IceWall SSO

Version 10.0

Session ID Generation Routine Developer's Manual

August 2010

Printed in Japan

HP Part No. B1544-97017

Rev.111007A

Notice

The information contained in this document is subject to change without notice.

Meticulous care has been taken in the preparation of this document, however, if a questionable or erroneous item, or an omission of content is found, please contact us.

Hewlett-Packard Japan, Ltd. shall not be liable for errors contained herein or for incidental or consequential damage in connection with the furnishing, performance or use of this document.

This manual and media, such as the CD-ROM supplied as a part of the product's package, are only to be used with this product.

– Table of Contents –

1 Introduction

This manual provides information required to develop the session ID generation routine to be integrated into the Authentication Module.

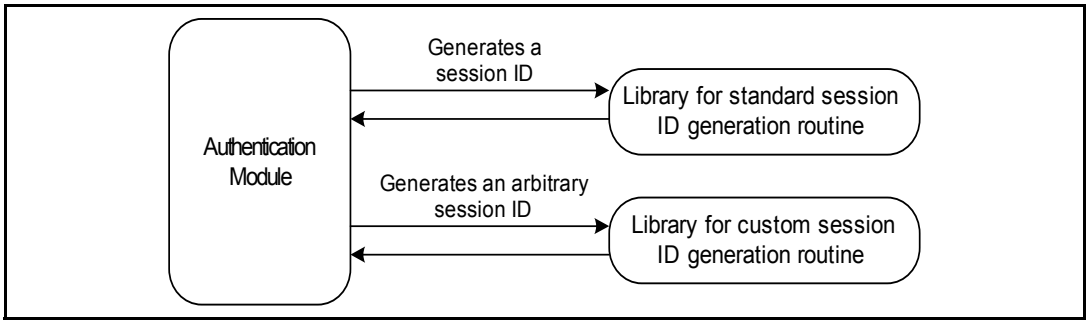
1.1 Version designations in the text

The table below gives the meanings of the version designations added to the text.

Designation	Meaning
10.0	An item added with the version enclosed in the square. In this case, the designation indicates the item was added with 10.0.
10.0	An item where the specification was changed or function added with the version enclosed in the oval. In this case, the designation indicates a specification change or added function with 10.0.

2 What is a Session ID Generation Routine?

The session ID generation routine is to enable the Authentication Module to generate arbitrary session IDs upon authentication, by making that part contained in a library.



The session ID generation routine is provided as a shared library. To change the session ID generation algorithm, you need to create a new shared library according to the development procedure.

The session ID generation routine can execute the following process.

• Generates arbitrary session ID.

You can give a meaning to each session ID or generate a session ID which is also effective in another security product or Web application.

3 Development Procedure

The following describes a general procedure for developing a session ID generation routine.

3.1 Determining Specification

Determine the specification according to “5 Remarks”.

3.2 Coding and Building

Based on the determined specification, code the session ID generation routine, and then build it into the shared library.

Use the skeleton source included in the development kit on coding.
You should build the shared library with the Makefile included in the development kit.

```
$ cd /opt/icewall-sso/developkit/certd/SidExit
$ make -f Makefile
```

3.3 Testing and Debugging

Test and debug the developed shared library using a test program and other tools. After the shared library has passed the test, integrate it into the Authentication Module and then perform an operation test.

Integrate the shared library into the Authentication Module in the following procedure.

(1) If the Authentication Module is running, stop it with the STOP command.

```
$ /opt/icewall-sso/certd/bin/end-cert
```

(2) Back up the session ID generation routine installed in standard configuration.

```
$ cd /opt/icewall-sso/lib
$ cp -p libCreateSidExit.sl libCreateSidExit.sl.bak
```

(3) Copy the created session ID generation routine to the target location.

```
$ cp -p /opt/icewall-sso/developkit/certd/SidExit/libCreateSid
Exit.sl /opt/icewall-sso/lib
```

(4) Start the Authentication Module with the START command.

```
$ /opt/icewall-sso/certd/bin/start-cert
```

4 Session ID Generation Routine Specifications

This section describes the specifications of the session ID generation routine.

- Interface function called by the Authentication Module
- Operation after generation of the session ID

IW_ExCreateSid

The following gives the specifications of the interface function.

Format

int IW_ExCreateSid(EX_CS_REQ* req, char* sessionid, char* userid, char* pass-word)

Argument

req: A request structure. Information of the Authentication Module is stored in this structure.

sessionid: A standard session ID (31 or 63 characters in length) generated by the Authentication Module. The session ID generated by this function is stored in this argument. When the authentication is successful, it becomes the session ID effective until the user logs out. Generates a session ID of **31 or 63 characters** in length. (10.0)

userid: The user ID entered upon logging in from the browser. This argument cannot be modified.

password: The password entered upon logging in from the browser. This argument cannot be modified.

Return value

The result of session ID generation is returned to the Authentication Module.
0 : Session ID generation was successful.
Other than 0: Session ID generation failed.

Restrictions

You can only modify sessionid among the arguments passed by this function. Operation is not guaranteed if you modify other arguments.

Generate the session ID under the following restrictions:

- Use a highly unique algorithm.
- When the session ID is 31 characters in length, you can only use 0-9 (0x30-0x39), A-Z (0x41-0x5a), and a-z (0x61-0x7a). When the session ID is 63 characters in length, you can only use 0-9 (0x30-0x39), A-Z (0x41-0x5a), a-z (0x61-0x7a), - (minus sign,0x2d), and _ (underscore, 0x5f). Operation is not guaranteed if you use other symbols and/or control codes. (10.0)

When the browser is notified of the session ID, one character for the Authentication Module identification key is added and the total length

becomes 32 or 64 characters. You can set the identification key to add with the CERTUNIQUEKEY parameter in the Authentication Module configuration file (cert.conf). In addition, the length (in bytes) of the generated session ID must match the setting of the SESSIONIDLEN parameter in the Authentication Module configuration file (cert.conf). 10.0

Remarks

If the generated session ID is not unique in the system, the Authentication Module calls this function again. In this case, the sessionid argument passes a newly generated standard session ID.

You can check the length of the session ID generated by the Authentication Module with strlen (*sessionid).

4.1 Behavior after session ID generation

The following describes the behavior of the Authentication Module, categorized by the execution result of the session ID generation routine.

(1) When the interface function returns 0
The Authentication Module continues the authentication process using the generated session ID.

(2) When the interface function returns other than 0
The following message appears in the error log of the Authentication Module and the browser shows the system error page.

Fatal: IW_ExCreateSid Error. UserID=[*User ID*] Result=[Return value from interface function] [EC31002-20097]

(3) If the length of the session ID stored in the sessionid argument is neither 31 characters nor 63
The following message appears in the error log of the Authentication Module and the browser shows the system error page. (10.0)

Fatal: Bad SessionID. UserID=[*User* ID] IW_INFO=[*Generated session* ID] [EC31003-20098]

(4) If the new session ID is a duplicate of another session ID
The following message appears in the access log of the Authentication Module and the session ID generation routine is called again. If a unique session ID cannot be generated after trying session ID generation for the number of times set by the COOOKIERETRY parameter, the browser shows the system error page.

Fatal: Ununique Cookie. UserID=[*User* ID] IW_INFO=[*Generated session* ID] [AC31004-25111]

(5) When the generated session ID does not match up to the byte length of the session ID generated by the Authentication Module
When the session ID generated by the session ID generation routine does not match up to the byte length of the session ID generated by the Authentication Module, the Authentication Module logs the following message into the error log and the browser shows the session ID generation error page.

Fatal: Bad SessionID. UserID=[*User* ID] IW_INFO=[*Generated session* ID] [EC31003-20098]

5 Remarks

Pay attention to the following when developing and integrating the session ID generation routine.

5.1 Area of possible negative impact with the session ID generation routine

Because the procedure executed within the session ID generation routine is closely related to the internal processing of the Authentication Module, when a system error occurs within the session ID generation routine, the Authentication Module may abort. Be sure to thoroughly test the developed library before integrating it into the Authentication Module.

Do not call exit() from the session ID generation routine because doing so terminates the running process of the Authentication Module.

5.2 Impact on Performance

Appending procedures to the session ID generation routine can cause negative impact to the performance to execute additional procedures. Determine the session ID generation logic and implement the code taking the influence on performance into consideration.

5.3 Memory Leak

Be sure to release the memory area allocated inside the session ID generation routine independently from the Authentication Module. Failure to do so may cause memory leak.

5.4 In order to generate a 63-byte session ID 10.0

In order to generate a 63 byte-length session ID, the request to the Authentication Module has to be based on ICP 2.0 and the SESSIONIDLEN parameter should be set to “64” in the Authentication Module configuration file (cert.conf).

5.5 Support

Support for the product is available only when the standard session ID generation routine is used. Support is not available if a custom session ID generation routine is used. Be sure to back up the standard library because it is necessary when you request support.

6 Restrictions

Note the following restrictions when developing the session ID generation routine. Without satisfying all of restrictions below on creating the session ID generation routine, the Authentication Module might be unable to start or show unstable behavior.

6.1 Standard Library Version to Link

The Authentication Module is designed to operate using libraries dynamically. However, if an archive library is linked to the session ID generation routine, the library linked to the session ID generation routine is used and a mismatch due to different library versions may occur. (This may cause a problem that cannot be resolved even when patches are applied to the OS.)

Note: In particular, you should be very careful about the state of the linked library, especially when your session ID generation routine uses a library such as Oracle, and others, that can be configured with different libraries depending on the installation enviromnent.

6.2 Compatibility

When using the session ID generation routine with a version different from the version with which the routine was created, the routine must be rebuilt. Therefore, when you have upgraded IceWall SSO, be sure to rebuild the session ID generation routine.

6.3 Threads

The Authentication Module uses threads for the internal processing and the session ID generation routine is called from threads. Therefore, be sure to use the thread-safe versions of the standard library functions. Also, when creating a user function, be sure to create it as a thread-safe version.

6.4 Linux

Be sure to add “-m64” to the compile option.
You can use “Linux Threads” and “Native POSIX Library Thread” for the thread for the Authentication Module.

7 Sample Source Code

The following gives a sample code of the session ID generation routine.

7.1 When the user ID is used as the session ID

- Use the first 31 characters of the user ID as the session ID.
- If the user ID is shorter than 31 characters, pad the trailing blank spaces with “0”s.

```
#include <string.h>

#include "iwcreatesid.h"

int IW_ExCreateSid ( req, sessionid, userid, password )
EX_CS_REQ *req;
char      *sessionid;
char      *userid;
char      *password;
{

    int length;

    strncpy( sessionid, userid, 31 );
    length = strlen( userid );

    if ( length < 31 ) {
        memset ( ( sessionid + length ), 0x30, ( 31- length ) );
    }

    return ( 0 );

}
```

8 Reference

The following shows the contents of the standard session ID generation routine development kit installed by default.
The development kit consists of the following directories and files.

Directories and files					Description
/opt/icewall-sso	/developkit	/certd	/SidExit	/iwcreatesid.c	Session ID generation routine development kit
				/iwcreatesid.h	
				/Makefile	

Note that the library created with this development kit has the same file name of the standard library installed by default. Be sure to back up the standard library before creating and integrating a new library.

8.1 Session ID Generation Routine Development Kit

The following shows the contents of each file of the standard session ID generation routine development kit installed by default.

8.1.1 Skeleton source (iwcreatesid.c)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "iwcreatesid.h"

/*------------------------------------------------------------------------*/
int IW_ExCreateSid ( req, sessionid, userid, password )
EX_CS_REQ *req;
char      *sessionid;
char      *userid;
char      *password;
{
   return( 0 );
}
```

8.1.2 Header file (iwcreatesid.h)

```
/*---------------------------------------------------------------------------*/
/* Create Sid Exit                                                           */
/*---------------------------------------------------------------------------*/
#ifndef IWCREATESID_H
#define IWCREATESID_H

/*---------------------------------------------------------------------------*/
/* API Structure                                                             */
/*---------------------------------------------------------------------------*/
typedef struct ex_cs_req EX_CS_REQ;

#endif /* #ifndef IWCREATESID_H */
```

8.1.3 Makefile (HP-UX: Makefile)

```
CC          = cc

CCFLAGS     = -D_UNIX -D_HPUX_SOURCE -D_POSIX_C_SOURCE=199506L -
Aa +e -D_FILE_OFFSET_BITS=64 -z +DD64 +z

LIBS        =

INCLUDE     = -I./

MAKEFILE    = Makefile

OBJS        = iwcreatesid.o

PROGRAM     = CreateSidExit

SRCS        = iwcreatesid.c

all:                    $(PROGRAM)

$(PROGRAM):             $(OBJS)
                        ld -b -z -o lib$(PROGRAM).sl $(OBJS) $(LIBS)
.c.o:
                        $(CC) $(CCFLAGS) $(INCLUDE) -c $(SRCS)
clean:
                        rm -f $(OBJS) lib$(PROGRAM).sl core
```

8.1.4 Makefile (Linux: Makefile)

```
CC          = gcc

CCFLAGS     = -m64 -fPIC -DLinux -D_FILE_OFFSET_BITS=64 -
D_LARGEFILE_SOURCE -D_GNU_SOURCE

LDFLAGS     = -m64 -fPIC

LIBS        =

INCLUDE     = -I./

MAKEFILE    = Makefile

OBJS        = iwcreatesid.o

PROGRAM     = CreateSidExit

SRCS        = iwcreatesid.c

all:                    $(PROGRAM)

$(PROGRAM):             $(OBJS)
                        gcc -shared $(LDFLAGS) -o lib$(PROGRAM).sl $(OBJS)
$(LIBS)
.c.o:
                        $(CC) $(CCFLAGS) $(INCLUDE) -c $(SRCS)
clean:
                        rm -f $(OBJS) lib$(PROGRAM).sl core
```