BaroPAM 가이드(C)

목차

1. BaroPAM 연동 API

1.1 연동 API 구성

BaroPAM 관련 Shared object(libbarokey-x.x.x.so)는 필드 또는 데이터 암복호화 및 **일회용 인증키**를 검증하는데 사용된다.

API구분	설명	위치
barokey.h	BaroPAM Header 파일	/usr/baropam/key
libbarokey-x.x.x.so	BaroPAM 모듈	/usr/baropam/key

필드 또는 데이터 암복호화에 사용되는 대칭 키(64byte)를 프로그램 내부에 고정되어 있으며, Shared object(libbarokey-x.x.x.so)를 사용하기 위해서는 반드시 Shared object file이 존재하는 디렉토리(/usr/baropam/key)를 Library path에 설정해야 한다.

Linux인 경우 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:**/usr/baropam/key**
HP-UX인 경우 export SHLIB_PATH=$SHLIB_PATH:**/usr/baropam/key**
AIX인 경우 export LIBPATH=$LIBPATH:**/usr/baropam/key**

BaroPAM에 대한 header 파일인 **barokey.h**은 다음과 같다.

```
#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
```

1.2 연동 API 함수

1) 암복호화 함수

① baro_encrypts 함수

- NAME
  baro_encrypts

- SYNOPSIS
  char *baro_encrypts(const char * data)

- DESCRIPTION
  데이터를 암호화하는 함수.

  data: 암호화할 데이터

- RETURN VALUES
  암호화한 데이터를 반환.

② baro_decrypts 함수

- NAME
  baro_decrypts

- SYNOPSIS
  char *baro_decrypts(const char * data)

- DESCRIPTION
  데이터를 복호화 하는 함수.

  data: 복호화할 데이터

- RETURN VALUES
  복호화한 데이터를 반환.

③ 데이터 암복호화 사용 예

```
#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]Wn", source_data );
    printf("encrypt data = [%s]Wn", encrypt_data);
    printf("decrypt data = [%s]Wn", decrypt_data);

    return 0;
}
```

2) 일회용 인증키 검증 함수

① verifyKEYL 함수(로그인-ID를 사용하는 경우)

- 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
  입력한 일회용 인증키가 맞는지 검증하는 함수

  login_id: 로그인-ID 항목에 입력한 ID를 설정.
  phone_no: 사용자별 스마트 폰 번호를 숫자만 설정.
  cycle_time: 사용자별로 지정한 일회용 인증키의 생성 주기(3~60초)를 설정.
  key_method: 일회용 인증키의 검증 방식(app1, app256, app384, app512: 앱)을 설정.
  auth_key: BaroPAM 앱에서 생성하여 입력한 일회용 인증키를 설정.

  만약, 사용자별로 스마트 폰 번호 및 개인별로 지정한 일회용 인증키의 생성 주기가 일회용 인증키의 생성기와 다른 경우 일회용 인증키가 달라서 검증에 실패할 수 있다. 반드시 정보를 일치시켜야 한다.

- RETURN VALUES
  성공 시에는 1을 반환하며, 실패 시는 0을 반환.

② verifyKEYP 함수(Secure key를 사용하는 경우)

- 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
  입력한 일회용 인증키가 맞는지 검증하는 함수

secure_key: 사용자 또는 카드별로 부여한 Secure key를 설정.
cycle_time: 사용자 또는 카드별로 지정한 **일회용 인증키**의 생성 주기(3~60초)를 설정
key_method: **일회용 인증키**의 검증 방식(app1, app256, app384, **app512**: 앱)을 설정.
auth_key: BaroPAM 앱에서 생성하여 입력한 **일회용 인증키**를 설정.

만약, 사용자별로 지정한 Secure key 및 **일회용 인증키**의 생성 주기가 **일회용 인증키**의 생성기와 다른 경우 **일회용 인증키**가 달라서 검증에 실패할 수 있다. 반드시 정보를 일치시켜야 한다.

- RETURN VALUES
성공 시에는 1을 반환하며, 실패 시는 0을 반환.

1.3 인증키 검증 부분

Sample program) verifyKEYL 함수(로그인-ID를 사용하는 경우)

```
#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];    // 로그인-ID
    const char *phone_no   = argv[2];    // 폰번호
    const char *cycle_time = argv[3];    // 생성주기
    const char *key_method = argv[4];    // 인증키 생성방식
          char *auth_key   = argv[5];    // 일회용 인증키

    // 일회용 인증키 검증
    long bauth_key = verifyKEYL(login_id, phone_no, cycle_time, key_method, auth_key);

    // 일회용 인증키 검증(성공)
    if (bauth_key == 1) {
        printf("Auth key success.Wn");
    // 일회용 인증키 검증(실패)
    } else {
        printf("Auth key faild.Wn");
    }
}
```

Sample program) verifyKEYP 함수(Secure key를 사용하는 경우)

```
#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];    // 생성주기
    const char *key_method = argv[3];    // 인증키 생성방식
          char *auth_key   = argv[4];    // 일회용 인증키

    // 일회용 인증키 검증
    long bauth_key = verifyKEYP(secure_key, cycle_time, key_method, auth_key);

    // 일회용 인증키 검증(성공)
    if (bauth_key == 1) {
        printf("Auth key success.₩n");
    // 일회용 인증키 검증(실패)
    } else {
        printf("Auth key faild.₩n");
    }
}
```

컴파일)

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

실행)

```
$ ./barokeys mc529@nurit.co.kr 01027714076 20 0 app512 490661
Auth key success.
```

참고) 실행시 다음과 같은 오류가 발생하는 경우

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

이런 경우 shared object file이 존재하지 않거나 shared object file가 존재하는 디렉토리가 Library path에 설정되어 있지 않아서 발생하므로 shared object file이 존재여부를 확인하고 shared object file가 존재하는 디렉토리(/usr/baropam/key)를 Library path에 설정해야 한다.

```
Linux인 경우 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/baropam/key
HP-UX인 경우 export SHLIB_PATH=$SHLIB_PATH:/usr/baropam/key
AIX인 경우   export LIBPATH=$LIBPATH:/usr/baropam/key
```

2) 일회용 인증키의 검증 모듈 사용 예(Tuxedo 환경)

예) verifyKEYL 함수(로그인-ID를 사용하는 경우)

```
#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);
}
```

예) verifyKEYP 함수(Secure key를 사용하는 경우)

```
#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);
}
```

컴파일)

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

참고) Tuxedo 서비스에서 **일회용 인증키**를 검증하는 경우, 컴파일 시 CFLAGS 옵션에 일회용 검증키 모듈을 "-L/usr/baropam/key -lbarokey-x.x.x" 설정해야 한다.

Tuxedo 서버 프로세스 기동시 다음과 같은 오류가 발생하는 경우

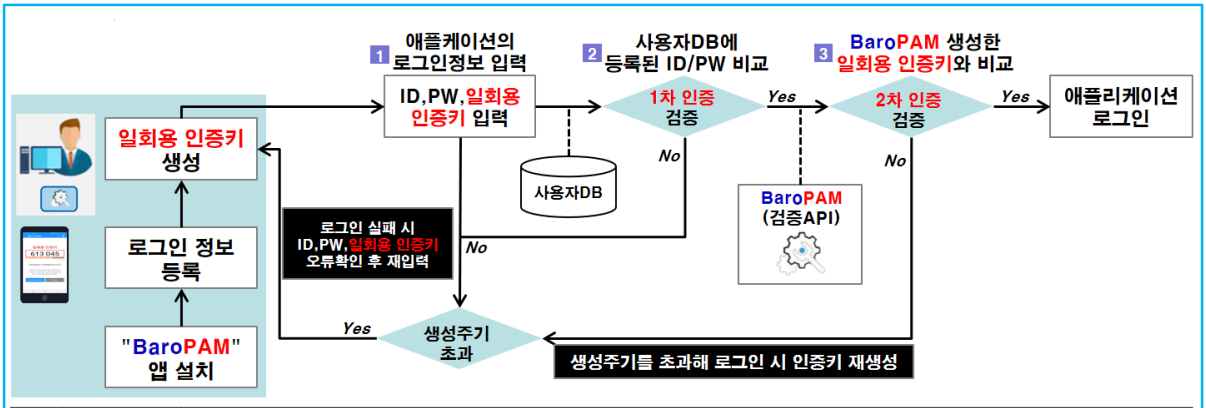
```
error while loading shared libraries: libbarokey-x.x.x.so: cannot open shared object file: No such file or directory
```

이런 경우 shared object file이 존재하지 않거나 shared object file가 존재하는 디렉토리가 Library path에 설정되어 있지 않아서 발생하므로 shared object file이 존재여부를 확인하고 shared object file가 존재하는 디렉토리(/usr/baropam/key)를 Library path에 설정해야 한다.

Linux인 경우 export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/baropam/key
HP-UX인 경우 export SHLIB_PATH=$SHLIB_PATH:/usr/baropam/key
AIX인 경우 export LIBPATH=$LIBPATH:/usr/baropam/key

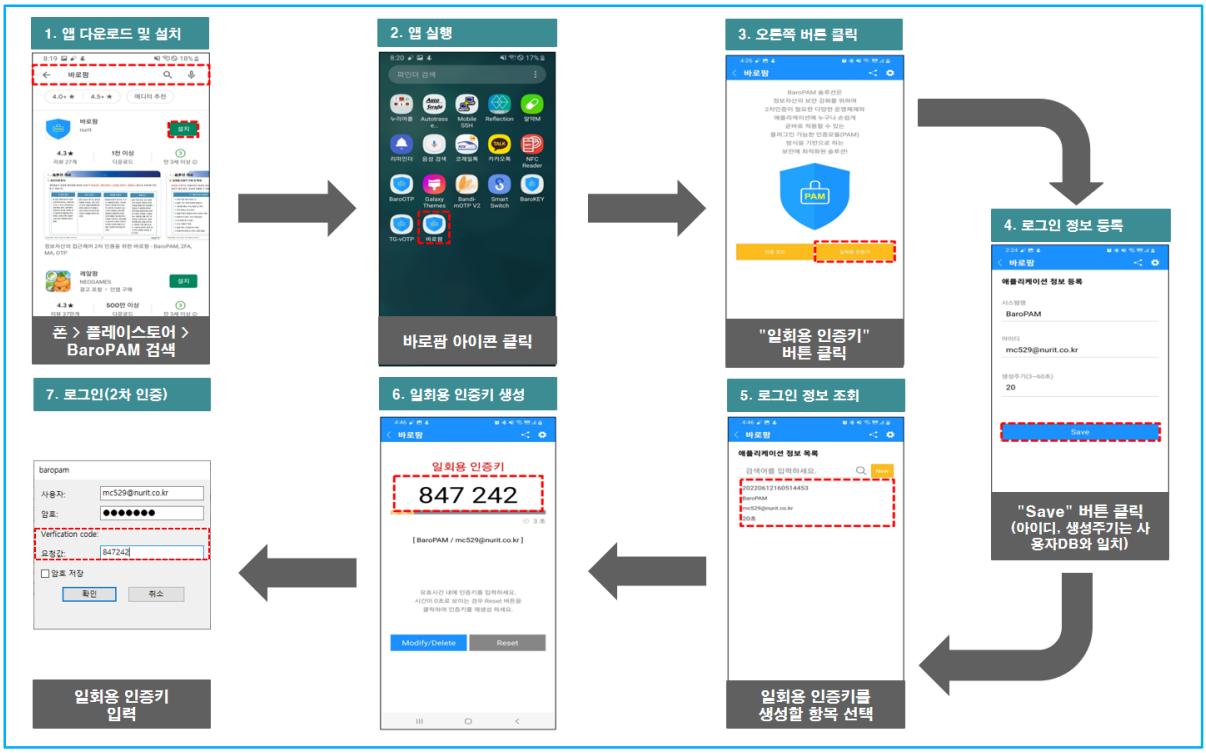
2. BaroPAM 적용

2.1 BaroPAM 적용 프로세스



No	구분	내용	설명
1	2차 인증 앱 (BaroPAM)	BaroPAM 앱 다운로드 및 설치	-안드로이드폰: 구글 플레이스토어 접속 및 "바로팜" 앱 다운로드 -아이폰: 애플 앱스토어 접속 및 "BaroPAMs" 앱 다운로드
2		BaroPAM 앱 실행	
3		"일회용 인증키" 버튼 선택	-실행된 앱 하단 오른쪽 버튼 선택
4		애플리케이션의 로그인 정보 등록	-시스템명: 애플리케이션 명칭(임의 등록 가능) -아이디: 애플리케이션의 로그인 -ID -생성주기: 3~60초 사이 입력
5		저장 후 목록 조회 시 나오는 항목 클릭, 일회용 인증키 생성	-아이폰 사용자는 본인 확인 절차를 거치지 않은 경우 본인 인증 절차를 진행
6		애플리케이션에 접속해 ID, PW, 일회용 인증키를 입력해 로그인	-아이디, 생성주기, 일회용 인증키를 이용해 2차 인증 검증

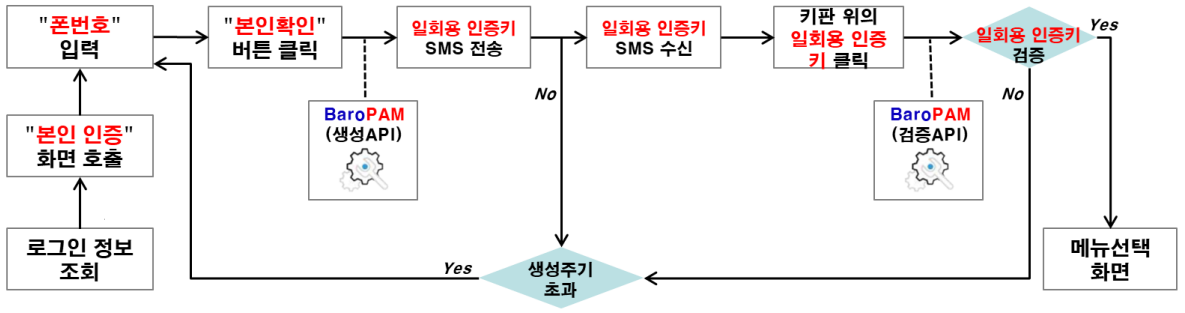
2.2 BaroPAM 적용 화면



2.3 본인확인 적용 프로세스

아이폰(iPhone)의 기기정보를 얻지 못해서 **2차 인증키(일회용 인증키)**를 생성하기 위해서 로그인 정보 항목을 선택했을 때 "**일회용 인증키**" 생성 화면으로 이동하지 않은 경우가 발생할 수 있다.

또한, 타인의 폰번호를 부정으로 사용하지 못하도록 하기 위해서 별도의 본인확인 기능을 적용할 필요가 있는데, "**BaroPAM**" 앱에서는 자체 알고리즘을 적용하여 자체적으로 본인확인 절차를 실행하고 있다.

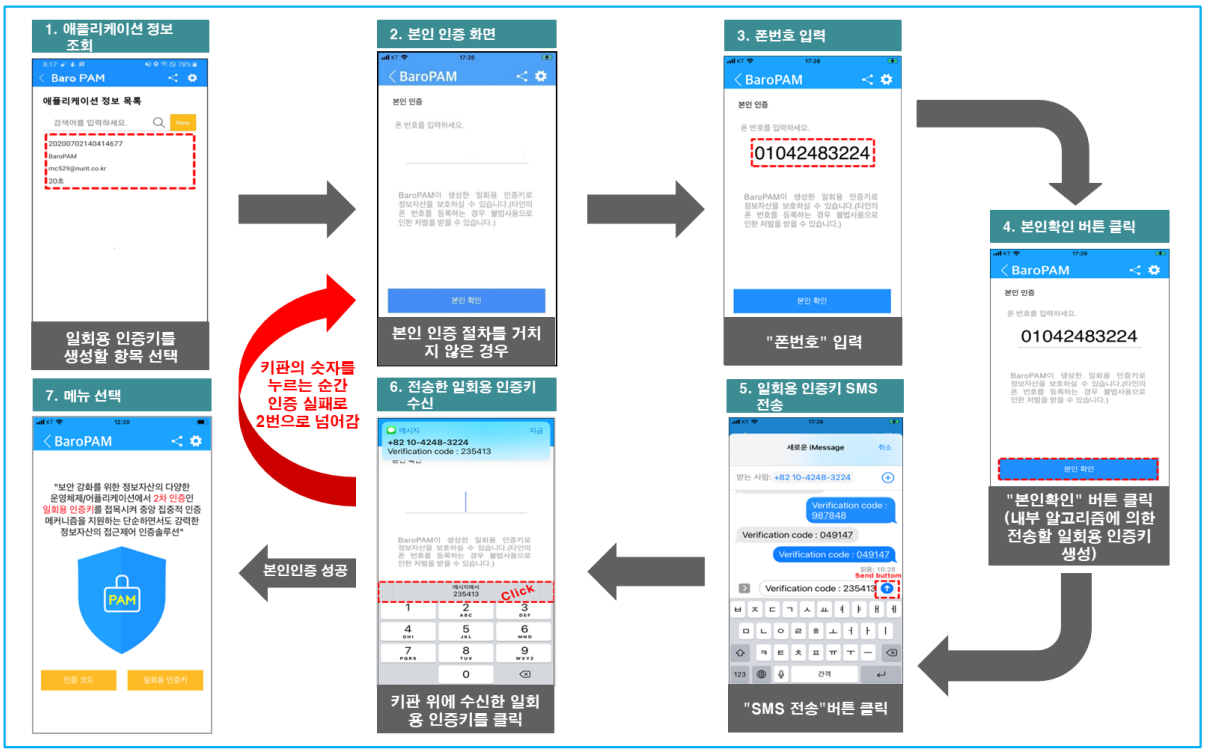


No	구분	내용	설명
1	아이폰 본인확인	로그인 정보 목록에서 일회용 인증키를 생성할 항목 클릭	-본인확인 절차를 거치지 않은 경우 본인인증 절차를 진행
2		"본인인증" 화면 호출	
3		"폰번호" 입력	-본인인증할 폰번호를 입력
4		"본인확인" 버튼 클릭	-내부 알고리즘에 의하여 "Verification code"인 일회용 인증키를 생성 -생성한 일회용 인증키를 전송하기 위한 SMS 연동 화면 호출
5		"Verification code" SMS 전송	-일회용 인증키를 확인한 후 "SMS 전송" 버튼을 클릭
6		"Verification code" SMS 수신	-본인 폰으로 전송한 일회용 인증키가 키판 위에 표시 -폰번호가 틀린 경우 타인 폰으로 일회용 인증키가 전송됨
7		키판 위의 "Verification code" 클릭	-내부 알고리즘에 의하여 일회용 인증키를 검증 -검증에 성공하면 메뉴선택 화면, 실패하면 본인인증 화면이 호출

2.4 본인확인 적용 화면

아이폰(iPhone)의 기기정보를 얻지 못해서 **2차 인증키(일회용 인증키)**를 생성하기 위해서 로그인 정보 항목을 선택했을 때 "**일회용 인증키**" 생성 화면으로 이동하지 않은 경우가 발생할 수 있다.

또한, 타인의 폰번호를 부정으로 사용하지 못하도록 하기 위해서 별도의 본인확인 기능을 적용할 필요가 있는데, "BaroPAM" 앱에서는 자체 알고리즘을 적용하여 자체적으로 본인확인 절차를 실행하고 있다.



참고) SMS로 전송한 OTA key가 수신은 되었는데 키판 위에 표시되지 않거나 SMS로 전송한 OTA key가 수신되지 않은 경우

아이폰의 "**암호 자동 완성 기능**"이 설정되지 않아서 발생한다. "BaroPAM" 앱을 설치한 후 iOS12 부터는 더욱 편리한 암호 자동 완성 기능을 반드시 설정해야 한다. (아이폰의 "**설정**" -> "**암호**" -> "**암호 자동 완성**" -> "**허용**")

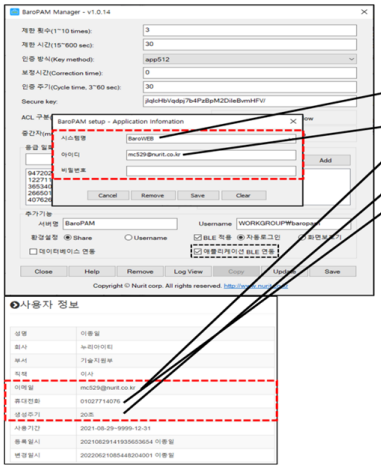
2.5 BaroPAM 앱 설치 및 정보 설정

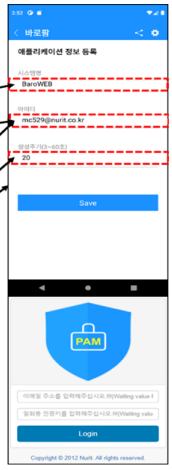
정보자산에 로그인 시 Verification code에 입력할 **일회용 인증키**의 생성기인 **BaroPAM** 앱의 다운로드 (https://play.google.com/store/apps/details?id=com.baro.pam)는 구글의 "**Play 스토어**"나 Apple의 "**App 스토어**"에서 가능하며, 설치는 일반 앱의 설치와 동일하다.

BaroPAM 앱 다운로드

Google Play

App Store





BaroPAM 앱은 Android 6.0 (Marshmalliw) API 23, iOS 13.0 이상에서 사용 가능하며, 가로보기 모드를 지원하지 않는다.

BaroPAM 앱을 설치한 후 BaroPAM 앱을 실행하여 메뉴 선택화면에서 "일회용 인증키" 버튼을 클릭하여 애플리케이션의 사용자 정보에 설정한 "인증주기, 아이디, 시스템명"을 BaroPAM 앱의 "애플리케이션 정보 등록" 화면에서 동일하게 입력해야 한다.

현상 : 안드로이폰 또는 아이폰의 날짜와 시간이 현재 시간과 차이가 발생하여 "일회용 인증키"가 맞지 않은 경우
원인 : 안드로이폰 또는 아이폰의 날짜와 시간을 네트워크에서 제공하는 시간을 사용하지 않아서 발생.
조치 : 안드로이폰인 경우는 폰의 "설정" -> "일반" -> "날짜 및 시간" -> "날짜 및 자동 설정"과 "시간대 자동 설정" -> "허용"
아이폰인 경우는 폰의 "설정" -> "날짜 및 시간" -> "자동으로 설정" -> "허용"

3. About BaroPAM

Version 1.0 - Official Release - 2016.12.1

http://www.nurit.co.kr

제 조 사 : 주식회사 누리아이티
등록번호 : 258-87-00901
대표이사 : 이종일
대표전화 : 02-2665-0119(영업문의/기술지원)
이 메 일 : mc529@nurit.co.kr
주 소 : 서울시 강서구 마곡중앙2로 15, 913호(마곡동, 마곡테크노타워2)