IceWall SSO

Version 10.0

Authentication DB Encryption Library Developer's Manual

August 2010

Printed in Japan

HP Part No. B1544-97018

Rev.111006A

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 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.

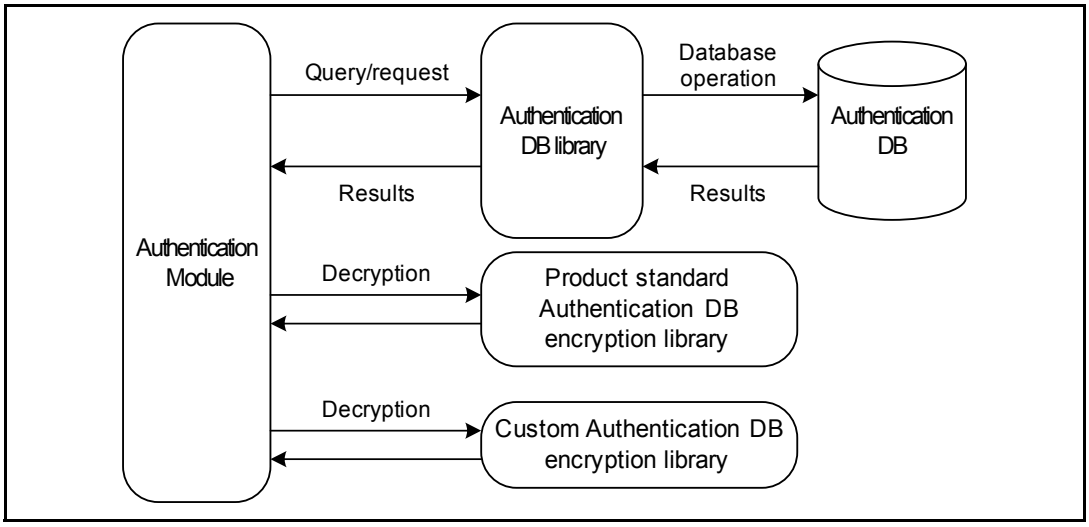
– Table of Contents –

1 Introduction

This manual provides information required to develop the Authentication DB encryption library integrated into the Authentication Module.

2 What is the Authentication DB Encryption Library?

The Authentication DB encryption library decrypts encrypted data which the Authentication Module obtained from the Authentication DB. (Note that it does not encrypt data.)



Among the additional columns set by the DBEXATTR parameter in the Authentication Module configuration file (cert.conf), the Authentication DB columns to be decrypted are those set by the DBCRYPTOATTR parameter.
Setting the IceWall SSO standard columns to the DBCRYPTOATTR parameter is not supported.

2.1 Product Standard Authentication DB Encryption Library

The library decrypts data encrypted by the encryption algorithm standard to the product.

If the DBCRYPTOTYPE parameter in the Authentication Module configuration file (cert.conf) is set to 1 or 3 and data obtained from the Authentication DB is prefixed with {IWCRYPTO128} or {IWCRYPTO256}, the data is decrypted assuming that it is encrypted by the encryption algorithm standard to the product.

2.2 Custom Authentication DB Encryption Library

The library decrypts data encrypted by a custom encryption algorithm.

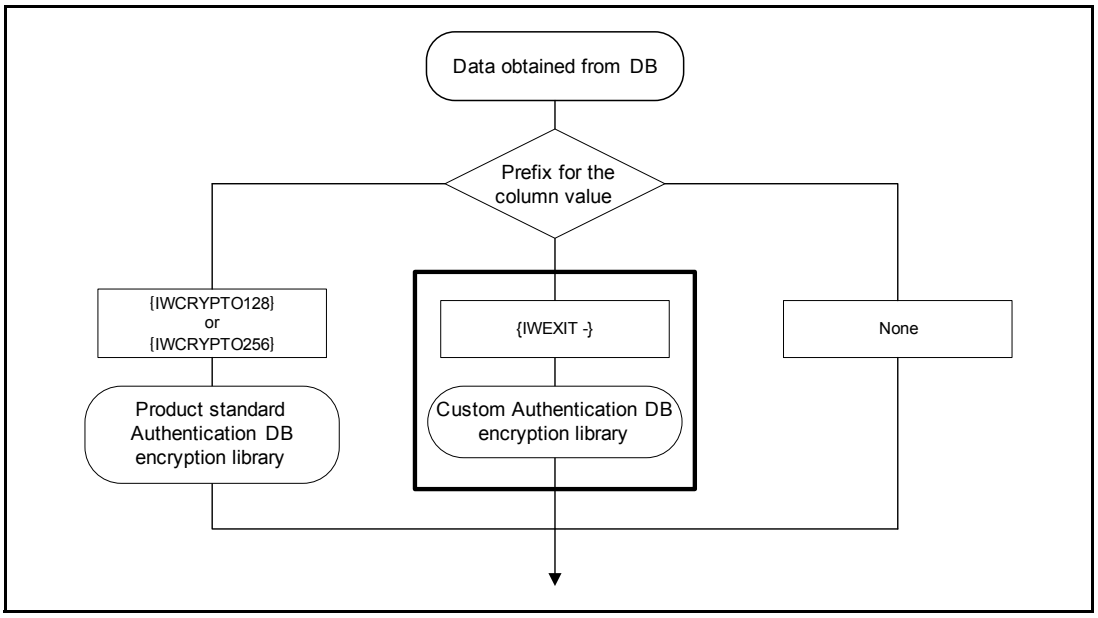
If the DBCRYPTOTYPE parameter in the Authentication Module configuration file (cert.conf) is set to 2 or 3 and data obtained from the Authentication DB is prefixed with {IWEXIT -}, the data is decrypted assuming that it is encrypted by a custom encryption algorithm.

3 Development Procedure

The following describes the general procedure for developing a custom Authentication DB encryption library.

3.1 Determining the Specification

Determine the encryption algorithm used for the Authentication DB encryption library.



Authentication DB Encryption Library Diagram

Although the Authentication DB encryption library only decrypts data, the data should be prefixed with {IWEXIT -}, i.e. prefixed with a string beginning with "{" and ending with "}", and the string enclosed between "{" and "}" starting with "IWEXIT", upon saving the encrypted data in the Authentication DB. If the prefix {IWEXIT -} is missing, the data is treated as unencrypted data. Note that the prefix {IWEXIT -} should not be greater than 64 bytes in length including "{ }."

3.2 Coding and Building

According to the determined specification, write the source code of the Authentication DB encryption library, and then build it into the shared library.

On coding, use the skeleton source code in the development kit. Build the library with the source code and the Makefile in the development kit.

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

3.3 Testing and Debugging

Test and debug the developed shared library using test programs and other tools. After the shared library has passed tests, integrate it into the Authentication Module and then perform functional tests.

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 Authentication DB encryption library installed by default.

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

(3) Copy the developed custom Authentication DB encryption library to the target location.

```
$ cp -p /opt/icewall-sso/developkit/certd/DBCryptoExit/libiwDB
CryptoExit.sl /opt/icewall-sso/lib
```

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

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

4 Custom Authentication DB Encryption Library Specification

This section describes the specification of the custom Authentication DB encryption library.

The custom Authentication DB encryption library defines the following:

- Interface function called by the Authentication Module
- Return values from the interface function (Normal and Abnormal only)

The following pages describe the specifications in detail.

CS_ExDBDecryptColumn

The following lists the specifications of the decryption interface function for the custom encryption algorithm.

Format

int CS_ExDBDecryptColumn(char* indata, int inlen, char outdata, int* outlen)**

Function

This function implements a process to decrypt the value of a column prefixed with {IWEXIT -}.
It receives the address and length of the encrypted data from indata and inlen, respectively, and then stores the address and length of the decrypted data in outdata and outlen, respectively.

Argument

indata : Address of the buffer where the data with the prefix to be decrypted is stored

inlen : Length of the data passed in indata

outdata : Address of the buffer where the decrypted data is stored
(Reserving the area for the outdata is performed within the Authentication DB encryption library, and releasing it is performed in the Authentication Module after obtaining the decrypted data.)

outlen : Length of the data stored in outdata

Return value

EXCS_RESULT_OK: Normal termination
EXCS_RESULT_NG“ Abnormal termination
When the library has returned EXCS_RESULT_NG, the Forwarder returns the system error page.

Restrictions

None

5 Considerations

Consider the following when developing and integrating the Authentication DB encryption library.

5.1 Area of Possible Negative Impact with the Authentication DB Encryption Library

Be sure to thoroughly test the developed library before integrating it into the Authentication Module. Do not call exit() from the Authentication DB encryption library; otherwise, it would result in terminating the running process of the module.

5.2 Performance Impact

Appending procedures to the Authentication DB encryption library can cause negative impact on the performance to execute additional procedures. Determine the decryption 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 Authentication DB encryption library independently from the Authentication Module. Failure to do so may cause memory leak.

5.4 Product Support Service

Product support service is available only when the product standard Authentication DB encryption library is used. Product support service is not available if a custom Authentication DB encryption library is used. Be sure to back up the product standard Authentication DB encryption library because it is required to receive product support service.

6 Restrictions

Note the following restrictions when developing the Authentication DB encryption library.
If all of the restrictions below are not satisfied during development, each process may not start or may show unstable behavior.

6.1 Restrictions Common to All OSes

Build the code as 64-bit binary.
The Authentication Module uses threads for the internal processing and the Authentication DB encryption library 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.

6.2 HP-UX 11i v3 (Itanium) Version

Be sure to add “+DD64” to the compile option.
Build the code on the same architecture as the processor on which the target Authentication Module to incorporate the DB encryption library runs on. Link 64-bit libraries only.

6.3 Linux Version

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 Development Tips

This section introduces a tip for developing the Authentication DB encryption library.

7.1 Implementing Multiple Encryption Algorithms

When implementing multiple encryption algorithms, the interface function can determine which algorithm to use by defining multiple prefixes {IWEXIT -} for each of the custom Authentication DB encryption algorithms.

8 Sample Source Code

The following gives a sample code of the Authentication DB encryption library.

8.1 Use prefix ({IWEXIT -}) to branch to one of the multiple decryption processes.

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

#include "csexdbcrypto.h"

extern int myDBDecrypt001(char *indata, int inlen, char **outdata, int *outlen);   /* *1 */
extern int myDBDecrypt002(char *indata, int inlen, char **outdata, int *outlen);   /*  *1 */
extern int myDBDecrypt003(char *indata, int inlen, char **outdata, int *outlen);   /*  *1 */

int CS_ExDBDecryptColumn(char *indata, int inlen, char **outdata, int *outlen)
{
   if (indata == NULL || inlen < 1 || outdata == NULL || outlen == NULL) {
      return EXCS_RESULT_NG;
   }

   /* Perform appropriate decryption procedure depending on prefix {IWEXITxxxx}. */
   if (strncasecmp("{IWEXITCRYPTO001}", indata, sizeof("{IWEXITCRYPTO001}") - 1) == 0) {
      /* Procedure of decryption method IWEXITCRYPTO001 */
      myDBDecrypt001(indata, inlen, outdata, outlen);
   } else if (strncasecmp("{IWEXITCRYPTO002}", indata, sizeof("{IWEXITCRYPTO002}") - 1) == 0)
{
      /* Procedure of decryption method IWEXITCRYPTO002 */
      myDBDecrypt002(indata, inlen, outdata, outlen);
   } else if (strncasecmp("{IWEXITCRYPTO003}", indata, sizeof("{IWEXITCRYPTO003}") - 1) == 0)
{
      /* Procedure of decryption method IWEXITCRYPTO003 */
      myDBDecrypt003(indata, inlen, outdata, outlen);
   } else {
      /* Unsupported encryption method.  Do not convert passed data and pass it to authentication
module. */
      *outdata = (char*)malloc(inlen+1);
      if (*outdata == NULL) {
         return EXCS_RESULT_NG;
      }

      memcpy(*outdata, indata, inlen);
      *(*outdata+inlen) = '\0';
      *outlen = inlen;
   }

   return EXCS_RESULT_OK;
}
```

*1 The functions myDBDecrypt001(), myDBDecrypt002(), and myDBDecrypt003() are not provided as APIs. They are just example functions to decrypt encrypted column values.

9 Reference

The following shows the contents of the product standard Authentication DB encryption library development kit installed by default.
The development kit consists of the following directories and files.

Directories and files					Description
/opt/icewall-sso	/developkit	/certd	/DBCryptoExit	/csexdbcrypto.c	Authentication DB Encryption Library Development Kit
				/csexdbcrypto.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.

9.1 Authentication DB Encryption Library Development Kit

9.1.1 Skeleton Source (csexdbcrypto.c)

The following source code returns a value obtained from the Authentication DB to the Authentication Module without conversion.

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

#include "csexdbcrypto.h"

int CS_ExDBDecryptColumn(char *indata, int inlen, char **outdata, int *outlen)
{
    if (indata == NULL || inlen < 1 || outdata == NULL || outlen == NULL) {
        return EXCS_RESULT_NG;
    }

    *outdata = (char*)malloc(inlen+1);
    if (*outdata == NULL) {
        return EXCS_RESULT_NG;
    }

    memcpy(*outdata, indata, inlen);
    *(*outdata+inlen) = '\0';
    *outlen = inlen;

    return EXCS_RESULT_OK;
}
```

9.1.2 Header file (csexdbcrypto.h)

```
#ifndef CSEX_DBCRYPTO_H
#define CSEX_DBCRYPTO_H

#define EXCS_RESULT_OK        0
#define EXCS_RESULT_NG        -1

#endif  /* #ifndef CSEX_DBCRYPTO_H */
```

9.1.3 Makefile (HP-UX Itanium: 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        = csexdbcrypto.o

PROGRAM     = iwDBCryptoExit

SRCS        = csexdbcrypto.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
```

9.1.4 Makefile (64-bit 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        = csexdbcrypto.o

PROGRAM     = iwDBCryptoExit

SRCS        = csexdbcrypto.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
```