

# **STANDARD FOR PRODUCT SECURITY ASSURANCE**

STM-0117 for General Use, Eighth Edition

**SONY**

## Terms of Use:

Copyright and all intellectual property rights in the content of this document are vested in Sony Corporation and reserved, unless otherwise indicated.

This document is the Sony Technical Manual, STM-0117 for General Use, Eighth Edition.

Copyright 2012 Sony Corp.

ALL RIGHTS RESERVED

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without the prior written permission of Sony Corporation.

## CONTENTS

1. PURPOSE .....	1
2. SCOPE.....	1
3. TERMS AND DEFINITIONS .....	1
4. STANDARDS .....	2
4.1 Certification and identification .....	2
4.1.1 Verification of server certificate on HTTPS connection.....	2
4.1.2 Confirmation method of URL of online resource to be retrieved or displayed .....	2
4.1.3 Password masking .....	2
4.1.4 Protection of network products.....	3
4.2 Improvement of product reliability.....	4
4.2.1 Prohibition of use of rootkit technology .....	4
4.2.2 Prohibition of open of unnecessary port .....	4
4.2.3 External library check.....	4
4.2.4 How to use wireless appliance for secure communication.....	5
4.2.5 Rejection of viruses .....	6
4.3 Basic technology of security implementation .....	7
4.3.1 Principles for vulnerability prevention on coding .....	7
4.3.2 Use-restricted function group 1 .....	12
4.3.3 Use-restricted function group 2 .....	14
4.3.4 Format string .....	18
4.3.5 ActiveX controls.....	19
5. REFERENCES .....	20
APPENDIX .....	21

## 1. PURPOSE

This Manual is a reference that describes the rules, guidelines, and other know-how from the viewpoint of security, as the standards to assure product security from design to mass production release. The purpose of this Manual is to be utilized by the persons concerned to each process, and to prevent security problems or vulnerability problems of the products.

## 2. SCOPE

This Manual applies to software that is required to meet the specifications or quality standards of Sony Corporation.

## 3. TERMS AND DEFINITIONS

"Vulnerability"

Vulnerability, as used in this Manual, refers to vulnerability in terms of product security, and it is associated with not only the problems with violation of copyright protection and encryption technologies for protecting specific information assets but also defects and faults that can cause menaces such as artificial attacks by malicious third parties and unintended uses.

"Buffer overflow"

A typical vulnerability, it occurs when a character string exceeding the buffer size reserved by the program is input. Not only may a buffer overflow cause the program to perform unexpected operations such as abnormal termination, but if abused, it may also permit any program created by a third party to be executed.

"Certificate chain"

The certificate chain is a sequence that can be traced from a CA (Certificate Authority) that gives a signature to the trusted CAs.

"Rootkit technology"

The rootkit is a program that hides a file installed in a system, or a running process. It is possibly installed as a kernel-mode driver by tampering the files composing an OS. Generally, the rootkit is used as a utility by a malicious third party to attack a target computer; it has backdoor function that helps invasion, data tampering function that deletes the log during unauthorized invasion, and concealment feature that hides the presence of attack program (process) from the monitor program of an OS.

"Development environment"

Development environment refers to the tools such as text editors, make programs, compilers, libraries, and SDK (Software Development Kit) used to develop programs. A product package containing development tools is called IDE (Integrated Development Environment).

## 4. STANDARDS

### 4.1 Certification and identification

Users and their opposite parties of communication must be certified and the information must be secured to protect the user-input data from malicious third parties. This section describes the knowledge and the technology about certification and identification of users.

#### 4.1.1 Verification of server certificate on HTTPS connection

A server certificate shall be verified when using HTTPS communication to connect a WEB service.

The following 5 items need to be verified.

- 1) The signature of a CA attached to the certificate is correct.
- 2) The certificate chain is correct.
- 3) The subject (the name of the object of certificate) is identical to the destination.
- 4) The certificate is in the valid period.
- 5) The certificate is effective (confirmation of the CRL: Certificate Revocation List).

If you find any uncertain items listed above, notify the user that the certificate was not valid and then terminate communications. As an alternative, you may give the user an option to continue communications at the user's risks that may arise due to the situation that made the certificate invalid. Keep in mind that even if you download a file from a server, it will constitute connecting a WEB service.

Regarding "5) The certificate is effective (confirmation of the CRL: Certificate Revocation List)," sample code for Windows is included in Appendix A at the back of this Manual.

#### 4.1.2 Confirmation method of URL of online resource to be retrieved or displayed

- 1) Provide users with a method to confirm the destination URL being displayed, if a feature to retrieve and display the online information is available. The aim is to enable users to confirm what site they are connecting.

Screen display or manual description is available as a method of confirmation.

- 2) In case to connect a WEB server, use HTTPS connection whenever possible.
  - 2.1) When establishing HTTPS connection, refer to Section 4.1.1 "Verification of server certificate on HTTPS connection."
  - 2.2) If the server certificate is properly verified, provide users a way to confirm that they are connecting the right site. Most web browsers show a padlock icon to emphasize security.
  - 2.3) Provide a way to check the verified server certificate. Especially when the verification result of the certificate is incorrect, and additional choice to continue communication is given to the users as described in "4.1.1 Verification of server certificate on HTTPS connection," try to help users' comprehension by highlighting the incorrect parts

#### 4.1.3 Password masking

A UI (User Interface) that receives a password shall display the user-input character as a special sign like \* (asterisk) or null character.

Do not display the user-input characters as they are with no time limit. The purpose is to reduce the risk of shoulder surfing.

A UI, in which the user confirms the password has been set, can prevent to give a clue of the number of the characters in password to a malicious third party by displaying the number of \* that differs from those of the input characters, or by showing only "Already set."

#### 4.1.4 Protection of network products

- 1) Function of deleting confidential data as a whole in an appliance  
It is required to have a function of deleting as a whole confidential data preserved and stored in an appliance that cover personal information, private content, and some sorts of data for using network services in order to prepare for sale, transfer to others or scrapping of an appliance. It is desirable to make statements about the following points on instruction manuals and online manuals and the like:
  - In the case of an appliance which can preserve account information relevant to charging for online settlement and others or enables users to preserve any information as they like, there are possible risks where some information remaining in the appliance after its sale, transfer or scrapping might be abused by a malicious third party.
  - Users are recommended to use the function of deleting data as a whole before its sale, transfer or scrapping.
- 2) Certification of users  
Consider the need of mounting a function of certifying users for products preserving confidential data.  
As for an embedded device, certification through input of four-digit numeral PIN code is common. In addition, it is required that users should be able to choose certification through password and passphrase stronger than PIN code because a device has come to preserve a large amount of sensitive information.  
When it is considered necessary to guard privacy among users of the products shared by multiple users, an addition of accounts and certification for each account shall be required.  
Also, it is to be desired that you should study the importance of secret information that an apparatus processes and the functions mounted in the apparatus comprehensively. Then, if an abuse of the apparatus picked up by a third party is considered to cause a serious damage, make the apparatus have functions of account lockout or deleting data (if there is a backup of it) activated by repeated certification failures, or remote lock for controlling access to the apparatus at a distant place.
- 3) Encryption  
If a product can connect with external appliances and services and can provide functions of reading and getting confidential data stored in the appliance, but is not protected by certification of users, encrypt and preserve confidential data preserved in the appliance. This is intended for a malicious third party not to restore easily the confidential data they extract from the appliance if they get it. In this regard, see that the strength of encryption is enough.
- 4) Certification of addressee and data encryption when sending information  
Certify addressee when sending personal information from appliances to an open network represented by the Internet. The purpose is to prevent personal information from being sent to an unexpected addressee. Give and receive personal information either by encrypting the information or through an encrypted communication path if sending the information via an unsecured appliance. It is important to encrypt communication by the appliances at both ends of communication in order to avoid danger of information leaks when an appliance whose security level is lower relays information.
- 5) Explanation of risk due to leakage of a user ID and password  
As for an appliance with a function of sending a user ID and password, elaborate on the risks caused by leakage of a user ID and password, and measures against it with instruction manuals and online manuals and the like.  
Urge users to stop using a short and easy password or using the same one for every service provided, in order to avoid harm users will suffer in an on-line settlement and so forth where they are robbed of their account information relevant to charging. Hardware suppliers and service providers should cooperate to protect users by sending an e-mail every time a settlement is made or their registration is altered or by providing users with their record so that users should be aware that "their accounts might have been attacked by spoofing."

## 4.2 Improvement of product reliability

Countermeasures are necessary to prevent malicious third parties from abusing users' product. This section describes the knowledge and technology to improve product reliability.

### 4.2.1 Prohibition of use of rootkit technology

- 1) Objects prohibited to be bundled with a product  
Inspect the files and the programs to be bundled to a product by using scanners<sup>(\*)</sup> provided by security software vendors, to make sure that the rootkit technology is NOT used. Never bundle the file or the program detected as a rootkit to the product. Because use of the rootkit technology will make the users uneasy and unpleasant, furthermore, it may be abused as a hiding place of computer virus or worms.
- 2) Install and uninstall  
In the case of a features that conceals execution from a system or a program that hides files and directories is indispensable in the product specification, explain its purpose to the users and obtain permission before install it. It must have the uninstall feature to delete the installed files or programs whenever the user wants to. Moreover, take measures to prevent abuse of the installed program by setting access control to the features.

<sup>(\*)</sup> The followings are the names and the URLs of the scanners available from the security software vendors.

- For Windows

Kaspersky Anti-Virus [http://usa.kaspersky.com/products\\_services/anti-virus.php](http://usa.kaspersky.com/products_services/anti-virus.php)

Norton Internet Security <http://www.symantec.com/norton/internet-security>

- For Linux (Unix-based system)

Not available as of now.

Items will be added at revision in the future.

### 4.2.2 Prohibition of open of unnecessary port

For the products that have Internet connection feature and open the port for listening, clarify the reason and necessity of each listen port based on the product specification. It is preferable to record and save the reason. If the reason is unclear or inadequate, or the alternatives exist, stop the listen port. To minimize the number of the listen ports reduces the chances of intrusion by attackers.

Execution of FTP server shall not be set at factory setting even for maintenance purpose, try to run it only when it is necessary.

If the OS has a console that can execute external commands, an open port can be checked by using netstat command. What is more, an open port can be checked from the outside of the object device by using nmap command. After executing these commands, the port that can not be explained to be essential for the product specifications must be shut.

### 4.2.3 External library check

When procuring software or libraries, including open sources from outside, confirm that they have no vulnerability.

If a patch to fix vulnerability is provided, apply it for sure.

If vulnerability is found in procured external libraries after they had been shipped as a part of the products, and it might result in damage to the users, take appropriate action such as provision of updater or notice to the users.

#### 4.2.4 How to use wireless appliance for secure communication

When mounting a wireless communication function into a shipped product, do not fail to incorporate communication method considered to be secure at the time of design. It is because the use of communication method weak in encryption strength during wireless communication would cause risk of a transmitted data being wiretapped by a malicious third party. Even if it is a communication method weak in encryption strength, however, the mounting itself may be accepted provided that it is treated as an option that users can decide to choose.

As for Wi-Fi connection, WPA2/AES is said to be secure at the time of issuing this STM. Therefore, this communication method must be incorporated. Since methods such as WEP and WPA/TKIP have proved to be decoded in a short time, there is a risk of users' communication being tapped or falsified or their wireless LAN being accessed illegally.

Because you have a difficulty in Wi-Fi connection between appliances and setting of encryption methods, it is required to be based on the WPS (Wi-Fi Protected Setup) standard. WPS is a standard the specification of which was established by the Wi-Fi Alliance for the purpose of connecting wireless LAN appliances and configuring its security settings easily.

Wi-Fi Alliance: <http://www.wi-fi.org/>

WPS: <http://www.wi-fi.org/wifi-protected-setup>

Moreover, an appliance working as a Wi-Fi access point must provide some communication methods different in encryption strength. It is required to make it possible to control access for cutting off communication between the appliances connected in different communication methods from each other.

#### **4.2.5 Rejection of viruses**

Software defined by the Scope of this Manual must be examined with detection tools provided by security software vendors, to confirm that it is not infected by viruses.

Such detection tools must be equipped with the latest engines and patterns.

More than one detection tool must be used because different tools can cover different fields.

### 4.3 Basic technology of security implementation

This section describes the coding knowledge and technology effective to prevent vulnerability.

#### 4.3.1 Principles for vulnerability prevention on coding

1) Reliability verification over external data

When using a data (such as communication data, files, and environment variable) given from outside, it is necessary to verify whether its value is within the assumed range and form or not. The (input) data given from outside of a program should not be used without verifying the reliability of its data. Because most of vulnerability problems are caused by the unexpected behavior of the program when the data to be processed has an unexpected form or value. In particular the data given from outside is dangerous, since it may artificially contain malicious data that operates program on its own.

2) Buffer length check

When processing a buffer, it is necessary to know the buffer length in order to assure the non-occurrence of access exceeding the buffer range. When processing a variable-length buffer, always treat the buffer pointer and the buffer size as a pair, and implement a range check process to prevent reading/writing the outside of the buffer range. If writing to the outside of the buffer space allocated by the program was done, vulnerability problem called buffer overflow may occur. This might cause serious damage such as an attacker takes control of the program.

3) Alternatives for use-restricted functions

3.1) Functions with use-restriction

Do not use the use-restricted functions written in Section 4.3.2 and 4.3.3. Instead, use the alternative functions provided in the same section 4.3.2 and 4.3.3. The use-restricted functions are the C language functions that tend to cause vulnerability problems.

3.2) Function interface and behavior specification of alternative function

When replacing the use-restricted functions in Section 4.3.2 and 4.3.3 with the alternative functions, check the function interface and behavior specification of the alternative functions to implement them in accord with the original behavior specification of the program. Because the behavior specification of the alternative functions slightly differs from that of the use-restricted functions.

3.3) Validation of parameter before alternative function call

It is essential to implement the countermeasures to validate the value of a parameter to be given to an alternative function, or to correct the value of a parameter to fit into the normal range, prior to an alternative function call in the program.

The most of the alternative functions in Microsoft environment have the "Parameter Validation" feature, therefore a vulnerability problem such as buffer overflow is detected just before its occurrence, and also has a feature to force termination of the program. These features are able to foil attacks from attackers, but forcibly terminate the program when an abnormal value is accidentally given to an alternative function in it.

To implement the countermeasures, it is necessary to understand in what situations the Parameter Validation feature works. Refer to the documents on alternative functions at the "MSDN Library" Web site, and confirm the descriptions about parameter validation.

However, validation/correction of the parameters to be given to the alternative functions can be omitted only if the forced termination of the program upon an abnormal data input is admitted by the product specification.

Refer to the web sites below for parameter validation feature.

- Parameter Validation

<http://msdn.microsoft.com/en-us/library/ksazx244.aspx>

- Security Enhancements in the CRT

<http://msdn.microsoft.com/en-us/library/8ef0s5kh.aspx>

## 4) Leverage of the latest development environments

## 4.1) Migration to new development environments

If old development environments are still used, consider to migrate to new development environments and carry out if it is possible. The reason is security-enhanced libraries or compilers are available in new development environments.

## 4.2) Security feature of compilers and libraries

Consider the security features implemented in compilers or libraries and utilize them if available.

## Examples of the security features

## - Microsoft C/C++ compiler (cl.exe)

Source code static analyzer (/analyze)

Buffer security checker (/GS)

ASLR support (/DynamicBase)

Data Execution Prevention compatibility (/NXCompat)

Safe Structured Exception Handling (/SafeSEH)

## - GNU C/C++ compiler (gcc, g++)

Buffer overflow detection (-fstack-protector, gcc 4.1 or later)

Integer overflow detection (-ftrapv, gcc 3.4 or later)

Buffer overflow detection at compile/execution time (-D\_FORTIFY\_SOURCE=2)

Unauthorized memory access detection at execution time (-g -fmudflap -lmudflap, gcc 4.0 or later)

## 5) CODING RULES TO REDEEM VULNERABILITY

Rules to redeem vulnerability incurred during coding are as follows:

## 5.1) Integer overflow

## Vulnerable points:

If integer variables related to the size of buffers are improperly treated by the program, it may generate vulnerable points to be attacked by a buffer overflow attack.

## Countermeasures:

- In dealing with integer variables related to the buffer size, be sure to take all of the following measures, unless they are not applicable,:

a) Use unsigned integer variables for the buffer size.

b) Use the same number of bits for the integer variables for the buffer size, if possible.

c) Use the largest number of bits for the integer variables unless there is specific necessity such as constrain of resources.

d) Check the range of integer variables, both before and after processing them, considering possible overflow of calculation results, inversion of signs, missing bits, etc.

e) Consider use of the following compiler options:

For # gcc: -ftrapv

For Visual C+: /RTCc

## 5.2) SQL injection

### Vulnerable points:

If SQL statements are improperly treated when the program manipulates database, it may generate vulnerable points to be attacked by an SQL injection attack. In the case of the SQL injection attack, its database may be broken, altered, or read out by outsiders.

### Countermeasures:

- To prevent an SQL injection attack when the program lets inputs handle the database, be sure to take one of the following measures a) to c):
  - a) When using inputs as parameters to complete SQL statements, use prepared statements for the database (or a placeholder function or a binding function).
  - b) Prepare a stored procedure at the database so that the program uses the stored procedure.
  - c) If an escape handling that can completely render the inputs harmless is available, the inputs may be used as parts of SQL statements after the handling.

The inputs must not be used as they are, just by being connected to SQL statements.

## 5.3) Command injection

### Vulnerable points:

An improper way of activating externally executable files as child processes may generate vulnerable points to be attacked by a command injection attack.

In the case of the command injection attack, an unexpected program brought in by an attacker may run or unexpected commands may be executed.

### Countermeasures:

- To prevent a command injection attack, be sure to take both of the following measures a) and b), unless they are not applicable, :
  - a) Specify the child program to be executed with the absolute path in order to eliminate the influence of environmental variables or the executable file paths.
  - b) If the program uses the inputs as parameters to activate externally executable files, make sure that the inputs are within the normal range for the parameters and they are provided in proper formats.
- If a program runs under the Unix OS environment, take the following measure c) in addition to a) and b):
  - c) Under the Unix OS environment, use exec-type function, not system() or popen(). Specifically use execl() or execve() that are not affected by environmental variables. When using an exec-type function, substitute the absolute path for the outer executable file for the first argument.
- If the program runs under the Windows OS environment, take both of the following measures d) and e) in addition to a) and b):
  - d) When using ShellExecute()-type API, give the target file the absolute path of the externally executable file, and substitute the command-line argument for the parameter.
  - e) When using CreateProcess()-type API, give the first argument the absolute path name for the externally executable file, and the command-line argument for the second parameter.

#### 5.4) Directory traversal attack

##### Vulnerable points:

When the program treats resources of external files, an improper generation of path may generate vulnerable points to be attacked by a directory traversal attack.

In the case of the directory traversal attack, the attacker may access the resource data in the system unfavorably.

##### Countermeasures:

- Take all of the following countermeasures a) to c):
- a) When using the inputs from outside, transform the path into an absolute path after merging the inputs.  
If there is a symbolic link, it is recommended that the target specified by the symbolic link should be transformed into an absolute path without the symbolic link.
- b) Make sure that the absolute path through the process described the above a) points at the possible directories.
- c) Keep using the absolute path being transformed in the program after a) and b).

#### 5.5) Environmental variables

##### Vulnerable points:

If the program has a procedure affected by environmental variables, it may generate unexpected vulnerable points due to the environmental variables contaminated by the attacker.

Particularly, when the externally executable files or libraries are loaded, environmental variables contaminated by the attacker may allow vicious executable files or libraries to be loaded.

##### Countermeasures:

- In executing the procedures affected by the environmental variables, be sure to do all of the following countermeasures a) to c), unless they are not applicable:
- a) Make sure that all the environmental variables which affect the process have the supposed values in executing the process.
- b) Designate the library with the absolute path when loading external libraries.
- c) Carry out the countermeasures described in Command Injection in loading externally executable files.

#### 5.6) Double free

##### Vulnerable points:

If the program written by C/C++ frees the memory blocks allocated dynamically by using malloc(), more than once, or new, etc, the heap area may be broken. Once the heap area is broken, the attacker may run any program at will depending on the environment.

##### Countermeasures:

- To prevent vulnerability incurred by repeated opening of memory blocks, take the following countermeasure a):
- a) Soon after opening (free(), delete) the memory blocks which are allocated dynamically by malloc() or new, give NULL to the pointer which has been pointing at the memory blocks.

## 5.7) Access control

### Vulnerable points:

If access permission to the resources inside the product system is improperly treated, the attacker may retrieve, alter, or broke the data. Particularly, files and directories should be properly protected by the access control function provided by the system embedded into products.

### Countermeasures:

- To prevent vulnerability that may bring about unexpected access to the resources, take all of the following countermeasures a) to c), unless they are not applicable:
  - a) Access permission should be limited as much as possible for the resources allocated in the system. Do not give unnecessary access permission to users or programs in the system.
  - b) If multiple programs (or users) with different types of permission share a specific resource, those programs (or users) should be grouped and access permission should be given for the group exclusively.
  - c) Do not place the resources which require access control on media that are not equipped with access control functions. For example, do not place the resources in the media whose file system has no access control function.

## 5.8) Information leakage from temporary files

### Vulnerable points:

If the program uses temporary files improperly, the data recorded on the temporary files may leak or they may generate vulnerable points to be attacked by a symbolic link attack.

### Countermeasures:

- When the program uses temporary files, be sure to take as many items of the following countermeasures a) to d) as possible:
  - a) Place temporary files in the directory prepared for them only. Do not place them in the common directory (such as /tmp for the Unix OS). This is to prevent the attacker from accessing to the temporary files easily.
  - b) With a), least privilege settings for the temporary files and their directories.
  - c) In generating temporary files, give them unpredictable file names. If the same file name has been used, it should be rejected. When the file name is easy to guess, it may give the attacker the opportunity to replace the files or attack them by a symbolic link attack. By checking the existing file names, this procedure will prevent the attacker from opening the prepared files or links.
  - d) Delete the temporary files when the program is turned off.

Items a) to d) are independent from each other and effective individually. The more items are used, the more secure the program will be.

- The Unix OS can use the following function to generate a temporary file safely:

```
int mkstemp(char *template);
```

The Windows OS needs to generate unpredictable file names by itself.

In the case of the Unix OS, it is effective to delete (unlink()) the temporary file soon after it opens (open(), etc). Even if the temporary file is unlink(), the process which has been opened (open(), etc) can access the temporary file. However, the other processes cannot access the temporary files and the attacker cannot use it.

## 5.9) Symbolic link attack

## Vulnerable points:

If the program opens files improperly, it may open symbolic links prepared by the attacker unintentionally. If the program open the symbolic links prepared by the attacker, it may cause unexpected file accesses to the files to leak, alter, or break.

## Countermeasures:

- According to the condition, please take one of these:
  - a) If the program does not need to accept symbolic links, make sure that the link is not a symbolic link when opening the file.
  - b) If the program needs to accept symbolic links, translate the path to open the file into one without any symbolic link and make sure that the path after the translation is authentic.

## 4.3.2 Use-restricted function group 1

The functions shown here often cause the buffer overflow vulnerability.

- 1) Use-restricted function group 1 (Windows: Visual Studio 2005 or later)  
If the target OS of the software is Windows, and its development environment is Visual Studio 2005 or later, do NOT use the use-restricted functions in the table below. Use of the alternative function corresponds to it is recommended.

No.	Use-restricted function	Alternative function
1	gets	gets_s
2	scanf	scanf_s
3	strcpy	strcpy_s
4	strcat	strcat_s
5	sprintf	sprintf_s

- 2) Use-restricted function group 1 (Windows: Visual Studio 2003 or earlier)  
If the target OS of the software is Windows, and its development environment is Visual Studio 2003 or earlier, do NOT use the use-restricted functions in the table below. Use of the alternative function corresponds to it is recommended.

No.	Use-restricted function	Alternative function
1	gets	fgets
2	scanf	sscanf
3	strcpy	strncpy
4	strcat	strncat
5	sprintf	snprintf

3) Use-restricted function group 1 (Linux)

If the target OS of the software is Linux, do NOT use the use-restricted functions in the table below. When the alternative function corresponds to it exists, use of it is recommended.

No.	Use-restricted function	Alternative function
1	gets	fgets
2	scanf	sscanf
3	strcpy	strncpy
4	strcat	strncat
5	sprintf	snprintf

4) Use-restricted function group 1 (Others)

If the target OS of the software is other than the above, do NOT use the use-restricted functions in the table below. When the alternative function corresponds to it exists, use of it is recommended.

No.	Use-restricted function	Alternative function
1	gets	fgets
2	scanf	sscanf
3	strcpy	strncpy
4	strcat	strncat
5	sprintf	snprintf

### 4.3.3 Use-restricted function group 2

Use frequency of the functions shown here is lower than the ones listed in "4.3.2 Use-restricted function group 1," but there is a similar risk. Even for the functions not shown here, design and implement them with consideration for security referring the information written in "Security Note" of MSDN. The list of C standard library functions that need special attention is included as reference information in Appendix B at the back of this Manual.

1) Use-restricted function group 2 (Windows: Visual Studio 2005 or later)

If the target OS of the software is Windows, and its development environment is Visual Studio 2005 or later, do NOT use the use-restricted functions in the table below. Use of the alternative function corresponds to it is recommended.

When C++ is used as a development language, the replacement from the use-restricted function to the alternative function can be partially done by the compilers. This automatic replacement is called the template overloads, and can be used by defining the following three macro variables respectively as 1:

`_CRT_SECURE_CPP_OVERLOAD_STANDARD_NAMES`

`_CRT_SECURE_CPP_OVERLOAD_STANDARD_NAMES_COUNT`

`_CRT_SECURE_CPP_OVERLOAD_SECURE_NAMES`

For details, refer to MSDN (<http://msdn.microsoft.com/en-us/library/ms175759.aspx>).

No.	Use-restricted function	Alternative function
1	<code>__werror</code>	<code>__werror_s</code>
2	<code>_access</code>	<code>_access_s</code>
3	<code>access</code>	<code>_access_s</code>
4	<code>_cgets</code>	<code>_cgets_s</code>
5	<code>cgets</code>	<code>_cgets_s</code>
6	<code>_cgetws</code>	<code>_cgetws_s</code>
7	<code>_chsize</code>	<code>_chsize_s</code>
8	<code>chsize</code>	<code>_chsize_s</code>
9	<code>_controlfp</code>	<code>_controlfp_s</code>
10	<code>_cprintf</code>	<code>_cprintf_s</code>
11	<code>cprintf</code>	<code>_cprintf_s</code>
12	<code>_cprintf_l</code>	<code>_cprintf_s_l</code>
13	<code>_cscanf</code>	<code>_cscanf_s</code>
14	<code>cscanf</code>	<code>_cscanf_s</code>
15	<code>_cscanf_l</code>	<code>_cscanf_s_l</code>
16	<code>_ctime32</code>	<code>_ctime32_s</code>
17	<code>_ctime64</code>	<code>_ctime64_s</code>
18	<code>_cwprintf</code>	<code>_cwprintf_s</code>
19	<code>_cwprintf_l</code>	<code>_cwprintf_s_l</code>
20	<code>_wscanf</code>	<code>_wscanf_s</code>
21	<code>_wscanf_l</code>	<code>_wscanf_s_l</code>
22	<code>_ecvt</code>	<code>_ecvt_s</code>
23	<code>ecvt</code>	<code>_ecvt_s</code>
24	<code>_fcvt</code>	<code>_fcvt_s</code>
25	<code>fcvt</code>	<code>_fcvt_s</code>
26	<code>_fprintf_l</code>	<code>_fprintf_s_l</code>
27	<code>_fscanf_l</code>	<code>_fscanf_s_l</code>
28	<code>_ftime</code>	<code>_ftime_s</code>
29	<code>_ftime32</code>	<code>_ftime32_s</code>
30	<code>_ftime64</code>	<code>_ftime64_s</code>
31	<code>_fwprintf_l</code>	<code>_fwprintf_s_l</code>
32	<code>_fwscanf_l</code>	<code>_fwscanf_s_l</code>
33	<code>_gcvt</code>	<code>_gcvt_s</code>

34	<code>gcvt</code>	<code>_gcvt_s</code>
35	<code>_getws</code>	<code>_getws_s</code>
36	<code>_gmtime32</code>	<code>_gmtime32_s</code>
37	<code>_gmtime64</code>	<code>_gmtime64_s</code>
38	<code>_i64toa</code>	<code>_i64toa_s</code>
39	<code>_i64tow</code>	<code>_i64tow_s</code>
40	<code>_itoa</code>	<code>_itoa_s</code>
41	<code>itoa</code>	<code>_itoa_s</code>
42	<code>_itow</code>	<code>_itow_s</code>
43	<code>_lfind</code>	<code>_lfind_s</code>
44	<code>lfind</code>	<code>_lfind_s</code>
45	<code>_localtime32</code>	<code>_localtime32_s</code>
46	<code>_localtime64</code>	<code>_localtime64_s</code>
47	<code>_lsearch</code>	<code>_lsearch_s</code>
48	<code>lsearch</code>	<code>_lsearch_s</code>
49	<code>_ltoa</code>	<code>_ltoa_s</code>
50	<code>ltoa</code>	<code>_ltoa_s</code>
51	<code>_ltow</code>	<code>_ltow_s</code>
52	<code>_makepath</code>	<code>_makepath_s</code>
53	<code>_mbccpy</code>	<code>_mbccpy_s</code>
54	<code>_mbccpy_l</code>	<code>_mbccpy_s_l</code>
55	<code>_mbcat</code>	<code>_mbcat_s</code>
56	<code>_mbcpy</code>	<code>_mbcpy_s</code>
57	<code>_mbslwr</code>	<code>_mbslwr_s</code>
58	<code>_mbslwr_l</code>	<code>_mbslwr_s_l</code>
59	<code>_mbsnbcats</code>	<code>_mbsnbcats_s</code>
60	<code>_mbsnbcats_l</code>	<code>_mbsnbcats_s_l</code>
61	<code>_mbsnbcpy</code>	<code>_mbsnbcpy_s</code>
62	<code>_mbsnbcpy_l</code>	<code>_mbsnbcpy_s_l</code>
63	<code>_mbsnbset</code>	<code>_mbsnbset_s</code>
64	<code>_mbsnbset_l</code>	<code>_mbsnbset_s_l</code>
65	<code>_mbsncat</code>	<code>_mbsncat_s *2</code>
66	<code>_mbsncat_l</code>	<code>_mbsncat_s_l *2</code>
67	<code>_mbsncpy</code>	<code>_mbsncpy_s *2</code>

Please refrain from copying or reproducing without permission.

68	<code>_mbsncpy_l</code>	<code>_mbsncpy_s_l *2</code>
69	<code>_mbsnset</code>	<code>_mbsnset_s</code>
70	<code>_mbsnset_l</code>	<code>_mbsnset_s_l</code>
71	<code>_mbsset</code>	<code>_mbsset_s</code>
72	<code>_mbsset_l</code>	<code>_mbsset_s_l</code>
73	<code>_mbstok</code>	<code>_mbstok_s</code>
74	<code>_mbstok_l</code>	<code>_mbstok_s_l</code>
75	<code>_mbstowcs_l</code>	<code>_mbstowcs_s_l *2</code>
76	<code>_mbsupr</code>	<code>_mbsupr_s</code>
77	<code>_mbsupr_l</code>	<code>_mbsupr_s_l</code>
78	<code>_mktemp</code>	<code>_mktemp_s</code>
79	<code>mktemp</code>	<code>_mktemp_s</code>
80	<code>_printf_l</code>	<code>_printf_s_l</code>
81	<code>_putenv</code>	<code>_putenv_s</code>
82	<code>putenv</code>	<code>_putenv_s</code>
83	<code>_scanf_l</code>	<code>_scanf_s_l</code>
84	<code>_searchenv</code>	<code>_searchenv_s</code>
85	<code>_snprintf</code>	<code>_snprintf_s *2</code>
86	<code>_snprintf_l</code>	<code>_snprintf_s_l *2</code>
87	<code>_snscanf</code>	<code>_snscanf_s</code>
88	<code>_snscanf_l</code>	<code>_snscanf_s_l</code>
89	<code>_snwprintf</code>	<code>_snwprintf_s *2</code>
90	<code>_snwprintf_l</code>	<code>_snwprintf_s_l *2</code>
91	<code>_snwscanf</code>	<code>_snwscanf_s</code>
92	<code>_snwscanf_l</code>	<code>_snwscanf_s_l</code>
93	<code>_creat</code>	<code>_sopen_s</code>
94	<code>_open</code>	<code>_sopen_s</code>
95	<code>_sopen</code>	<code>_sopen_s</code>
96	<code>creat</code>	<code>_sopen_s</code>
97	<code>open</code>	<code>_sopen_s</code>
98	<code>sopen</code>	<code>_sopen_s</code>
99	<code>_splitpath</code>	<code>_splitpath_s</code>
100	<code>_sprintf_l</code>	<code>_sprintf_s_l</code>
101	<code>_sscanf_l</code>	<code>_sscanf_s_l</code>
102	<code>_strdate</code>	<code>_strdate_s</code>
103	<code>_strerror</code>	<code>_strerror_s</code>
104	<code>_strlwr</code>	<code>_strlwr_s</code>
105	<code>strlwr</code>	<code>_strlwr_s</code>
106	<code>_strlwr_l</code>	<code>_strlwr_s_l</code>
107	<code>_strncat_l</code>	<code>_strncat_s_l *2</code>
108	<code>_strncpy_l</code>	<code>_strncpy_s_l *2</code>
109	<code>_strnset</code>	<code>_strnset_s</code>
110	<code>strnset</code>	<code>_strnset_s</code>
111	<code>_strnset_l</code>	<code>_strnset_s_l</code>
112	<code>_strset</code>	<code>_strset_s</code>
113	<code>strset</code>	<code>_strset_s</code>
114	<code>_strset_l</code>	<code>_strset_s_l</code>
115	<code>_strtime</code>	<code>_strtime_s</code>
116	<code>_strtok_l</code>	<code>_strtok_s_l</code>
117	<code>_strupr</code>	<code>_strupr_s</code>
118	<code>strupr</code>	<code>_strupr_s</code>

119	<code>_strupr_l</code>	<code>_strupr_s_l</code>
120	<code>_swprintf_l</code>	<code>_swprintf_s_l</code>
121	<code>_swscanf_l</code>	<code>_swscanf_s_l</code>
122	<code>_ui64toa</code>	<code>_ui64toa_s</code>
123	<code>_ui64tow</code>	<code>_ui64tow_s</code>
124	<code>_ultoa</code>	<code>_ultoa_s</code>
125	<code>ultoa</code>	<code>_ultoa_s</code>
126	<code>_ultow</code>	<code>_ultow_s</code>
127	<code>_umask</code>	<code>_umask_s</code>
128	<code>umask</code>	<code>_umask_s</code>
129	<code>_vcprintf</code>	<code>_vcprintf_s</code>
130	<code>_vcprintf_l</code>	<code>_vcprintf_s_l</code>
131	<code>_vcwprintf</code>	<code>_vcwprintf_s</code>
132	<code>_vcwprintf_l</code>	<code>_vcwprintf_s_l</code>
133	<code>_vfprintf_l</code>	<code>_vfprintf_s_l</code>
134	<code>_vfwprintf_l</code>	<code>_vfwprintf_s_l</code>
135	<code>_vprintf_l</code>	<code>_vprintf_s_l</code>
136	<code>_vsnprintf</code>	<code>_vsnprintf_s *2</code>
137	<code>_vsnprintf_l</code>	<code>_vsnprintf_s_l *2</code>
138	<code>_vsnwprintf</code>	<code>_vsnwprintf_s *2</code>
139	<code>_vsnwprintf_l</code>	<code>_vsnwprintf_s_l *2</code>
140	<code>_vsprintf_l</code>	<code>_vsprintf_s_l</code>
141	<code>__vswprintf_l</code>	<code>_vswprintf_s_l</code>
142	<code>_vswprintf_l</code>	<code>_vswprintf_s_l</code>
143	<code>_vwprintf_l</code>	<code>_vwprintf_s_l</code>
144	<code>_waccess</code>	<code>_waccess_s</code>
145	<code>_wasctime</code>	<code>_wasctime_s</code>
146	<code>_wcserror</code>	<code>_wcserror_s</code>
147	<code>_wcslwr</code>	<code>_wcslwr_s</code>
148	<code>wcslwr</code>	<code>_wcslwr_s</code>
149	<code>_wcslwr_l</code>	<code>_wcslwr_s_l</code>
150	<code>_wcsncat_l</code>	<code>_wcsncat_s_l *2</code>
151	<code>_wcsncpy_l</code>	<code>_wcsncpy_s_l *2</code>
152	<code>_wcsnset</code>	<code>_wcsnset_s</code>
153	<code>wcsnset</code>	<code>_wcsnset_s</code>
154	<code>_wcsnset_l</code>	<code>_wcsnset_s_l</code>
155	<code>_wcsset</code>	<code>_wcsset_s</code>
156	<code>wcsset</code>	<code>_wcsset_s</code>
157	<code>_wcsset_l</code>	<code>_wcsset_s_l</code>
158	<code>_wcstok_l</code>	<code>_wcstok_s_l</code>
159	<code>_wcstombs_l</code>	<code>_wcstombs_s_l *2</code>
160	<code>_wcsupr</code>	<code>_wcsupr_s</code>
161	<code>wcsupr</code>	<code>_wcsupr_s</code>
162	<code>_wcsupr_l</code>	<code>_wcsupr_s_l</code>
163	<code>_wctime</code>	<code>_wctime_s</code>
164	<code>_wctime32</code>	<code>_wctime32_s</code>
165	<code>_wctime64</code>	<code>_wctime64_s</code>
166	<code>_wctomb_l</code>	<code>_wctomb_s_l</code>
167	<code>_wfopen</code>	<code>_wfopen_s</code>
168	<code>_wfreopen</code>	<code>_wfreopen_s</code>
169	<code>_wgetenv</code>	<code>_wgetenv_s</code>

Please refrain from copying or reproducing without permission.

170	_wmakepath	_wmakepath_s
171	_wmktemp	_wmktemp_s
172	_wprintf_l	_wprintf_s_l
173	_wputenv	_wputenv_s
174	_wscanf_l	_wscanf_s_l
175	_wsearchenv	_wsearchenv_s
176	_wcreat	_wsopen_s
177	_wopen	_wsopen_s
178	_wsopen	_wsopen_s
179	_wsplitpath	_wsplitpath_s
180	_wstrdate	_wstrdate_s
181	_wstrtime	_wstrtime_s
182	_wtmpnam	_wtmpnam_s
183	asctime	asctime_s
184	bsearch	bsearch_s
185	clearerr	clearerr_s
186	ctime	ctime_s
187	fopen	fopen_s
188	fprintf	fprintf_s
189	freopen	freopen_s
190	fscanf	fscanf_s
191	fwprintf	fwprintf_s
192	fwscanf	fwscanf_s
193	getenv	getenv_s
194	gmtime	gmtime_s
195	localtime	localtime_s
196	mbsrtowcs	mbsrtowcs_s *2
197	mbstowcs	mbstowcs_s *2
198	memcpy	memcpy_s
199	memmove	memmove_s
200	printf	printf_s
201	qsort	qsort_s
202	rand	rand_s *1
203	sscanf	sscanf_s
204	strerror	strerror_s
205	strncat	strncat_s *2
206	strncpy	strncpy_s *2
207	strtok	strtok_s
208	swprintf	swprintf_s
209	swscanf	swscanf_s
210	tmpfile	tmpfile_s
211	tmpnam	tmpnam_s
212	vfprintf	vfprintf_s
213	vfwprintf	vfwprintf_s
214	vprintf	vprintf_s
215	vsnprintf	vsnprintf_s *2
216	vsprintf	vsprintf_s
217	vswprintf	vswprintf_s
218	vwprintf	vwprintf_s
219	wcrtomb	wcrtomb_s
220	wcscat	wcscat_s

221	wscpy	wscpy_s
222	wcsncat	wcsncat_s *2
223	wcsncpy	wcsncpy_s *2
224	wcsrtombs	wcsrtombs_s *2
225	wcstok	wcstok_s
226	wcstombs	wcstombs_s *2
227	wctomb	wctomb_s
228	wmemcpy	wmemcpy_s
229	wmemmove	wmemmove_s
230	wprintf	wprintf_s
231	wscanf	wscanf_s

- \*1 rand\_s() can be used only in the software targeting Windows XP or later.
- \*2 If "\_TRUNCATE" is specified as a parameter, a string exceeding the length of a destination buffer is cutoff. An error check or other feature is needed to prevent malfunction of the program caused by a truncated string.

2) Use-restricted function group 2 (Windows: Visual Studio 2003 or earlier)  
 If the target OS of the software is Windows, and its development environment is Visual Studio 2003 or earlier, do NOT use the use-restricted functions in the table below. Use of the alternative function corresponds to it is recommended.

No.	Use-restricted function	Alternative function
1	<code>_getws</code>	<code>fgetws</code>
2	<code>_mbscat</code>	<code>_mbsncat, _mbsnbcac</code>
3	<code>_mbscopy</code>	<code>_mbsncpy, _mbsnbcpy</code>
4	<code>swprintf</code>	<code>_snwprintf</code>
5	<code>vsprintf</code>	<code>_vsnprintf</code>
6	<code>vswprintf</code>	<code>_vsnwprintf</code>
7	<code>wcscat</code>	<code>wcsncat</code>
8	<code>wscopy</code>	<code>wcsncpy</code>

Note) Visual Studio 2003 or earlier has a few alternative functions that contribute security improvement, that is why the table is short.

3) Use-restricted function group 2 (Linux)  
 If the target OS of the software is Linux, do NOT use the use-restricted functions in the table below. When the alternative function corresponds to it exists, use of it is recommended.

No.	Use-restricted function	Alternative function
1	<code>vsprintf</code>	<code>vsnprintf</code>
2	<code>wcscat</code>	<code>wcsncat</code>
3	<code>wscopy</code>	<code>wcsncpy</code>

Note) Linux (glibc) has a few alternative functions that contribute security improvement, that is why the table is short.

4) Use-restricted function group 2 (Others)  
 If the target OS of the software is other than the above, do NOT use the use-restricted functions in the table below. When the alternative function corresponds to it exists, use of it is recommended.

No.	Use-restricted function	Alternative function
1	<code>vsprintf</code>	<code>vsnprintf</code>
2	<code>wcscat</code>	<code>wcsncat</code>
3	<code>wscopy</code>	<code>wcsncpy</code>

#### 4.3.4 Format string

The codes for products shall follow the restrictions below.

- 1) Never use unvalidated inputs from outside as a format string.
  - 1.1) Do not pass the inputs from outside of the program to a format string of printf/scanf family functions without validation. Because if the input from outside has a type specifier, the program will be out of control of the programmer.

Example of prohibition:

```
FILE* fp;
char buf[256];
fgets(buf, sizeof(buf), fp);
printf_s(buf); /* Do not give an external input to a format string */
Instead, give the input to a format with a type specifier "%s", or substitute with puts.
printf_s("%.255s", buf);
```

- 1.2) Specify string constant to format string of printf/scanf family functions. The security can be enhanced by preventing operation and change from outside to the format string.

Example:

```
static const char format[] = "Error (%d): %s\n";
...
printf_s(format, errno, description);
```

- ~~2) Never use the type specifier "n"~~

This clause, 2) Never use the type specifier "n" is deleted, because there is no risk as far as the programmers knowingly use the type specifier "n" to format string. However, when input from outside is given to the format string, be careful not to mix the type specifier "n."

- 3) The output by the type specifier "s" specifies accuracy  
When passing a format string that contains the type specifier "s" to sprintf function family, specify accuracy that matches the buffer size that outputs the value.

Example:

```
char *key, *value;
char buf[32];
/* 32 = 15 + 1 ('=') + 15 + 1 ('\0') */
sprintf_s(buf, sizeof(buf)/sizeof(buf[0]),
"%15s=%15s", key, value); /* Specify accuracy. */
```

- 4) Specify width of the input by type specifier "c" or "s"  
When passing a format string that contains the type specifier "c" or "s" to scanf function family, specify width that matches the buffer size that receives the value.

Example:

```
char buf[20];
sscanf_s(input, "%19s", buf, sizeof(buf)/sizeof(buf[0]));
```

#### 4.3.5 ActiveX controls

ActiveX controls have various vulnerabilities. Even locally installed ActiveX controls can be operated via Web pages, and are used as entry path by the attackers.

Verify the value of an input from outside the control before use, and make sure the value is within the range assumed at design stage. Most vulnerability problems can be avoided by checking the input values and eliminating the illegal inputs.

- 1) Limitation on execution site  
Limit the Web sites that can launch ActiveX controls used only on specific sites. Limiting the execution to the local computers only is effective defense that prevents misuse from the Web pages on the Internet.  
To implement the above mentioned measures, SiteLock Template for ActiveX Controls (<http://www.microsoft.com/downloads/details.aspx?FamilyID=43cd7e1e-5719-45c0-88d9-ec9ea7fefbcb&displaylang=en>) provided by Microsoft is available.
  - 2) Digital signatures  
The digital signatures enable the users to check illegal falsification and the publisher of the file. The digital signatures shall be properly used to provide authorized files to the users.
    - 2.1) Addition of digital signature  
Add the digital signature to ActiveX controls.  
The digital signature shall be added to all ActiveX controls, even to the ones which are not delivered through network, and which do not have network connection feature, because they can be launched from Web pages.
    - 2.2) Certificate for digital signature  
Purchase a certificate for the digital signatures from a certificate authority that is well-known and its root certificate has been pre-installed in personal computers. For example, VeriSign, Inc., Entrust, Inc., GlobalSign, Inc., and Verizon Business (Cybertrust).
    - 2.3) Construction of original certificate authority and use of original certificate  
Avoid constructing an original certificate authority, and use of the certificates issued by the original certificate authority. This is because there are problems in the management of certificate authority and distribution of the certificates to the users.
  - 3) Safe For Initialization/Scripting mark  
Do not mark an ActiveX control as Safe for Scripting and Initialization, if it executes any one of the following processes.
    - I. Arbitrarily specifies <sup>(\*)</sup> the information stored in a local computer, a file system on a network, registry settings, a camera, or an USB device in order to write, read, create, detect or delete it.
    - II. Discloses private information such as a private key, a password, and a document.
    - III. Excessively consumes time and the resource in a memory or a disk area.
    - IV. Executes a system call including file execution, which has the possibility of damage.When "Safe For Initialization/Scripting" is not marked, even if an unauthorized site is hosting an ActiveX control, the ActiveX container will alert users before a malicious operation is performed
- (\*) "Arbitrarily specifies" means capability to manipulate the name of a path or data by some input at execution time.

## 5. REFERENCES

- Deprecated CRT Functions  
<http://msdn.microsoft.com/en-us/library/ms235384.aspx>
- Security Enhancements in the CRT  
<http://msdn.microsoft.com/en-us/library/8ef0s5kh.aspx>
- Parameter Validation  
<http://msdn.microsoft.com/en-us/library/ksazx244.aspx>
- Fortify Taxonomy: Software Security Errors  
<https://www.fortify.com/vulncat/>

## APPENDIX

### APPENDIX A

- 1) Sample code for verification of server certificate on HTTPS connection  
 To confirm CRL using WinHTTP, use WinHttpSetOption API and activate WINHTTP\_ENABLE\_SSL\_REVOCATION feature of the request object (this feature is inactive by default). Below is the sample code to activate the CRL confirmation feature and analyze the errors related to CRL if they exist.

```
#include <stdio.h>
#include <windows.h>
#include <winhttp.h>
#pragma comment(lib, "winhttp")

struct callback_param_t
{
    HINTERNET hInet;
    DWORD dwErrCert;
};

static VOID CALLBACK SyncCallback(HINTERNET, DWORD_PTR, DWORD, PVOID, DWORD);

DWORD ConnectHTTPSFunc(LPCWSTR pswzServerName,
                      LPCWSTR pswzObjectName,
                      LPDWORD lpdwErrCert)
{
    DWORD dwErr = ERROR_SUCCESS;
    HINTERNET hSession = NULL;
    HINTERNET hConnect = NULL;
    HINTERNET hRequest = NULL;

    if(NULL == lpdwErrCert)
    {
        *lpdwErrCert = 0;
    }

    hSession = ::WinHttpOpen(0,
        WINHTTP_ACCESS_TYPE_NO_PROXY,
        WINHTTP_NO_PROXY_NAME,
        WINHTTP_NO_PROXY_BYPASS,
        0);

    if(NULL == hSession)
    {
        dwErr = ::GetLastError();
    }
    else
    {
        hConnect = ::WinHttpConnect(hSession,
            pswzServerName,
            INTERNET_DEFAULT_HTTPS_PORT,
            0);

        if(NULL == hConnect)
        {
            dwErr = ::GetLastError();
        }
        else
        {
            // Use WINHTTP_FLAG_SECURE flag to verify CRL
            hRequest = ::WinHttpOpenRequest(hConnect,
                NULL,
                pswzObjectName,
                0,
                WINHTTP_NO_REFERER,
                WINHTTP_DEFAULT_ACCEPT_TYPES,
                WINHTTP_FLAG_SECURE);

            if(NULL == hRequest)

```

Please refrain from copying or reproducing without permission.

```

    {
        dwErr = ::GetLastError();
    }
    else
    {
        DWORD dwOpt = WINHTTP_ENABLE_SSL_REVOCATION;

        const BOOL bSetOptionResults = ::WinHttpSetOption(hRequest,
            WINHTTP_OPTION_ENABLE_FEATURE,
            &dwOpt,
            sizeof(dwOpt));

        if(!bSetOptionResults)
        {
            dwErr = ::GetLastError();
        }
        else
        {
            callback_param_t param;
            param.hInet = hRequest;
            param.dwErrCert = 0;

            const WINHTTP_STATUS_CALLBACK isCallback
            = ::WinHttpSetStatusCallback(hRequest,
                SyncCallback,
                WINHTTP_CALLBACK_FLAG_SECURE_FAILURE,
                0);

            if(WINHTTP_INVALID_STATUS_CALLBACK == isCallback)
            {
                dwErr = ::GetLastError();
            }
            else
            {
                const BOOL bSendResults = ::WinHttpSendRequest(hRequest,
                    WINHTTP_NO_ADDITIONAL_HEADERS,
                    0,
                    WINHTTP_NO_REQUEST_DATA,
                    0,
                    0,
                    reinterpret_cast<DWORD_PTR>(&param));

                if(!bSendResults)
                {
                    dwErr = ::GetLastError();

                    // Value is set to lpdwErrCert, if an error occurred in CRL check.
                    if(lpdwErrCert)
                    {
                        *lpdwErrCert = param.dwErrCert;
                    }
                }
                else
                {
                    // Place additional code here.
                    // For instance, receive response
                }
            }
        }
        ::WinHttpCloseHandle(hRequest);
    }
    ::WinHttpCloseHandle(hConnect);
}
::WinHttpCloseHandle(hSession);
}

return dwErr;
}

static VOID CALLBACK SyncCallback(HINTERNET inet,
    DWORD_PTR context,
    DWORD status,
    PVOID information,
    DWORD informationLength)

```

```

{
    callback_param_t &p = *reinterpret_cast<callback_param_t*>(context);
    const DWORD flag = reinterpret_cast<DWORD>(information);

    if((0 != context) &&
        (inet == p.hInet) &&
        (WINHTTP_CALLBACK_STATUS_SECURE_FAILURE == status) &&
        (sizeof(DWORD) == informationLength))
    {
        p.dwErrCert = flag;
    }
}

int main(int argc, char **argv)
{
    DWORD dwErrCert = 0;
    DWORD dwErr = ConnectHTTPSFunc(L"www.sample.com", L"/", &dwErrCert);

    if((ERROR_SUCCESS != dwErr) && (0 != dwErrCert))
    {
        if(dwErrCert & WINHTTP_CALLBACK_STATUS_FLAG_CERT_REV_FAILED)
        {
            puts("Certification revocation checking has been enabled, "
                "but the revocation check failed to verify whether "
                "a certificate has been revoked. The server used "
                "to check for revocation might be unreachable.");
        }
        if(dwErrCert & WINHTTP_CALLBACK_STATUS_FLAG_INVALID_CERT)
        {
            puts("SSL certificate is invalid.");
        }
        if(dwErrCert & WINHTTP_CALLBACK_STATUS_FLAG_CERT_REVOKED)
        {
            puts("SSL certificate was revoked.");
        }
        if(dwErrCert & WINHTTP_CALLBACK_STATUS_FLAG_INVALID_CA)
        {
            puts("The function is unfamiliar with the Certificate "
                "Authority that generated the server's certificate.");
        }
        if(dwErrCert & WINHTTP_CALLBACK_STATUS_FLAG_CERT_CN_INVALID)
        {
            puts("SSL certificate common name (host name field) "
                "is incorrect, for example, if you entered "
                "www.microsoft.com and the common name on the "
                "certificate says www.msn.com.");
        }
        if(dwErrCert & WINHTTP_CALLBACK_STATUS_FLAG_CERT_DATE_INVALID)
        {
            puts("SSL certificate date that was received from the "
                "server is bad. The certificate is expired.");
        }
        if(dwErrCert & WINHTTP_CALLBACK_STATUS_FLAG_SECURITY_CHANNEL_ERROR)
        {
            puts("The application experienced an internal error "
                "loading the SSL libraries.");
        }
        if(dwErrCert & WINHTTP_CALLBACK_STATUS_FLAG_CERT_WRONG_USAGE)
        {
            puts("WINHTTP_CALLBACK_STATUS_FLAG_CERT_WRONG_USAGE");
        }
    }

    return dwErr;
}

```

## APPENDIX B

### 1) C standard library functions need special attention to buffer operation

The table below lists the standard library functions of C language (ISO/IEC 9899:1999, JIS X 3010:2003), to which security enhancement by buffer boundary check is proposed in ISO/IEC TR 24731-1.

Especially, when developing a product that can not use the alternative functions equivalent to ISO/IEC TR 24731-1, read the specifications and the manuals of these functions to understand their behavior, and pay attention not to make vulnerability of the buffer overflow.

asctime
bsearch
ctime
fopen
fprintf
freopen
fscanf
fwprintf
fwscanf
getenv
gets
gmtime
localtime
mbsrtowcs
mbstowcs
memcpy
memmove
printf
qsort
scanf
snprintf
sprintf
sscanf
strcat
strcpy
strerror
strncat
strncpy
strtok
swprintf

swscanf
tmpfile
tmpnam
vfprintf
vfscanf
vfwprintf
vfwscanf
vprintf
vscanf
vsprintf
vsprintf
vsscanf
vswprintf
vswscanf
vwprintf
vwscanf
wcrtomb
wcscat
wcscpy
wcsncat
wcsncpy
wcsrtombs
wcstok
wcstombs
wctomb
wmemcpy
wmemmove
wprintf
wscanf

## APPENDIX C ANALYSIS OF SOURCE CODE WITH THE SUPPORT OF COMPILER

- 1) A static analysis of source code by Microsoft C compiler  
You can use source code static analyzer as a feature in the C compiler (c1.exe) attached to Microsoft Windows SDK.

When you add /analyze to starting up options of the compiler, it does boundary check of buffer or a matching test between a string and the parameter or argument, etc, and lists the positions of source codes which vulnerability is attributed to.

For example, input c1.exe /analyze /c test.c in a command line shell of SDK in order to analyze the following source code, test.c.

```
/* test.c -- for demonstrate /analyze compiler option */
#include <stdio.h>
static char g_buf[10];
char test(size_t i)
{
    if(i >= 10)
        fprintf_s(stderr, "out of range");
    return g_buf[i];
}
```

After analyzing test.c, the following warning message is output.

```
warning C6385: Invalid data: accessing 'g_buf', the readable size is
'10' bytes, but '11' bytes might be read: Lines: 8, 9, 10
```

Although the function test outputs an error message when the value of parameter i is out of the boundary, it warns that there is a problem in accessing the ith element of g\_buf regardless of the value of i.

Also, even in Visual Studio (IDE), you can use a feature of static analyzer by opening "property of project" and adding /analyze to "additional option" of "composition property |C/C++ | command line".

Since you have only to set up compiler options to use it, please be sure to incorporate it in setup of the development environment as a measure against vulnerability.

- 2) SAL annotations by Microsoft C compiler  
A static analysis of source code by Microsoft C compiler is made based on the SAL (Microsoft Standard Sourcecode Annotation Language, <http://msdn.microsoft.com/en-us/library/ms235402.aspx>) annotation.

Its main purpose is boundary check of buffer. Some annotations are about a matching test between a string and the parameter argument and detection of a failure of checking return value.

The SAL annotations can be used not only in API provided by Microsoft but in a source code you are developing. The annotated source code is tested by the compiler from both mounting side (to be called) and client side (to call). If you put the annotation in the proclamation of the header file, the compiler detects problems very precisely without changing the way the existing mounting works. Examples of major SAL annotations are given below for your reference in development.

`_In_z_`

A string of letters that the function will take in to read and that will terminate with the characters '\0'.  
e.g. void f(\_In\_z\_ const TCHAR\* filename);

`_In_count_(size)`

An arrangement that the function will take in to read. It has the number of elements specified by an argument.

e.g. int sum(\_In\_count\_(nr) const int\* array, size\_t nr);

\_In\_bytecount\_(size)

The buffer that the function will take in to read from. It has the number of bytes specified by an argument.

e.g. void process(\_In\_bytecount\_(size) const unsigned char\* image, size\_t size);

\_Out\_z\_cap\_(size)

The buffer that the function will take in to write to. It may write up to the number of elements specified by an argument therein.

The buffer will terminate with the characters '\0' when it is out of control of the function.

e.g. void getFilename(\_Out\_z\_cap\_(length) TCHAR\* name, size\_t length);

\_Out\_cap\_(size)

The buffer that the function will take in to write to. It may write up to the number of elements specified by an argument therein.

e.g. void sort(\_Out\_cap\_(nr) int\* result, \_In\_count\_(nr) const int\* array, size\_t nr);

\_Out\_cap\_c\_(size)

The buffer that the function will take in to write to. It may write up to the number of elements specified by a constant expression therein.

e.g. void getWorkdir(\_Out\_cap\_c\_(MAX\_PATH) TCHAR\* path);

\_Out\_bytecap\_(size)

The buffer that the function will take in to write to. It may write up to the number of bytes specified by an argument therein.

e.g. void getBinary(\_Out\_bytecap\_(size) unsigned char\* value, size\_t size, \_In\_z\_ TCHAR\* key);

\_Deref\_out\_z\_

The pointer taking in buffer that the function will obtain and return.

e.g. void dupStr(\_Deref\_out\_z\_ TCHAR\*\* newStr, \_In\_z\_ const TCHAR\* fromStr);

\_Inout\_z\_cap\_(size)

The buffer that the function may read from and write to. It may read and write up to the number of elements specified by an argument therein. The function will be given an argument that terminates with the characters '\0' and the result will terminate as well.

e.g. void truncate(\_Inout\_z\_cap\_(size) TCHAR\* path, size\_t size);

\_Ret\_z\_

The return value of the function that will terminate with the characters '\0'.

e.g. \_Ret\_z\_ const TCHAR\* getName(ID id);

\_Printf\_format\_string\_

The argument is a string of the function in the style of printf.

e.g. void DebugTrace(\_Printf\_format\_string\_ const TCHAR\* format, ...);

\_Scanf\_s\_format\_string\_

The argument is a string of the function in the style of scanf\_s.

e.g. void readData(\_Scanf\_s\_format\_string\_ const TCHAR\* format, ...);

## APPENDIX D PRECAUTIONS FOR ADJUSTING CODES

### 1) Checklist

The checklist below includes precautions to be noted when you start to write codes, replace use-restricted functions with alternative ones, and deal with vulnerability problem called buffer overflow. This can be called a summary of the explanations made later. You are requested to refer to it together with the explanations for the purpose of writing codes securely.

The buffer size of a copy destination buffer is secured to be larger than the number of elements of a copy source.

The size (which is specified by every argument replaced) of a copy destination buffer is smaller than a size secured actually.

The buffer is managed by using the structure having pointer and size as members. Delivery between buffers is not made by just a pointer without any information about a buffer size.

The buffer is managed by using container class C++. Delivery between buffers is not made by just a pointer without any information about a buffer size.

`std::vector` is used as the buffer for writing a string of C++. As the buffer for writing a string, `std::string` must not be used.

The size variable delivered to the string function must be an unsigned integer such as `size_t` or `unsigned long`. An integer with a minus sign must not be delivered.

The type of the variable (character type or string type) to be delivered to the generic string function matches the requested type (TCHAR or LPTSTR).

Macros for counting the number of elements are used. In Windows, the macro, `countof` is used. In other platforms, the following macros are used: `#define LENGTH(array) sizeof(array) / sizeof((array)[0])`

The function for counting the number of elements C++ is used. `:template<typename T, size_t N> inline size_t length_of(T (&)[N]) { return N; }`

In `strncpy_s(dst, dst_size, src, src_len)`, it says "`dst_size == _countof(dst)`" and "`dst_size > src_len`". "`dst_size`" and "`src_len`" are not the same constant or variable.

When reading characters and strings by the function, `sscanf`, buffer size is limited using width specifiers ("`%7s`", etc. for arrangement of char type which is eight element long).

NULL termination characters is taken into account when copying by `strncpy`. \*NULL character is not added if number of letters copied are equal to the length of arrangement as a copy destination.

The buffer size which deserves the number of NULL characters at termination that are surely added by `strncat` is taken into account. \*"number of letters copied" is added after NULL characters at termination as a copy destination.

When extracting characters ("`%c`", "`%C`") and strings ("`%s`", "`%S`") by `sscanf_s`, a pointer and buffer size are given as a corresponding parameter.

The `void *` type argument is given to `memXXX`, and the buffer size is specified by a "number of bytes".

The `char *` type argument is given to `strXXX`, and the buffer size is specified by a "number of elements".

The `char *` type argument is given to `mbsXXX`, and the buffer size is specified by a "number of elements".

The `wchar_t *` type argument is given to `wcsXXX`, and the buffer size is specified by a "number of elements".

The `TCHAR *` (`TCH*`, `LPTSTR`, `LPCTSTR`, `PTSTR`, `PCTSTR`, etc.) type argument is given to `tcsXXX`, and the buffer size is specified by a "number of elements".

### 2) Put together a pointer and size of buffer.

The major factor causing a problem of buffer overflow when using C/C++ language lies in the fact that only a pointer pointing to the head is given to the function, and processed without the limitation of buffer size or terminus.

In C/C++ language, a pointer and size of buffer can be put together by using a container class (`std::vector` and `std::list`, etc.) or a string class (`std::string`) of STL. Especially, for a new part of mounting it is recommended to use a container class or a string class of STL willingly. Also, design in the way that a pointer is picked out from a container class or a string class and given only when a pointer is requested in the API of a standard library or platform.

Examples:

```
std::vector<char> buf(MAX_PATH);
strncpy(&buf.at(0), dir, MAX_PATH);
```

- \* The pointer which is returned by the `c_str` function of `std::string` is the `const char *` type and not rewritable. In other words, it can be used as a copy source of the `strncpy` function, but must not be used as a copy destination.

Although you can describe `strncpy (&buf.at(0), dir, MAX_PATH);` using `std::string`, it should not be done. The problem of mobility can arise because C++ standard does not assure that the internal memory area is seamless.

If the buffer of rewritable string is necessary, you had better use `std::vector<char>`, `std::vector<wchar_t>`, `std::vector<TCHAR>`.

In the case of C language, it is useful to prepare a structure putting together the size and pointer of buffer and design to put together them using the structure when using buffer on a new part of mounting.

Examples:

```
typedef struct container_tag {
    size_t size;
    void *p;
} container_t;

container_t malloc_container(size_t size) {
    container_t v = { size, malloc(size) };
    if (!v.p) v.size = 0;
    return v;
}

void free_container(container_t v) {
    free(v.p);
}

...
container_t v = malloc_container(strlen(filepath) + 1);
strncpy(v.p, filepath, v.size);
```

### 3) Specifications of C standard functions to be noted

`strncpy`:

Copy the range from `src` of the second argument to the letter `n` of the third one at the maximum to `dst` of the first argument. Note that there are some cases where it does not terminate with NULL characters. For `strlen(src) < n`, copy `strlen(src)` letter of `src` to `dst`. Moreover, fill the range up to the `n`th element of `dst` with NULL characters.

For `strlen(src) >= n`, copy `n` letter of `src` to `dst`. In this case, `dst` does not terminate with NULL character.

`strncat`:

Add the range from `src` of the second argument to the letter `n` of the third one at the maximum to `dst` of the first argument and add NULL characters. Note that `n` is not the size of `dst` as a copy destination but the number of letters added at the end of it. Also, it is necessary to confirm before the call that the size of `dst` is larger than `strlen(dst) + n + 1` because NULL terminated characters are added.

For `strlen(src) < n`, copy `strlen(src)` letter of `src` to after the `strlen(dst)`th element of `dst` and add NULL characters.

For `strlen(src) >= n`, copy `n` letter of `src` to after the `strlen(dst)`th element of `dst` and add NULL characters.

`sprintf`:

The first argument `dst` terminates surely with NULL characters. Write `n` letters specified by the second argument at the maximum. If the string of characters formatted after the third argument exceeds `n` letters, the range to `n-1` letter is copied to `dst`. The `n-1`st element of `dst` becomes NULL characters.

`scanf`, `sscanf`:

Be careful of using `scanf` and `sscanf` when a string of the second argument includes type specifiers `s` or `c`.

This is because they analyze a string of the first argument and write "multiple characters" in the buffer they refer to in the argument on and after the third argument. Careless use of them can cause buffer overflow.

In this regard, it is necessary to use `sscanf` and inspect a string to be input in advance or to restrict it within the specified width. (For example, replace `%s` which is the form corresponding to the buffer of 16 letters with `%15s`.)

```
char buf[256];
char mystr[128];
sscanf(buf, "%s", mystr);
/* As buf can store a longer string than mystr, it can cause buffer overflow. */
```

```
char buf[16];
char mystr[16];
sscanf(buf, "%s", mystr);
/* This function can be used safely if '\0' and blanks are included before the 16th letter of buf. If buf is spoiled by buffer overflow, which is to be checked in the first place, mystr can overflow. */
```

```
sscanf(buf, "%15s", mystr);
/* Using width specifier is safe as writing over the size of mystr is prohibited. As a security countermeasure, it is enough but there is a possibility of missing inputs of buf. In order to prevent malfunction due to the missing, it is recommended to confirm that buf[strlen(mystr)] is a type of blanks. */
```

```
sscanf_s(buf, "%s", mystr, _countof(mystr));
/* When using the development environment according to TR 24731-1, use of sscanf_s is the safest measure. */
```

- 4) Precautions for extracting a letter or a string of letters by `sscanf_s`  
 Type specifier `c/C` or `s/S` in a format string of letters such as specifies two of a) a pointer at the head of the buffer and b) the number of elements of the corresponding buffer by the following parameters.

```
char c, bufc[...];
wchar_t wc, bufw[...];
scanf_s("%c %s %C %S",
&c, 1, bufc, _countof(bufc),
&wc, 1, bufw, _countof(bufw));
```

\* `countof` is used only for an arrangement of local and global variables in functions or an arrangement in the structure or class.

- 5) A number of letters and buffer size in the process of a string of letters  
 Codes such as `strncpy_s(d, len, s, len);` specifying the same value for a number of letters to be copied and buffer size as a copy destination are probably incorrect.  
 Note that especially "number of letter" in the function for string operation does not include NULL termination characters ('\0'). For example, in `strncpy_s`, a number of letters specified by the fourth argument as a copy source does not include NULL termination characters. On the other hand, the specification stipulates that the second argument should specify buffer size enough to store letters including NULL termination characters. In the example of `strncpy_s` at the beginning, if `strlen(s)` is equal to `len`, it fails because there is no vacancy for adding NULL termination characters in `d`, the arrangement as a copy destination, the size of which is `len`.

Even if the second and fourth argument seem to specify different variables, it is recommended to note and confirm that the same value is not used when substituting just before as can be seen in `size_t dst1 = len; strncpy_s(d, dst1, s, len);`.

In addition, if `len` is a signed integer (eg. `int` or `long`), it is necessary to confirm that the value is not minus just before calling a function. In a standard C specification, `size_t` as a type of unsigned integer is used for a buffer size. When substituting a signed integer for an unsigned integer, the sign bit loses meaning and is treated as one of numerical value. A minus number is treated as a giant integer. As a result, it tries to copy too many elements and comes to give a function the buffer with capacity much larger than its actual size. It leads to an occurrence of buffer overflow.

If `len` is an unsigned integer, pay attention to the subtraction for this variable. If the result of subtraction is below 0, it becomes a giant integer unexpectedly through integer overflow. It also leads to an occurrence of buffer overflow.

- 6) Buffer type given to the function treating generic string  
 Confirm that buffer type of a string given to the Windows generic string function (a function having a `t`-prefix such as `tcscopy`, etc. :mostly) is `TCHAR*` or `LPTSTR`.
- 7) Unit of buffer size specified by the function treating a string  
 The buffer size of `memcpy`, `memmove` is specified by the byte. Other string functions and `wmemcpy`, etc. are specified by the element.  
 Under Windows, `_countof` can be used as a macro for counting the number of elements of arrangement. Note that it can not be used for a pointer.

The macros for counting the number of elements of arrangement in C language can be defined as follows.

```
#define LENGTH(array) sizeof(array) / sizeof((array)[0])
```

It is recommended to define and use a template function for getting the length of arrangement in C++ . It can prevent a mistake of using `LENGTH`, macro for a pointer.

```
template<typename T, size_t N> inline size_t length_of(T (&)[N]) { return N; }
```

8) Examples of mistakes

To specify the value bigger than the buffer size proclaimed for the buffer size given to a function.

e.g. `char buf[16]; ... ; snprintf(buf, 256, ...)`

It is correct to give the number of elements. `snprintf(buf, _countof(buf), ...)`, etc.

To use `sizeof` when you should count the number of buffer elements.

e.g. `TCHAR buf[N]; _tcscpy_s(buf, sizeof(buf), ...)`

It is correct to give the number of elements. `_tcscpy_s(buf, _countof(buf), ...)`, etc.

When building generic letters for Windows with unicode character set, use of `sizeof` is regarded as specification of double buffer size.

To give the arrangement of `char` or `wchar_t` and pointer to the generic string function for Windows.

e.g. `char buf[6]; _stprintf_s(buf, _T("%02d:%02d"), major, minor);`

It is correct to match types of function argument. Change to `TCHAR buf[6];` or use `sprintf_s`.

In this example, a type mismatch happens when building with unicode character set, When matching types forcibly using cast, buffer overflow occurs due to a size mismatch.

To substitute `sscanf` for `scanf` simply.

e.g. substitute `char buf[8]; sscanf(instr, "%s", buf);` for `char buf[8]; scanf("%s", buf);`

It is correct to restrict input length. For example, change to `sscanf(instr, "%7s", buf);` or use `sscanf_s`, etc.

When inputting "12345678 90" through standard input using codes before substitution, overflow occurs. Even if after substitution, when the content of `instr` is "12345678 90", overflow still occurs.

(Note)

This document is subject to change without prior notice, as a result of a revision or modifications on the STM-0117, the Sony Technical Manual titled "Standard for Product Security Assurance."

Standard for Product Security Assurance

STM-0117 for General Use, Eighth Edition

Enforced 2012.03.01

Issued by Secretariat of the Sony Technical Standards, Sony Corporation