JISEC

Issued on March 4, 2013

Vulnerability Assessment Guide for Developers

It has become a major issue for developers to ensure that the security functions of IT products properly work as well as to eliminate flaws (vulnerabilities) that can be exploited for attacks against the IT products.
An international standard for IT security evaluation, Common Criteria, defines the vulnerability assessment methodology.
This document explains important notes when performing searches to detect vulnerabilities and conducting tests to confirm vulnerabilities.

Information Security Certification Office
Security Center, Technology Headquarters, IPA
www.ipa.go.jp/security/jisec/

- Table of Contents -

1 Introduction

It has become a major issue for developers to eliminate flaws that can be exploited for attacks (i.e., vulnerabilities) from IT products.

An international standard for IT security evaluation, Common Criteria (hereinafter referred to as "CC") defines the evaluation methodology for examining the security functions in the evaluated IT product to ensure that it does not have any vulnerability that can be exploited. This document is prepared for introducing the vulnerability assessment methodology of the CC, as well as for explaining important notes when you perform searches to detect vulnerabilities and conduct the penetration testing to confirm the possibility of being exploited for attacks.

The concept for the CC vulnerability assessment will be effective not only for evaluators conducting the CC evaluation, but also for security-related reviews and quality tests for IT products among the developers. We hope that this document will help the readers to understand the CC and serve as a useful reference in various efforts for improving the security of IT products.

1.1 Target readers of this document

The target readers of this document are assumed to be developers engaged in the design, implementation, and testing of the security functions in IT products, as well as evaluators conducting CC-based vulnerability assessment.

As an explanation of the CC standards, this document is described primarily for evaluators conducting CC-based evaluations. However, the details of vulnerability assessment should be taken into account in the course of development. Developers can apply the contents of this document to their own IT product development by interpreting the "evaluators" described in this document as developers themselves.

1.2 Organization of this document

This document consists of the following six chapters.

■ Chapter 1 Introduction

This chapter explains the objectives and the target readers of this document.

■ Chapter 2 Overview of vulnerability assessment

This chapter explains the overview of the CC vulnerability assessment as a whole.

■ Chapter 3 Attack potential

This chapter explains "attack potential," which is the pass/fail verdict criteria of the CC vulnerability assessment, indicating concrete attack scenarios with the calculation example of their attack potential.

■ Chapter 4 Vulnerability search

This chapter explains the overview of the vulnerability search required for the CC, as well as the reference information and the important notes in association with the search for vulnerabilities in practice.

■ Chapter 5 Penetration testing

This chapter explains the reference information and the important notes with respect to the penetration testing for determining whether or not the assumed vulnerabilities exist in reality.

■ Chapter 6 Conclusion

This chapter explains the important notes with respect to the contents explained in this document as a whole.

1.3 Common Criteria standards documents

The evaluation criteria and evaluation methodology of this guide are based on the standards documents listed in Table 1-1 and Table 1-2 below. The evaluation criteria and evaluation methodology are referred to as "CC" and "CEM," respectively, in their abbreviations.

Table 1-1 CC/CEM standards documents (Japanese translation versions)

CC/CEM version 3.1 Release 4 (CC/CEM v3.1 Release 4)
Evaluation criteria: Common Criteria for Information Technology Security Evaluation (CC version 3.1 Release 4)
Part 1: Introduction and general model Version 3.1 Revision 4 [Japanese version 1.0]
Part 2: Security functional components Version 3.1 Revision 4 [Japanese version 1.0]
Part 3: Security assurance components Version 3.1 Revision 4 [Japanese version 1.0]
Evaluation methodology: Common Methodology for Information Technology Security Evaluation (CEM Version 3.1 Release 4)
Evaluation methodology Version 3.1 Revision 4 [Japanese version 1.0]

Table 1-2 CC/CEM standards documents (Original versions)

CC/CEM v3.1 Release 4		
Evaluation criteria: Common Criteria for Information Technology Security Evaluation (CC v3.1 Release 4)		
Part 1: Introduction and general model	Version 3.1	Revision 4
Part 2: Security functional components	Version 3.1	Revision 4
Part 3: Security assurance components	Version 3.1	Revision 4
Evaluation methodology: Common Methodology for Information Technology Security Evaluation (CEM v3.1 Release 4)		
Evaluation methodology	Version 3.1	Revision 4

This document is based on the contents stated in the sections of CEM [1] below, which are taken from the CC and CEM standards documents.

- CEM, "15 Class AVA: Vulnerability assessment"
- CEM, "Annex B Vulnerability Assessment (AVA)"

1.4 Terms and definitions

Table 1-3 shows the terms used in this document in relation to the CC and CEM.

Table 1-3 CC/CEM terms and definitions

Terms	Explanation
CC (Common Criteria)	The international standard ISO/IEC 15408 for evaluating whether an IT product has been properly designed and the design has been accurately implemented from the viewpoint of information security.
CEM (Common Evaluation Methodology)	An evaluation methodology defined for conducting the CC-based security evaluation in a homogeneous manner. Items that should be evaluated and the perspectives of evaluation to satisfy the CC standards are defined.
EAL (Evaluation Assurance Level)	The degree of the assurance in the CC-based security evaluation. Seven levels from EAL 1 to EAL 7 are defined. The higher the EAL is, the wider scope of design information of the product is evaluated strictly.
Security Target	A document describing the evaluated IT product for the CC-based security evaluation. It includes the description of the evaluation scope of the IT product, the assumptions, the security functions of the evaluation target, and the evaluation assurance level. The Security Target is prepared by the developer of the IT product on the basis of the requirements of the procurement personnel of the IT product.

2 Overview of vulnerability assessment

This chapter explains the overview of the CC vulnerability assessment as a whole.

2.1 Characteristics of the CC vulnerability assessment

In a CC-based security evaluation for an IT product, the evaluator examines the IT product and its documentation including the design documents and user guidance to evaluate whether or not the security functions have been accurately implemented, as well as whether or not the product has any vulnerabilities.

Apart from the CC evaluation, there are a large number of commercial-based security examination services in the field of vulnerability assessment. However, the methods and quality of those examinations vary depending on the service provider. In the CC, on the other hand, the evaluation methodology has been determined so that the same results can be derived from as long as evaluations are conducted by evaluators at the accredited Evaluation Facilities. In addition, the evaluation methodology is considered to avoid the evaluators' excessive demands for the security functions of the products more than necessary for the procurement personnel of the products.

Therefore, the CC vulnerability assessment has some differences from typical security examination services. The major differences are as follows:

■ Consideration to developer design information

In typical security examination services, black box tests are conducted based primarily on the already-known vulnerability information. In the CC, on the other hand, the evaluator examines design documents, etc., presented by the developer, in addition to the already-known vulnerability information, before screening out the vulnerabilities that the products may have, and then confirm such vulnerabilities in testing.

The documentation that the developer shall present has been prescribed for each evaluation assurance level (EAL specified in the Security Target) required by the procurement personnel of the products. Taking a design document of a product as an example, only the external interface specifications are evaluated at EAL 1, the design inside the product is added at EAL 2 or higher, and the source codes are further added at EAL 4 or higher; the higher the EAL is, the wider scope of documentation is evaluated in detail.

■ **Consideration to the attack potential required (difficulty in attacks)**

In general, the degree of attacks that the security functions of a product should be resistant to varies depending on the intended usage, etc., by the procurement personnel. Allowing for such circumstances, pass/fail verdicts of the CC are determined to the vulnerabilities detected in the evaluation with consideration given to the difficulty in being exploited for actual attacks and the requirements from the procurement personnel of the product.

More specifically, the CC prescribes the attacker's potentials that the evaluated product shall be resistant to for each evaluation assurance level (EAL specified in the Security Target) required by the procurement personnel. For instance, even if the evaluated product has a vulnerability, the product can pass the CC evaluation as long as the attack potential required for an attacker to succeed in attacks by exploiting the vulnerability exceeds the reference value prescribed for each of the EALs (i.e., it is difficult to succeed in attacks by exploiting the vulnerability in practice), meaning that the evaluated product has the resistance necessary for attacks.

■ **Consideration to the assumptions**

In the CC, the assumptions to the operational environment for the procurement personnel of the product (assumptions specified in the Security Target) are taken into consideration. For instance, even if the evaluated product has a problem that can be technically regarded as a vulnerability, the product can pass the CC evaluation as long as the attacks exploiting the problem can be prevented by means of operational measures in accordance with the assumptions.

■ **Limitation to vulnerabilities**

In the CC, vulnerability means the compromise of the security functions required by the procurement personnel of the product (security functions specified as the evaluation target in the Security Target). For instance, even if the evaluated product has a problem that may cause a denial-of-service, the problem will not be regarded as a vulnerability in the CC evaluation as long as the problem does not compromise the security functions specified in the Security Target.

2.2 Flow of the CC vulnerability assessment

Figure 2-1 shows the flow of the CC-based vulnerability assessment.

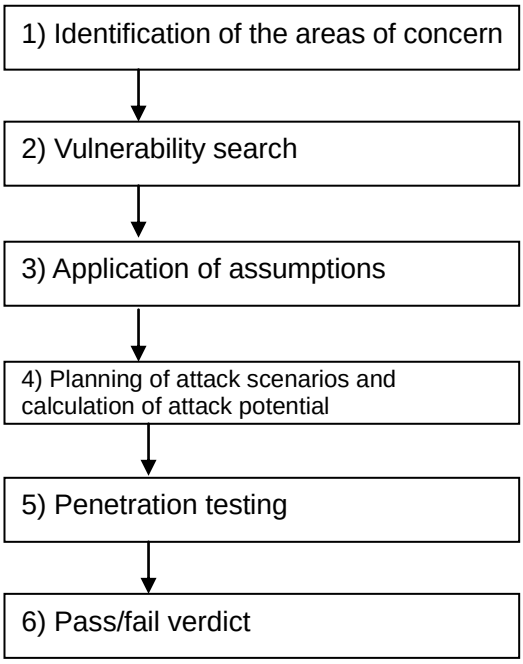

Figure 2-1 Flow of vulnerability assessment

The overview of each item in Figure 2-1 is as follows:

1) Identification of the areas of concern

An "area of concern" denotes a part that the evaluator recognizes the necessity to investigate the vulnerability in detail in the design and implementation of the evaluated products. In the CC evaluation, the evaluator examines the documentation, including the Security Target, design documents, and user guidance, prior to vulnerability analysis. In the examination, the evaluator identifies potential processing in which security problems may occur. Such processing parts, regarded as "areas of concern," will be used as the input to the subsequent vulnerability search section.

2) Vulnerability search

The evaluator searches for potential vulnerabilities that the evaluated products may have on the basis of the "areas of concern" in the evaluated products and adopted techniques, etc., to make a list of such vulnerabilities. The vulnerability search is conducted in a combination of the search for generally-known vulnerabilities (hereinafter referred to as "public domain vulnerabilities") using search

engines on the Internet, and the analysis of the documentation of the evaluated products.

Note that the evaluator may additionally identify another potential "area of concern" in which security problems may occur while analyzing the documentation in the vulnerability search. In such a case, the evaluator searches for vulnerabilities in a combination of the search for public domain vulnerabilities and the analysis of the documentation with respect to the additionally identified "area of concern" as well.

3) Application of assumptions

The evaluator analyzes the possibilities of the vulnerabilities listed during the vulnerability search to see whether or not they are applicable to the evaluated products in operational environments in which the assumptions are fulfilled. Vulnerabilities that will not occur in the operational environment, in which the assumptions are fulfilled, are excluded from the target of the analysis in this stage even if they may technically exist in the product. The rest of the vulnerabilities becomes the target of the subsequent analysis.

4) Planning of attack scenarios and calculation of attack potential

With respect to the vulnerabilities that the evaluated products may have, the evaluator plans attack scenarios exploiting such vulnerabilities and calculates the attack potential of the attacker required for executing those attack scenarios (hereinafter referred to as "attack potential"). The attack scenarios will be confirmed in the subsequent penetration testing. Note that the penetration testing does not need to be conducted when the attack potential required for exploiting the vulnerabilities obviously exceeds the reference value prescribed for each of the EALs (i.e., it is obviously difficult to succeed in attacks).

5) Penetration testing

The evaluator plans the test items on the basis of the attack scenarios and carries out the penetration testing for examining whether or not the assumed vulnerabilities are actually contained in the evaluated products.

6) Pass/fail verdict

The products will pass the CC evaluation when the evaluated products in the operational environment, in which the assumptions are fulfilled, have no applicable vulnerabilities or when the penetration testing did not succeed.

The success of the penetration testing on the evaluated products means that the products have the assumed vulnerabilities in reality. The pass or fail of the CC evaluation in this case will be determined by comparing the value of attack potential required for exploiting the vulnerabilities and the reference value prescribed to each of the EALs. The evaluator recalculates the value of attack potential in view of the efforts involved from the planning of the penetration testing to the success of the attacks, and then determines a pass or fail verdict.

3 Attack potential

This chapter explains attack potential that is used for the pass/fail verdict of the CC vulnerability assessment.

3.1 Definition of attack potential

An attack potential is a numerically expressed attacker's potential that is required for executing attack scenarios for exploiting vulnerabilities. An attack potential is expressed as the sum of the numerical values calculated for each of the five factors in Table 3-1. Note that the "attack" in Table 3-1 includes all the attacker's efforts, such as the attacker's discovery of the vulnerabilities, the device of the attack methods for exploiting the vulnerabilities, and the success of attacks in practice.

Table 3-1 Factors of attack potential

Factor	Description
Elapsed time	It refers to the time required for the attack. The value is weighted in accordance with the elapsed time, such as "less than one day" (value: 0), "between one day and one week" (value: 1), "between one week and two weeks" (value: 2), and "between two weeks and one month" (value: 4).
Specialist expertise	It refers to the generic technical knowledge required for the attack. The value is weighted in accordance with the level of knowledge, such as "layman" (value: 0), "proficient person" (value: 3), and "expert" (value: 6).
Knowledge of evaluation target	It refers to the knowledge in the design and operation of the target product that is required for the attack. The value is weighted in accordance with the difficulty in obtaining the product information, such as "public information" (value: 0), "restricted information" (value: 3), and "sensitive information" (value: 7).
Window of opportunity	It refers to the access opportunity to the target product that is required for the attack. The value is weighted in accordance with the difficulty involved in accessing the product without the attack being noticed until the success of the attack, such as "unnecessary/unlimited access" (value: 0), "easy access" (value: 1), "moderate access" (value: 4), and "difficult access" (value: 10).
Equipment	It refers to the software or hardware required for the attack. The value is weighted in accordance with the difficulty in obtaining the equipment, such as "standard equipment" (value:0), "specialized equipment" (value: 4), and "bespoke equipment" (value: 7).

For the details of attack potentials, refer to CEM [1], "Annex B.4 Calculating attack potential."

3.2 Pass/fail verdict of vulnerability assessment

In the CC, the value of attack potential that a product shall be resistant to even when it is attacked is prescribed to each of the EALs. For instance, it is "Basic" (value: 0 to 9) at EAL 1 to EAL 3, and "Enhanced-Basic" (value: 10 to 13) at EAL 4.

The pass or fail verdict of the CC vulnerability assessment is determined by comparing the attack potential required for exploiting the vulnerabilities detected in the CC evaluation with the reference value prescribed to each of the EALs.

- When the attack potential required for exploiting vulnerabilities exceeds the reference value

  This means that it is difficult for attackers to succeed in attacks by exploiting the vulnerabilities. Even when such vulnerabilities are detected in a product, the product can pass the CC evaluation in spite of the existence of the vulnerabilities in the product, because the reference value that the product shall be resistant to has been satisfied.

- When the attack potential required for exploiting vulnerabilities falls short of the reference value

  This means that it is easy to succeed in attacks. If such vulnerabilities are detected in a product, the product will fail the CC evaluation, because the reference value that the product shall be resistant to has not been satisfied.

3.3 Examples of the calculation of attack potentials and pass/fail verdicts

This section explains the calculation of attack potentials and pass/fail verdicts of vulnerability assessment with concrete examples.

- Example 1

  An expert who is conversant in IT technology would be able to succeed in an attack within two weeks by modifying and applying an attacking tool available on the Internet.

Table 3-2 shows an example of calculation of the attack potential in this case. Note that the most unfavorable conditions for the attacked product are applied to the factors that are not explicitly described in the example 1.

Table 3-2 Calculation example of attack potential (when a modified tool is used)

Factor		Value
Elapsed time	Two weeks or less	2
Specialist expertise	Expert	6
Knowledge of evaluation target	Public	0
Window of opportunity	Unlimited access	0
Equipment	Specialized (modified tool)	4
Total (attack potential)		12

Table 3-2 indicates that the attack potential required for exploiting this vulnerability is 12 (="Enhanced-Basic"), thus resulting in the pass/fail verdict of vulnerability assessment as follows:

- At EAL 1 to EAL 3 (must be resistant to "Basic" attacks): Pass
- At EAL 4 (must be resistant to "Enhanced-Basic" attacks): Fail

■ Example 2

An experienced person would be able to succeed in an attack with ease by obtaining and applying an attack tool published on the Internet.

Table 3-3 shows an example of calculation of the attack potential in this case. Note that the most unfavorable conditions for the attacked product are applied to the factors that are not explicitly described in the example 2.

Table 3-3 Calculation example of attack potential (when a public domain tool is used)

Factor		Value
Elapsed time	Less than one day	0
Specialist expertise	Proficient	3
Knowledge of evaluation target	Public	0
Window of opportunity	Unlimited access	0
Equipment	Standard (tool obtained on the Internet)	0
Total (attack potential)		3

Table 3-3 indicates that the attack potential required for exploiting this vulnerability is 3 (="Basic"), thus resulting in the pass/fail verdict of vulnerability assessment as follows:

- At EAL 1 to EAL 3 (must be resistant to "Basic" attacks): Fail
- At EAL 4 (must be resistant to "Enhanced-Basic" attacks): Fail

3.4 Important notes for calculating attack potentials

The following matters have to be taken into account for calculating attack potentials.

■ Consideration to the availability of the information and tools for attacks

The difference between the example 1 and the example 2 in the previous section is whether an attacking tool directly applicable to the product is available on the Internet or not. In this way, the difficulty in succeeding in an attack (attack potential) significantly varies depending on whether or not the information about attack tools as well as the concrete methods and procedures for exploiting vulnerabilities is available on the Internet, regardless of whether the tool itself uses a sophisticated method. Therefore, the evaluator has to calculate the attack potential after thoroughly checking on the Internet, etc., for the availability of public information and tools for attacks.

■ Consideration to multiple attack scenarios for a single vulnerability

There is no guarantee that there is only a single attack method for a vulnerability. In many cases, there are several attack methods. Taking password analysis and an attack to a product-specific vulnerability as examples, there can be several attack scenarios as follows:

- Password analysis
  - A layman enters and analyzes passwords over time by means of a brute-force attack.
  - A proficient person analyzes passwords in a short time using many PCs.

- Attack to product-specific vulnerability
  - An expert carries out reverse engineering with the aid of public information and develops an attack code over time.
  - An expert obtains secret information and develops an attack code in a short time.

The value of attack potential varies because the weights assigned to each of the attack potential factors are different depending on the attack scenarios. Therefore, the evaluator has to calculate the attack potentials required for executing each of the attack scenarios after screening out all the possible attack scenarios for exploiting the vulnerability, and then determine a verdict as to whether the evaluated products have prescribed resistance to all the attack scenarios.

■ **Consideration to up-to-date information**

The attack potential required for exploiting vulnerabilities may vary with time. For instance, the required attack potential will decrease when "improved hardware performance has reduced the elapsed time for the attack," or when "a new attack tool has been made available." Therefore, even when the evaluator has an experience with a similar evaluation in the past, the evaluator has to conduct an evaluation on the basis of up-to-date information at the time of the evaluation, rather than conducting it on the basis of the past experience.

4 Vulnerability search

This chapter explains the vulnerability search that is required in the CC for identifying the vulnerabilities that the evaluated products may have.

4.1 Vulnerability search method

The ultimate goal of the CC evaluation is to ensure that the evaluated products have no vulnerabilities that can be exploited. Therefore, it is important for the evaluator to eliminate insufficient considerations as much as possible when searching for vulnerabilities that the product may have. To that end, the following vulnerability search methods have been employed in the CC.

■ **Public domain vulnerability search**

In the CC, it is required for the evaluator to search for generally-known vulnerabilities on the basis of the product area and the evaluator's areas of concern. This method is aimed at preventing insufficient considerations to vulnerabilities.

■ **Vulnerability search through documentation analysis**

In the CC, it is required for the evaluator to analyze the design information of the evaluated products in view of general perspectives on vulnerabilities for hypothesizing the possibility of the vulnerabilities. This method can detect not only generally-known vulnerabilities but also vulnerabilities that are dependent on product-specific design or implementation.

■ **Vulnerability search by the flaw hypothesis methodology (at EAL 4 or higher)**

In the CC, it is required for the evaluator to use the "flaw hypothesis methodology" [3], which is a generally-known vulnerability search method, at EAL 4 or higher. By applying the flaw hypothesis methodology in documentation analysis, it is expected to further reduce insufficient considerations.

The following sections explain each vulnerability search method in detail.

4.2 Public domain vulnerability search

This section explains the tips and important notes with respect to the information sources in public domain vulnerability searches and the information to search for.

(1) Information sources for public domain vulnerabilities

The information about public domain vulnerabilities can be obtained by searching books and various information published on the Internet. Typical information sources include the following:

■ Internet in general
- Search by means of search engines

■ Vulnerabilities discovered in individual products
- JVN iPedia (Vulnerability Countermeasure Information Database) [4]
- CVE (Common Vulnerabilities and Exposures) [5]
- Others

■ General vulnerabilities independent of the product
- CWE (Common Weakness Enumeration) [6]
- IPA: How to Secure Your Web Site [7]
- IPA: Secure Programming Course [8]
- Others

■ Information from the viewpoint of attacks
- Exploit Database [9]
- Others

(2) Search information of public domain vulnerabilities

The following are the perspectives and keywords that should be taken into account for searching information sources for public domain vulnerabilities.

■ **Product area**

By searching for keywords relevant to the product area, it is expected to obtain the information about vulnerabilities in the functions specific to the product area and vulnerabilities that tend to occur in the product area. Possible search keywords for such a purpose include the following:

- Product type

  Firewall, DBMS, and multi-function printer are examples of the applicable keywords.

- Similar products

  The names of existing products of the evaluated products, the product series, and competitors' products are also candidates for search keywords. In addition, the security information of the product and the software update information provided from the product vendor itself are also applicable keywords.

- Origin of derivative products

  Some products incorporate another product as it is or with some customization. The name of the original product of such derivatives can also be used as a search keyword. For instance, when the SSL/TLS functions are implemented in the evaluated products and "openssl" is used for the implementation, the vulnerabilities in "openssl" should be searched for.

■ **Adopted technology**

By searching for information in the technology area adopted by the product, it is expected to obtain the information about vulnerabilities that tend to occur in the technology area. The evaluator should search for specific technology names and product names in association with, for example, security technologies and network protocols, as well as the components, program execution environments, and implementation techniques adopted in the product and the data format used in the product.

■ **Areas of concern**

Examples of the areas of concern for the evaluator include the following:

- Areas in which a large number of vulnerabilities are known (e.g., Web interface, input processing)
- Product-specific specifications, functions, and interfaces
- Complicated specifications and functions, etc.

(3) Important notes with public domain vulnerability searches

The following matters have to be taken into account for searching for public domain vulnerabilities.

■ **Consideration to vulnerabilities in other products**

The objective of public domain vulnerability searches is to reduce insufficient considerations. Therefore, when information about a vulnerability in a different

product from the evaluated product is obtained as a result of a public domain vulnerability search, the evaluator should hypothesize possible vulnerabilities that the product may have on the basis of the idea, "Now that this vulnerability has been found in a different product, it may be applicable to the said product," rather than considering, "Since this vulnerability has been found in a different product, it will not be applicable to the evaluated product."

■ **Elimination of the bias in the information sources**

By searching throughout the Internet, the evaluator can obtain a large quantity of information regarding vulnerabilities, including papers on vulnerabilities and presentation materials at conferences. On the other hand, the evaluator can obtain vulnerability information efficiently by searching on sites specialized in vulnerability information such as CVE [5]. It should be noted, however, that relying only on specific sites may hinder the discovery of vulnerability information that can be easily obtained by searching on other information sources.

■ **Repeated searches**

The evaluator may have to repeat searches, in a way such as searching for further relevant information using some information obtained from a search result. For instance, only the outlines of vulnerabilities are published in the majority of vulnerability information on a product-by-product basis, whereas the details of them are not published. In that case, the evaluator may obtain additional information by searching on information sources from the standpoint of attacks or various forums on vulnerabilities, etc., on the basis of the product name, keywords relevant to the vulnerability, etc.

4.3 Vulnerability search through documentation analysis

This section explains the tips and important notes with respect to the hypothesis of vulnerabilities by analyzing documentation, such as the design documents and user guidance of the products.

(1) Perspectives of vulnerabilities

In the CC, it is required for the evaluator to search for vulnerabilities in view of the general perspectives of vulnerabilities shown in Table 4-1.

Table 4-1 General perspectives of vulnerabilities

	Category	Description
(a)	General vulnerabilities with respect to the product type	It includes general vulnerabilities obtained by means of public domain vulnerability searches.
(b)	Bypassing	It is applicable to the cases where the attacker can evade the application of the security functions. It also includes the cases where confidential data is retrieved.
(c)	Tampering	It is applicable to the cases where the attacker executes processing that is not originally intended or suspending the security functions by modifying the executable codes or data of the security functions.
(d)	Direct attacks	It is applicable to the cases where the attacker directly attacks a mechanism of password authentication, etc., with repeated attempts or a similar method.
(e)	Monitoring	It is applicable to the cases where the attacker monitors the behaviors of the product or transmitted and received data to obtain confidential information protected by the product.
(f)	Misuse	It is applicable to the cases where the users cannot securely manage or use the product because the user guidance of the product is not properly described or requires significantly burdensome operation management for maintaining the security.

For the details of those categories, refer to CEM [1] "Annex B.2.1 Generic vulnerability guidance."

Additionally, at EAL 2 or higher, it is required for the evaluator to consider the security architecture description presented by the developer as documentation. The security architecture is a mechanism that protects the security functions of a product from being bypassed or tampered. The contents are important information for vulnerability searches. For the details of security architecture, refer to the following document.

- IPA: Security Architecture Guide for Developers [2]

(2) Examples of search procedures based on the perspectives of vulnerabilities

In the CC, although it is required for the evaluator to consider the perspectives of vulnerabilities in the previous section, no specific methods are prescribed. For better understanding for the readers of this document, this section introduces an example procedure with which the documentation is analyzed to search for vulnerabilities.

In this search procedure, the evaluator analyzes the documentation and searches for the vulnerabilities that the product may have from the perspectives of vulnerabilities in Table 4-1 with respect to each of the security functions and interfaces of the product. The flow of the vulnerability search in this case is as follows.

(I) Specification of the security functions and interfaces of the product

The evaluator analyzes the documentation, such as the design documents and user guidance of the product, for making a list of the security functions and interfaces of the product.

Even an interface that seems to have nothing to do with the security functions may have a vulnerability that can compromise the security functions, such as buffer overflow. The list covers all the interfaces so that such vulnerabilities can also be detected.

(II) Hypothesis of vulnerabilities

The evaluator hypothesizes the vulnerabilities by applying the perspectives in Table 4-1 with respect to each of the security functions and interfaces in the list.

For instance, the possibilities of vulnerabilities shown in Table 4-2 are considered when applying the perspectives in Table 4-1 to the password authentication function.

Table 4-2 Application example of the perspectives of vulnerabilities

	Category	Example of vulnerability
(a)	General vulnerabilities with respect to the product type	There is a concern that attack methods, such as "rainbow attack," obtained from public domain vulnerabilities may succeed.
(b)	Bypassing	With Web systems, there is a concern that access may succeed without authentication by specifying the URL directly. In addition, there is a concern that authentication may succeed by using a password that is obtained from the storage location of passwords in some way.
(c)	Tampering	There is a concern that an attack using a buffer overflow or SQL injection may succeed by entering unauthorized data, leading to the execution of processing that is not originally intended. In addition, there is a concern that authentication may succeed by modifying or overwriting the file in which passwords are stored in some way.
(d)	Direct attacks	There is a concern that repeated attempts of various passwords may lead to a success of authentication.
(e)	Monitoring	There is a concern that passwords transmitted to the network may be wiretapped.
(f)	Misuse	There is a concern that the products may be operated without noticing that functions to reinforce authentication functions, such as the limited number of password attempts, have been disabled due to improper functions or user guidance of the products.

(III) Analysis of documentation

The evaluator examines the documentation to analyze whether or not the hypothesized vulnerabilities are applicable. When the documentation includes the security architecture description, the evaluator should also consider the protection mechanisms. The hypothesized vulnerabilities will fall under either of the following:

■ When the countermeasures against the vulnerabilities cannot be determined

The possibility that the product has the hypothesized vulnerabilities cannot be

denied. As explained in Chapter 2, the existence or absence of the vulnerabilities will be confirmed by means of the penetration testing also in view of the attack potential required for exploiting the vulnerabilities.

■ When the countermeasures against the vulnerabilities have been taken

For instance, this is applicable to the cases where the countermeasure by the encryption of network communications has been taken against monitoring. In this case, the evaluator examines the documents of the test results carried out by the evaluator or developer to confirm whether or not the behaviors of the countermeasure described in the design documents of the product (in this example, password protection by means of the encryption of network communications) have already been tested. If the test is not sufficient, the evaluator carries out an operation check test to verify the behaviors of the countermeasure.

The conclusion drawn from those confirmations is that since countermeasures have been taken to the analyzed vulnerability (monitoring, in this example), the product does not have the vulnerability. However, additional analysis described in Section (IV) below is required for the countermeasure against the vulnerability (encryption of network communications, in this example).

(IV) Analysis to vulnerability countermeasures

Even though a countermeasure had been taken to the hypothesized vulnerabilities, a problem, for example, in the way of implementing the countermeasure would pose a risk that attackers might compromise the security functions of the product by exploiting the problem. Therefore, the evaluator conducts the analyses in Sections (II) and (III) to the vulnerability countermeasure of the product as well for the purpose of searching for flaws and weaknesses in the countermeasure. For instance, when countermeasures against monitoring have been taken by means of the encryption of network communications, the analyses in Sections (II) and (III) will be conducted on the encryption of the network communications.

The protection mechanisms stated in the security architecture description are included in the target of this analysis.

(V) Analysis of all the security functions and interfaces

The evaluator conducts the analyses in Sections (II) to (IV), regarding all the

security functions and interfaces of the product.

(3) Important notes in documentation analysis

The following notes have to be taken into account for analyzing documentation to search for vulnerabilities.

(3-1) Analysis of source codes

The evaluator will be able to conduct more detailed analysis, including the implementation level that is not described in the design documents, by referring to the source codes of the product. The following are the perspectives of vulnerability search that are distinctive of source codes.

■ **Detailed analysis of data and processing that the security functions depend on**

In the source codes, the evaluator focuses attention on, for example, the data that the security functions access or process, or branch processing in the security functions. The evaluator analyzes whether or not the product can be used in such a manner as to set that data to abnormal or unauthorized values or to mislead the branch conditions.

■ **Analysis focusing on public domain vulnerabilities in coding**

When analyzing source codes, the evaluator considers problems in coding that can be the causes of vulnerabilities. Typical vulnerabilities resulting from coding can be obtained through the public domain vulnerability search explained in Section 4.2. For instance, the following problems are known:

- Buffer overflow (stack, heap)
- Integer overflow
- Unauthorized use of memory (e.g., the use of freed memory, double free of memory)
- Race condition, etc.

■ **Analysis focusing on the compiler**

In the CC at EAL 4 or higher in which the source codes are evaluated, the evaluator evaluates the implementation-dependent syntax in the programming language and the specifications of the compiler before analyzing whether or not the source codes have vulnerabilities resulting from the compiler.

Depending on the way of coding that the developer uses, the complier may generate codes that are different from the intention of the developer, thereby posing vulnerabilities. Taking C language as an example, the following problems are known:

- Optimization

  The optimization processing of the compiler may change the order of processes or delete redundant processes. It may cause adverse effects when the developer describes codes that control hardware or when they intentionally describe redundant codes because of a security requirement.

- Sign of the char type

  According to the standards of C language, the sign of the char type with which "signed" or "unsigned" is not explicitly specified is regarded to be implementation dependent. If the sign of the char type that the developer assumes is different from that of the compiler, it will lead to a different result from the intention of the developer in comparison between and addition/subtraction of values exceeding 127. This may adversely affect the security functions depending on the way that char-type variables are used.

■ **Hidden functions and options**

Some products may have functions or options that are not stated in the design documents or user guidance. In other cases, the debugging function used in the product development may remain in the final product by mistake. Such functions may have vulnerabilities because they have not been sufficiently examined and tested by the developer as to whether they have security problems or not.

Note that, in the CC, the evaluator analyzes the source codes in comparison with the specification documents and user guidance of the product prior to vulnerability assessment so that they can evaluate whether or not the functions of the product have been accurately implemented. In most cases, such hidden functions and options are detected in the course of this evaluation.

■ **Complicated processing**

Depending on source codes, the processing can be complicated and hard to understand. When the evaluator finds a complicated process, they may hypothesize the existence of vulnerabilities that is hard to detect only with source code analysis, and confirm the existence or absence of such vulnerabilities by means of testing.

(3-2) Verdict on the possibilities of vulnerabilities

In documentation analysis, the evaluator not only newly detects the possibilities of vulnerabilities, but also determines that the product is free from the possibility of the occurrence of vulnerabilities that are uncertain as to the correspondence to the product. At that time, extra caution should be exercised not to determine the absence of problems easily only because countermeasures have been taken against the vulnerabilities. The vulnerabilities may leave room for being exploited by attacks owing to insufficient considerations in the design of the countermeasures or coding errors.

For instance, when the product checks special characters and keywords to avoid their adverse effects during the processing of entered character strings, there are concerns over the following problems.

■ **Insufficient checks**

There can be cases where it lacks some of the characters to be checked in the design documents or programs. It should be noted that blacklist systems are generally insufficient and easy to lack something.

■ **Insufficient considerations to encoding**

In some cases, characters can be expressed in a variety of formats, such as hexadecimal notation, % notation, and UTF-7/8. There can be cases where such considerations are insufficient in the design documents or programs.

■ **Insufficient considerations to the difference in the interpretations between the time of checking and using character strings**

Special characters can be interpreted differently between the time of checking character strings and the time of using them in practice. For instance, there is a possibility that a NULL character or a line feed code in the middle of a character string is regarded as a character when the character string is being checked, whereas it is ignored when the character string is used in practice. There can be cases where such considerations are insufficient in the design documents or programs. Extra caution should be taken on such differences in the interpretation when the evaluated product passes data to another IT product, such as Web browser and DBMS that processes SQL.

4.4 Vulnerability search by the flaw hypothesis methodology

In the CC evaluation at EAL 4 or higher, it is required to use the flaw hypothesis methodology in search for vulnerabilities by analyzing documentation. This section explains the overview of the flaw hypothesis methodology.

(1) Overview of the flaw hypothesis methodology

The flaw hypothesis methodology is a generally-known vulnerability search method. The CC does not include the explanation of the flaw hypothesis methodology. For the details of the flaw hypothesis methodology, refer to the following reference:

- Clark Weissman, "Penetration Testing" [3]

The flaw hypothesis methodology consists of the four stages as follows:

1) Flaw generation
   A hypothesis on the suspected flaw is generated.

2) Flaw confirmation
   Whether the hypothesis on the flaw is correct or not is confirmed.

3) Flaw generalization
   Typical weaknesses that can cause the confirmed flaw are analyzed.

4) Flaw elimination
   The confirmed flaw is reported. (The developer should address it in some way.)

Figure 4-1 shows the flow of documentation analysis by the flaw hypothesis methodology.

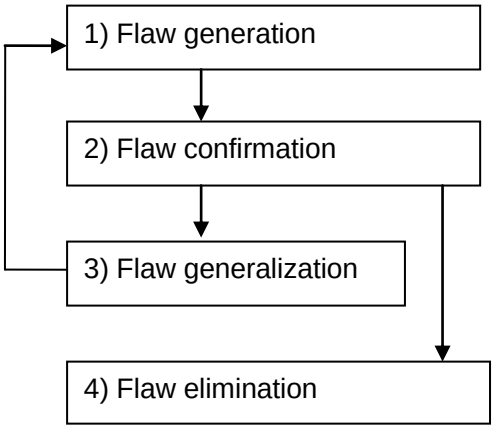

Figure 4-1 Analysis flow by the flaw hypothesis methodology

(2) Details of the flaw hypothesis methodology

This section explains the four stages of the flaw hypothesis methodology.

1) Flaw generation

The evaluator analyzes the design information (security policies that the product should realize, external interface specifications, internal product design documents, and source codes) and user guidance of the product to hypothesize flaws in the product as well as flaws in operations. Examples of the perspectives when hypothesizing flaws include the following:

- Flaws similar to flaws that have already been discovered
- Discrepancies between the security policies that the product should realize and the design and implementation
- Discrepancies between the design and the implementation of the product
- Imperfections in the architecture or functions of the product
- Implementation errors
- Flaws caused by the development work (e.g., remains of debugging function)
- Usage that is likely to cause a problem

These concrete implementation methods are almost the same as the contents already explained in "4.2 Public domain vulnerability search" and "4.3 Vulnerability search through documentation analysis."

2) Flaw confirmation

The evaluator analyzes the source codes and completed test results, etc., to determine whether or not the hypothesized flaws are applicable to the evaluated products.

In addition, the evaluator may conduct some tests to confirm hypotheses under analysis. The tests in this case, having different purposes than the penetration testing that is used to determine whether or not the vulnerabilities can be exploited in the end, also include tests designed for confirming the specifications. On the basis of the verification results of the hypotheses, the evaluator continues to explore the possibilities of the flaws by developing and modifying the hypotheses. Proceeding with analyses in parallel with tests, the evaluator can not only gain a deeper understanding of the product but also verify the hypotheses faster than the detailed analysis of the source codes.

3) Flaw generalization

The evaluator analyzes the confirmed flaws for "flaw generalization" and then uses the generalized flaws as the inputs to the analysis in Stage 1) for devising new hypotheses. Flaw generalization includes the following views:

■ **Specification of the mechanism causing a flaw**

The evaluator analyzes the confirmed flaw for specifying the mechanism (weakness) that causes the flaw. The weakness can be applicable to other processes as well.

As an example, suppose a flaw that allows attackers to bypass the access control function is confirmed in the access control function of the product. Rather than recognizing this as "bypassing of the access control function," the evaluator should go one step further and analyze the mechanism that causes the flaw.

As an example, suppose the cause of the bypassing of the access control function has been found to be an exploitation of the time difference between the checking of the access privilege and the execution of actual access ("Time-of-check to Time-of-use"). In that case, the generalized weakness is a "Time-of-check to Time-of-use" problem, posing a risk that the weakness may be applicable not only to the access control function but also to the entire processing that verifies the validity of data.

■ **Generalization of the mechanism causing a flaw**

For the mechanism (weakness) that causes the flaw, the evaluator further considers the hierarchical concept of the weakness, which is more generalized.

For instance, "Race condition" (falls under CWE-362 in CWE [6]) is a possible hierarchical weakness to "Time-of-check to Time-of-use" (falls under CWE-367 in CWE [6]). Hence, searching for flaws in the product from the more generalized perspective of "Race condition" may lead to the discovery of other flaws, such as race condition on shared data.

■ **Combination of flaws**

Even when a product has multiple flaws that do not so seriously affect the security on an individual basis, a combination of some of the multiple flaws may have a serious effect.

As an example, suppose that the following two flaws (weaknesses) have been discovered in a product.

- Interface A

  There is a possibility that attackers can access important data protected by the product. To that end, it is required to modify a configuration file, but there is no means available to modify the configuration file.

- Interface B

  There is a possibility that attackers can modify some files in the product. However, they cannot access the important data protected by the product.

In this case, when the evaluator analyzes interface A and interface B separately, the analysis may derive a conclusion that the product has no vulnerabilities that can adversely affect the security of the product.

However, combining those two interfaces will allow the evaluator to hypothesize, "Attacker may be able to access the important data protected by the product by modifying the configuration file of interface A through interface B."

4) Flaw elimination

The evaluator reports the discovered flaws of the product and their recommended measures to the developer. The developer analyzes the details of the flaws, examines the modification to the product and workaround plans, and

eliminates the flaws.

4.5 Search for detailed information regarding vulnerabilities

The public domain vulnerability search is used not only for the purpose of identifying general vulnerabilities that the product may have, but also for the purpose of obtaining further detailed information about the vulnerabilities that have been hypothesized to exist in the product. The following explains the publicly available information search for the latter case.

(1) Verdict on whether vulnerabilities can be exploited or not

As already explained in Chapter 3 "Attack potential," whether or not the attack method is known will significantly affect the verdict on whether the vulnerability can be exploited or not. When the attack codes or tools are published, in particular, attackers can relatively easily practice an attack even if it seems to be technically difficult. In addition, the information for such attacks is also necessary for the evaluator to carry out the penetration testing.

Therefore, it is required to search for public domain information extensively as to see whether or not concrete information has been published for attacking vulnerabilities that have been discovered through public domain vulnerability search or documentation search.

(2) Disproof of the developer's assertions and countermeasures

When some countermeasures have been implemented to the security of the product, it is required for the evaluator to analyze whether or not the countermeasures have any vulnerability from the viewpoint of attackers, rather than simply assuming that "there should not be any problems by virtue of the countermeasures." It is required to search for public information extensively in such disproof of security measures as well.

The following is an example of a case where public domain vulnerability searches are repeatedly conducted for the disproof of security measures.

■ **Example: Disproof of measures for the leakage of cryptographic key**

Suppose a case where the evaluated product provides an encryption function, and the documentation does not provide any measures to retrieve the

cryptographic key. The evaluator attempts the disproof of the measures from the same viewpoint as attackers.

i) Is there really no method to read out the cryptographic key?

Searching on the Internet will allow the evaluator to obtain tips for the ways to readout.

- Device files in the operating system (e.g., /dev/mem)
- Core dump files of processes
- Secondary storage devices (areas for paging or swapping)
- Read-out from DRAM

Even if the contents of the memory can be read out, however, it seems to be difficult to specify and retrieve the cryptographic key from a massive amount of data recorded in the memory.

ii) Is it really difficult to specify the cryptographic key in the memory?

Searching on the Internet will allow the evaluator to obtain papers on the algorithm for specifying the cryptographic key in memory.

Lest We Remember: Cold Boot Attacks on Encryption Keys, Proc. 17th USENIX Security Symposium, 2008

However, although it is possible logically, it might be difficult to practice.

iii) Is it really possible to practice the contents of the research paper?

Searching on the Internet further will allow the evaluator to find the existence of "*aeskeyfind*," a program to demonstrate the contents of the research paper.

As described above, even when there is a barrier against attacks, the evaluator should continue to search from the perspective of exploring the possibility of defeating the barrier. The evaluator will devise the most easy-to-execute attack scenario after conducting a sufficient amount of searching. Then, the evaluator will calculate its attack potential and conduct the penetration testing as necessary.

5 Penetration testing

This chapter explains the penetration testing for confirming whether the vulnerabilities that the evaluated products may have actually exist or not.

5.1 Overview of penetration testing

As a result of vulnerability analysis, the vulnerabilities that the evaluated products may have are listed. The evaluator conducts the penetration testing to determine whether or not the evaluated products have those vulnerabilities in reality.

Before conducting the penetration testing, the evaluator has to devise appropriate testing methods in view of the possibilities of various attacks against the intended vulnerabilities. In the CC, no concrete methods for the penetration testing are prescribed. By searching on the Internet, the evaluator can obtain the explanation of the penetration testing methods and various kinds of information, including penetration testing tools. The following are example references:

- O'Reilly Japan
  Metasploit: The Penetration Tester's Guide [10]
- OWASP Testing Project: OWASP Testing Guide [12]

5.2 Important notes in penetration testing

This section explains the tips and important notes for devising or conducting the penetration testing.

(1) Consideration to attack variations

There can be a large number of variations in attacks against the intended vulnerabilities. Therefore, even when the developer has taken countermeasures against the intended vulnerabilities, it is concerned that attacks capable of evading the countermeasures may exist. After searching for public domain vulnerabilities, the evaluator has to conduct the penetration testing with due consideration of the variations in those attacks.

Taking cross site scripting as an example, a large number of variations are known, such as the encoding of the character code and the HTML tags to be used. Refer to the following for the details.

- OWASP: XSS Filter Evasion Cheat Sheet [13]

(2) Utilization of examination tools

To examine a large number of variations in attacks, the use of the corresponding examination tools will allow for efficient examinations.

By searching on the Internet, the evaluator can find a large number of examination tools. In the selection of examination tools, the evaluator has to pay attention to whether or not the examination tool can sufficiently examine the vulnerabilities that the evaluator wishes to examine. If evaluators cannot conduct the intended examination with any examination tools, the evaluators have to devise the examination methods by themselves.

Some examination tools have a structure in which common functions required for examination are provided as a framework and concrete examination data (attack codes) can be added as necessary. By searching on the Internet for the examination data (attack codes) for such examination tools, the evaluator may obtain the examination data (attack codes) for the intended vulnerabilities.

(3) Verdict on the success or failure of attacks

When planning the details of the penetration testing, the evaluator has to pay attention not only to the examination data (attack data) to be entered to the evaluated products, but also to the verdict as to whether the conducted tests (attacks) succeeded or failed. In some cases, the success of an attack may be overlooked in spite of the successful result of the attack because it is difficult to determine whether the attack succeeded or failed by the response from the product.

The case with SQL injection is considered as an example. Cases of successful login that are used as easy-to-understand examples in the majority of explanations are actually quite rare. As a matter of fact, the result in most cases is something like a simple error message display, which is difficult to determine whether the SQL injection succeeded or failed. It should be noted that there is a possibility that the SQL has been executed in the product even when an error message is shown.

Even if it is difficult to determine whether an attack succeeded or failed, the evaluator may determine whether an attack succeeded or failed using logs, etc., in the product when they can obtain cooperation from the developer. Otherwise, the evaluator has to determine whether the attack succeeded or failed from the same viewpoint as attackers. Example methods include executing a time-consuming command to observe the difference in the response times between successful

attempts and failed attempts.

In case of SQL injection, an attack method called "blind SQL injection" is known. For the details of the examination method, refer to "OWASP Testing Guide" [12].

(4) Port scanning examination

In general, the examination using a tool for port scanning is conducted for investigating TCP/IP open ports. The following matters have to be taken into account for the port scanning examination.

■ **Consideration to dynamic open ports**

When conducting port scanning, the evaluator has to pay attention to the timing of the execution as well. Depending on the functions of the evaluated products, some ports may be opened in the middle of the operation with the use of the product. Therefore, there is a possibility that such dynamically opened ports cannot be detected depending on the timing of executing port scanning.

■ **Examination when unexpected open ports are detected**

In the CC, the developer provides the external interface specifications of the evaluated products, and the evaluation is conducted on the basis of the provided specifications. Therefore, the open ports of the evaluated products have been fixed before the examination. Nevertheless, if unexpected open ports were detected, the imperfection of the documentation submitted as the external interface specifications, rather than vulnerabilities, should be suspected.

■ **Examination when legal open ports are detected**

When the open ports in accordance with the external interface specifications are detected, the evaluator cannot determine that the product has no vulnerabilities only for the reason that the interfaces are in accordance with the specifications. The evaluator has to conduct a follow-up examination of the possibility of vulnerabilities in network protocols as explained below.

(5) Examination of network protocols

Some network protocols may include functions that can be exploited for attacks by the specifications themselves or setting errors.

Taking an FTP server as an example, setting errors of the access privilege of the target file, unexpected directory access by means of the CD command, unexpected program execution by means of the SITE EXEC command, exploitation of the PORT command (FTP bounce attack), etc., fall under this category. In addition, when a maintenance account is set up in a product, unauthorized login by using the account is also a concern.

Having investigated such weaknesses in the protocols during the vulnerability search, the evaluator confirms that there is no possibility of being exploited by conducting the penetration testing.

(6) Examination of implementation flaws

The product may have vulnerabilities caused by flaws in the implementation, including buffer overflow. When conducting an examination in which the source codes of the products cannot be referred, the evaluator hypothesizes flaws in the implementation and confirms them by means of testing.

A method called "fuzzing" is known as a method for testing such obscure vulnerabilities. With respect to various fields such as a command and data format of network protocols, for instance, the evaluator enters data that can cause a buffer overflow or data that includes special characters which can cause a malfunction to the evaluated product for confirming the response. Thus, the evaluator can examine whether or not the product has vulnerabilities. For fuzzing, refer to the following reference:

- IPA: Guidebook for the practical use of fuzzing [14]

6 Conclusion

In this document, the overview of the CC vulnerability assessment is explained, including important notes for conducting the search of vulnerabilities and the penetration testing in practice.

In the CC, the evaluation methodology has been designed so that homogeneous evaluation results can be obtained, and thus the prescribed evaluation methodology can be regarded as the "best practice." However, to implement the evaluation methodology in practice, it is necessary to search on the Internet and other sources sufficiently and utilize the obtained information appropriately.

Especially, not only vulnerability information but also various attack tools and attack codes have been published on the Internet. When there exists an attack tool, etc., it is very dangerous that even attacks that require advanced knowledge and techniques can be relatively easily executed. Extra caution should be taken on such up-to-date information when vulnerability assessment is practiced.

In addition, the period of time from the discovery of a vulnerability to the known availability of the corresponding attack code is getting shorter recently, posing a problem known as the so-called "zero-day attack." Therefore, when vulnerabilities are detected in products, it is desired that developers should counter the vulnerabilities promptly after due consideration not only of the difficulties in exploiting the vulnerabilities at the time of detection, but also of the possible future changes in the difficulties.

Note that basic knowledge of IT technology is required for understanding the details of vulnerabilities and devising the corresponding penetration testing. For instance, the following reference will be informative.

- O'Reilly Japan:
  Hacking: The Art of Exploitation, 2nd Edition [15]

Bibliography

A list of the references mentioned in this document is shown below. The URLs are available at the present time of writing.

■ Throughout the document

[1] *Common Methodology for Information Technology Security Evaluation: Evaluation methodology, Version 3.1 Revision 4, September 2012, CCMB-2012-09-004, (November 2012, Japanese version 1.0)*
http://www.ipa.go.jp/security/jisec/cc/index.html
The standard document for the evaluation methodology of vulnerabilities explained in this document.

[2] *IPA: Security Architecture Guide for Developers,*
http://www.ipa.go.jp/security/jisec/apdx.html#ADV_ARC_GUIDE
This document explains security architecture, which is a mechanism for protecting security functions of IT products from attacks. In the CC evaluation, the developers are required to design and implement their IT products in a way that they can pass vulnerability assessment. Security architecture corresponds to the details of design and implementation focusing on vulnerability measures.

■ Chapter 4　Vulnerability search

[3] *Clark Weissman, "Penetration Testing," Information Security: An Integrated Collection of Essays, Essay 11, pp.269-296, IEEE Computer Society Press, 1995*
This book is an explanation of the flaw hypothesis methodology written by its advocate.

[4] *JVN iPedia (Vulnerability Countermeasure Information Database), http://jvndb.jvn.jp/*
This database provides vulnerability countermeasure information for software products used in Japan.

[5] *CVE (Common Vulnerabilities and Exposures), http://cve.mitre.org/*
This site provides the information of vulnerabilities found in individual products. The registered information is assigned with a CVE-ID, allowing you to uniquely identify what vulnerability in what product the information is for.

[6] *CWE (Common Weakness Enumeration), http://cwe.mitre.org/*
This site provides the information on the types of software vulnerabilities (weaknesses).

The registered information is assigned with a CWE-ID, allowing you to uniquely identify the type of software vulnerabilities (weaknesses).

CWE can be regarded as a set of common weaknesses organized in a hierarchical structure that are derived by applying "flaw generalization" contained in the flaw hypothesis methodology to vulnerabilities discovered in the world.

[7] *IPA: How to Secure Your Web Site,*
http://www.ipa.go.jp/security/vuln/websecurity.html

This document explains typical vulnerabilities in Web applications and the countermeasures against them.

[8] *IPA: Secure Programming Course,*
http://www.ipa.go.jp/security/awareness/vendor/programming/

This training course explains the countermeasures against typical software vulnerabilities that should be taken in the entire development process including design, implementation, and testing.

[9] *Exploit Database, http://www.exploit-db.com/*

This database provides various information regarding vulnerabilities and their attack methods.

■ Chapter 5　Penetration testing

[10] *David Kennedy et al., Metasploit: The Penetration Tester's Guide,*
O'Reilly Japan, 2012

This book explains the general ways of the penetration testing with a focus on a common tool, Metasploit.

[11] *Metasploit, http://www.metasploit.com/*

This is the official website of Metasploit, one of the vulnerability examination tools. This site provides various modules for Metasploit and concrete information for examining various vulnerabilities.

[12] *OWASP Testing Project, OWASP Testing Guide v3,*
https://www.owasp.org/index.php/OWASP_Testing_Project
- Original version: https://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf
- Japanese translation: https://www.owasp.org/images/1/1e/OTGv3Japanese.pdf

This testing guide describes concrete methods to examine vulnerabilities in Web applications.

[13] *OWASP: XSS Filter Evasion Cheat Sheet,*
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
This sheet describes various notations that can be used for cross site scripting attacks.

[14] *IPA: Guidebook for the practical use of fuzzing,*
http://www.ipa.go.jp/security/vuln/fuzzing.html
This document explains the overview and the way to practice fuzzing, one of the techniques for detecting vulnerabilities.

■ Chapter 6　Conclusion

[15] *Jon Erickson, Hacking: The Art of Exploitation, 2nd Edition,*
O'Reilly Japan, 2011
This book explains the technical mechanisms and actual verification programs of basic vulnerabilities, such as buffer overflows, shellcodes, attacks via networks, and password cracking, as well as their attack methods.

Vulnerability Assessment Guide for Developers

March 4, 2013 First edition

Author and Publisher Information-Technology Promotion Agency, Japan (IPA)

Writers Information Security Certification Office