BaroPAM integration API(C-en)

Nurit Co., Ltd.

Lee Jongil

January 20, 2024

1. Integration API configuration

BaroPAM related shared object (libbarokey-6.x.x.so) is used to verify field or data encryption and decryption and one-time authentication keys.

API class	Description	Dirctory
barokey.h	BaroPAM Header file	/usr/baropam/key
libbarokey-x.x.x.so	BaroPAM Module	/usr/baropam/key

The symmetric key (64 bytes) used for field or data encryption/decryption is fixed inside the program, and in order to use the shared object (libbarokey-x.x.x.so), the directory where the shared object file exists (/usr/baropam/key) must be It must be set in the library path.

```
For Linux  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/baropam/key
For HP-UX  export SHLIB_PATH=$SHLIB_PATH:/usr/baropam/key
For AIX    export LIBPATH=$LIBPATH:/usr/baropam/key
```

The header file **barokey.h** for BaroPAM is as follows.

```
#ifndef BAROKEY_INCLUDED
#define BAROKEY_INCLUDED

#include <stdlib.h>

#ifdef __cplusplus
extern "C" {
#endif

char *baro_encrypts(const char *data);
char *baro_decrypts(const char *data);

long  get_timestamps(void);
long  getRemainingTime(const char *cycle_time);
char *getSecureKeyCreate(void);
```

```
char *generateKEYL(const char *login_id, const char *phone_no, const char *cycle_time, const char *key_method);
char *generateKEYP(const char *secure_key, const char *cycle_time, const char *key_method);
long  verifyKEY(const char *login_id, const char *phone_no, const char *cycle_time, const char *key_method, const char *tkey);
long  verifyKEYL(const char *login_id, const char *phone_no, const char *cycle_time, const char *key_method, const char *tkey);
long  verifyKEYP(const char *secure_key, const char *cycle_time, const char *key_method, const char *tkey);

#ifdef __cplusplus
}
#endif

#endif
```

2. Integration API function

1) Encryption and decryption function

① baro_encrypts function

- NAME

  baro_encrypts

- SYNOPSIS

  char *baro_encrypts(const char * data)

- DESCRIPTION

  Function to encrypt data.

  data: Data to be encrypted

- RETURN VALUES

  Return encrypted data.

② baro_decrypts function

- NAME

  baro_decrypts

- SYNOPSIS

  char *baro_decrypts(const char * data)

- DESCRIPTION

  Function to decrypt data.

  data: Data to be decrypted

- RETURN VALUES

  Return decrypted data.

③ Data encryption/decryption usage example

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

#include "/usr/baropam/key/barokey.h"

int main(int argc, char *argv[]) {
      const char *source_data = argv[1];
      const char *encrypt_data;
      const char *decrypt_data;

      encrypt_data = baro_encrypts(source_data );
      decrypt_data = baro_decrypts(encrypt_data);

      printf("Source   data = [%s]₩n", source_data );
      printf("encrypt data = [%s]₩n", encrypt_data);
      printf("decrypt data = [%s]₩n", decrypt_data);

      return 0;
}
```

2) One-time authentication key verification function

① verifyKEYL function(When using login-ID)

- NAME

  verifyKEYL

- SYNOPSIS

  long verifyKEYL(const char *login_id, const char *phone_no, const char *cycle_time, const char *key_method, const char *auth_key)

- DESCRIPTION

  A function that verifies whether the entered one-time authentication key is correct.

  login_id: Set the ID entered in the login-ID field of the login screen.
  phone_no: Login-ID set user's smart phone number only with numbers.
  cycle_time: Set the generation cycle (3~60 seconds) of one-time authentication key specified for each user.
  key_method: Set the one-time authentication key verification method (app1, app256, app384, app512: app).
  auth_key: Set the one-time authentication key created and entered in the BaroPAM app on the login screen.

  If the generation period of the smart phone number for each user and the one-time authentication key designated for each individual is different from the generator of the one-time authentication key, verification may fail because the one-time authentication key is different. You must match the information.

- RETURN VALUES

  On success, 1 is returned, and on failure, 0 is returned.

② verifyKEYP function(When using secure key)

- NAME

  verifyKEYP

- SYNOPSIS

  long verifyKEYP(const char *secure_key, const char *cycle_time, const char *key_method, const char *auth_key)

- DESCRIPTION

  A function that verifies whether the entered one-time authentication key is correct.

Secure_key: Set the secure key given for each user.

cycle_time: Set the generation cycle (3~60 seconds) of one-time authentication key specified for each user.

key_method: Set the one-time authentication key verification method (app1, app256, app384, app512: app).

auth_key: Set the one-time authentication key created and entered in the BaroPAM app on the login screen.

If the generation cycle of the secure key and one-time authentication key specified for each user, verification may fail because the one-time authentication key is different. You must match the information.

- RETURN VALUES

On success, 1 is returned, and on failure, 0 is returned.

3. Authentication key verification part

1) Example of using authentication key verification module (Linux environment)

Sample program) verifyKEYL function (when using login-ID)

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

#include "/usr/baropam/key/barokey.h"

int main(int argc, char *argv[]) {
    const char *login_id   = argv[1];    // Login-ID
    const char *phone_no   = argv[2];    // Phone number
    const char *cycle_time = argv[3];    // Cycle time
    const char *key_method = argv[4];    // Key method
          char *auth_key   = argv[5];    // Auth key

    // Authentication key verification
    long bauth_key = verifyKEYL(login_id, phone_no, cycle_time, key_method, auth_key);

    // Authentication key verification (success)
    if (bauth_key == 1) {
        printf("Auth key success.Wn");
    // Authentication key verification (failure)
    } else {
```

```
        printf("Auth key faild.Wn");
    }
}
```

Sample program) verifyKEYP function (when using secure key)

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

#include "/usr/baropam/key/barokey.h"

int main(int argc, char *argv[]) {
    const char *secure_key = argv[1];    // Secure key
    const char *cycle_time = argv[2];    // Cycle time
    const char *key_method = argv[3];    // Key method
          char *auth_key   = argv[4];    // Auth key

    // Authentication key verification
    long bauth_key = verifyKEYP(secure_key, cycle_time, key_method, auth_key);

    // Authentication key verification (success)
    if (bauth_key == 1) {
        printf("Auth key success.Wn");
    // Authentication key verification (failure)
    } else {
        printf("Auth key faild.Wn");
    }
}
```

Compilation)

```
$ gcc -o barokeys barokeys.c -L/usr/baropam/key - lbarokey-x.x.x
```

Execution)

```
$ ./barokeys mc529@hanmail.net 01027714076 20 0 app512 490661
Auth key success.
```

Note) If the following error occurs during execution

```
./barokeys: error while loading shared libraries: libbarokey.so: cannot open shared object
file: No such file or directory
```

In this case, it occurs because the shared object file does not exist or the directory where the shared object file exists is not set in the Library path. Therefore, check the existence of the shared object file and the directory where the shared object file exists

(/usr/baropam/key) Should be set in the Library path.

```
For Linux  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/baropam/key
For HP-UX  export SHLIB_PATH=$SHLIB_PATH:/usr/baropam/key
For AIX    export LIBPATH=$LIBPATH:/usr/baropam/key
```

2) Example of using authentication key verification module (Tuxedo environment)

Example) verifyKEYL function (when using login-ID)

```
#include <stdio.h>
#include <ctype.h>
#include <atmi.h>       /* TUXEDO Header File */
#include <userlog.h>    /* TUXEDO Header File */

#include "/usr/baropam/key/barokey.h"

TOUPPER(TPSVCINFO *rqst) {
    const char *login_id   = "mc529@hanmail.net";
    const char *phone_no   = "01027714076";
    const char *cycle_time = "20";
    const char *key_method = "app512";
          char *auth_key   = rqst->data;

    long bauth_key = verifyKEYL(login_id, phone_no, cycle_time, key_method, auth_key);

    sprintf(rqst->data, "%d", bauth_key);
    /* Return the transformed buffer to the requestor. */
    tpreturn(TPSUCCESS, 0, rqst->data, 0L, 0);
}
```

Example) verifyKEYP function (when using secure key)

```
#include <stdio.h>
#include <ctype.h>
#include <atmi.h>       /* TUXEDO Header File */
#include <userlog.h>    /* TUXEDO Header File */

#include "/usr/baropam/key/barokey.h"

TOUPPER(TPSVCINFO *rqst) {
    const char *secure_key = "jlqlcHbVqdpj7b4PzBpM2DileBvmHFV/";
    const char *cycle_time = "20";
    const char *key_method = "app512";
          char *auth_key   = rqst->data;

    long bauth_key = verifyKEYP(secure_key, cycle_time, key_method, auth_key);

    sprintf(rqst->data, "%d", bauth_key);
    /* Return the transformed buffer to the requestor. */
```

```
    tpreturn(TPSUCCESS, 0, rqst->data, 0L, 0);
}
```

Compilation)

```
$ CFLAGS="-L/usr/baropam/key - lbarokey-x.x.x" buildserver -v -o simpserv -f simpserv.c -s TOUPPER
```

Note) When verifying the one-time authentication key in Tuxedo service, the one-time verification key module "-L/usr/baropam/key -lbarokey-x.x.x" must be set in the CFLAGS option when compiling.

If the following error occurs when starting the Tuxedo server process

```
error while loading shared libraries: libbarokey-x.x.x.so: cannot open shared object file: No such file or directory
```

In this case, it occurs because the shared object file does not exist or the directory where the shared object file exists is not set in the Library path. Therefore, check the existence of the shared object file and the directory where the shared object file exists (/usr/baropam/key) Should be set in the Library path.

```
For Linux  export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/baropam/key
For HP-UX  export SHLIB_PATH=$SHLIB_PATH:/usr/baropam/key
For AIX    export LIBPATH=$LIBPATH:/usr/baropam/key
```