesis papers. The complete functionality was constructed by integrating all validated functionalities into the automation tool, originally containing only some functional modules. The proposed framework in this study enables the derivation of entire and comprehensive real-world security guidelines, overcoming the limitations of the existing arbitrary and partial approach.

Security guidelines for the product are generated by associating functional security vulnerabilities with the latest vulnerabilities like CVE [18] and CVSS scores, resulting in the creation of automated and intelligent security vulnerability guidelines tailored to each product. The security guidelines produced through this framework are practical and applicable in real-world scenarios, allowing for responses based on security priorities. To validate the proposed framework, the entire software system was configured using Microsoft's automation tool based on VPN software [15–17], incorporating verified functionalities and vulnerabilities. Additionally, it intelligently generates on-demand security vulnerability guidelines, and each module underwent validation through vulnerability scenarios. The final set of 121 security guidelines with associated vulnerability scores was generated.

In this research, VPN software was employed for validating the framework. The rationale for choosing VPN is its well-established and widespread use, making it anticipated that many individuals

will comprehend and be able to apply this framework. Therefore, VPN software was selected as the subject of our experiment. VPN encompasses the overall functionalities of information security software, including features such as information flow control, virtual channels, audit logs, identification and authentication, security management, transmission data protection, self-testing, secure session management, VPN client protection, secure interaction between VPN clients and servers, and encryption support. In this paper, we implemented and validated all these features in the automated tool, intelligently enabling the generation of customized guidelines for the entire system, subsystems, and selected modules.

In this paper, Chapter 2 presents studies related to the proposed security guideline derivation framework as well as those related to threat modeling. Chapter 3 specifically describes the methodology proposed in this study and each of its stages. Chapter 4 applies this study's proposed framework to a VPN to analyze vulnerabilities and formulate security guidelines. Finally, Chapter 5 presents this study's conclusions and describes future research.

2 Related Works

2.1 Studies on the Formulation of Security Guidelines

Typical security guidelines include the Common Criteria (CC) [19]. CC provides guidelines to establish general concepts and principles for performing information technology (IT) security evaluations of products and systems based on the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) 15408. To receive CC certification, the organization developing the product preemptively prepares a description of the security target, an outline of the product's security functions, and an assessment of potential threats. Then, the organization verifies that the product's security characteristics satisfy the previously defined standard by performing penetration testing on the product. It then prepares a report on the results of this examination. Finally, the prepared documents and the products are sent to the CC evaluation organization to verify that all the security guidelines are satisfied [20]. CC certification provides a framework through which the product's security features and relevant guarantees can be specified. Each country has CC-based security certification organizations to satisfy its differing security guidelines [21,22]. However, CC certification provides only security features and guarantees for IT security products. Moreover, it is difficult to examine the security of IoT devices or general software such as e-commerce software. This study proposes a security guideline methodology that can be applied to all general products. Studies [23–25] have been conducted on the establishment of various security guidelines in addition to those for evaluation and certification. A study [23] determined that the key causes of data security breaches in the medical industry include human error and malicious result reporting. The study proposed education to improve the information security awareness of organizations through security guidelines. In that study, the advantages and disadvantages of existing information security guidelines were compared, and an effective information security awareness program for organizations was developed to provide efficient security education. Another study [24] developed a context-based ontology to establish security guidelines to increase the interoperability and mutual interests between smart-home providers, users, and smart devices. Here, the cases were divided as follows: one wherein security management is performed automatically without the user's involvement, and another wherein the user manages security according to the guidelines. Specific guidelines were provided for each case. Another study [25] proposed security guidelines for the design of Automatic Teller Machine (ATM) interfaces. This study observed that although ATM interfaces have designs that are specialized for usability, there are no specific guidelines related to security. Therefore, security guidelines for ATM interface design and the

definition of each guideline were proposed using security guidelines based on existing regulations and recommended guidelines for existing interfaces. As demonstrated, studies have established and applied security guidelines through a variety of methodologies. The present study proposes a vulnerability analysis and security guideline establishment methodology that can be used by product managers who are not security experts, for more general products.

Recent research on evaluating the security of Artificial Intelligence-Machine Learning (AI-ML)-based systems [10] proposed the STRIDE-AI methodology, an asset-centric approach. By adapting Microsoft's STRIDE approach to the AI-ML domain, it provides tailored threat modeling. The proposed methodology offers efficient security controls for ML practitioners to protect ML assets. The researchers validated the effectiveness of their approach in addressing security threats by utilizing a publicly available use case based on scenarios. The software architecture provided in this research is somewhat partial or broad, limiting its applicability to real-world product guidelines and focusing more on the verification of research methods. Another research [11] extended the Microsoft STRIDE threat model to identify threats from a user's perspective. This model integrates aspects of threat modeling relevant to end-users, such as valuable assets, attack targets, and attack methods. For the validation of this study, user-centric threat modeling was utilized to verify a broad architecture at Level 0 of a data flow diagram (DFD) using a use case, but detailed design at Level 2 was not fully constrained. Similar limitations exist in previous vulnerability analysis studies [12–15], which are confined to partial functions and are primarily applicable for the verification of only their research, not fully covering the entire functionality.

2.2 Threat Modeling

This study proposes a vulnerability analysis and security guideline establishment methodology that uses threat modeling. Various threat modeling studies have been performed. The STRIDE threat modeling approach [26] developed by Microsoft is a computer security threat identification methodology based on the security development lifecycle. In STRIDE threat modeling, vulnerabilities are grouped into six categories: spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege. In addition, STRIDE threat modeling provides accurate guidelines that can be used by security experts and general users. The Process for Attack Simulation and Threat Analysis (PASTA) threat modeling approach [27] was developed by VerSprite. It provides a stagewise process for analyzing the threats to key assets and understanding the status of vulnerabilities to establish an organization's overall security strategy. It analyzes security threats to products and to all the business aspects of the organization that manages and develops products. Linkability, Identifiability, Non-repudiation, Detectability, Disclosure of information, Unawareness, and Non-compliance (LIDDUN) threat modeling [28] is applied to mitigate threats to personal information in systems. It consists of three stages: model the system, induce threats, and manage the threats. Here, the threat-inducing stage is grouped into seven threat categories: linkability, identifiability, non-repudiation, detectability, disclosure of information, unawareness, and non-compliance. Another type of threat modeling is the DREAD threat modeling approach [29] developed by Microsoft, which calculates risk scores. The threat level can be defined by the sum of the individual risk scores. Here, the risk scores are as follows: three points for High, two points for Medium, and one point for Low. The threat level is specified as 5–15 points based on the sum of the risk scores. The Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) threat modeling approach [30] was developed by the Software Engineering Institute (SEI). This methodology consists of three stages: construct an asset-based threat profile, identify vulnerabilities in each asset's infrastructure, and establish a security strategy. Here, the current security status and key assets are identified. The security strategy

is established through various individuals within the organization. Threat Vulnerability and Risk Analysis (TVRA) threat modeling [31] was developed by the European Telecommunications Standards Institute (ETSI). This methodology analyzes threats to communication systems and evaluates risks. It uses 10 stages to identify key assets and evaluate scenarios and risks according to the threats identified. Other threat methodologies include Visual, Agile, and Simple Threat modeling (VAST) [32] and System-Theoretic Process Analysis for Security (STPA-Sec) [33]. This study compared and analyzed each of the threat modeling approaches, and selected the one most suitable for general use.

3 Proposed Framework

3.1 Overall Proposed Framework

This section specifically describes the framework proposed in this research, the methods used in each stage, and the results of an analysis of these stages. The overall framework proposed in this study is shown in Fig. 1. First, the product functional specifications, operating environments, software architectures, reports, thesis papers, etc., are analyzed to examine the overall flow of product usage. Here, end-to-end points need to be analyzed to examine the actual environment that is used and the specific process that is applied during usage. Second, it is necessary to develop a DFD to apply most of the threat modeling approaches based on product analysis. A DFD shows the data flow direction, data types, data modifications, etc., during product usage. Threat modeling can be applied through various methods such as text and surveys. However, most threat modeling approaches analyze product vulnerabilities based on DFDs. The points where vulnerabilities may occur and those that need to be protected can be verified by applying threat modeling based on the generated DFDs. In addition, the data categories and types where vulnerabilities occur can be analyzed and corrected. Then, the vulnerabilities are analyzed based on the product's functions and operating environments. The vulnerability analysis uses threat modeling, threat libraries, and attack-scenario methods. Third, threat libraries are constructed to analyze the vulnerability analysis results obtained via threat modeling and the actual current vulnerabilities that are identified. Threat libraries may collect known vulnerabilities from various public data such as Common Vulnerabilities and Exposure (CVE), Common Weakness Enumeration (CWE), and Open Web Application Security Project (OWASP). In this study, the vulnerabilities collected by threat libraries were combined with the results of an analysis by threat modeling to present more specific risks. Fourth, attack scenarios were generated based on these vulnerabilities to understand the attack routes, etc., that employ the specific vulnerabilities. The generated DFD is verified and modified by comparing the vulnerability lists of modules used according to the attack scenarios Fifth, then, the security vulnerabilities were automatedly generated based on the final DFD, considering average scores of CVSS. Finally, the security guideline was established by combining vulnerability scores. The proposed framework intelligently enables the automatic generation of security guidelines for specific scopes within the entire system, including subsystems, modules, and selected components. The framework proposed in this study can be applied to most general products. It is effective for analyzing vulnerabilities and formulating security guidelines.

The framework we propose enables automated customized vulnerability analysis for new software products. The proposed framework suggests a method to collect overall functionality information for the software (features, environment, architecture, development reports, papers). It proposes how automated tools, often limited to certain security vulnerability features, can generate additional security modules tailored to the software. It suggests a validation process for the generated overall functionalities and vulnerabilities through attack scenarios. It measures the significance by assigning

average CVSS scores for validated functionalities and vulnerabilities, categorized by function, module, and subsystem. Through automated systems, it creates intelligent security vulnerability guidelines for the entire system, subsystems, and selected modules.

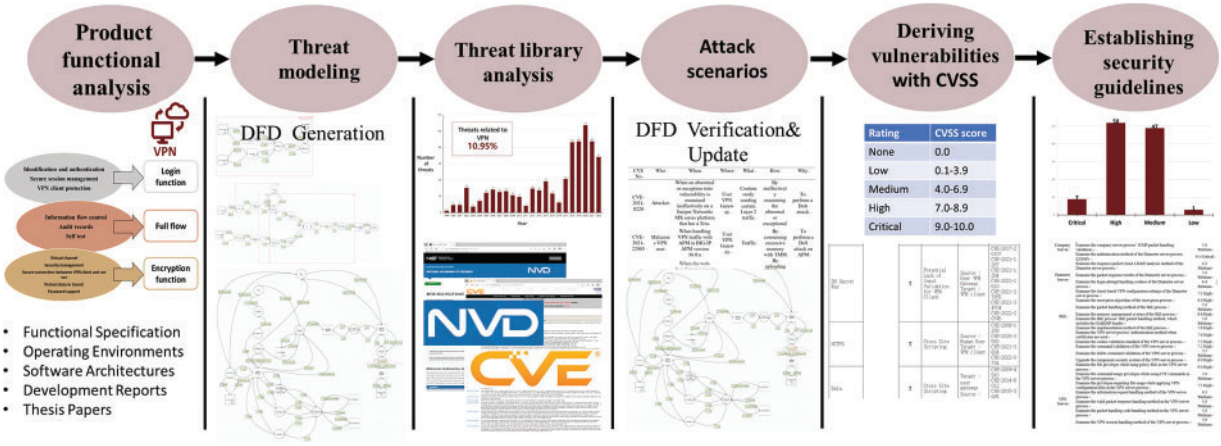


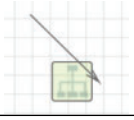
Figure 1: Overall process of the proposed framework

3.2 Product Functional Analysis and Data Flow Diagram (DFD)

We derived comprehensive and specific software functionalities by utilizing technical documents, certification authorities, published papers, and reports provided by the company. The identified functionalities were extracted based on gathered information, and a system encompassing the overall functionalities was derived through cross-verification. To examine products, this study analyzed their functions and operating environments. For products under development, threat modeling can be applied through predefined functions. For developed products, threat modeling is applied based on their implemented and usable functions. The product functions include all the parts that the users utilize as they employ the services. The product analysis must examine the environment in which the product is used and managed. This requires an analysis of the operating environment. The product's operating environment refers to the overall environment that aids and manages the operation of actual services. DFDs are generated to apply threat modeling based on the analyzed product functions and operating environments. A DFD shows the transformations that occur as the data flows through the processes within the software. It is subdivided into stages to depict data flows and functions in detail. Here, the stages consist of Level 0, Level 1, and Level 2. Level 0 is a context diagram. It shows a simple and basic outline of the entire system or processes of the product. Level 1 divides the results of Level 0 into sub-processes and analyzes these to depict the key functions performed by the system. The final stage, Level 2, adds detail to the sub-processes in Level 1 to depict more specific elements of the DFD. A DFD is composed of the symbols shown in Table 1. It shows the product's systems and sub-processes.

A DFD is depicted by these elements. Users can apply the software, and the results can be depicted by a single diagram. Conventional security tests depict the security results of certain specific functions. However, DFDs can be used to perform vulnerability analysis on all the functions.

Table 1: Descriptions and symbols for elements of data flow diagrams

Item	Description	Symbol
External entity	Used in data input or output via the role of data generation and consumption. Indicates the start or end point of data generation in the course of handling a process, and is interrelated with a process.	
Process	A process for converting input data into the desired data and outputting it. The process cannot generate data. Follows the principle of processing input data sequentially and always processing the data.	
Data store	A set of information that stores data. Data is only stored and does not undergo separate processing such as conversion. Depicted as two parallel lines.	
Data flow	Represents the interface between elements. The names of data values, etc., should be assigned. Depicted as a directional arrow.	

3.3 Threat Modeling, Library Analysis, and Attack Scenarios

3.3.1 Threat Modeling

Threat modeling enables the protection of valuable resources, identification and resolution of threats within products, and comprehension of the communication between processes. Here, threat modeling can structure and depict the information that influences the security of an application program. It can be widely applied to software, applications, systems, networks, distributed systems, Internet of Things (IoT) devices, business processes, etc. Threat modeling is a sequential process that identifies threats and then, defines countermeasures to prevent and resolve these. Here, threats include malicious attacks, incidental causes such as device errors, and potential threats that have not been identified. Various approaches to threat modeling have been studied and developed to identify and resolve such threats. Each of these approaches has advantages and disadvantages as shown in Table 2.

This study proposes a framework for using STRIDE threat modeling. It can be applied to general products, perform vulnerability analysis in a variety of circumstances (rather than particular ones), and provide specific guidelines. STRIDE threat modeling analyzes vulnerabilities based on product DFDs. Here, the analysis results are grouped into six categories. The major categories can be determined. The processes and systems where threats occur can be identified by analyzing vulnerabilities based on DFDs.

Table 2: Comparison of threat modeling approaches according to usage features

Function/threat modeling	STRIDE	PASTA	LIDDUN	DREAD	VAST	TVRA	STPA-sec
Application in general circumstances	O	O		O			
Applicable to various non-business products/systems	O		O	O	O		O
Provides specific guidelines	O	O	O	O			
Provides public automated analysis tools	O		O				
Can assign threat levels				O		O	O
Applicable to complete products (not in the development stage)	O		O		O	O	

3.3.2 Threat Libraries and Attack Scenarios

A threat library is developed to provide vulnerability analysis results that are more specific than those provided by STRIDE threat modeling. The threat library consists of threats in the developed software. It includes publicly known vulnerabilities. This study proposes a threat library configuration based on CVE. CVE provides a list of publicly known vulnerabilities. For the specific vulnerabilities that are established via the threat library, vulnerabilities that were analyzed through STRIDE threat modeling can be examined, and more specific security guidelines can be presented. The results of the STRIDE vulnerability analysis and the constructed threat library are analyzed to generate attack scenarios for understanding attack routes, attack times, etc. The attack scenarios illustrate vulnerabilities based on the principles of the cyber kill chain, analyzing the threats in terms of who, where, when, how, what, and why, stemming from the exploitation of weaknesses. The existing DFDs are validated by comparing them with the anticipated outcomes from the attack scenarios. DFDs are updated in areas identified with issues, leading to the generation of validated DFDs.

3.4 Establishing Security Guidelines with Scores

A vulnerability list for the product is generated through the created DFDs. This vulnerability list is integrated with CVSS scores, and average scores are generated based on categories such as the entire system, subsystems, modules, etc., corresponding to the functional aspects of the product. Ultimately, a comprehensive security guideline is formulated through a thorough analysis of vulnerabilities in the

product. Security guidelines are criteria for certifying the security of an application program. These can stabilize and improve the security of the product. In particular, security should be maintained and ensured as vulnerabilities and software functions are developed rapidly. Conventional methodologies perform expensive intrusion testing or provide guidelines only for certain functions related to security modules. Therefore, it is difficult to secure all the functions, and security certification becomes expensive. This study used a methodology based on public vulnerabilities and threat modeling. The security guidelines derived from these include all the functions of the software as a whole and can be used without security experts or attack simulators. Thereby, these can ensure security inexpensively. In addition, the products that are developed and maintained using the security guidelines proposed in this study are safe from the increasing number of new vulnerabilities and potential vulnerabilities that are not public. This study applied the proposed methodology to a VPN and formulated security guidelines for the VPN.

4 VPN Vulnerability Analysis and Security Guideline Formulation

4.1 VPN Analysis

We applied the framework proposed in this study to a VPN to demonstrate the practicality of the framework and verify the actual process of application [34–37]. A VPN extends a public network so that it behaves similarly to a private network. This enables the use of the public network such that the computing device functions as if it is connected directly to a private network. Recently, due to COVID-19, the number of companies permitting work from home has increased, and non-face-to-face technologies such as remote classes and remote meetings have grown. There has been a growth in VPN technology as well. This has enabled users to access companies' internal networks from home. This study analyzed vulnerabilities and formulated security guidelines for VPNs using the proposed framework.

4.1.1 VPN Operating Environment and Functions

The user applies the VPN client at home to connect to the company's computers. At this time, it is feasible to access the company's internal network via the VPN tunnel. The VPN comprises the operating environment shown in Fig. 2. Here, the VPN's security functions include information flow control, virtual channels, audit records, identification and authentication, security management, protection of transmitted data, self-testing, secure session management, protection of VPN clients, securing connections between VPN clients and servers, and password support. Identification and authentication, secure session management, and VPN client protection are secure functions upon logging in. Information flow control, audit records, and self-testing are functions that are secure throughout VPN usage. Other security functions include encryption to secure the VPN. Information flow control and virtual channels are the chief functions of the VPN. Information flow control permits only users with verified integrity to gain access. It must provide functions that permit communication only between communication partners specified by an authorized administrator. In addition, the gateway must block a variety of incoming/outgoing information according to rules set by the administrator and send only the permitted traffic through. Virtual channels provide a function that generates virtual variations for secure communication with communication partners. These should be capable of ending sessions when abnormal termination or access is detected.

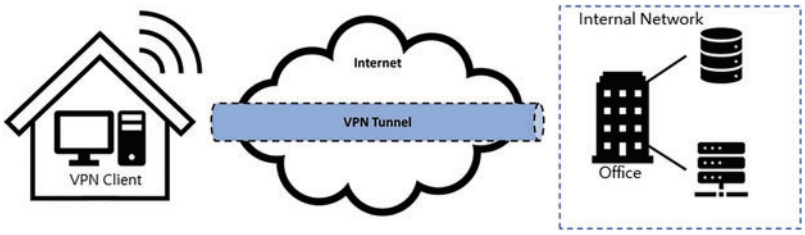


Figure 2: Operating environment of a VPN

4.1.2 VPN Data Flow Diagram

This study generated a two-stage DFD based on the VPN's functions. The Threat Modeling Tool (TMT) provided by Microsoft was used to generate the DFD. The TMT essentially provides DFD elements. The user can conveniently generate the desired elements, etc., and perform STRIDE analysis. The TMT is a customizable automation tool that can modify unanalyzed STRIDE vulnerability results. A DFD that incorporates VPN functions includes a login function and an encryption function.

In a VPN, the IP addresses are modified at the server VPN gateway shown in Fig. 3. The server VPN gateway is connected to the VPN server so that user audit records and administrator-specified rules can be set and the IP addresses of various users can be modified. At this time, the VPN server is connected to an SQL database to generate and store users' usage times, types, etc. The data that is transmitted to the server VPN gateway can be transmitted to the company via a modified IP address. Because the gateway is an individual device rather than a process, it is depicted as a data storage element. The login proposed in this study is authenticated via two-factor authentication (2FA) [37] as shown in Fig. 4. 2FA provides better security through a two-stage authentication. A typical example of this is the OTP authentication performed after logging in.

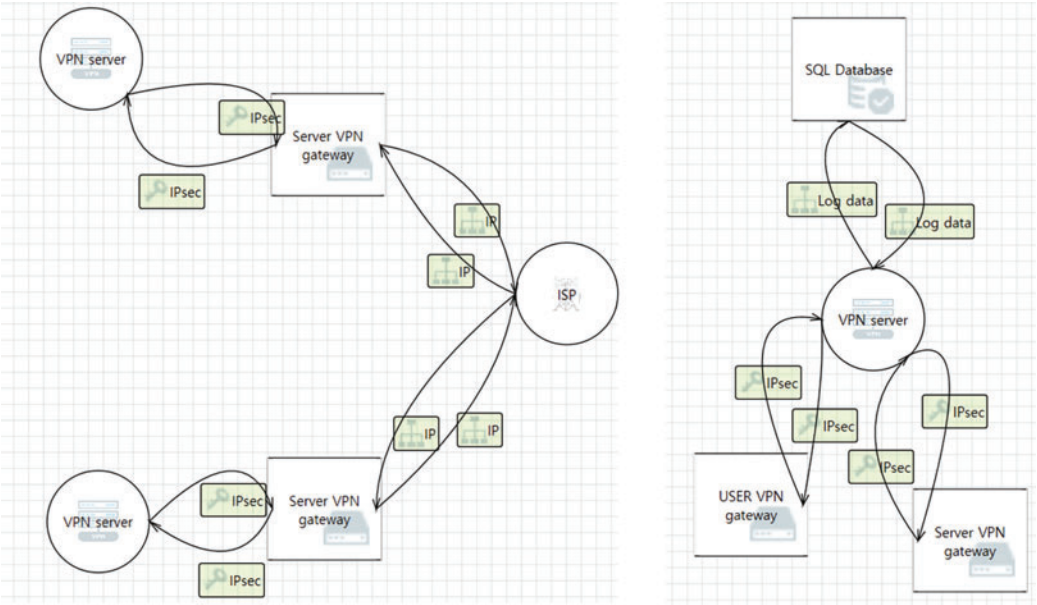


Figure 3: DFD of a VPN. Left: IP address alteration via VPN gateway. Right: database connected to VPN server

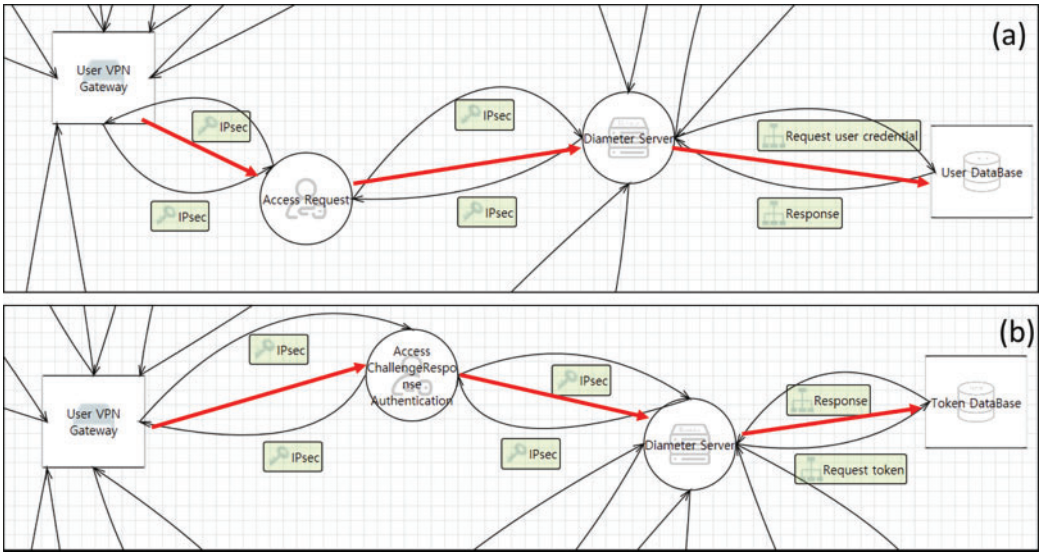


Figure 4: DFD of 2FA authentication: (a) first authentication, (b) second authentication

This study used an authentication protocol that performs authentication using a Diameter server. The User DB stores account information such as the IDs, passwords, names, etc., of users. The Token DB stores biometric information such as fingerprints and voice information, enabling 2FA. Users enter their information via the VPN client. A verification process occurs on the Diameter server to examine whether the information is correct. The VPN designed in this study is encrypted by IPsec [38,39]. A DFD of the encryption is shown in Fig. 5.

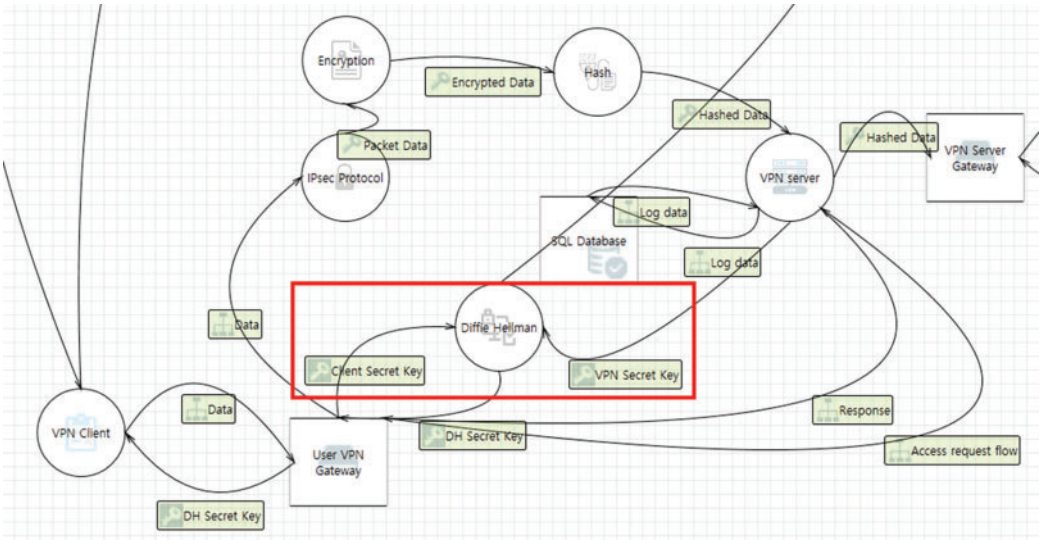


Figure 5: DFD of encryption

For encryption, the client's server key is entered, and encryption is performed using the Diffie–Hellman algorithm. At this time, a new secret key is generated to allow for a secure connection between the VPN server and the client. An AH is used to packetize the data. The key is encrypted and decrypted

via DES. Thereby, the transmitted data is protected. In addition, the transferred data is rejected when the key for the decrypted results at the final destination differs from the key sent earlier.

4.2 VPN Vulnerability Analysis

A total of 121 vulnerabilities were identified in the STRIDE analysis results that used the DFD designed in this study. The identified vulnerabilities included 14 spoofings, 20 tamperings, 1 repudiation, 31 information disclosures, 30 denials of service, and 25 elevations of privilege vulnerabilities. To examine the STRIDE analysis results and make them more specific, 803 CVEs were analyzed to examine 207 CVEs related to this study. Then, the STRIDE results and CVE results were linked as shown in Table 3.

Table 3: STRIDE analysis results and CVE matching results

DFD element type	Specific element	Interaction	STRIDE	STRIDE description	CVE
Process	VPN client	HTTPS	**S**	Spoofing the Human User Process	CVE-2020-5145
		Data	**S**	Spoofing the User VPN Gateway Process	CVE-2018-0333 CVE-2020-3583 CVE-2020-5137 CVE-2021-34704 CVE-2022-0342 CVE-2022-20745
(Omitted)					
Process	Diameter server	IPsec	**D**	Potential Excessive Resource Consumption for Access Request or Diameter Server	CVE-2002-1100
	Company server	HTTPS	**D**	Potential Excessive Resource Consumption for Company Gateway or Company Wed Server	CVE-2003-0260
	Diameter server	Request token	**I**	Authorization Bypass	CVE-2019-6143
		IPsec	**I**	Weak Access Control for a Resource	CVE-2013-5510
		IPsec	**D**	Potential Excessive Resource Consumption for Access Request or Diameter Server	CVE-2008-0915
		IPsec	**D**	Potential Process Crash or Stop for Diameter Server	CVE-2008-2733
	Encryption	Encrypted Data	**I**	Weak Authentication Scheme	CVE-2007-2332
(Omitted)					

(Continued)

Table 3 (continued)

DFD element type	Specific element	Interaction	STRIDE	STRIDE description	CVE
		Access request flow	**T**	The VPN server Data Store Could Be Corrupted	CVE-2017-12319
		Access request flow	**I**	Weak Authentication Scheme	CVE-2002-1099
		Data	**I**	Weak Authentication Scheme	CVE-2019-16651
		Access request flow	**D**	Potential Excessive Resource Consumption for User VPN Gateway or VPN server	CVE-2002-1101 CVE-2021-22985
Data storage	User VPN gateway	Data	**D**	Potential Excessive Resource Consumption for VPN Client or User VPN Gateway	CVE-2005-3409 CVE-2021-0228 CVE-2001-0428
		Data	**D**	Potential Process Crash or Stop for User VPN Gateway	CVE-2002-1102 CVE-2013-3415 CVE-2017-12319
		Access request flow	**D**	DoS via Buffer overflow	CVE-2021-45991 CVE-2004-0699
	Company web gateway	Decrypted Data	**D**	Potential Process Crash or Stop for Company Gateway	CVE-2017-12319
		Decrypted Data	**T**	The Company Gateway Data Store Could Be Corrupted	CVE-2017-12319
(Omitted)					
Data flow	Decrypted data flow	Decrypted Data	**D**	Data Flow Decrypted Data Is Potentially Interrupted	CVE-2018-0154

To make the CVE analysis results more specific, an attack scenario was prepared, and the attack route, etc., was analyzed. This is shown in Table 4.

Table 4: CVE attack scenario analysis results (several typical cases among several analyses)

CVE No.	Who	When	Where	What	How	Why
CVE-2021-0228	Attacker	When an abnormal or exception state vulnerability is examined ineffectively on a Juniper Networks MX series platform that has a Trio-based MPC deployed in an EVPN-VXLAN configuration	User VPN Gateway	Continuously sending certain Layer 2 traffic	By ineffectively examining the abnormal or exceptional state	To perform a DoS attack

(Continued)

Table 4 (continued)

CVE No.	Who	When	Where	What	How	Why
CVE-2020-27569	Remote attacker	When Aviatrix VPN Client 2.8.2 and previous versions are used	SQL database	Logs	By committing an error related to writing to a world-writable location	To obtain write access permission to all the system files
CVE-2021-22985	Malicious VPN user	When handling VPN traffic with APM in BIG-IP APM version 16.0.x	User VPN Gateway	Traffic	By consuming excessive memory with TMM	To perform a DoS attack on APM
CVE-2021-1297	Unauthenticated remote attacker	When the web-based management interface of Cisco Small Business RV160, RV160W, RV260, RV260P, and RV260W VPN routers is used	VPN client	System directory	By uploading specially-made files through an insufficient validation examination	To overwrite any files on the system
(Omitted)						
CVE-2001-0428	Attacker	When a Cisco VPN 3000 series device is used	User VPN Gateway	Specially altered IP packets that include IP option settings with ineffective options	By transmitting to a vulnerable device	To consume 100% of CPU resources and deny the device's services
CVE-2001-1499	Attacker	When CheckPoint VPN-1.4.1SP4 (which uses SecuRemote) is applied	Access request	Different error messages regarding valid IDs and inappropriate IDs	By an attacker determining a valid ID and performing a brute-force password attack	To acquire account information
CVE-2002-0047	Attacker	When the CIPE VPN package is used	VPN server	Small sequential packets in Crypto IP Encapsulation (CIPE)	By an error in the packet handling code	To stop the program
CVE-2002-1447	Attacker	When using the Cisco VPN Client for the Unix platform	Access request	User-provided arguments in VPN client commands	By insufficient bound examination and vulnerabilities	To cause a buffer overflow and obtain privileges on the system
CVE-2002-0853	Attacker	When the Cisco VPN client is used	IKE	IKE packets that have been altered by a remote attacker to have a payload with a length of zero	By sending these to vulnerable VPN clients and increasing the CPU usage to 100%	To consume CPU resources and deny service

4.3 VPN Security Guidelines

The final security guidelines were formulated based on the STRIDE vulnerability identification results and CVE analysis results. The security guidelines formulated in this study are for VPN security. The importance of each guideline is indicated by its CVSS, which is its CVE vulnerability score. This is shown in Table 5.

Table 5: Security guideline derivation

Element	Security guideline	CVSS
	Examine whether the VPN client's user personal information is encrypted when transmitted.	5.0 Medium
	Verify whether the VPN client executable file's functions have been altered.	7.2 High
	Examine whether the existing service configuration environment has been organized effectively in the VPN client.	7.2 High
	Verify whether the password is entered while executing the VPN client with administrator privileges.	7.2 High
	Examine whether there are errors related to privileges at the program level in the VPN client.	6.7 Medium
	Restrict the VPN client installation file path, and verify the execution file path while running the client.	7.4 High
	Examine whether the correct absolute path of the dynamic library that is used during VPN client execution is entered in the configuration file.	7.9 High
	Examine the privileges in the VPN client's program file execution and configuration settings.	7.6 High
	Examine whether the basic user can overwrite the binary file in the VPN client.	8.8 High
	Examine the execution files and configuration files of the dynamic libraries and external libraries that are executed while connecting via the VPN client.	7.3 High
	Verify the restrictions on the access range that is executed with administrator privileges while running the VPN client.	9.4 Critical
	Verify whether there have been no modifications to the path of the configuration file required to use the VPN client.	9.8 Critical
	Verify the values while passing parameters of functions that are used while running the VPN client.	7.4 High
	Examine whether the VPN client cannot extend the kernel.	7.8 High

(Continued)

Table 5 (continued)

Element	Security guideline	CVSS
	Verify whether continuous authentication is feasible to prevent hijacking while using the VPN client.	7.8 High
	Verify whether the VPN client's HTTP/S requests and responses are examined by the validation mechanism.	7.8 High
	Examine the security control related to privileges during communication between the VPN client and server.	8.8 High
	Examine scripts while executing VPN client processes.	5.5 Medium
	Verify the presence of a mechanism that detects whether the user has directly performed execution in the VPN client.	6.3 Medium
	Verify the validity of the VPN connection name received by the VPN client.	6.4 Medium
	Verify whether there are restrictions on the value and length of strings transmitted to the VPN client.	7.8 High
	Examine whether access privileges are restricted for each user type in the VPN client.	8.1 High
VPN client	Verify the target that accesses the tunnel before the user replies to the VPN client.	4.0 Medium
	Examine the IP address of the tunnel generated in the VPN client process.	6.5 Medium
	Examine whether the IPC message has been altered by performing validation in the VPN client.	5.5 Medium
	Examine the "Delete service data and reports" function in the VPN client.	7.8 High
	Examine the validation of the certificate received while connecting from the VPN client to a VPN device.	8.2 High
	Examine the TCP filter so that TCP packets are not altered in the VPN client.	5.0 Medium
	Examine the management session's VPN delete requests in the VPN client.	5.5 Medium
	Examine the privilege range of the management interface files that are stored by the VPN client process.	9.8 Critical

(Continued)

Table 5 (continued)

Element	Security guideline	CVSS
	Examine whether the validation of the management interface text in the VPN client is appropriate.	8.8 High
	Examine whether CLI commands have been altered in the VPN client.	7.8 High
	Examine the access privilege range of director files while executing the VPN client process.	7.6 High
	Examine whether file read requests have been altered in the VPN client.	7.5 High
	Examine whether there are accounts with no passwords that have administrator privileges in the VPN client.	9.8 Critical
	Examine whether two-factor authentication has been downgraded to one-factor authentication in the VPN client.	6.5 Medium
	Examine the appropriateness of the authentication access mechanism while accessing the VPN client process.	9.8 Critical
	Examine whether the initial VPN request was performed multiple times in the VPN client.	7.5 High
	Examine the connection list (hostname, path or cookie list, etc.) that is sent to the gateway in the VPN client.	5.0 Medium
	Examine the settings of the traffic status confirmation module used in the VPN client.	7.1 High
	Verify the drivers used by the VPN client.	2.1 Low
	Verify whether there are errors in the update function of the VPN client.	7.8 High
	Verify whether there are restrictions on the amount of memory that can be used by the VPN client.	5.5 Medium
	Verify whether the previous process ended normally while setting up the network of the VPN client process.	4.4 Medium
	Examine the data that was sent by the user in the VPN client process system.	5.5 Medium
	Upgrade the component security system of the VPN client process.	8.0 High
Access request	Examine whether the replies to VPN client login attempt failures are encrypted.	5.3 Medium

(Continued)

Table 5 (continued)

Element	Security guideline	CVSS
	Examine the function for restricting the number of VPN client logins.	6.4 Medium
	Examine the function for restricting the length during VPN client login input.	5.0 Medium
	Examine the function for restricting the length of VPN client user profile names.	7.2 High
Company server	Examine the company server process' ICMP packet handling validation.	5.0 Medium
Diameter server	Examine the authentication method of the Diameter server process (LDAP).	9.1 Critical
	Examine the response packet (AAA LDAP) analysis method of the Diameter server process.	4.3 Medium
	Examine the packet response results of the Diameter server process.	5.0 Medium
	Examine the login-attempt handling cookies of the Diameter server process.	6.4 Medium
	Examine the client-based VPN configuration settings of the Diameter server process.	7.1 High
	Examine the encryption algorithm of the encryption process.	9.0 High
IKE	Examine the packet handling method of the IKE process.	5.0 Medium
	Examine the memory management system of the IKE process.	8.6 High
	Examine the IKE process' IKE packet handling method, which includes the ISAKMP header.	5.0 Medium
	Examine the implementation method of the IKE process.	7.8 High
VPN server	Examine the VPN server process' authentication method when certificates are used.	7.4 High
	Examine the cookie validation standard of the VPN server process.	7.5 High
	Examine the command validation of the VPN server process.	7.2 High
	Examine the delete command validation of the VPN server process.	5.5 Medium
	Upgrade the component security system of the VPN server process.	8.0 High
	Examine the file privileges while using policy files in the VPN server process.	9.0 High

(Continued)

Table 5 (continued)

Element	Security guideline	CVSS
	Examine the command usage privileges while using FTF commands in the VPN server process.	5.0 Medium
	Examine the privileges regarding file usage while applying VPN configuration files in the VPN server process.	7.5 High
	Examine the information request handling method of the VPN server process.	4.3 Medium
	Examine the valid packet response handling method in the VPN server process.	5.0 Medium
	Examine the packet handling code handling method in the VPN server process.	5.0 Medium
	Examine the VPN session handling method of the VPN server process.	5.9 Medium
	Examine the error handling method of the VPN server process.	5.0 Medium
	Examine the HTTP packet handling method of the VPN server process.	7.8 High
	Examine the TCP packet handling method of the VPN server process.	7.1 High
	Examine the XEE payload handling method of the VPN server process.	7.5 High
	Examine the password helper parameter handling method of the VPN server process.	7.8 High
	Examine the IP address handling method of the VPN server process.	8.8 High
	Examine the HTTP request handling method of the VPN server process.	6.1 Medium
Company gateway	Examine the BGP packet handling method when the company gateway's BGP session is used.	5.9 Medium
Server VPN gateway	Examine the message handling method in the server VPN gateway.	7.7 High
	Examine the message input and output handling method in the server VPN gateway.	5.5 Medium
	Examine the traffic and traffic handling verification method in the server VPN gateway.	6.4 Medium
	Examine the data transmission method in the server VPN gateway.	5.2 Medium
	Examine the parameter verification method in the server VPN gateway.	7.6 High
	Examine the packet verification method in the server VPN gateway.	5.0 Medium

(Continued)

Table 5 (continued)

Element	Security guideline	CVSS
	Examine the system connection verification method in the server VPN gateway.	5.0 Medium
	Examine the packet format verification method in the server VPN gateway.	6.6 Medium
	Examine the IKE packet handling method in the server VPN gateway.	6.6 Medium
	Examine the request handling method for services within authenticated devices in the server VPN gateway.	6.7 Medium
User VPN gateway	Examine the handling method in the options of the user VPN gateway.	5.0 Medium
	Examine the web page HTML verification method in the user VPN gateway.	5.0 Medium
	Examine the data and parameter handling method in the user VPN gateway.	6.5 Medium
	Examine the ID handling method used by the VPN client in the user VPN gateway	5.0 Medium
	Examine the transmission data handling method in the user VPN gateway.	7.0 High
	Examine the verification method for client connections in the user VPN gateway.	6.4 Medium
Company web gateway	Examine the data and parameter handling methods in the company web gateway.	5.9 Medium
User database	Examine the SQL verification method used by the database in the user database.	7.5 High
	Examine the authentication key verification method in the user database.	7.2 High
	Examine the verification method for parameters that can elevate privileges in the user database.	8.3 High
	Examine the verification method for valid IDs in the user database.	9.3 Critical
SQL database	Examine the log data handling method in the SQL database.	7.5 High
	Examine the SQL injection parameter verification method in the SQL database.	7.5 High
Decrypted data flow	Examine whether data transmission to the device is verified in the decrypted data flow.	7.5 High

5 Conclusion

The primary contribution of this research is the design of a framework that enables the creation of entire and comprehensive real-world security guidelines, encompassing the systematic vulnerability

analysis of practical software using automated tools within a vulnerability assessment framework. This approach facilitates more efficient and rapid analysis by introducing a structured method for discovering, classifying, and prioritizing vulnerabilities. It plays a crucial role in enhancing the security posture of organizations and mitigating the risk of exploitation. The distinctive features of this approach are expected to contribute to the development of more realistic and effective security measures, serving as a practical and systematic solution in contrast to existing vulnerability assessment methods.

This study proposes a vulnerability analysis framework suitable for software or systems. The framework was used to intelligently establish security guidelines in products with threat scores using an automated tool, verify functionalities of the software, and integrate the latest vulnerability content and scores. The entire software system and its usage environment (including the software's features and operating environment) were analyzed. Threat modeling was applied to identify vulnerabilities therein. The results of vulnerability identification by threat modeling can help identify potential threats and anticipate new threats.

The security guidelines formulated ultimately can secure all general software features, rather than being limited to certain special features. The proposed framework can also generate customized security guidelines for specific sections, modules, and algorithms in the entire software product. By identifying the primary security vulnerabilities in the entire software system, the security guidelines generated through the proposed framework help to reinforce specifically vulnerable areas. In addition, the suggested framework can be applied across various fields to develop structured and systematic security guidelines.

The framework proposed in this study leverages the Microsoft STRIDE threat modeling tool. However, effective utilization of this tool requires extensive training, and a drawback is that users must possess prior knowledge to comprehend and deploy it. Additionally, the validation process for adding new modules to the tool, such as those on the Microsoft STRIDE tool, may present challenges and prove time-consuming. To address this limitation, our future focus will be on defining reusability to save time and creating easily accessible common security modules.

Acknowledgement: The authors acknowledge the researchers at the Electronics and Telecommunications Research Institute (ETRI) in South Korea who supported this research.

Funding Statement: This work is the result of commissioned research project supported by the Affiliated Institute of ETRI (2022-086) received by Junho Ahn. This research was supported by the National Research Foundation of Korea (NRF) Basic Science Research Program funded by the Ministry of Education (No. 2020R1A6A1A03040583) and this work was supported by Korea Institute for Advancement of Technology (KIAT) Grant funded by the Korea government (MOTIE) (P0008691, HRD Program for Industrial Innovation).

Author Contributions: Research Design: D. Kim, N. Kim, J. Ahn; Data Collection: D. Kim, J. Ahn; Data Analysis and Results Integration: D. Kim, J. Ahn; Drafting of the Manuscript: D. Kim, J. Ahn; Visualization: D. Kim, J. Ahn. All authors reviewed and validated the final manuscript.

Availability of Data and Materials: The data used in the research can be made available to the corresponding author upon request within the authorized scope by ETRI.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] X. Zhang, C. Xia, T. Ma, L. Zhang, and Z. Jin, "Optimizing energy-latency tradeoff for computation offloading in SDIN-enabled MEC-based IIoT," *KSII Trans. Internet. Inf.*, vol. 16, no. 12, pp. 4081–4098, Dec. 2022. doi: 10.3837/tiis.2022.12.017.

[2] L. Li, H. Chi, K. Xie, and D. Chan, "Mobility-sensitive multicast protocol in NEMO," *KSII Trans. Internet Inf. Syst.*, vol. 16, no. 6, pp. 1994–2017, Jun. 2022. doi: 10.3837/tiis.2022.06.012.

[3] X. Liang, Y. Wu, Y. Huang, D. W. K. Ng, P. Li and Y. Yao, "Performance optimization and analysis on P2P mobile communication systems accelerated by MEC servers," *KSII Trans. Internet. Inf. Syst.*, vol. 16, no. 1, pp. 188–210, Jan. 2022. doi: 10.3837/tiis.2022.01.011.

[4] Datareportal, "Digital 2022: Global overview report," 2022. Accessed: Apr. 03, 2023. [Online]. Available: https://datareportal.com/reports/digital-2022-global-overview-report

[5] Precedence Research, "Capacitor market-global industry analysis, size, share, growth, trends, regional outlook, and forecast 2023-2032," 2023. Accessed: Apr. 05, 2023. [Online]. Available: https://www.precedenceresearch.com/capacitor-market

[6] ReportLinker, "Software products global market report 2023," 2023. Accessed: Apr. 07, 2023. [Online]. Available: https://www.reportlinker.com/p06246415/Software-Products-Global-Market-Report.html

[7] CVE Details, "Browse vulnerabilities by date," 2023. Accessed: Mar. 03, 2023. [Online]. Available: https://www.cvedetails.com/browse-by-date.php

[8] M. Anderson and F. Bajak, *Cyberattack on US Pipeline is Linked to Criminal Gang*. New York, NY, USA: AP News, 2021. Accessed: Apr. 15, 2023. [Online]. Available: https://apnews.com/article/europe-hacking-government-and-politics-technology-business-333e47df702f755f8922274389b7e920

[9] R. Lakshmanan, *Samsung Admits Data Breach that Exposed Details of Some U.S. Customers*. Venice, CA, USA: The Hacker News, 2022. Accessed: Mar. 19, 2023. [Online]. Available: https://thehackernews.com/2022/09/samsung-admits-data-breach-that-exposed.html

[10] L. Mauri and E. Damiani, "Modeling threats to AI-ML systems using STRIDE," *Sens.*, vol. 22, no. 17, pp. 6662, Sep. 2022. doi: 10.3390/s22176662.

[11] P. Datta, S. Sartoli, L. F. Gutierrez, F. Abri, A. S. Namin and K. S. Jones, "A user-centric threat model and repository for cyber attacks," in *37th ACM/SIGAPP Symp. Appl. Comput. (SAC '22)*, New York, NY, USA, 2022, pp. 1341–1346.

[12] I. Oh *et al.*, "Derivation of security requirements of smart TV based on STRIDE threat modeling," *J. Korea Institute Inform. Secur. Cryptol.*, vol. 30, no. 2, pp. 213–230, Apr. 2020.

[13] E. Park and S. Kim, "Derivation of security requirements of smart factory based on STRIDE threat modeling," *J. Korea Institute Inform. Secur. Cryptol.*, vol. 27, no. 6, pp. 1467–1482, Dec. 2017.

[14] S. Shim, S. Im, H. Ryu, S. Jun, and T. Ki, "Threat analysis of the smart doorlock systems using threat modeling," *J. Korean Institute Commun. Inform. Sci.*, vol. 45, pp. 11, Nov. 2020.

[15] M. H. M. Zaharuddin, R. A. Rahman, and M. Kassim, "Technical comparison analysis of encryption algorithm on site-to-site IPSec VPN," in *2010 Int. Conf. Comput. Appl. Indust. Electron.*, Kuala Lumpur, Malaysia, 2010, pp. 641–645.

[16] H. Mao, L. Zhu, and H. Qin, "A comparative research on SSL VPN and IPSec VPN," in *2012 8th Int. Conf. Wireless Commun., Netw. Mobile Comput.*, Shanghai, China, 2012, pp. 1–4.

[17] B. K. Chawla, O. P. Gupta, and B. K. Sawhney, "A review on IPsec and SSL VPN," *Int. J. Sci. Eng. Res.*, vol. 5, no. 11, pp. 21–24, Nov. 2014.

[18] CVE, "Common vulnerabilities and exposures," 2023. Accessed: Mar. 25, 2023. [Online]. Available: https://cve.mitre.org/

[19] Common Criteria, "CC:2022 release 1," 2022. Accessed: March 12, 2023. [Online]. Available: https://www.commoncriteriaportal.org/cc/

[20] Teron Labs, "CC evaluation institute," 2023. Accessed: March 18, 2023. [Online]. Available: https://www.teronlabs.com/
[21] Canadian Centre for Cyber Security, "Canadian common criteria scheme," 2023. Accessed: Apr. 21, 2023. [Online]. Available: https://www.cyber.gc.ca/en/tools-services/common-criteria
[22] NIAP, "National information assurance partnership," 2023. Accessed: May 03, 2023. [Online]. Available: https://www.niap-ccevs.org/
[23] L. H. Yeo and J. Banfield, "Human factors in electronic health records cybersecurity breach: An exploratory analysis," *Perspect. Health. Inf. Manag.*, vol. 19, no. 2, pp. 1–10, 2022.
[24] Y. I. Khan and M. U. Ndubuaku, "Ontology-based automation of security guidelines for smart homes," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, Singapore, 2018, pp. 35–40.
[25] F. Falconi, C. Zapata, A. Moquillaza, and F. Paz, "Security guidelines for the design of ATM interfaces," in *AHFE, 2020 Virtual Conf.*, Verlag, San Diego, USA, Springer, 2020, pp. 265–271.
[26] Microsoft, "Microsoft threat modeling," 2022. Accessed: Feb. 03, 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats
[27] T. UcedaVelez and M. M. Morana, *Risk centric threat modeling: process for attack simulation and threat analysis*. NJ. USA: Wiley Publishing, 2015. [Online]. Available: https://dl.acm.org/doi/10.5555/2834500.
[28] K. Wuyts and W. Joosen, *LINDDUN Privacy Threat Modeling: A Tutorial*. Leuven, Belgium: Department of Computer Science, KU Leuven, 2015. Accessed: Feb. 09, 2023. [Online]. Available: https://kuleuven.limo.libis.be/discovery/search?query=any,contains,LIRIAS1652885&tab=LIRIAS&search_scope=lirias_profile&vid=32KUL_KUL:,Lirias&offset=0
[29] M. Curphey, J. Scambray, and E. Olson, *Improving Web Application Security*. USA: Microsoft, 2003. Accessed: Feb. 15, 2023. [Online]. Available: https://download.microsoft.com/download/d/8/c/d8c02f31-64af-438c-a9f4-e31acb8e3333/threats_countermeasures.pdf
[30] C. Alberts, A. Dorofee, J. Stevens, and C. Woody, *Introduction to the OCTAVE Approach*. PA, USA: Software Engineering Institute, Carnegie Mellon University, 2003. Accessed: Feb. 02, 2023. [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=51546
[31] ETSI, "European Telecommunications Standard Institute: Telecommunications and internet converged services and protocols for advanced networking (TISPAN); Methods and protocols; Part 1: Method and proforma for threat, risk, vulnerability analysis, ETSI TS 165-1 V5.2.3," 2017. Accessed: Mar. 02, 2023. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/102100_102199/10216501/04.02.03_60/ts_10216501v040203p.pdf
[32] ThreatModeler, "Threat modeling methodologies: What is VAST?," Accessed: Oct. 09, 2018. [Online]. Available: https://threatmodeler.com/threat-modeling-methodologies-vast/
[33] W. Young and N. Leveson, "Systems thinking for safety and security," in *29th Annu. Comput. Secur. Appl. Conf.*, New Orleans LA. USA, ACM, 2013, pp. 1–8.
[34] J. Zhang, "Research on key technology of VPN protocol recognition," in *2018 IEEE Int. Conf. Safety Produce Inf. (IICSPI)*, Chongqing, China, 2018, pp. 161–164.
[35] Y. k. Kang, D. W. Kim, T. W. Kwon, and J. R. Choi, "An efficient implementation of hash function processor for IPSEC," in *IEEE Asia-Pacific Conf. ASIC*, Taipei, Taiwan, 2002, pp. 93–96.
[36] Z. Wang and L. Cao, "Implementation and comparison of two hash algorithms," in *Int. Conf. Comput. Inf. Sci.*, Shiyang, China, 2013, pp. 721–725.
[37] P. N. Thanh and K. Kim, "Implementation of open two-factor authentication service applied to virtual private network," in *Int. Conf. Inf. Netw. 2013 (ICOIN)*, Bangkok, Thailand, 2013, pp. 135–140.
[38] B. Santoso, A. Sani, T. Husain, and N. Hendri, "VPN site to site implementation using protocol L2TP and ipsec," *Teknokom*, vol. 4, no. 1, pp. 30–36, 2021. doi: 10.31943/teknokom.v4i1.59.
[39] A. A. Al-khatib and R. Hassan, "Impact of IPSec protocol on the performance of network real-time applications: A review," *Int. J. Netw. Secur.*, vol. 20, no. 5, pp. 811–819, 2018.