

패스워드 안전성 검증 라이브러리 매뉴얼

2012. 12.

[목 차]

1. 개요	1
1.1. 배경 및 목적	1
2. 환경 설정	2
2.1. 라이브러리	2
2.1.1. 윈도우버전	2
2.1.1.1. dll ,dll_jndi, lib_test	2
2.1.2. 리눅스버전	4
2.1.2.1. lib	4
2.1.2.2. lib_jndi	9
2.1.2.3. lib_test	15
2.1.3. 유닉스버전	16
2.1.3.1. lib	16
2.1.3.2. lib_jndi	19
2.1.3.3. lib_test	21
3. 실행 방법	22
3.1. 개요	22
3.1.1. 목적	22
3.1.2. 범위	22
3.2. 실행 순서	23
3.2.1. LIB_001 : 환경설정 로드	23
3.2.2. LIB_002 : 패스워드 검증	27
3.2.3. LIB_003 : 개인정보 사전파일 생성	30
3.2.4. LIB_004 : 개인정보 사전파일 제거	32
3.2.5. LIB_005 : 사용자 사전파일 생성	33
3.2.6. LIB_006 : 사전 파일에서 데이터 추출	35
부록 1. 패스워드 검증 라이브러리 API 설명	38
부록 2. 에러코드 및 대처방안	79
부록 3. JDK 설치 방법	80

패스워드 안전성 검증 라이브러리 매뉴얼

1. 개요

1.1 배경 및 목적

최근 발생한 유명 포털사이트 해킹사고 등으로 인해 웹사이트 자체에 대한 보안과 사용자의 안전한 패스워드 관리에 대해 이슈화되고 있다. 특히, 패스워드의 경우, 가장 기본적인 사용자 인증 수단으로 적은 시스템 구축비용, 구현 및 사용자 편의성 등을 이유로 다른 인증 방식에 비해 웹사이트에서 가장 널리 사용되고 있어 패스워드 관리에 대한 중요성이 더욱 부각되고 있다.

한국인터넷진흥원이 '07년 20대 남녀 대학생 패스워드 이용현황을 조사한 결과에 따르면, 대부분이 영소문자 및 숫자의 2가지 문자 구성으로 된 8자리 이하의 패스워드를 사용하고 있었으며, 전체의 64.5%가 6자리 이하의 패스워드를 사용하는 등 현 시점에서의 컴퓨팅 파워를 고려했을 때 대부분이 취약한 패스워드를 사용하고 있는 것으로 나타났다. 또한, 은행, 포털 등 103개 주요 웹사이트에서의 패스워드 이용 현황에 대해 조사 결과에 따르면, 23%의 웹사이트에서 8자리 이하의 패스워드를 사용하고 있어 패스워드 정책상의 문제점을 드러내고 있었다.

국내 패스워드 사용현황의 문제점에 대한 대책을 제시하기 위해 한국인터넷진흥원에서는 '07년 정보통신부와 함께 안전한 패스워드 생성기준과 관리방법을 제시하는 『패스워드 선택 및 이용 가이드』를 발간하였으며, 가이드에 따른 입력된 패스워드의 안전성 강도를 확인할 수 있는 『패스워드 안전성 검증 소프트웨어』를 개발하여 보급 하고 있다.

본 문서는 한국인터넷진흥원에서 보급하는 『패스워드 안전성 검증 소프트웨어』에 대한 윈도우 및 리눅스 시스템에서 적용할 수 있게 소스코드 및 라이브러리를 배포하고 있다. 해당 라이브러리 및 소스코드를 활용하여 기관 및 업체에게 패스워드에 대한 안전성 검증을 위해 개발되었다. 본 문서의 2장에서는 윈도우즈 및 리눅스용 환경 구성 방법을 설명하고 있으며, 3장에서는 실행 시 사용할 수 있는 주요 기능과 세부 함수 및 라이브러리에 대해 설명하고 있다.

2. 환경 설정

본 문서는 패스워드안정성 검증 라이브러리 의 각 버전 별로 소스코드 환경 설정, 컴파일 을 할 수 있도록 만든 문서이다.

2.1. 라이브러리

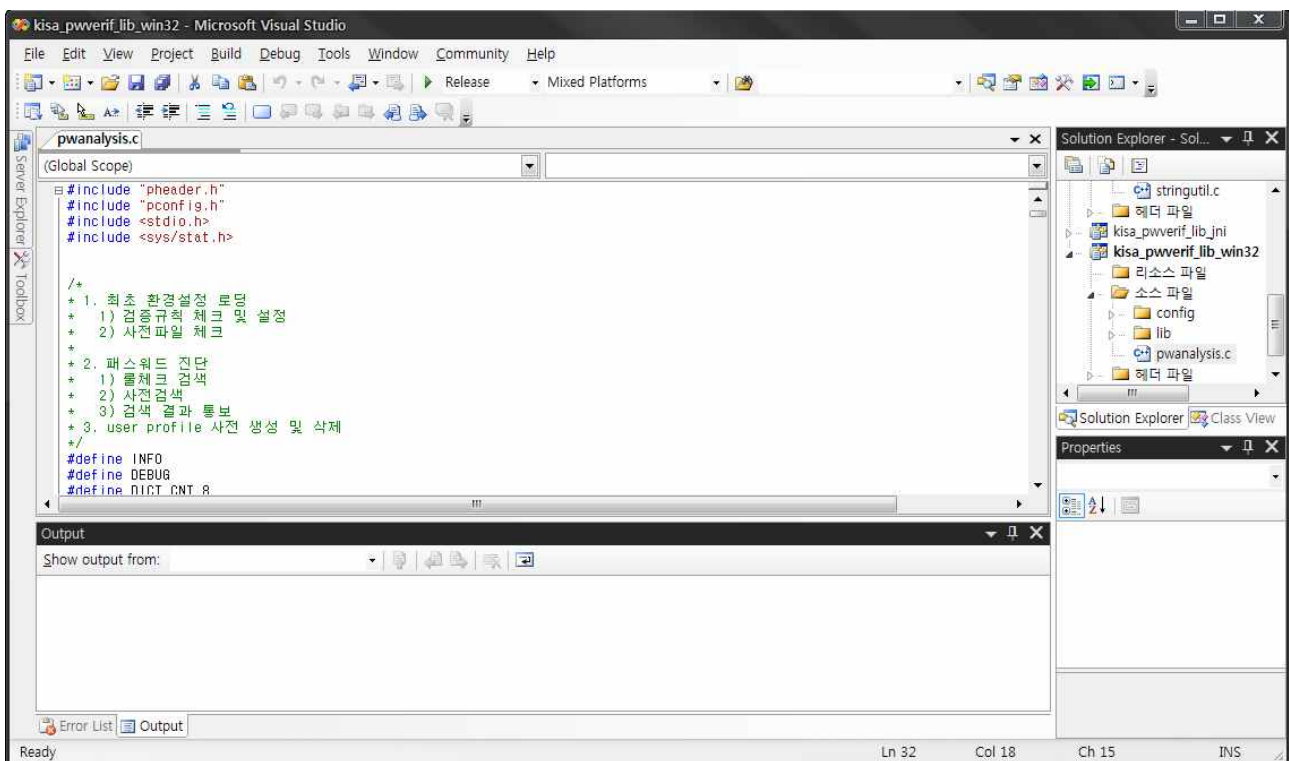
2.1.1. 윈도우 버전

2.1.1.1. dll, dll_jni, lib_test

Visual Studio 2005 에서 패스워드 안정성 검증 라이브러리 프로젝트를 빌드 하여 라이브러리, jni, 테스트 프로그램을 만들어 사용하는 방법이다.

1) 프로젝트 실행

kisa_pwverif_lib_win32 폴더의 kisa_pwverif_lib_win32.sln을 실행하여 프로젝트를 실행한다.



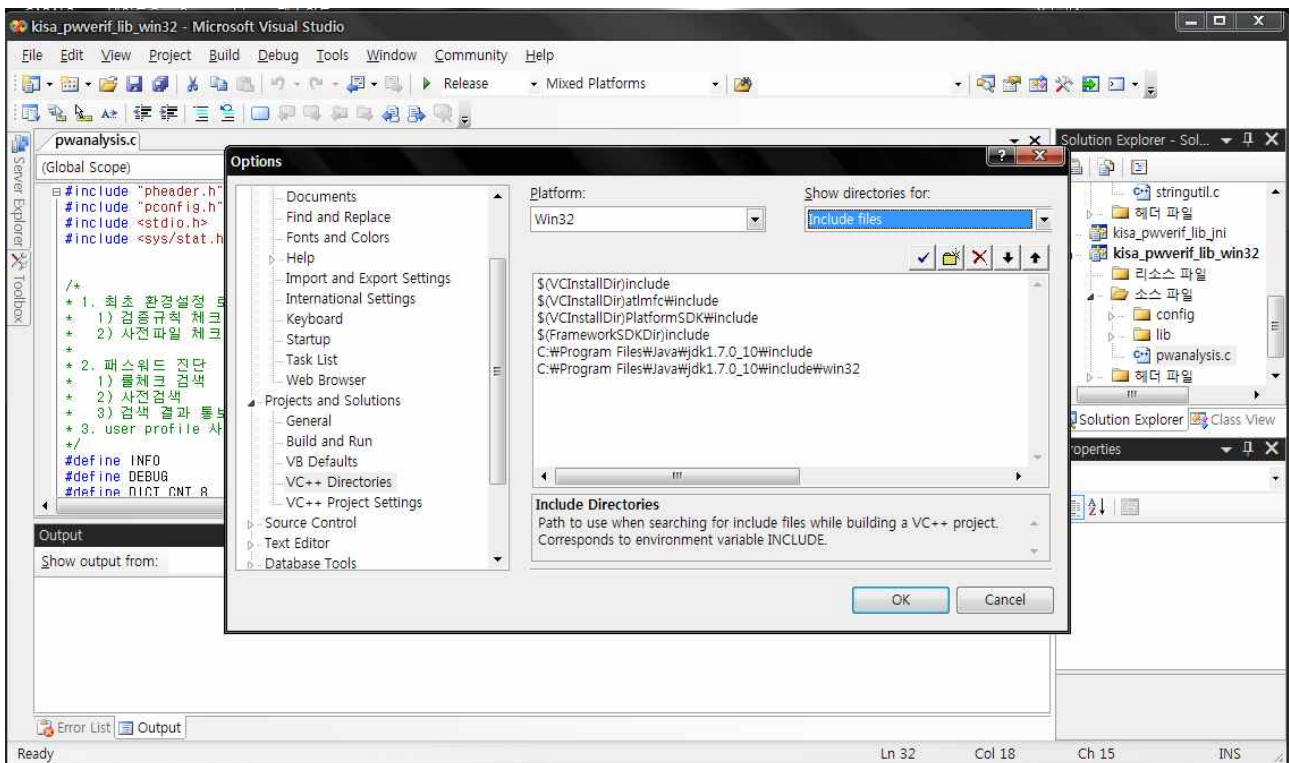
< 프로젝트 실행 >

2) JDK 설치

JDK 설치 는 부록 3을 참고 하여 설치한다.

3) Visual Studio JDK 경로 등록

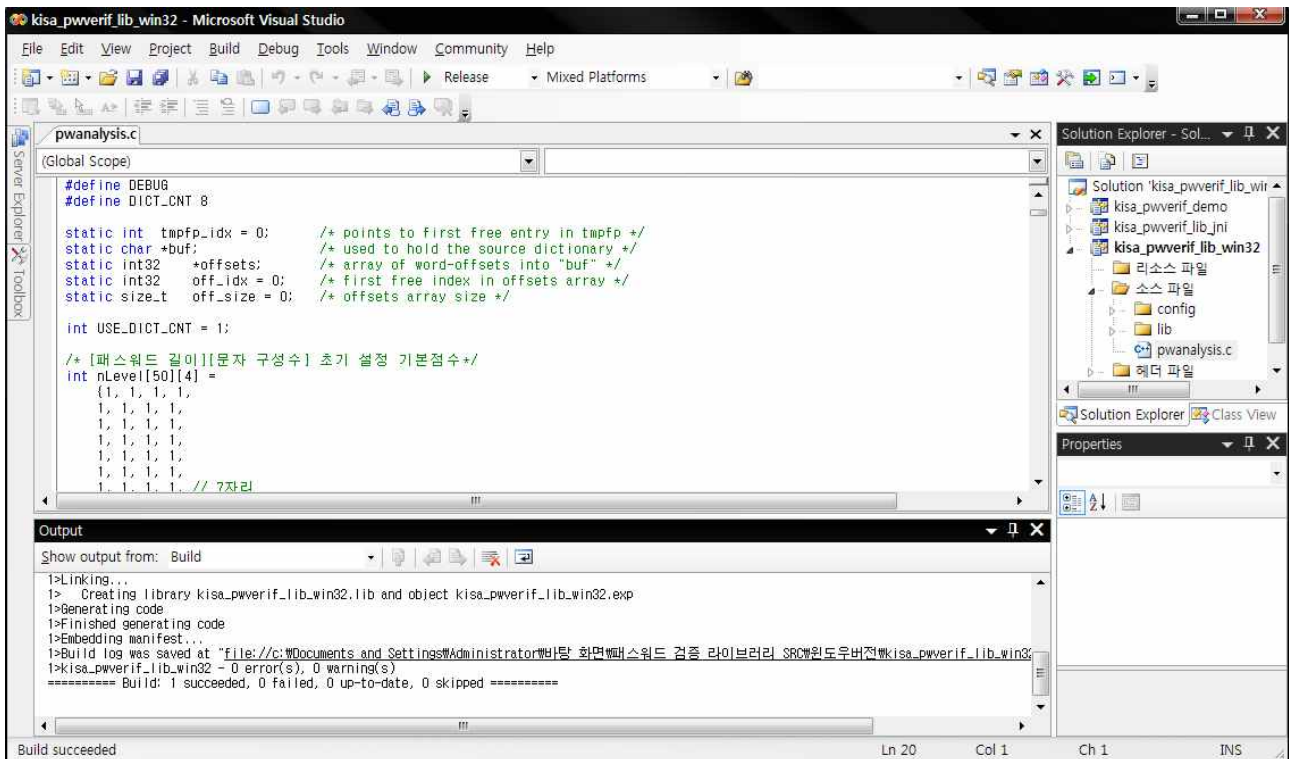
JDK 설치경로를 등록 한다. Tool -> option -> Projects and Solutions -> VC++ Directories 위치에 다음 그림과 같이 Show Directories for : Include files를 선택하고 JDK include 경로를 등록한다.



< JDK 등록 >

4) 플랫폼 클린 후 빌드

JDK 등록 후 kisa_pwverif_lib_win32 프로젝트 캄파일시 라이브러리, jni 라이브러리, 테스트 프로그램이 빌드 된다.



<프로젝트 빌드>

2.1.2. 리눅스 버전

2.1.2.1. dll

1) 라이브러리 만들기

다음은 패스워드 안정성 검증 라이브러리를 만들어서 사용하는 방법이다. 라이브러리는 다른 플랫폼 에도 적용 할 수 있도록 하기위해 공유 라이브러리를 사용하여 만들도록 하겠다.

```

1
2 CC=gcc
3 CFLAGS= -shared -static
4
5 INCS= -I.
6
7 SRCS= gstring.c  glist.c  gtable.c  gconfig.c  stringutil.c  rules.c  dictmanger.c
      pwverifier.c pwanalysis.c
8 OBJS= gstring.o  glist.o  gtable.o  gconfig.o  stringutil.o  rules.o  dictmanger.o

```

```

    pwverifier.o pwanalysis.o
9
10 LIBNAME=libpwverif.so
11
12 all: $(LIBNAME)
13
14 $(LIBNAME): $(OBJS)
15     $(CC) ${OBJS} -o $@ $(CFLAGS)
16
17 .c.o: $(SRCS)
18     $(CC) $(CFLAGS) -c $.c $(INCS)
19
20 clean:
21     rm -f *.o $(LIBNAME) core *~
22
23 install: $(EXE)
24     $(CP) ./$(LIBNAME) ../$(LIBNAME)
25
26 CP      = cp
27 MAKE    = make
28 RM      = rm -f
29 MV      = mv -f

```

<라이브러리 Makefile>

make를 실행하기 위해서는 컴파일 옵션을 수정해야 한다. 먼저 3번 라인의 옵션을 포함해야 한다. -shared는 공유 라이브러리를 사용해서 링크 하라는 옵션 이다. 그 리고 10번 라인의 libpwverif.so 라이브러리의 파일명을 적는다.

```

./make
gcc gstring.o glist.o gtable.o gconfig.o stringutil.o rules.
pwanalysis.o -o libpwverif.so -shared -static
kdhee@kdhee:~/kisa$ ls -al libpwverif.so
-rwxrwxr-x 1 kdhee kdhee 76514 12월 20 00:40 libpwverif.so
kdhee@kdhee:~/kisa$ make

```

<라이브러리 빌드>

라이브러리를 빌드(Make) 하면 libpwverif.so 파일이 생성 되는 것을 확인 할 수 있다.

2) 라이브러리 연동

빌드 된 공유 라이브러리를 사용하기 위해서는 공유 라이브러리의 헤더 파일과 공유라이브러리를 추가해야 한다. 라이브러리를 등록 하는 방법은 여러 가지가 있으나 가장 간단한 방법으로 한다. 원칙적으로 사용자가 만들어 쓰는 라이브러리는 /usr/local/lib 에 넣어 쓰는 것을 권장 하지만, 현재 위치를 그냥 명시 하는 방법을 쓴다.

```
#vi /etc/ld.so.conf
```

<라이브러리 연동 - ld.so.conf 파일 편집기 오픈>

```
include ld.so.conf.d/*.conf
```

<라이브러리연동 - ld.so.conf 파일에 include 추가>

3) 테스트 프로그램 빌드

공유 라이브러리를 포함하여 테스트 파일을 만들기 위해서는 Makefile 에 필요한 헤더 파일 경로와 라이브러리를 추가해야 한다.

```
1 .SUFFIXES : .cc .o
2
3 CC = gcc
4
5 INC = -I ../ #<- include 되는 헤더 파일의 패스를 추가한다.
6 LIBS = -lpwverif #<- 링크할 때 필요한 라이브러리를 추가한다.
7 LIB_PATH = -L ./
8 CFLAGS = -g $(INC)
9
10
11 SRCS = test.c # <- 소스 파일의 이름을 적는다.
12 OBJS = $(SRCS:.cpp=.o)
13
14 TARGET = password_test
15
16 all : $(TARGET)
17
```



```

18 $(TARGET) : $(OBJS)
19     $(CC) -o $@ $(OBJS) $(LIB_PATH) $(LIBS) $(CFLAGS)
20
21 dep :
22     gccmakedep $(INC) $(SRCS)
23
24 clean :
25     rm -rf $(TARGET) core
26
27 new :
28     $(MAKE) clean
29     $(MAKE)
30

```

<테스트 프로그램 Makefile>

Makefile의 5번 라인에 "pconfig.h", "pheader.h" 의 include 경로를 포함해야 한다. 현재 헤더파일이 ./(현재경로)에 있기 때문에 -I 옵션을 포함하여 추가 한다. 그리고 5번 라인에 링크에 필요한 라이브러리를 추가 한다.

현재 공유 라이브러리 파일이 libpwverif.so 이기 때문에 -lpwverif 로 추가해야 한다. 11번 라인에 컴파일 할 소스 파일과 14번 라인에 실행 파일의 이름을 TARGET 으로 적는다.

4) 라이브러리 소스 파일 적용

소스 파일 구성은 main() 함수에서 LoadConfigFile()을 호출하여 ./conf경로의 pwdverif.ini 파일을 호출하여 사전 파일을 load 하는 기능이다.

```

#include "pheader.h"
#include "pconfig.h"
#include <stdio.h>
#include <sys/stat.h>

int main()
{

```

```

char *rcode;
rcode = LoadConfigFile();

printf("configfile =>%s\n",rcode);

return 0;
}

```

<라이브러리 적용 테스트 코드>

파일명은 test.c 로 생성하여 함수 호출에 필요한 헤더를 추가해야 한다.
먼저 Makefile 에 LoadConfigFile() 에 대한 헤더의 경로를 패스를 추가했기 때문에
1,2번 라인과 같이 헤더를 추가하여 사용해야 한다.

이 코드는 테스트 코드로 작성했기 때문에 LoadConfigFile(); 함수만 호출하여 return
값을 확인하는 정도로 프로그래밍 하였지만 실무에서는 필요한 기능에 따라 함수들을
호출하여 사용하면 된다.

다음은 test.c 파일을 빌드 한 그림이다.

```

./make
kdhee@kdhee:~/kisa/so_test$
kdhee@kdhee:~/kisa/so_test$
kdhee@kdhee:~/kisa/so_test$
kdhee@kdhee:~/kisa/so_test$
kdhee@kdhee:~/kisa/so_test$
kdhee@kdhee:~/kisa/so_test$
kdhee@kdhee:~/kisa/so_test$ make
make: Warning: File `test.c' has modification time 1e+04 s in the future
gcc -o password_test test.c -L ./ -lpwverif -g -I ../
make: warning: Clock skew detected. Your build may be incomplete.
kdhee@kdhee:~/kisa/so_test$ ls
Makefile  conf      include  password_test  pwjni.h  test.h
Makefile_ dict5   libpwverif.so  pwjni.c      test.c
kdhee@kdhee:~/kisa/so_test$

```

<test.c 빌드>

컴파일에 필요한 헤더 경로와 컴파일 옵션이 적용되어 빌드 되는 것을 확인 할 수 있다.
make을 실행하면 현재 경로에 password_test 실행 파일이 생성되고 ./password를 실행하면 정상적으로 실행이 된다.

2.1.2.2. dll_jndi

리눅스에서 java jni 를 사용하기 위해서는 먼저 JDK 를 설치해야 한다.
JDK는 <http://www.oracle.com> 에서 최신 버전으로 다운로드 하면 된다.

1) JDK 설치 및 환경설정

다운 받은 JDK 파일을 압축을 푼다.

```
# tar xvzf jdk-7u10-linux-i586.tar.gz
```

<압축 풀기>

jdk1.7.0_10 폴더가 생성되면 /usr/lib/jvm/ 경로에 폴더를 복사한다.

```
# sudo cp jdk1.7.0_10 /usr/lib/jvm/ -Rfa
```

<JDK 폴더 복사>

복사된 JDK 를 사용하기 위해서는 PATH를 설정해야 한다.

```
#vi ~/.bashrc  
export JAVA_HOME=/usr/lib/jvm/jdk1.7.0_10/  
PATH=$PATH:$JAVA_HOME/bin:
```

<JDK PATH 설정>

2) JDK 설치 확인

```
#java -version
```

3) JNI 이용한 라이브러리 컴파일

라이브러리 컴파일을 위해서는 Java JDK 1.7.0 버전이 설치되어 있어야 하며 OS 환경변수가 추가되어 있어야 한다.

4) Native Method를 선언하는 자바 클래스(PwVerifJNI.java)를 작성한다.

```
package kisalib;
```

```

import java.util.*;

public class PwVerifJNI
{
    // 원시 메소드 선언 (Native method declaration) - 함수앞에 native 키워드 명시
    public native String init();
    public native String createProfileDictionary(String load,String save);
    public native String verify(String pwd);

    //호출할 라이브러리 위치
    private static String libPath = "./lib/libpwverif.so";

    // 사전파일이 존재하는 디렉토리 경로
    private static String DicPath = "./kisa/dicts/";
    //사전 파일 Profile.pwi 생성 디렉토리 경로
    //사전 파일 Profile.pwd 생성 디렉토리 경로
    //사전 파일 Profile.hwm 생성 디렉토리 경로
    private static String ProFilePath = "./kisa/dicts/Profile";

    //사전 파일 load data
    private static String tmpFile= "./dicts/Profile";

    //임시 test passwd
    private static String dict_pwd= "12Beavis";

    static
    {
        // Load library - 호출할 dll 파일명 명시
        System.loadLibrary("libpwverif.so");
    }

    public static void main(String[] argv)
    {
        String retval = null;
    }
}

```

```
PwVerifJNI nt = new PwVerifJNI();

retval = nt.init();
System.out.println("Env init Result[ " + retval + "]);

retval = nt.createProfileDictionary(tmpFile,ProFilePath);
System.out.println("Env create Result[ " + retval + "]);

retval = nt.verify(dict_pwd);
System.out.println("pwd verifier Result[ " + retval + "]);
}
}
```

<PwVerifJNI Java클래스>

5) java파일 컴파일 하여 클래스파일 생성한다. 아래 명령을 실행하면 현재 디렉토리 아래 ksalib 라는 디렉토리가 만들어지며 해당 디렉토리에 PwVerifJNI.class 파일이 생성된다.

```
# javac -d . PwVerifJNI.java
```

6) javah 명령어로 컴파일된 클래스로부터 c/c++을 위한 헤더파일(PwVerifJNI.h)을 생성한다. 명령 실행 후 현재 디렉토리에 PwVerifJNI.h 파일이 생성된다.

```
# javah -jni -o PwVerifJNI.h ksalib.PwVerifJNI
```

5). C언어로 Native Method를 구현한다.

```
#include "pheader.h"
#include <jni.h>          // JNI interface 헤더 파일
#include "PwVerifJNI.h"  // JNI로 만든 Native Code 헤더파일

/*
 * Class:      PwVerifJNI
 * Method:     init
 * Signature:  (Ljava/lang/String;)Ljava/lang/String;
 */
JNIEXPORT jstring JNICALL Java_ksalib_PwVerifJNI_init
    (JNIEnv *env, jobject obj)
{
    char czResult[512] = {0x00};
    char *pResult;

    pResult = LoadConfigFile();

    strcpy(czResult, pResult);

    return (*env)->NewStringUTF(env, czResult);
}
```

```

/*
 * Class:      PwVerifJNI
 * Method:     verify
 * Signature:  (Ljava/lang/String;)Ljava/lang/String;
 */

JNIEXPORT jstring JNICALL Java_kisalib_PwVerifJNI_verify
(JNIEnv *env, jobject obj, jstring passwd)
{
    const char* strPasswd = (*env)->GetStringUTFChars(env, passwd, 0);
    char czResult[512] = {0x00};
    char *pResult = VerifyPW((char*)strPasswd);
    (*env)->ReleaseStringUTFChars(env, passwd, strPasswd);

    strcpy(czResult, pResult);

    return (*env)->NewStringUTF(env, czResult);
}

/*
 * Class:      PwVerifJNI
 * Method:     createProfileDictionary
 * Signature:  (Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
 */

JNIEXPORT jstring JNICALL Java_kisalib_PwVerifJNI_createProfileDictionary
(JNIEnv *env, jobject obj, jstring type, jstring filepath)
{
    char czResult[512] = {0x00};
    const char* strType = (*env)->GetStringUTFChars(env, type, 0);
    const char* strFilePath = (*env)->GetStringUTFChars(env, filepath, 0);
    char *pResult;

    if (strType[0] == 'C')
        pResult = CreateDict((char*)strFilePath, PROFILE_DICT);
    else
        pResult = DeleteDict(PROFILE_DICT);
}

```

```

strcpy(czResult, pResult);
return (*env)->NewStringUTF(env, czResult);
}

```

<Native Method>

7). JNI를 통해 C/C++ 코드 함수를 호출하기 위한 기본 구조 설계를 마쳤으므로 리눅스 동적라이브러리 파일(.so) 생성을 위해 소스파일을 다음과 같이 컴파일 한다.

- PwVerifJNI.cpp와 PwVerifJNI.cpp 를 각각 컴파일 후 동적라이브러리를 생성 한다
- 컴파일 시 동적라이브러리로 컴파일하기 위해 Makefile 에 "-shared" 옵션을 넣는다.
- 컴파일 시 자바 라이브러리를 가져오기 위해 Makefile "-I" 옵션으로 JDK include 위치를 지정 한다

Makefile 작성

```

#java jdk 설치 경로
JAVA_HOME=/usr/lib/jvm/jdk1.7.0_10/

#컴파일러 설정
CC=gcc

#컴파일 옵션 추가
CFLAGS= -shared -static

#jdk 헤더 경로 추가
INCS= -I. -I${JAVA_HOME}/include
INCS+=-I${JAVA_HOME}/include/linux

#소스 파일 추가
SRCS= gstring.c glist.c gtable.c gconfig.c stringutil.c rules.c dictmanger.c
pwverifier.c pwanalysis.c PwVerifJNI.c

#헤더 경로 추가
OBJS= gstring.o glist.o gtable.o gconfig.o stringutil.o rules.o dictmanger.o
pwverifier.o pwanalysis.o PwVerifJNI.o

#컴파일후 생성된 공유 라이브러리 이름 설정
LIBNAME=libpwverif.so

all: $(LIBNAME)

$(LIBNAME): $(OBJS)
$(CC) ${OBJS} -o $@ $(CFLAGS)

```

<Native Method Make File>

컴파일 실행

```
#make clean; make
```

8) 컴파일을 마치면 컴파일을 실행한 디렉토리에 Linux용 동적 라이브러리가 생성 된다.

※ 라이브러리명 : libpwverif.so

```
java PwVerifJNI
```

<자바 테스트 실행>

2.1.2.3. lib_test

다음은 libpwverif.so 공유 라이브러리를 이용하여 test.c 파일을 빌드 한 내용이다
내용은 ./password_test 를 실행하여 LoadConfigFile()함수를 이용하여 사전 파일
데이터를 읽어 들이는 프로그램이다.

실행한 내용은 다음 그림과 같다.

```
kdhee@kdhee:~/kisa/so_test$ ./password_test
[PWD_VERIF_DICT]dict.file.prefix.1=./dicts/names_dict
[PWD_VERIF_DICT]dict.file.prefix.2=./dicts/loanword_dict
[PWD_VERIF_DICT]dict.file.prefix.3=./dicts/economy_dict
[PWD_VERIF_DICT]dict.file.prefix.4=./dicts/stock_dict
[PWD_VERIF_DICT]dict.file.prefix.5=./dicts/it_dict
[PWD_VERIF_DICT]dict.file.prefix.6=./dicts/chat_dict
[PWD_VERIF_DICT]dict.file.prefix.7=./dicts/fadwords_dict
[PWD_VERIF_DICT]dict.file.prefix.8=default
[PWD_VERIF_RULE]pwd_verif_rule.min_len=6
[PWD_VERIF_RULE]pwd_verif_rule.min_diff=4
[PWD_VERIF_RULE]pwd_verif_rule.max_step=3
[PWD_VERIF_RULE]pwd_verif_rule.postfix_num=2
[PWD_VERIF_RULE]pwd_verif_rule.prefix_num=2
[PWD_VERIF_SEARCH]pwd_verif_search.pw_postfix_num=1
[PWD_VERIF_SEARCH]pwd_verif_search.pw_prefix_num=1
[PWD_VERIF_SEARCH]pwd_verif_search.pw_min_rept=1
[PWD_VERIF_SEARCH]pwd_verif_search.pw_reverse=1
configfile =>RET_CD=0000:PW_LEN=6:CTS_NCR=3:REPT_NCR=3:NCHAR=4:USE_DICT_CNT=7:USE_PDI
CT=00
kdhee@kdhee:~/kisa/so_test$ █
```

<라이브러리 테스트>

빌드 된 공유 라이브러리로 링크 되어 실행 파일이 정상적으로 사전 목록에서 데이터를 불러오는 것을 확을 할 수 있다.

2.1.3. 유닉스 버전

2.1.3.1. dll

1) 라이브러리 만들기

다음은 유닉스의 공유 라이브러리를 만들어 사용하는 방법이다. 유닉스도 다른 플랫폼에 사용 하기위해 공유 라이브러리를 만들도록 하겠다.

```
1
2 CC=gcc
3 CFLAGS= -G
4 #CFLAGS= -shared -static
5 INCS= -I.
6
7 SRCS= gstring.c  glist.c  gtable.c  gconfig.c  stringutil.c  rules.c  dictmanger.c
      pwverifier.c pwanalysis.c
8 OBJS= gstring.o  glist.o  gtable.o  gconfig.o  stringutil.o  rules.o  dictmanger.o
      pwverifier.o pwanalysis.o
9
10 LIBNAME=libpwverif.so
11
12 all: $(LIBNAME)
13
14 $(LIBNAME): $(OBJS)
15     $(CC) ${OBJS} -o $@ $(CFLAGS)
16
17 .c.o: $(SRCS)
18     $(CC) $(CFLAGS) -c *.c $(INCS)
19
20 clean:
21     rm -f *.o $(LIBNAME) core *~
22
23 install: $(EXE)
24     $(CP) ./$(LIBNAME) ../$(LIBNAME)
25
26 CP      = cp
27 MAKE    = make
28 RM      = rm -f
29 MV      = mv -f
```

<유닉스 Makefile>

솔라리스에서 공유 라이브러리로 만들기 위해서는 컴파일 옵션을 변경해야 한다. Makefile 의 3번 라인에 CFLAGS= -G 을 추가하고 기존의 4번 라인 #CFLAGS=-shared -static 은 주석을 해야 한다. 그리고 생성되는 라이브러리의 파일 이름으로 NAME=libpwverif.so 을 추가해준다.

```
#make clean; make
```

```
pwanalysis.c:1050:3: warning: incompatible implicit declaration of built-in function 'free'
pwanalysis.c: In function 'InsertWord':
pwanalysis.c:1116:18: warning: incompatible implicit declaration of built-in function 'realloc'
pwanalysis.c:1122:4: warning: incompatible implicit declaration of built-in function 'free'
gcc gstring.o glist.o gtable.o gconfig.o stringutil.o rules.o dictmanger.o pwverifier.o pwanalysis.o -o libpwv
so -G
kdhee@solaris:~/kisa$ ls -al libpwverif.so
-rwxr-xr-x 1 kdhee staff 69240 12월 21일 06:19 libpwverif.so
kdhee@solaris:~/kisa$
```

<라이브러리 빌드>

라이브러리를 빌드(make) 하면 libpwverif.so 파일이 생성 되는 것을 확인 할 수 있다.

3) 테스트 프로그램 빌드

공유 라이브러리를 포함하여 테스트 파일을 만들기 위해서는 Makefile 에 필요한 헤더 파일 경로와 라이브러리를 추가해야 한다.

```
1 .SUFFIXES : .cc .o
2
3 CC = gcc
4
5 INC = -I ../ #<- include 되는 헤더 파일의 패스를 추가한다.
6 LIBS = -lpwverif #<- 링크할 때 필요한 라이브러리를 추가한다.
7 LIB_PATH = -L ./
8 CFLAGS = -g $(INC)
9
10
11 SRCS = test.c # <- 소스 파일의 이름을 적는다.
12 OBJS = $(SRCS:.cpp=.o)
13
14 TARGET = password_test
15
```

```

16 all : $(TARGET)
17
18 $(TARGET) : $(OBJS)
19     $(CC) -o $@ $(OBJS) $(LIB_PATH) $(LIBS) $(CFLAGS)
20
21 dep :
22     gccmakedep $(INC) $(SRCS)
23
24 clean :
25     rm -rf $(TARGET) core
26
27 new :
28     $(MAKE) clean
29     $(MAKE)
30

```

<테스트 프로그램 Makefile>

Makefile의 5번 라인에 "pconfig.h", "pheader.h" 의 include 경로를 포함해야 한다. 현재 헤더파일이 ./(현재경로)에 있기 때문에 -I 옵션을 포함하여 추가 한다. 그리고 5번 라인에 링크에 필요한 라이브러리를 추가 한다. 현재 공유 라이브러리 파일이 libpwverif.so 이기 때문에 -lpwverif 로 추가해야 한다. 11번 라인에 컴파일 할 소스 파일과 14번 라인에 실행 파일의 이름을 TARGET 으로 적는다.

4) 라이브러리 소스 파일 적용

소스 파일 구성은 main() 함수에서 LoadConfigFile()을 호출하여 ./conf경로의 pwdverif.ini 파일을 호출하여 사전 파일을 load 후 임의의 패스워드 검증 후 tmp.txt 파일을 dicts/profile 경로에 사전파일을 추가하는 기능이다.

```

#include "pheader.h"
#include "pconfig.h"
#include <stdio.h>
#include <sys/stat.h>

```

```

int main()
{
    char *rcode;
    rcode = LoadConfigFile();
    printf("configfile =>%s\n",rcode);

    rcode = verivfPW("12qwert");
    printf("configfile =>%s\n",rcode);

    rcode = CreateDict("tmp.txt","dicts/profile");
    printf("configfile =>%s\n",rcode);

    return 0;
}

```

<라이브러리 적용 테스트 코드>

파일명은 sample.c 로 생성하며 함수 호출에 필요한 헤더를 추가해야 한다.
 먼저 Makefile 에 LoadConfigFile(),verivfPW(), CreateDict(),에 대한 헤더의 경로를 패스를 추가했기 때문에 "pheader.h","pconfig.h" 헤더를 추가하여 사용해야 한다.
 이 코드는 테스트 코드로 작성했기 때문에 LoadConfigFile(),verivfPW(), CreateDict(),함수를 호출 하여 사전파일 config file load 와 임의의 password 입력, tmp.txt 파일을 입력하여 profile사전파일을 테스트 하는 목적으로 생성하는 코드이다.

2.1.3.2. dll_jndi

유닉스의 java jni 사용하는 과정은 리눅스 와 동일하다. 참고로 소스에 포함된 Makefile_jndi 파일을 Makefile 로 변환 하여 빌드하면 jni 공유 라이브러리로 사용할 수 있다.

org_kisa_pwverif_lib_PwVerifJni.h , pwveriflib.c 이 jni 와 연동된 파일이다.
 Makefile 에 "jni.h" PATH 경로만 설정해 주면 리눅스와 동일하게 빌드가 될 것이다.
 참고로 다음은 PwVerifJNI.Java 소스 파일의 클래스 내용이다.

```

package org.kisa.pwverif.lib;

public class PwVerifJNI
{
    public native String init();
    public native String createProfileDictionary(String load,String save);
    public native String verify(String pwd);

    System.loadLibrary("libpwverifJNI.so");

    public static void main(String[] argv)
    {
        String retval = null;
        PwVerifJNI nt = new PwVerifJNI();

        retval = nt.init();
        System.out.println("Env init Result[ " + retval + "]);
    }
}

```

```

        retval = nt.createProfileDictionary("tmp.txt","profile");
        System.out.println("Env create Result[ " + retval + "]);

        retval = nt.verify("12Beavis");
        System.out.println("pwd verifier Result[ " + retval + "]);
    }
}

```

먼저 생성한 libpwverifJNI.so 파일을 System.loadLibrary("libpwverifJNI.so"); 함수를

이용하여 C소스의 동적 라이브러리 파일을 사용하면 JAVA 소스에서 c함수를 호출하여 사용 할 수 있다.

2)java class 파일을 실행한다.

```
#java PwVerifJNI
```

<자바 테스트 실행>

2.1.3.3. lib_test

다음은 libpwverif.so 공유 라이브러리를 이용하여 test.c 파일을 빌드 한 내용이다
내용은 ./password_test를 실행한 내용은 다음 그림과 같다.

```
Env Load [PWD_VERIF_DICT]dict.file.prefix.1=./dicts/names_dict
[PWD_VERIF_DICT]dict.file.prefix.2=./dicts/loanword_dict
[PWD_VERIF_DICT]dict.file.prefix.3=./dicts/economy_dict
[PWD_VERIF_DICT]dict.file.prefix.4=./dicts/stock_dict
[PWD_VERIF_DICT]dict.file.prefix.5=./dicts/it_dict
[PWD_VERIF_DICT]dict.file.prefix.6=./dicts/chat_dict
[PWD_VERIF_DICT]dict.file.prefix.7=./dicts/fadwords_dict
[PWD_VERIF_DICT]dict.file.prefix.8=./dicts/test
[PWD_VERIF_RULE]pwd_verif_rule.min_len=6
[PWD_VERIF_RULE]pwd_verif_rule.min_diff=4
[PWD_VERIF_RULE]pwd_verif_rule.max_step=3
[PWD_VERIF_RULE]pwd_verif_rule.postfix_num=2
[PWD_VERIF_RULE]pwd_verif_rule.prefix_num=2
[PWD_VERIF_SEARCH]pwd_verif_search.pw_postfix_num=1
[PWD_VERIF_SEARCH]pwd_verif_search.pw_prefix_num=1
[PWD_VERIF_SEARCH]pwd_verif_search.pw_min_rept=1
[PWD_VERIF_SEARCH]pwd_verif_search.pw_reverse=1

Env Load Result[RET_CD=0000:PW_LEN=6:CTS_NCR=3:REPT_NCR=3:NCHAR=4:USE_DICT_CNT=8:USE_PDICT=00]
Password Safety Verification Lib Test
1. Env Load
2. Verify Password
3. Personal Dict Create
4. Personal Dict Delete
5. User Dict Create
6. Pull data out of dict
7. Exit
```

<라이브러리 테스트>

빌드 된 공유 라이브러리로 링크되어 실행 파일이 정상적으로 사전 목록에서 데이터를 불러오는 것을 확을 할 수 있다.

3. 실행 방법

3.1. 개요

본 문서는 패스워드 안정성 검증 S/W에 대한 검증 라이브러리 및 진단 소프트웨어 시험 절차서 이다.

3.1.1. 목적

본 문서의 목적은 다음과 같다.

- 1) 패스워드 안정성 검증 라이브러리 대한 명확한 검증 절차 기술
- 2) 패스워드 안정성 검증 S/W 테스트 프로그램 으로 검증을 위한 상세 가이드 제공
- 3) 각 테스트 케이스의 검증 결과를 판단할 수 있는 자료 제공

3.1.2. 범위

리눅스, 유닉스, 윈도우 의 패스워드 안정성 검증 라이브러리 테스트 인터페이스가 동일하므로 리눅스 테스트 프로그램을 기준으로 작성 하였다.

본 문서의 범위는 다음과 같다.

- 1) 패스워드 안정성 검증 라이브러리 테스트 프로그램 사용 절차 기록
- 2) 본 문서의 시험 절차서 상 필요한 테스트 케이스의 검증 절차 기록
- 3) 각 시험 절차서의 검증 결과 , 예외 상황 발생 처리 및 그 결과 기록

다음과 같은 사항은 본 문서에 포함되지 않는다.

- 1) 본 시험 절차서에 포함되지 않은 테스트 항목
- 2) 각 테스트 케이스와 연관관계가 있으나 각 테스트 케이스에 명시되지 않은 단위 기능과 세부 기능에 해당하는 부분

3.2. 실행 순서

케이스 ID	케이스 명	시스템 명
LIB_001	환경 설정 정보 로드	진단 라이브러리
LIB_002	패스워드 검증	진단 라이브러리
LIB_003	개인정보 사전 파일 생성	진단 라이브러리
LIB_004	개인정보 사전 파일 제거	진단 라이브러리
LIB_005	사용자 사전 파일 생성	진단 라이브러리
LIB_006	사전 파일에서 데이터 추출	진단 라이브러리

3.2.1. LIB_0001 : 환경 설정 정보 로드

케이스 ID /케이스명	LIB_001 : 환경 설정 정보 로드	시스템 명	검증 라이브러리
입력데이터 / 테스트 방법 / 수행 결과			
<p>1. 입력 데이터</p> <p>검증 라이브러리 리눅스 테스트 프로그램 pwverif 위치에 ./conf/pwdverif.ini 를 확인 한다.</p> <p>pwdverif.ini 파일 내용은 다음과 같다.</p> <pre>##### # 1. 사전 종류 정의 #####</pre>			

[PWD_VERIF_DICT]

dict.file.prefix.1=./dicts/names_dict
dict.file.prefix.2=./dicts/loanword_dict
dict.file.prefix.3=./dicts/economy_dict
dict.file.prefix.4=./dicts/stock_dict
dict.file.prefix.5=./dicts/it_dict
dict.file.prefix.6=./dicts/chat_dict
dict.file.prefix.7=./dicts/fadwords_dict

#####

2. 사전검증 규칙 설정

#####

[PWD_VERIF_RULE]

패스워드 최소길이 -- default - 6

pwd_verif_rule.min_len=6

패스워드 중복되는 문자 제외한 수 구성 - 4

pwd_verif_rule.min_diff=4

패스워드 연속문자 체크 기본값 -- default - 3

pwd_verif_rule.min_step=3

패스워드 뒤 숫자길이 설정(1,2) - default 2(0~99), 1:(0~9)

pwd_verif_rule.postfix_num=2

패스워드 앞 숫자길이 설정(1,2) - default 2(0~99), 1:(0~9)

pwd_verif_rule.prefix_num=2

패스워드 반복문자열 체크

pwd_verif_rule.min_rept=3

#####

3. 사전 검색 옵션 설정 - 사용 1, 미사용 - 0

1) 문자열 반복 사용 유무

2) 패스워드 뒤에 숫자 붙이기 사용 유무 체크

3) 패스워드 앞에 숫자 붙이기 사용 유무 체크

4) 문자열 뒤집기 사용 유무

#####

[PWD_VERIF_SEARCH]

반복문자열 사용 유무

pwd_verif_search.pw_min_rept=1

패스워드 뒤에 숫자 붙이기 사용 유무

pwd_verif_search.pw_postfix_num=1

패스워드 앞에 숫자 붙이기 사용 유무

pwd_verif_search.pw_prefix_num=1

문자열 뒤집기 사용 유무

pwd_verif_search.pw_reverse=1

#####

4. 사전파일 생성시 원복 삭제 유무 - 삭제 1, 미삭제 - 0

#####

[PWD_VERIF_ETC]

pwd_create.orgfile.delete=1

2. 테스트 절차

1). pwdverif.ini 확인 후 pwdverif 프로그램을 실행한다.

2). Select Number에 1을 입력 후 엔터를 실행한다.

3). 출력 내용을 확인 한다.

수행 결과

```
Password Safety Verification Lib Test

1. Env Load
2. Verify Password
3. Personal Dict Create
4. Personal Dict Delete
5. User Dict Create
6. Pull data out of dict
7. Exit

Select Number => 1

Env Load [PWD_VERIF_DICT]dict.file.prefix.1=./dicts/names_dict
[PWD_VERIF_DICT]dict.file.prefix.2=./dicts/loanword_dict
[PWD_VERIF_DICT]dict.file.prefix.3=./dicts/economy_dict
[PWD_VERIF_DICT]dict.file.prefix.4=./dicts/stock_dict
[PWD_VERIF_DICT]dict.file.prefix.5=./dicts/it_dict
[PWD_VERIF_DICT]dict.file.prefix.6=./dicts/chat_dict
[PWD_VERIF_DICT]dict.file.prefix.7=./dicts/yjkim
[PWD_VERIF_RULE]pwd_verif_rule.min_len=6
[PWD_VERIF_RULE]pwd_verif_rule.min_diff=4
[PWD_VERIF_RULE]pwd_verif_rule.max_step=3
[PWD_VERIF_RULE]pwd_verif_rule.postfix_num=2
[PWD_VERIF_RULE]pwd_verif_rule.prefix_num=2
[PWD_VERIF_SEARCH]pwd_verif_search.pw_postfix_num=1
[PWD_VERIF_SEARCH]pwd_verif_search.pw_prefix_num=1
[PWD_VERIF_SEARCH]pwd_verif_search.pw_min_rept=1
[PWD_VERIF_SEARCH]pwd_verif_search.pw_reverse=1

Env Load Result[RET_CD=0000:PW_LEN=6:CTS_NCR=3:REPT_NCR=3:NCHAR=4:USE_DICT_CNT=8:USE_PDICT=00]
```

< 환경설정 로드 결과 값>

1. 사전종류 정보 로드됨.
2. 패스워드 최소길이 설정 값 로드됨.
3. 패스워드에 포함될 최소 문자수 개수 로드됨.
4. 패스워드 연속문자 개수로드됨.
5. 패스워드 뒤에 숫자 1개 또는 2개 붙이기 설정 값 로드됨.
6. 패스워드 앞에 숫자 1개 또는 2개 붙이기 설정 값 로드됨.
7. 패스워드 뒤에 숫자 1개 또는 2개 붙이기 사용유무 로드됨.
8. 문자열 뒤집기 검색사용 유무 설정 값 로드됨.
9. 문자열 반복 사용 유무 설정 값 로드됨.
10. 정상적인 결과 리턴 값 확인.

3.2.2. LIB_0002 : 패스워드 검증

케이스 ID /케이스명	LIB_002 : 패스워드 검증	시스템 명	검증 라이브러리
입력데이터 / 테스트 방법 / 수행 결과			
<p>1. 입력 데이터</p> <p>검증 하고자 하는 패스워드를 입력 한다.</p> <p>2. 테스트 방법</p> <p>1). 테스트 케이스 LIB-001을 선 수행 한다. 2). Select Number 2를 입력 후 검증 패스워드를 입력한다. 3). 짧은 단어 패스워드 , 숫자/영문 조합 패스워드를 입력 하여 결과 값을 비교한다.</p> <div data-bbox="276 1111 1318 1503" data-label="Code-Block"> <pre> Password Safety Verification Lib Test 1. Env Load 2. Verify Password 3. Personal Dict Create 4. Personal Dict Delete 5. User Dict Create 6. Pull data out of dict 7. Exit Select Number => 2 input password : testprogram </pre> </div> <p>< 짧은 단어 패스워드 검색></p>			

```

Password Safety Verification Lib Test

1. Env Load
2. Verify Password
3. Personal Dict Create
4. Personal Dict Delete
5. User Dict Create
6. Pull data out of dict
7. Exit

Select Number => 2

input password : votmdnjen2144

```

< 숫자/영문 조합 패스워드 검색>

4). 출력 결과를 확인 한다.

수행 결과

```

Password Safety Verification Lib Test

1. Env Load
2. Verify Password
3. Personal Dict Create
4. Personal Dict Delete
5. User Dict Create
6. Pull data out of dict
7. Exit

Select Number => 2

input password : testprogram
[1] BASIC JUMSU[FWLEN=11][CHARACTERS=1]=[2]
[2] [MINDIFF=4],[junk=9][JUMSU=2]
[3] [USEMINREPT=1],[MINREPT=3][repcnt=0][JUMSU=2]
[4] [MINSTEP=3],[SeqChar=1][JUMSU=2]
[DictFindPW][DICT=./dicts/default_dict]
[INFO]DICT SEARCH FILE_NAME ./dicts/default_dict FOUND
[5] [DICT_SEARCH=FOUND],[JUMSU=1]
[6] [Personal DICT_SEARCH=NOT USED],[JUMSU=1]
[input password=testprogram],[Search Result=RET_CD=0000:RT_LEVEL=1:FW_LEN=11:PW_NCR=1:REPT_NCR=0:CTS_NCR=1:DICT_FD_CNT=1:PDICT_FD=XX]

```

< 짧은 단어 패스워드 검색 결과>

```

input password : votmdnjem2144
[1] BASIC JUMSU[PWLEN=13][CHARACTERS=2]=[3]
[2] [MINDIFF=4],[junk=11][JUMSU=3]
[3] [USEMINREPT=1],[MINREPT=3][repcnt=0][JUMSU=3]
[4] [MINSTEP=3],[SeqChar=1][JUMSU=3]
[DictFindPW][DICT=./dicts/default_dict]
[INFO]USEREVERSE[1]
[INFO]POSTFIX SEARCH[100]
[INFO]PREFIX SEARCH[100]
[INFO]DICT SEARCH FILE_NAME ./dicts/default_dict, NOT FOUND
[DictFindPW][DICT=./dicts/names_dict]
[INFO]USEREVERSE[1]
[INFO]POSTFIX SEARCH[100]
[INFO]PREFIX SEARCH[100]
[INFO]DICT SEARCH FILE_NAME ./dicts/names_dict, NOT FOUND
[DictFindPW][DICT=./dicts/loanword_dict]
[INFO]USEREVERSE[1]
[INFO]POSTFIX SEARCH[100]
[INFO]PREFIX SEARCH[100]
[INFO]DICT SEARCH FILE_NAME ./dicts/loanword_dict, NOT FOUND
[DictFindPW][DICT=./dicts/economy_dict]
[INFO]USEREVERSE[1]
[INFO]POSTFIX SEARCH[100]
[INFO]PREFIX SEARCH[100]
[INFO]DICT SEARCH FILE_NAME ./dicts/economy_dict, NOT FOUND
[DictFindPW][DICT=./dicts/stock_dict]
[INFO]USEREVERSE[1]
[INFO]POSTFIX SEARCH[100]
[INFO]PREFIX SEARCH[100]
[INFO]DICT SEARCH FILE_NAME ./dicts/stock_dict, NOT FOUND
[DictFindPW][DICT=./dicts/it_dict]
[INFO]USEREVERSE[1]
[INFO]POSTFIX SEARCH[100]
[INFO]PREFIX SEARCH[100]
[INFO]DICT SEARCH FILE_NAME ./dicts/it_dict, NOT FOUND
[INFO]USEREVERSE[1]
[INFO]POSTFIX SEARCH[100]
[INFO]PREFIX SEARCH[100]
[INFO]DICT SEARCH FILE_NAME ./dicts/it_dict, NOT FOUND
[DictFindPW][DICT=./dicts/chat_dict]
[INFO]USEREVERSE[1]
[INFO]POSTFIX SEARCH[100]
[INFO]PREFIX SEARCH[100]
[INFO]DICT SEARCH FILE_NAME ./dicts/chat_dict, NOT FOUND
[DictFindPW][DICT=./dicts/yjkim]
[INFO]USEREVERSE[1]
[INFO]POSTFIX SEARCH[100]
[INFO]PREFIX SEARCH[100]
[INFO]DICT SEARCH FILE_NAME ./dicts/yjkim, NOT FOUND
[5] [DICT_SEARCH=NOT FOUND],[JUMSU=3]
[6] [Personal DICT_SEARCH=NOT USED],[JUMSU=3]
[input password=votmdnjem2144],[Search Result=RET_CD=0000:RT_LEVEL=3:FW_LEN=13:FW_NCR=2:REPT_NCR=0:CTS_NCR=1:DICT_FD_CNT=9:PDICT_FD=XX]

```

< 숫자/영문 조합 패스워드 검색>

1. 출력결과 두 조건 모두 [1] 항목에서 패스워드 길이 및 문자 구성수를 체크하고 해당 하는 기본 점수를 부여받음.
2. [2] 항목에서 중복 문자 있는지 체크 결과 출력됨.
3. [3] 항목에서 반복문자 체크 결과 출력됨.
4. [4] 항목에서 연속문자 체크 결과 출력됨.
5. 짧은 단어 패스워드는 default_dict 사전에서 검색 되 점수 감점 출력됨.
6. 영문/숫자 조합 패스워드는 8개 사전에서 검색되지 않고 상급 점수가 출력됨.
7. 정상적인 최종 검색 결과 리턴 값을 확인함.

3.2.3. LIB_0003 : 개인정보 사전 파일 생성

케이스 ID /케이스명	LIB_003 : 개인정보 사전 파일 생성	시스템 명	검증 라이브러리
입력데이터 / 테스트 방법 / 수행 결과			
<p>1. 입력 데이터</p> <p>개인정보의 내용이 포함하고 있는 파일을 pwdverif 실행 파일이 있는 경로에 생성한다.</p> <p>[tmp.txt] 내용</p> <pre> program test yjkim 8282 monitoring play love 01046465858 027557777 kimyoungjae daniel </pre> <p>2. 테스트 절차</p> <p>1). Select Number 3 을 입력 입력한다.</p>			


```

Password Safety Verification Lib Test

1. Env Load
2. Verify Password
3. Personal Dict Create
4. Personal Dict Delete
5. User Dict Create
6. Pull data out of dict
7. Exit

Select Number => 3

```

수행 결과

```

Password Safety Verification Lib Test

1. Env Load
2. Verify Password
3. Personal Dict Create
4. Personal Dict Delete
5. User Dict Create
6. Pull data out of dict
7. Exit

Select Number => 3

Personal Dict Create :Current Dir, tmp.txt [Create Personal Dict]

Personal Dict Create Result[RET_CD=0000]

```

<개인정보 사전 파일 생성 결과 값>

```

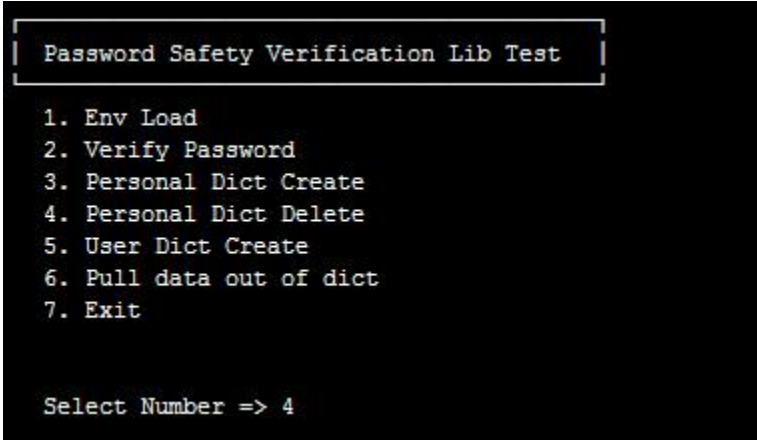
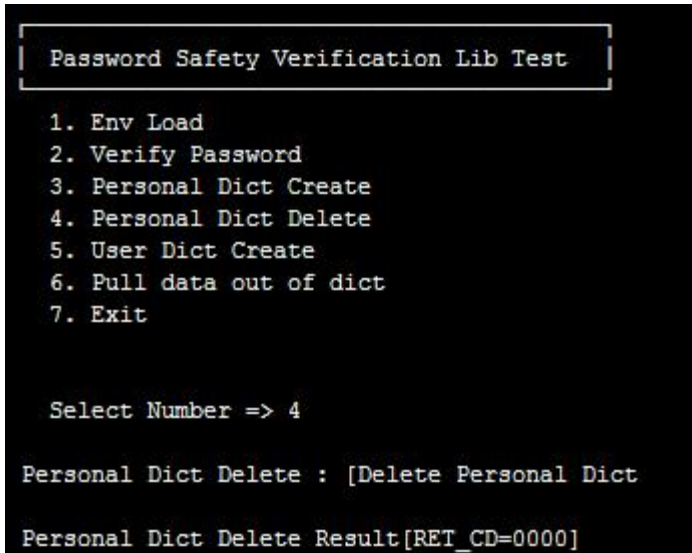
yjkim@kdhee:~/kisa/dicts$ ls
chat_dict.hwm      default_dict.pwi  enfl.pwd          it_dict.hwm       loanword_dict.pwi  profile.pwd       yjkim.hwm
chat_dict.pwd      economy_dict.hwm  enfl.pwi          it_dict.pwd       names_dict.hwm     profile.pwi       yjkim.pwd
chat_dict.pwi      economy_dict.pwd  fadwords_dict.hwm it_dict.pwi       names_dict.pwd     stock_dict.hwm    yjkim.pwi
default_dict.hwm   economy_dict.pwi  fadwords_dict.pwd loanword_dict.hwm  names_dict.pwi     stock_dict.pwd
default_dict.pwd   enfl.hwm         fadwords_dict.pwi loanword_dict.pwd  profile.hwm        stock_dict.pwi

```

<profil 사전 파일이 생성 결과 값>

- 1. 정상적인 결과 리턴 값 확인.
- 2. profile 사전 파일을 생성함.

3.2.4. LIB_0004 : 개인정보 사전 파일 제거

케이스 ID /케이스명	LIB_004 : 개인정보 사전 파일 제거	시스템 명	검증 라이브러리
입력데이터 / 테스트 방법 / 수행 결과			
<p>1. 입력 데이터</p> <p>2. 테스트 절차</p> <p>1). Select Number 4 을 입력 입력한다.</p>			
			
수행 결과			
 <p>< 개인정보 사전 파일 제거 결과 값></p>			

```
yjkim@kdhee:~/kisa/dicts$ ls
chat_dict.hwm  default_dict.pwd  economy_dict.pwi  fadwords_dict.hwm  it_dict.pwd  loanword_dict.pwi  stock_dict.hwm  yjkim.pwd
chat_dict.pwd  default_dict.pwi  enfl.hwm  fadwords_dict.pwd  it_dict.pwi  names_dict.hwm  stock_dict.pwd  yjkim.pwi
chat_dict.pwi  economy_dict.hwm  enfl.pwd  fadwords_dict.pwi  loanword_dict.hwm  names_dict.pwd  stock_dict.pwi
default_dict.hwm  economy_dict.pwd  enfl.pwi  it_dict.hwm  loanword_dict.pwd  names_dict.pwi  yjkim.hwm
```

< profil 사전 파일이 생성 결과 값>

1. 정상적인 결과 리턴 값 확인.
2. profil 사전 파일을 제거함.

3.2.5. LIB_0005 : 사용자 사전 파일 생성

케이스 ID /케이스명	LIB_005 : 사용자 사전 파일 생성	시스템 명	검증 라이브러리
입력데이터 / 테스트 방법 / 수행 결과			
<p>1. 입력 데이터</p> <p>개인정보의 내용이 포함하고 있는 파일을 pwdverif 실행 파일이 있는 경로에 생성한다.</p> <p>[yjkim] 내용 yjkim enflenfl apple 75-3 337-13 Kimyoungjae iphone</p>			

energy

test123

2828

2. 테스트 절차

- 1). Select Number 5 을 입력 입력한다.
- 2). Dictionary Name yjkim 을 입력한다.
- 3). Dictionary PREFIX Name ./dicts/yjkim 을 입력한다.

Password Safety Verification Lib Test

1. Env Load
2. Verify Password
3. Personal Dict Create
4. Personal Dict Delete
5. User Dict Create
6. Pull data out of dict
7. Exit

Select Number => 5

User Dict Create

Dictionary Name :yjkim

Dictionary PREFIX Name(./dicts/it_dict) :./dicts/yjkim

수행 결과

Password Safety Verification Lib Test

1. Env Load
2. Verify Password
3. Personal Dict Create
4. Personal Dict Delete
5. User Dict Create
6. Pull data out of dict
7. Exit

Select Number => 5

User Dict Create

Dictionary Name :yjkim

Dictionary PREFIX Name(./dicts/it_dict) :./dicts/yjkim
[Create Personal Dict]

Result[RET_CD=0000]

< 개인정보 사전 파일 생성 결과 값>

```
yjkim@kdhee:~/kisa/dicts$ ls
chat_dict.hwm  default_dict.pwi  enfl.pwd  it_dict.hwm  loanword_dict.pwi  profile.pwd  yjkim.hwm
chat_dict.pwd  economy_dict.hwm  enfl.pwi  it_dict.pwd  names_dict.hwm  profile.pwi  yjkim.pwd
chat_dict.pwi  economy_dict.pwd  fadwords_dict.hwm  it_dict.pwi  names_dict.pwd  stock_dict.hwm  yjkim.pwi
default_dict.hwm  economy_dict.pwi  fadwords_dict.pwd  loanword_dict.hwm  names_dict.pwi  stock_dict.pwd
default_dict.pwd  enfl.hwm  fadwords_dict.pwi  loanword_dict.pwd  profile.hwm  stock_dict.pwi
```

< yjkim 사전 파일이 생성 결과 값>

1. 정상적인 결과 리턴 값 확인.
2. yjkim 사전 파일을 생성함.

3.2.6. LIB_0006 : 사전 파일에서 데이터 추출

케이스 ID /케이스명	LIB_006 : 사전 파일에서 데이터 추출	시스템 명	검증 라이브러리
입력데이터 / 테스트 방법 / 수행 결과			
<ol style="list-style-type: none"> 1. 입력 데이터 <ol style="list-style-type: none"> 1) 추출 하고자 하는 사전 이름을 입력 한다. 2) 사전 파일의 경로를 입력 한다. 2. 테스트 절차 <ol style="list-style-type: none"> 1). Select Number 6 을 입력 입력한다. 2). Save File Name yjkim 을 입력한다. 3). Dictionary PREFIX Name ./dicts/yjkim 을 입력한다. 			

Password Safety Verification Lib Test

1. Env Load
2. Verify Password
3. Personal Dict Create
4. Personal Dict Delete
5. User Dict Create
6. Pull data out of dict
7. Exit

Select Number => 6

Pull data out of dict

Save File Name :yjkim

Dictionary Prefix Name(./dicts/it_dict) :./dicts/yjkim

수행 결과

Password Safety Verification Lib Test

1. Env Load
2. Verify Password
3. Personal Dict Create
4. Personal Dict Delete
5. User Dict Create
6. Pull data out of dict
7. Exit

Select Number => 6

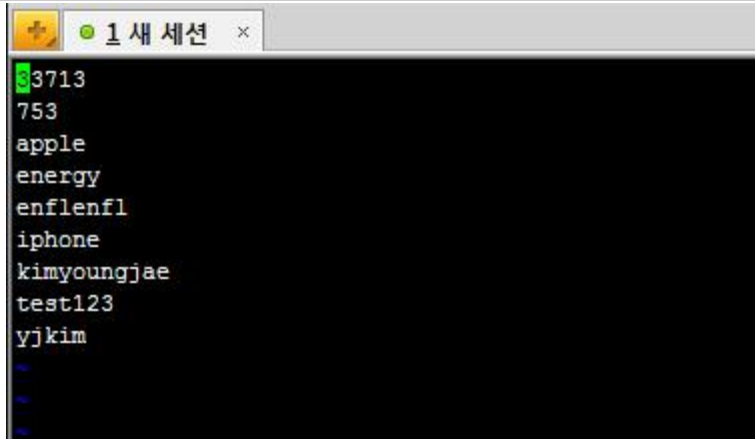
Pull data out of dict

Save File Name :yjkim

Dictionary Prefix Name(./dicts/it_dict) :./dicts/yjkim

Save Succcess

< yjkim사전 파일 추출 결과 값>

A terminal window titled "1 새 세션" (1 New Session) with a yellow icon. The terminal output shows a list of files extracted from a dictionary for the user "yjkim". The files are listed one per line: 3713, 753, apple, energy, enflenfl, iphone, kimyoungjae, test123, and yjkim. Each line is preceded by a green cursor icon. The terminal background is black, and the text is white.

```
3713
753
apple
energy
enflenfl
iphone
kimyoungjae
test123
yjkim
```

< yjkim사전 파일 추출 결과 값>

1. 정상적인 결과 리턴 값 확인.
2. yjkim 사전 파일을 생성함.

부록 1. 패스워드 검증 라이브러리 API 설명

1. 패스워드 검증 라이브러리

1.1. 인터페이스

■ API 호출 및 결과 값 반환

요청에 대한 결과 코드는 반드시 존재해야 하며 실패 시에는 에러코드를 반환한다.

■ 코드 구성

- RET_CD = 성공(0000)/실패(0000 이외 값) 반환
- 상세항목 구분은 “:” 된다.

1.2. 주요 API 및 소스 코드

패스워드 검증 라이브러리 주요 API 함수는 아래와 같다.

API 명	기능 설명
<code>char *LoadConfigFile</code>	■ 환경정보 파일을 읽고 검색할 사전종류 설정, 기본 규칙 설정, 검색 옵션 설정을 수행 한다.
<code>char *CreateDict</code>	■ 원본 사전파일을 진단 라이브러리에서 사용 가능한 사전파일(바이너리 -pwi,pwd,hwm)로 생성하는 기능을 수행한다.
<code>char *VerifyPW</code>	■ 패스워드에 대한 기본 규칙검사 및 사전정보를 수행 하고 진단 결과를 반환 한다.
<code>char *DeleteDict</code>	■ 사전 파일 삭제를 수행하고 결과를 반환 한다.

< 주요 API 목록 >

1.2.1. LoadConfigFile 함수

■ 기능

라이브러리 내 위치 한 “./conf/pwdverif.ini” 환경정보 파일을 읽어 설정된 사전 규칙, 사전 파일, 검색 옵션을 설정 한다.

■ 입력

없음

■ 결과 출력

항목	코드	값	비고
결과 코드	RET_CD	0000 또는 에러코드	
패스워드 최소길이	PW_LEN	6	
패스워드 연속문자 수	CTS_NCR	4	
문자열 반복 최소 수	REPT_NCR	4	
문자구성 수	NCHAR	4	
사전검색 사용 유무	USE_DICT	00(사용), 01(미사용)	
개인정보 사전검색	USD_PDICT	00(사용), 01(미사용)	

< 환경설정 로드 결과 값>

■ 패스워드 안정성 검증 SW 기본 환경설정

1. 사전종류 설정 (7개 설정)
2. 사전검증 규칙 설정
 - 1) 패스워드 길이 -- default - 6
 - 2) 패스워드 중복되는 문자 제외한 문자 구성 수
 - 3) 연속문자 개수 -- default - 3
 - 4) 패스워드 뒤 숫자 길이 설정(1,2) - default 2
 - 5) 패스워드 앞 숫자 길이 설정(1,2) - default 1
 - 6) 패스워드 반복문자 최소 수 - default 3
3. 사전 검색 옵션 설정 - 사용 1, 미사용 - 0
 - 1) 문자열 반복 사용 유무
 - 2) 패스워드 뒤에 숫자 붙이기 사용 유무 체크 - 사용하려면 반드시 검증규칙에 패스워드 뒤 숫자 길이 설정 되어 있어야 함
 - 3) 패스워드 앞에 숫자 붙이기 사용 유무 체크 - 사용하려면 반드시 검증규칙에 패스워드 앞 숫자 길이 설정 되어 있어야 함
 - 4) 문자열 뒤집기 사용 유무

#####

1. 사전 종류 정의

#####

[PWD_VERIF_DICT]

1. 인명사전

2. 외래어

3. 경제

4. 주식

5. it

6. 채팅어

7. 유행어

dict.file.prefix.1=./dicts/names_dict

dict.file.prefix.2=./dicts/loanword_dict

dict.file.prefix.3=./dicts/economy_dict

dict.file.prefix.4=./dicts/stock_dict

dict.file.prefix.5=./dicts/it_dict

dict.file.prefix.6=./dicts/chat_dict

dict.file.prefix.7=./dicts/fadwords_dict

#####

2. 사전검증 규칙 설정

#####

[PWD_VERIF_RULE]

패스워드 최소길이 -- default - 6

pwd_verif_rule.min_len=6

패스워드 중복되는 문자 제외한 수 구성 - 4

pwd_verif_rule.min_diff=4

패스워드 연속문자 체크 기본 값 -- default - 3

pwd_verif_rule.min_step=3

패스워드 뒤 숫자 길이 설정(1,2) - default 2(0~99), 1:(0~9)

pwd_verif_rule.postfix_num=2

패스워드 앞 숫자 길이 설정(1,2) - default 2(0~99), 1:(0~9)

pwd_verif_rule.prefix_num=2

패스워드 반복문자열 체크

pwd_verif_rule.min_rept=3

```
#####
```

```
# 3. 사전 검색 옵션 설정 - 사용 1, 미사용 - 0
```

```
# 1) 문자열 반복 사용 유무
```

```
# 2) 패스워드 뒤에 숫자 붙이기 사용 유무 체크
```

```
# 3) 패스워드 앞에 숫자 붙이기 사용 유무 체크
```

```
# 4) 문자열 뒤집기 사용 유무
```

```
#####
```

```
[PWD_VERIF_SEARCH]
```

```
# 반복문자열 사용 유무
```

```
pwd_verif_search.pw_min_rept=1
```

```
# 패스워드 뒤에 숫자 붙이기 사용 유무
```

```
pwd_verif_search.pw_postfix_num=1
```

```
# 패스워드 앞에 숫자 붙이기 사용 유무
```

```
pwd_verif_search.pw_prefix_num=1
```

```
# 문자열 뒤집기 사용 유무
```

```
pwd_verif_search.pw_reverse=1
```

```
#####
```

```
# 4. 사전파일 생성시 원본 삭제 유무 - 삭제 1, 미삭제 - 0
```

```
#####
```

```
[PWD_VERIF_ETC]
```

```
pwd_create.orgfile.delete=1
```

■ 소스코드

```
CRACKLIB_WIN32_API
char *LoadConfigFile()
{
    static char result[STRINGSIZE];
    PWDICT *pwp;
    int16 i;

    cfg = gcfg_create(5,3);

    if( gcfg_load(cfg, "./conf/pwdverif.ini") )
    {
        return "RET_CD=1001";    //1001, config file not found
    }
}
```

```

}

dict[0] = DEFAULT_CRACKLIB_DICT;

if (!(pwp = PWOOpen (dict[0], "rb")))
{
    return "RET_CD=1002"; //default dict not found
}

PWCclose(pwp);

dict[1] = gcfg_get(cfg, "PWD_VERIF_DICT", "dict.file.prefix.1", "");
dict[2] = gcfg_get(cfg, "PWD_VERIF_DICT", "dict.file.prefix.2", "");
dict[3] = gcfg_get(cfg, "PWD_VERIF_DICT", "dict.file.prefix.3", "");
dict[4] = gcfg_get(cfg, "PWD_VERIF_DICT", "dict.file.prefix.4", "");
dict[5] = gcfg_get(cfg, "PWD_VERIF_DICT", "dict.file.prefix.5", "");
dict[6] = gcfg_get(cfg, "PWD_VERIF_DICT", "dict.file.prefix.6", "");
dict[7] = gcfg_get(cfg, "PWD_VERIF_DICT", "dict.file.prefix.7", "");

USE_DICT_CNT = 1;

for (i = 1; i < DICT_CNT; i++)
{
    if (strlen(dict[i]))
    {
        if (!(pwp = PWOOpen (dict[i], "rb")))
        {
            return "RET_CD=1003"; //default dict not found
        }

        PWCclose(pwp);

        USE_DICT_CNT++;
    }
}

MINLEN = gcfg_getInt(cfg, "PWD_VERIF_RULE", "pwd_verif_rule.min_len", 6);
MINDIFF = gcfg_getInt(cfg, "PWD_VERIF_RULE", "pwd_verif_rule.min_diff", 4);
MINSTEP = gcfg_getInt(cfg, "PWD_VERIF_RULE", "pwd_verif_rule.min_step", 3);

POSTFIXNUM = gcfg_getInt(cfg,
    "PWD_VERIF_RULE", "pwd_verif_rule.postfix_num", 1);
PREFIXNUM = gcfg_getInt(cfg,
    "PWD_VERIF_RULE", "pwd_verif_rule.prefix_num", 0);

MINREPT = gcfg_getInt(cfg, "PWD_VERIF_RULE", "pwd_verif_rule.min_rept", 0);

```

```

USEPOSTFIXNUM = gcfg_getInt(cfg, "PWD_VERIF_SEARCH",
    "pwd_verif_search.pw_postfix_num", 1);
USEPREFIXNUM = gcfg_getInt(cfg, "PWD_VERIF_SEARCH",
    "pwd_verif_search.pw_prefix_num", 0);
USEREVERSE = gcfg_getInt(cfg, "PWD_VERIF_SEARCH",
    "pwd_verif_search.pw_reverse", 0);
USEMINREPT = gcfg_getInt(cfg, "PWD_VERIF_SEARCH",
    "pwd_verif_search.pw_min_rept", 0);
DELOGRFILE = gcfg_getInt(cfg, "PWD_VERIF_ETC",
    "pwd_create.orgfile.delete", 0);

//반복문자열 사용 유무 -- 사용하지 않을 경우 0으로 설정
if (!USEMINREPT)
{
    MINREPT = 0;
}

//-----
// 검색옵션 설정
//-----
if (USEPOSTFIXNUM)
{
    SetPostFixCnt(POSTFIXNUM);
}

if (USEPREFIXNUM)
{
    SetPreFixCnt(PREFIXNUM);
}

if (USEREVERSE)
{
    SetUseReverse(1);
}

sprintf(result, "RET_CD=0000:PW_LEN=%d:CTS_NCR=%d:
    REPT_NCR=%d:NCHAR=%d:USE_DICT_CNT=%d:", MINLEN,
    MINSTEP, MINREPT, 4, USE_DICT_CNT);

if (pwp = PWOpen (PROFILE_DICT, "rb"))
{
    USEPROFILE = 1;
    strcat(result, "USE_PDICT=01");
    PWCclose(pwp);
}

```

```

        else
        {
            strcat(result, "USE_PDICT=00");
        }

        return result;
    }

```

■ 사전 종류 추가 방법

1. pwdverif.ini 파일 에 dict.file.prefix.X 을 추가한다.
2. pwanalysis.c 파일 에 DICT_CNT를 사용할 사전만큼 수정 한다.

1.2.2. CreateDict 함수

■ 기능

일반 텍스트 모드의 사전 파일을 검증 라이브러리 내에서 사용하는 사전 파일로 구성하는 역할을 수행한다. 사전 파일로 구성 되면 인덱스를 이용한 빠른 검색을 할 수 있다는 장점이 있다. 단, 기존에 사전이 오픈되어 있을 경우 생성 불가능 하다.

- 생성 사전 파일 -

- *.pwi : 패스워드 사전 파일의 인덱스.
- *.pwd : 패스워드 사전 파일의 데이터.
- *.hwm : 패스워드 사전 파일의 사이즈.

■ 입력

char *filename : 텍스트 모드의 사전 파일 이름

char *dict_prefix : 생성될 사전의 prefix 이름

■ 소스코드

```

CRACKLIB_WIN32_API
char *CreateDict(char *filename, char * dict_prefix)
{
    PWDICT *pwp;
    char buffer[STRINGSIZE];

```

```

long lCount = 0;
FILE *file;
char *fname = "make.txt";
FILE *tmpfile, *orgfile;
int ret = 0;

if( ret = SortFile(filename, fname) )
{
    if(ret == 1)
        return "RET_CD=3001";
    else
        return "RET_CD=3002";
}
else
{
    if((file = fopen(fname, "r"))==NULL)
    {
        return "RET_CD=3003";
    }
    else
    {
        if (!(pwp = PWOpen(dict_prefix, "wb")))
        {
            return "RET_CD=3004";
        }

        while (fgets(buffer, STRINGSIZE, file) != NULL)
        {
            buffer[STRINGSIZE - 1] = '\0';
            lCount++;

            Chop(buffer);

            if (!buffer[0]) continue;

            PutPW(pwp, buffer);

```

```

        }
        fclose(file);
        PWCclose(pwp);
    }
}
USEPROFILE = 1;

if (tmpfile = fopen(fname, "r"))
{
    fclose(tmpfile);
    remove(fname);
}

if (DELORGFIL)
{
    if (orgfile = fopen(filename, "r"))
    {
        fclose(orgfile);
        remove(filename);
    }
}
return "RET_CD=0000";
}

```

■ 출력

항목	코드	값	비고
결과 코드	RET_CD	0000 또는 에러코드	

<사전생성 결과 값>

1.2.3. VerifyPW 함수

■ 기능

패스워드에 대해 기본 규칙검사 및 사전정보를 수행하고 진단 결과를 리턴 한다.

- 패스워드 검증 방법 -

1. 입력된 패스워드 길이 체크.
2. 문자 구성 수 체크.
3. 1,2 조합에 해당되는 기본 값 조회
4. 입력된 패스워드 반복 문자열 체크 - 설정 값보다 작을 시 기본점수에서 차감.
5. 입력된 패스워드 연속문자 체크 - 설정 값보다 작을 시 기본점수에서 차감.
6. 사전 검색
설정된 사전종류 만큼 검색 - 발견 시 나머지 사전 검색 중지.
사전에서 검색되었을 때 기본점수에서 차감.
7. 개인정보 사전검색
개인정보 사전이 있을 경우 사전 검색 - 사전 검색되었을 때 기본점수에서 차감.
8. 최종 등급 판정.

■ 입력

`char *pwd` : 검증 하고자 하는 패스워드

■ 소스코드

```
CRACKLIB_WIN32_API
char *VerifyPW(char * instring)
{
    int32 i, fdCount;
    char *ptr;
    char *jptr;
    char junk[STRINGSIZE];
    char *password;
    char rpassword[STRINGSIZE];
```

```

int nTotalSum = 0;
int nCharacters = 0;
int LEVEL = 0;
char ret_code[STRINGSIZE] = {0x00};
char result[STRINGSIZE];
int16 bNum = 0, bAlpha1=0, bAlpha2=0, bOther=0, repcnt = 0;
char strTmp[TRUNCSTRINGSIZE] = {0x00};

if (cfg == NULL)
    return "RET_CD=2001";

if (strlen(instring)-1 > TRUNCSTRINGSIZE)
    return "RET_CD=2000"; //password's size is too big

memset(rpassword, 0x00, STRINGSIZE);
strncpy(rpassword, instring, TRUNCSTRINGSIZE);

password = rpassword;

if (strlen(password) != 0)
{
    for (i = 0; i < strlen(password); i++)
    {
        if ((bNum != 1) && strchr(numberal, password[i]))
        {
            bNum = 1;
            nCharacters++;
        }
        if ((bAlpha1 != 1) && strchr(lowercase, password[i]))
        {
            bAlpha1 = 1;
            nCharacters++;
        }
        if ((bAlpha2 != 1) && strchr(uppercase, password[i]))
        {
            bAlpha2 = 1;

```

```

        nCharacters++;
    }
    if ((bOther != 1) && strchr(symbol, password[i]))
    {
        bOther = 1;
        nCharacters++;
    }
}

i = (int32)strlen(password);

if (i > 50)
{
    i = 49;
}

nTotalSum = nLevel[i][nCharacters-1];

/*-----
// 1. 입력된 패스워드 길이 체크
// default(6) - 6자리 미만의 패스워드
-----*/
snprintf(strTmp,sizeof(strTmp), "PW_LEN=%d", strlen(password));
strTmp[sizeof(strTmp)-1] = '\\0';
strncat(ret_code, strTmp, sizeof(ret_code)-1);

/*-----
// 2. 입력된 패스워드 문자수
// 중복문자 필터한 수, 대소구분
// aaABO - 4 --- 사용의미 없음
-----*/
jptr = junk;
*jptr = '\\0';

for (i = 0; i < STRINGSIZE && password[i]; i++)

```

```

{
    if (!strchr(junk, password[i]))
    {
        *(jptr++) = password[i];
        *jptr = '\0';
    }
}

if (strlen(junk) < (unsigned)MINDIFF)
{
    nTotalSum--;
}

/*-----
// 2-1. 입력된 패스워드 문자구성수
// 대/소/특수기호/숫자
-----*/
snprintf(strTmp, sizeof(strTmp), ":PW_NCR=%d", nCharacters);
strTmp[sizeof(strTmp)-1] = '\0';

if (strlen(ret_code)+strlen(strTmp) > sizeof(ret_code))
    return "RET_CD=2009";

strncat(ret_code, strTmp, sizeof(ret_code)-1);

strcpy(password, Lowercase(password));

Trim(password);

/*-----
// 3. 입력된 패스워드 반복문자열 체크
//
-----*/
if (USEMINREPT)
{
    repcnt = RepeteChar(password, 2); //기본 2

```

```

        if (repcnt > MINREPT)
            nTotalSum--;

        snprintf(strTmp, sizeof(strTmp), ":REPT_NCR=%d", repcnt);
    }
    else
    {
        snprintf(strTmp, sizeof(strTmp), ":REPT_NCR=%d", 0);
    }

    strTmp[sizeof(strTmp)-1] = '\0';
    strncat(ret_code, strTmp, sizeof(ret_code)-1);

    /*-----
    // 4. 입력된 패스워드 연속문자
    //
    -----*/
    i = 0;
    ptr = password;
    while (ptr[0] && ptr[1])
    {
        if ((ptr[1] == (ptr[0] + 1)) || (ptr[1] == (ptr[0] - 1)))
        {
            i++;
        }
        ptr++;
    }

    // 연속문자 결과 셋팅
    snprintf(strTmp, sizeof(strTmp), ":CTS_NCR=%d", i);
    strTmp[sizeof(strTmp)-1] = '\0';
    strncat(ret_code, strTmp, sizeof(ret_code)-1);

    if (i > (unsigned)MINSTEP)
        nTotalSum--;

```

```

/*-----
// 5. 사전검색
//   설정된 사전종류만큼 비교.
-----*/
fdCount = 0;

for (i = 0; i < DICT_CNT; i++)
{
    char *found;

    if (!strlen(dict[i]))
        continue;

    found = DictFindPW(password, dict[i]);

    fdCount++;

    if (found)
    {
        break;
    }
}

if (fdCount == DICT_CNT)
{
    fdCount++;
    snprintf(strTmp, sizeof(strTmp), ":DICT_FD_CNT=%d", fdCount);
}
else
{
    nTotalSum--;
    snprintf(strTmp, sizeof(strTmp), ":DICT_FD_CNT=%d", fdCount);
}

strTmp[sizeof(strTmp)-1] = '\0';

```

```

strncat(ret_code, strTmp, sizeof(ret_code)-1);

/*-----
// 6. 개인정보 사전검색
// (사전에 검출 결과 00:없다,01:있다, xx 사용안함)
-----*/
if (USEPROFILE)
{
    char *find;

    printf ("user profile dict filename %s\n", PROFILE_DICT);

    find = DictFindPW(password, PROFILE_DICT);

    if (find) // found dict
    {
        nTotalSum--;
        strcat(ret_code, ":PDICT_FD=01");

        printf("profile dict filename %s, %s\n", PROFILE_DICT, find);
    }
    else
    {
        //개인정보 사전 검색이 안되었을 때 사전파일에서 직접 비교
        if (DictFullSearch(password, PROFILE_DICT) > -1)
        {
            strcat(ret_code, ":PDICT_FD=00"); // not found
        }
        else
        {
            strcat(ret_code, ":PDICT_FD=01");
        }
    }
}
else
{

```

```

        strcat(ret_code, ":PDICT_FD=XX"); //개인정보 없을 때 기본값(00)
    }

    /*-----
    // 7. 결과(0000) 4자리
    //  -- 성공 0000,
    //  -- 실패 에러코드
    -----*/

    if (nTotalSum >= nRange[3][0] && nTotalSum <= nRange[3][1])
        LEVEL = 4; //" HIGHEST"
    else if (nTotalSum >= nRange[2][0] && nTotalSum <= nRange[2][1])
        LEVEL = 3; //" HIGH"
    else if (nTotalSum >= nRange[1][0] && nTotalSum <= nRange[1][1])
        LEVEL = 2; //" MEDIUM"
    else if (nTotalSum >= nRange[0][0] && nTotalSum <= nRange[0][1])
        LEVEL = 1; //" LOW"
    else
        LEVEL = 1; //" LOW"

    snprintf(strTmp, sizeof(strTmp), "RET_CD=0000:RT_LEVEL=%d:", LEVEL);
    strTmp[sizeof(strTmp)-1] = '\0';

    strncpy(result, strTmp, sizeof(ret_code)-1);

    if (sizeof(result) < strlen(ret_code) + strlen(result))
        return "RET_CD=2009";
    else
        return strcat(result, ret_code);
}

```


■ 출력

항목	코드	값	비고
결과 코드	RET_CD	0000 또는 에러코드	
진단결과 등급	RT_LEVEL	1: 하 등급 2: 중 등급 3: 상 등급 4: 최 상급	
요청 비밀번호 최소길이	PW_LEN	6	
비밀번호 문자 수	PW_NCR	4	
문자열 반복 수	REPT_NCR	4	
연속문자 수	CTS_NCR	4	
사전검색 사용 유무	DICT_FD	00(사용), 01(미사용)	
개인정보 사전검색	PDICT_FD	00(사용), 01(미사용)	

< 비밀번호 진단 결과 값 >

1.2.4. DeleteDict 함수

■ 기능

사전 파일 삭제를 수행하고 결과를 리턴 한다.

■ 입력

char *dict_prefix : 삭제 하고자 하는 사전파일 prefix 이름

■ 소스코드

```
CRACKLIB_WIN32_API char *DeleteDict(char *prefix)
{
    FILE *dfp;
    FILE *ifp;
    FILE *wfp;

    char iname[STRINGSIZE];
    char dname[STRINGSIZE];
```

```

char wname[STRINGSIZE];

snprintf(iname, sizeof(iname)-1, "%s.pwi", prefix);
iname[sizeof(iname)-1] = '\\0';
snprintf(dname, sizeof(dname), "%s.pwd", prefix);
dname[sizeof(dname)-1] = '\\0';
snprintf(wname, sizeof(wname), "%s.hwm", prefix);
wname[sizeof(wname)-1] = '\\0';

if (dfp = fopen(dname, "r"))
{
    fclose(dfp);
    remove(dname);
}

if (ifp = fopen(iname, "r"))
{
    fclose(ifp);
    remove(iname);
}

if (wfp = fopen(wname, "r"))
{
    fclose(wfp);
    remove(wname);
}

USEPROFILE = 0;

return "RET_CD=0000";
}

```

■ 출력

항목	코드	값	비고
결과 코드	RET_CD	0000 또는 에러코드	

< 사전 삭제 결과 값>

1.3. 기타 함수

앞서 소개한 함수들 외에도 다음과 같은 함수들이 사용되고 있다.

1.3.1. gcfg_create

■ 기능

gconfig_t 구조 체를 생성한다.

■ 입력

int capacity : 초기에 생성할 gconfig_t의 필드의 수

int increment : gconfig_t에 최대 수용 필드 수에 도달했을 경우 증가시킬 필드 수

■ 사용 구조체

```

/*
    Dynamic List 구현 구조체
*/
struct _GLIST_ST
{
    void **array;
    int idx;
    int capacity;
    int increment;
};

/**

```

Dynamic List 구현 구조체에 대한 자료형 정의

```
*/  
typedef struct _GLIST_ST glist_t;  
  
struct _GTABLE_ST  
{  
    glist_t *list;  
};  
typedef struct _GTABLE_ST gtable_t;
```

■ 소스코드

```
gconfig_t*  
gcfg_create(int capacity, int increment)  
{  
    gconfig_t *cfg = NULL;  
  
    cfg = malloc(sizeof(gconfig_t));  
    if (cfg == NULL)  
        return NULL;  
  
    cfg->list = glist_create(capacity, increment);  
  
    return cfg;  
}
```

■ 출력

생성된 gconfig_t 구조체

1.3.2. gcfg_load

■ 기능

경로에 지정된 파일로부터 데이터를 불러들인다.

■ 입력

gconfig_t* cfg : 불러들일 gconfig 구조체 포인터

char* path : 불러들일 파일 경로

■ 소스코드

```
int gcfg_load(gconfig_t* cfg, char* path)
{
    char buff[MAX_GCONFIG_LINE+1];
        char sectName[MAX_SECT_NAME_LEN+1];
    char* name;
    char* value;
    char* ptr;

    gconfigsect_t *cursect = NULL;

    int len;
    int wrapFlag = 0;
    FILE* fp;

    if (cfg == NULL)
        return -1;

    fp = fopen(path, "rt");
    if (fp == NULL)
    {
        return -1;
    }

    while(!feof(fp) && fgets(buff, sizeof(buff), fp) != NULL)
```

```

{
    if (buff[0] == '#' || (!wrapFlag && giswhitespace(buff[0])))
    {
        continue;
    }

    if (buff[0] == '[')
    {
        strncpy(sectName, buff, MAX_SECT_NAME_LEN);
        gtrim(sectName);
        if (sectName[strlen(sectName)-1] == ']')
        {
            memmove(sectName, &sectName[1],
                    strlen(sectName)-1);
            sectName[strlen(sectName)-2] = '\\0';
            gtrim(sectName);
            cursect = gcfg_findsection(cfg, sectName);
            if (cursect == NULL)
            {
                cursect = gconfigsect_create(sectName);
                gcfg_putsection(cfg, cursect);
            }
        }
    }

    if (wrapFlag)
    {
        if (giswhitespace(buff[0]))
        {
            gtrim(buff);

            if ((len = (int)strlen(buff)) > 0)
            {
                value = (char*)realloc(value, strlen(value)+len+1);
                strcat(value, buff);
            }
        }
    }
}

```

```

        if (value[strlen(value)-1] == '\\')
            value[strlen(value)-1] = '\0';
    }

    if (len == 0 || buff[len-1] != '\\')
    {
        wrapFlag = 0;
        if (cursect != NULL && cursect->table != NULL)
        {
            gtbl_put(cursect->table, name, value);
        }
    }
    continue;
}
else
{
    wrapFlag = 0;
}
}

if ((ptr = (char*)strchr(buff, '=')) == NULL)
    continue;

len = (int)strlen(buff)-(int)strlen(ptr)+1;
name = malloc(len+1);
strncpy(name, buff, len);
name[len-1] = '\0';
grtrim(name);

len = (int)strlen(buff)-(int)strlen(name)-2;
value = malloc(len+1);
strncpy(value, ptr+1, len);
value[len] = '\0';
grtrim(value);

if (strlen(value) >= 1 && value[strlen(value)-1] == '\\')

```

```

        {
            value[strlen(value)-1] = '\\0';
            wrapFlag = 1;
        }
        else
        {
            if (cursect != NULL && cursect->table != NULL)
            {
                gtbl_put(cursect->table, name, value);
            }
        }
    }
    fclose(fp);

    return 0;
}

```

■ 출력

성공 0을 출력 , 실패 -1 을 출력한다.

1.3.3. PWOpen

■ 기능

라이브러리에서 사용할 사전파일을 열기위한 함수로, 사전 파일은 총 3개로 구성되어 있으며, 확장자만 다르게 구분되어 있다.

■ 입력

`char *prefix` : 사전파일의 접두어

`char *mode` : 사전파일을 열 때의 옵션

■ 소스코드

```

PWDict *
PWOpen(char *prefix, char *mode)

```



```

{
    int use64 = 0;
    static PWDICT pdesc;
    static PWDICT64 pdesc64;
    char iname[STRINGSIZE];
    char dname[STRINGSIZE];
    char wname[STRINGSIZE];
    FILE *dfp;
    FILE *ifp;
    FILE *wfp;

    if (pdesc.header.pih_magic == PIH_MAGIC)
    {
        fprintf(stderr, "%s: another dictionary already open,
                     force closed \n", prefix);
        return ((PWDICT *) 0);
    }

    memset(&pdesc, '\0', sizeof(pdesc));
    memset(&pdesc64, '\0', sizeof(pdesc64));

    sprintf(iname, "%s.pwi", prefix);
    sprintf(dname, "%s.pwd", prefix);
    sprintf(wname, "%s.hwm", prefix);

    if (!(pdesc.dfp = fopen(dname, mode)))
    {
        return ((PWDICT *) 0);
    }

    if (!(pdesc.ifp = fopen(iname, mode)))
    {
        fclose(pdesc.dfp);
        perror(iname);
        return ((PWDICT *) 0);
    }
}

```

```

if (pdesc.wfp = fopen(wname, mode))
{
    pdesc.flags |= PFOR_USEHWMS;
}

ifp = pdesc.ifp;
dfp = pdesc.dfp;
wfp = pdesc.wfp;

if (mode[0] == 'w')
{
    pdesc.flags |= PFOR_WRITE;
    pdesc.header.pih_magic = PIH_MAGIC;
    pdesc.header.pih_blocklen = NUMWORDS;
    pdesc.header.pih_numwords = 0;

    fwrite((char *) &pdesc.header, sizeof(pdsc.header), 1, ifp);
} else
{
    pdesc.flags &= ~PFOR_WRITE;

    if (!fread((char *) &pdesc.header, sizeof(pdsc.header), 1, ifp))
    {
        fprintf(stderr, "%s: error reading header\n", prefix);

        pdesc.header.pih_magic = 0;
        fclose(ifp);
        fclose(dfp);
        if(wfp)
            fclose(wfp);
        return ((PWDICT *) 0);
    }

    if ((pdsc.header.pih_magic == 0) || (pdsc.header.pih_numwords == 0))
    {

```

```

/* uh-oh. either a broken "64-bit" file or a garbage file. */
rewind (ifp);
if (!fread((char *) &pdsc64.header, sizeof(pdsc64.header), 1, ifp))
{
    fprintf(stderr, "%s: error reading header\n", prefix);
    pdsc.header.pih_magic = 0;
    fclose(ifp);
    fclose(dfp);
    if(wfp)
    {
        fclose(wfp);
    }
    return ((PWDICT *) 0);
}
if (pdsc64.header.pih_magic != PIH_MAGIC)
{
    /* nope, not "64-bit" after all */
    fprintf(stderr, "%s: error reading header\n", prefix);

    pdsc.header.pih_magic = 0;
    fclose(ifp);
    fclose(dfp);
    if(wfp)
    {
        fclose(wfp);
    }
    return ((PWDICT *) 0);
}

pdsc.header.pih_magic = (int32)pdsc64.header.pih_magic;
pdsc.header.pih_numwords = (int32)pdsc64.header.pih_numwords;
pdsc.header.pih_blocklen = (int32)pdsc64.header.pih_blocklen;
pdsc.header.pih_pad = (int32)pdsc64.header.pih_pad;
use64 = 1;
}

```

```

if (pdesc.header.pih_magic != PIH_MAGIC)
{
    fprintf(stderr, "%s: magic mismatch\n", prefix);

    pdesc.header.pih_magic = 0;
    fclose(ifp);
    fclose(dfp);
    if(wfp)
    {
        fclose(wfp);
    }
    return ((PWDICT *) 0);
}

if (pdesc.header.pih_numwords < 1)
{
    fprintf(stderr, "%s: invalid word count\n", prefix);

    pdesc.header.pih_magic = 0;
    fclose(ifp);
    fclose(dfp);
    if(wfp)
    {
        fclose(wfp);
    }
    return ((PWDICT *) 0);
}

if (pdesc.header.pih_blocklen != NUMWORDS)
{
    fprintf(stderr, "%s: size mismatch\n", prefix);

    pdesc.header.pih_magic = 0;
    fclose(ifp);
    fclose(dfp);

```

```

        if(wfp)
        {
            fclose(wfp);
        }
        return ((PWDICT *) 0);
    }

    if (pdesc.flags & PFOR_USEHWMS)
    {
        int i;

        if (use64)
        {
            if (fread(pdesc64.hwms, 1, sizeof(pdesc64.hwms), wfp)
                != sizeof(pdesc64.hwms))
            {
                pdesc.flags &= ~PFOR_USEHWMS;
            }
            for (i = 0; i < sizeof(pdesc.hwms)
                / sizeof(pdesc.hwms[0]); i++)
            {
                pdesc.hwms[i] = (int32)pdesc64.hwms[i];
            }
        }
        else if (fread(pdesc.hwms, 1, sizeof(pdesc.hwms), wfp)
            != sizeof(pdesc.hwms))
        {
            pdesc.flags &= ~PFOR_USEHWMS;
        }
    }

    }

    return (&pdesc);
}

```

■ 출력

사전 파일을 가리키는 포인터

1.3.4. PWClose

■ 기능

사전파일을 가리키는 포인터를 입력 받아 PWOOpen 함수로 열었던 사전 파일을 닫는 함수이다.

■ 입력

PWDICT *pwp : 사전파일

■ 소스코드

```
int PWClose(PWDICT *pwp)
{
    if (pwp->header.pih_magic != PIH_MAGIC)
    {
        return (-1);
    }

    if (pwp->flags & PFOR_WRITE)
    {
        pwp->flags |= PFOR_FLUSH;
        PutPW(pwp, (char *) 0);    /* flush last index if necess */

        if (fseek(pwp->ifp, 0L, 0)) //-----data size info file
        {
            return (-1);
        }

        if (!fwrite((char *) &pwp->header, sizeof(pwp->header), 1,
                    pwp->ifp)) //--- data header save
        {
```

```

        return (-1);
    }

    if (pwp->flags & PFOR_USEHWMS)
    {
        int i;
        for (i=1; i<=0xff; i++)
        {
            if (!pwp->hwms[i])
            {
                pwp->hwms[i] = pwp->hwms[i-1];
            }
        }
        fwrite(pwp->hwms, 1, sizeof(pwp->hwms), pwp->wfp);
    }
}

fclose(pwp->ifp); //data size info
fclose(pwp->dfp); //data info
if(pwp->wfp)
{
    fclose(pwp->wfp); //index info
}

pwp->header.pih_magic = 0;

return (0);
}

```

■ 출력

파일을 닫은 결과 값 (0 : 정상 , -1 : 오류)

1.3.5. FinPW

■ 기능

입력받은 패스워드를 PWOpen으로 열었던 사전파일에서 찾는 기능을 하는 함수이다. 사전 파일에 포함된 단어의 수를 바이너리 검색하여 패스워드가 사전에 포함 되었는지를 검사한다.

■ 입력

PWDICT* pwp : 사전파일

char* string : 검증할 패스워드

■ 소스코드

```
int32 FindPW(PWDICT *pwp, char *string)
{
    register int32 lwm;
    register int32 hwm;
    register int32 middle;
    register char *This;
    int idx;

    if (pwp->flags & PFOR_USEHWMS)
    {
        idx = string[0] & 0xff;
        lwm = idx ? pwp->hwms[idx - 1] : 0;
        hwm = pwp->hwms[idx];
    }
    else
    {
        lwm = 0;
        hwm = PW_WORDS(pwp) - 1;
    }
}
```



```

/* if the high water mark is lower than the low water mark,
   something is screwed up */
if ( hwm < lwm )
{
    lwm = 0;
    hwm = PW_WORDS(pwp) - 1;
}

for (;;)
{
    int cmp;

    middle = lwm + ((hwm - lwm + 1) / 2);

    This = GetPW(pwp, middle);
    if ( ! This )
    {
        return(PW_WORDS(pwp));
    }

    cmp = strcmp(string, This);
    if (cmp == 0)
    {
        return(middle);
    }

    if (middle == hwm)
    {
        break;
    }

    if (cmp < 0)
    {
        hwm = middle;
    }
}

```

```

    }
    else if (cmp > 0)
    {
        lwm = middle;
    }
}
return (PW_WORDS(pwp));
}

```

■ 출력

사전파일 중 입력한 패스워드 위치를 반환한다.

1.3.6. GetPW

■ 기능

FindPW()에서 예상되는 사전파일의 위치를 입력하여 GetPW()함수를 호출하면, GetPW()함수는 인덱스 파일을 읽어 들이고 인덱스 파일의 값에 따라 실제 사전 파일에서 데이터를 가져온 뒤 입력된 위치에 맞는 텍스트를 리턴 한다.

■ 입력

PWDICT* pwp : 사전파일

int32 number : 사전파일 중 입력한 패스워드의 예상 위치

■ 소스코드

```

char *
GetPW(PWDICT *pwp, int32 number)
{
    int32 datum;
    register int i;
    register char *ostr;
    register char *nstr;
    register char *bptr;

```

```

char buffer[NUMWORDS * MAXWORDLEN];
static char data[NUMWORDS][MAXWORDLEN];
static int32 prevblock = 0xffffffff;
int32 thisblock;
static prevnumber = 0;

thisblock = number / NUMWORDS;

if(_PWIsBroken64(pwp->ifp))
{
    int64 datum64;
    if (fseek(pwp->ifp, sizeof(struct pi_header64) +
        (thisblock * sizeof(int64)), 0))
    {
        return ((char *) 0);
    }

    if (!fread((char *) &datum64, sizeof(datum64), 1, pwp->ifp))
    {
        return ((char *) 0);
    }
    datum = (int32)datum64;
}
else
{
    if (fseek(pwp->ifp, sizeof(struct pi_header) +
        (thisblock * sizeof(int32)), 0))
    {
        return ((char *) 0);
    }

    //-----
    if (!fread((char *) &datum, sizeof(datum), 1, pwp->ifp))
    {
        return ((char *) 0);
    }
}

```

```

}

//-----
if (fseek(pwp->dfp, datum, 0))
{
    return ((char *) 0);
}

if (!fread(buffer, 1, sizeof(buffer), pwp->dfp))
{
    return ((char *) 0);
}

prevblock = thisblock;
bptr = buffer;
for (ostr = data[0]; *(ostr++) = *(bptr++)); /* nothing */
ostr = data[0];

for (i = 1; i < NUMWORDS; i++)
{
    nstr = data[i];
    strcpy(nstr, ostr);

    if (strlen(nstr) == 0) continue;

    ostr = nstr + *(bptr++);

    while ((*ostr++) = *(bptr++));

    ostr = nstr;
}

return (data[number % NUMWORDS]);
}

```

■ 출력

사전파일 중 입력한 패스워드 위치의 텍스트

1.3.7. PutPW

■ 기능

PWDICT 사전 파일에 입력된 string을 삽입하고, PWDICT 사전 파일의 관련 정보를 갱신하는 역할을 수행하는 함수이다.

■ 입력

PWDICT* pwp : 사전파일

char* string : 사전파일에 삽입할 텍스트

■ 소스코드

```
int PutPW(PWDICT *pwp, char *string)
{
    if (!(pwp->flags & PFOR_WRITE))
    {
        return (-1);
    }

    if (string)
    {
        strncpy(pwp->data[pwp->count], string, MAXWORDLEN);
        pwp->data[pwp->count][MAXWORDLEN - 1] = '\0';

        pwp->hwms[string[0] & 0xff]= pwp->header.pih_numwords;

        ++(pwp->count);
        ++(pwp->header.pih_numwords);
    }
    else if (!(pwp->flags & PFOR_FLUSH))
```

```

{
    return (-1);
}

if ((pwp->flags & PFOR_FLUSH) || !(pwp->count % NUMWORDS))
{
    int i;
    int32 datum;
    register char *ostr;
    int32 total = 0;
    int n = 0;

    datum = (int32) ftell(pwp->dfp);
    //----- dtp data size 조회

    fwrite((char *) &datum, sizeof(datum), 1, pwp->ifp);
    //----- ifp에 데이터 크기 저장

    if (fputs(pwp->data[0], pwp->dfp) < 0) {
        return (-1);
    }

    if (putc(0, pwp->dfp) < 0) {
        return (-1);
    }

    ostr = pwp->data[0];

    for (i = 1; i < NUMWORDS; i++)
    {
        register int j;
        register char *nstr;
        nstr = pwp->data[i];

        if (nstr[0])
        {

```

```

        for(j= 0; ostr[j] && nstr[j] && (ostr[j] == nstr[j]); j++);

        if ( putc(j & 0xff, pwp->dfp) < 0) {
            return (-1);
        }

        if( fputs(nstr + j, pwp->dfp) < 0) {
            return (-1);
        }
    }
    putc(0, pwp->dfp);
    ostr = nstr;
}

memset(pwp->data, '\0', sizeof(pwp->data));
pwp->count = 0;
}
return (0);
}

```

■ 출력

사전파일에 텍스트를 삽입한 결과 (0: 정상 , -1 : 오류)

1.3.8. _PWIsBroken64

■ 기능

패스워드 검증 라이브러리에서 사용하는 PWDICT 변수가 64비트 인지 체크하는 함수이다.

■ 입력

FILE *ifp : 인덱스 파일을 가리키는 파일 포인터

■ 소스코드

```
static int _PWIsBroken64(FILE *ifp)
{
    PWDICT64 pdesc64;
    rewind(ifp);
    if (!fread((char *) &pdesc64.header, sizeof(pdesc64.header), 1, ifp))
    {
        return 0;
    }

    return (pdesc64.header.pih_magic == PIH_MAGIC);
}
```

■ 출력

64비트 PWDICT 여부 (1 : 64비트 , 0 : 32비트)

부록 2. 장애 관리

1. 장애관리

요청에 대한 결과 성공(0000)외 에러코드 정의는 다음과 같다.

1.1.1. 에러코드 및 대처

분류	에러 코드	설명	처리
환경설정	1001	환경설정 파일 없음	conf 폴더 pwdverf.ini 파일존재 확인, 또는 실행 경로 확인
	1002	default 사전 없음	dicts 폴더에 default.pwi, pwd,hwm 사전 파일 확인
	1003	환경설정 파일에 설정된 사전파일이 존재하지 않음	사전파일 확인 및 신규생성
패스워드 진단	2001	환경설정 정보 로드가 안 됨	패스워드 진단 함수 이전에 환경로드 함수 호출
사전 생성	3001	사용자 사전 파일 없음	등록할 사용자 사전 존재 유무 확인
	3002	사용자 사전 소팅 에러	사용자 파일 내용 오류 확인
	3003	소팅된 파일 생성 오류	사용자 파일 내용 오류 확인
	3004	사전 오픈 에러	이미 사전 파일이 오픈 되어 있는 경우

<에러코드 및 대처 >

부록 3. JDK 설치 방법

1. 윈도우 JDK 다운로드 및 설치

1.1. 다운로드

다음 주소로 들어간 후 JDK를 설치한다.

<http://www.oracle.com/us/sun/index.html>

1.2. 다운로드 방법

1) 사이트 접속후 Downloads -> Javs SE -> Download JDK선택

ORACLE® (Sign In/Register for Account | Help) United States ▾ Communities ▾ I am a... ▾ I want to... ▾ Secure Search 🔍

Products and Services Downloads Store Support Education Partners About Oracle Technology Network ▾

Downloads 선택

Java

- Java EE & GlassFish
- Java ME
- Java SE
- Java Runtime Environment (JRE)
- JavaFX
- Sun Downloads: A-Z Listing

Java SE 선택

Java Platform, Standard Edition

Java SE 6 Update 23
This release includes performance improvements and bug fixes. [Learn more ▶](#)

What Java Do I Need? You must have a copy of the JRE (Java Runtime Environment) on your system to **run** Java applications and applets. To **develop** Java applications and applets, you need the **JDK** (Java Development Kit), which includes the JRE.

	Download JDK	Download JRE
JDK 6 Docs	JRE 6 Docs	
Installation Instructions	Installation Instructions	
ReadMe	ReadMe	
ReleaseNotes	ReleaseNotes	
Oracle License	Oracle License	
Third Party Licenses	Third Party Licenses	
Supported System Configurations	Supported System Configurations	

Download JDK 선택

2) Platform 을 Windows로 선택

Provide Information, then Continue to Download

Select Platform and Language for your download:

Platform: Windows ▾


Language: Multi-language

☒ I agree to the [Java SE Development Kit 6u23 License Agreement](#).

Continue »

3) jdk-6u23-windows-i586.exe 선택 후 다운로드 실행

Download Information and Files

 **Get the latest [Java Runtime Environment](#) to use [Sun Download Manager](#)**

Internet Explorer Users: Check the top of this page for a "Java(TM) Web Start ActiveX Control" message in the information bar. If it appears, click it to finish detecting your Java version.

We were unable to detect a recent version of Java Runtime Environment (JRE) on your system. With the latest JRE, you can automatically download, install, and run [Sun Download Manager](#) (SDM) directly from this page. We highly recommend SDM to easily manage your downloads (pause, resume, restart, verify, and more). Visit [java.com](#) for the latest JRE.

Instructions: Click the file name to start the download.

Available Files

File Description and Name	Size
Java SE Development Kit 6u23	76.32 MB
 jdk-6u23-windows-i586.exe	

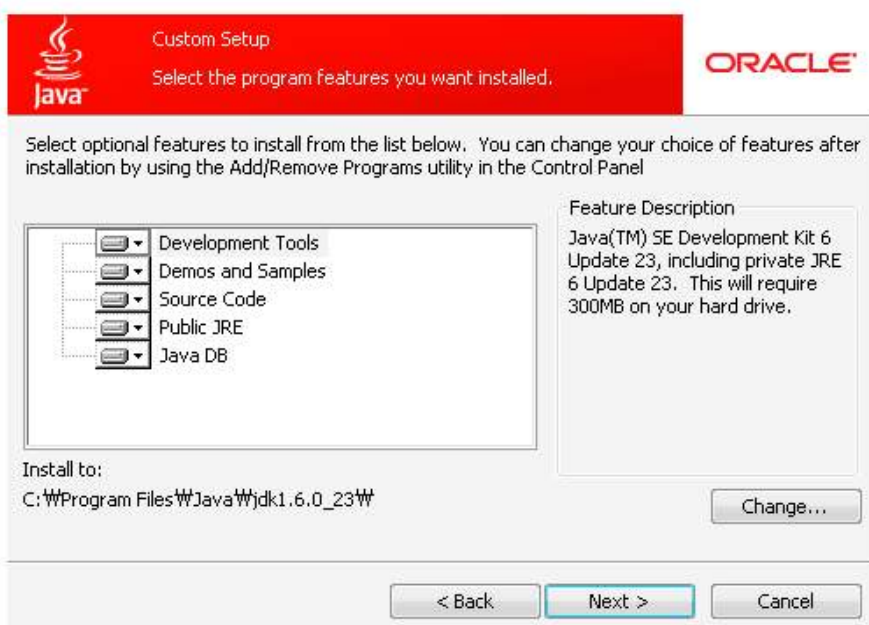
Notes:

- For download problems or questions, please see the [Download FAQs](#).
- If you logged in first, you can complete this download any time in the next 30 days. Just visit your [Download History](#).
- For Customer Service, contact [Download Customer Service](#).

1.3. 설치

1) jdk-6u23-windows-i586.exe 실행

2) "Next >" 단추 선택





3) "Finish" 단추 선택 및 설치 완료



2. 리눅스 JDK 다운로드 및 설치

JDK 설치에는 Ubuntu 12.10 Desktop 32비트 버전이며 설치 방법은 openjdk 설치 방법과 oracle-java jdk를 설치하는 방법이 있으며 다음과 같이 필요한 방법으로 선택하여 설치하면 된다.

2.1. apt-get 다운로드

1) apt-get 으로 openjdk 설치

```
$ sudo apt-get install openjdk-7-jdk
```

2) apt-get 으로 oracle-java7 jdk 설치하기

```
$ sudo add-apt-repository ppa:webupd8team/java
```

3) apt-get 으로 oracle-java7 jdk 설치하기

```
$ sudo apt-get update  
$ sudo apt-get install oracle-java7-installer
```

2.2. Java 홈페이지에서 다운로드 및 설치

1) 웹 브라우저 주소 창에 java.oracle.com 입력

웹 브라우저 주소 창에 java.oracle.com 입력

2) 사이트 접속후 Downloads -> Javs SE -> Download JDK선택

Oracle Technology Network > Java > Java SE > Downloads

Overview Downloads Documentation Community Technologies Training

Java SE Downloads

Latest Release Next Release (Early Access) Embedded Use Previous Releases

Java Platform (JDK) 7u10 JavaFX 2.2.4 JDK 7u10 + NetBeans

Here are the Java SE downloads in detail:

Java Platform, Standard Edition		
Java SE 7u10	JDK	JRE

3) Accept License Agreement 선택후 Linux x86 다운로드 선택

News

- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

Java SE Development Kit 7u10

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

☒ Accept License Agreement ☐ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	106.63 MB	jdk-7u10-linux-i586.rpm
Linux x86	92.97 MB	jdk-7u10-linux-i586.tar.gz
Linux x64	104.75 MB	jdk-7u10-linux-x64.rpm
Linux x64	91.71 MB	jdk-7u10-linux-x64.tar.gz
Mac OS X x64	143.46 MB	jdk-7u10-macosx-x64.dmg
Solaris x86 (SVR4 package)	135.61 MB	jdk-7u10-solaris-i586.tar.Z
Solaris x86	91.97 MB	jdk-7u10-solaris-i586.tar.gz
Solaris SPARC (SVR4 package)	135.79 MB	jdk-7u10-solaris-sparc.tar.Z
Solaris SPARC	95.3 MB	jdk-7u10-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	22.86 MB	jdk-7u10-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.57 MB	jdk-7u10-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	22.64 MB	jdk-7u10-solaris-x64.tar.Z
Solaris x64	15.02 MB	jdk-7u10-solaris-x64.tar.gz
Windows x86	88.72 MB	jdk-7u10-windows-i586.exe
Windows x64	90.36 MB	jdk-7u10-windows-x64.exe
Linux ARM v6v7 Soft Float ABI	65.07 MB	jdk-7u10-linux-arm-sfp.tar.gz

Developer Training
Documentation
Java.com
Java.net
Student Developers
Tutorials

Java magazine Get it now for FREE! Subscribe Today

Java We're Hiring! Join the Java Development Team!

4) 다운로드 파일의 폴더로 이동

```
$ cd ~  
$ cd download/
```

5) jdk 설치

먼저 압축을 해제한 후 설치할 폴더로 복사를 한다.

```
$ tar xvfz jdk-7u10-linux-i586.tar.gz  
$ mkdir -p /usr/lib/jvm  
$ sudo cp /jdk-7u10-linux-i586 /usr/lib/jvm/ -Rfa
```

6) java 추가하기

```
$ sudo update -alternatives --install "/usr/bin/java" "java"  
"/usr/lib/jvm/jdk1.7.0_10/bin/java" 1
```

7) javac 추가하기

```
$ sudo update -alternatives --install "/usr/bin/javac" "javac"  
"/usr/lib/jvm/jdk1.7.0_10/bin/javac" 1
```

8) javaws 추가하기

```
$ sudo update -alternatives --install "/usr/bin/javac" "javaws"  
"/usr/lib/jvm/jdk1.7.0_10/bin/javaws" 1
```


7) 버전 확인

```
$ java --version
```

```
Error: A fatal exception has occurred. Program will exit.  
kdhee@kdhee:~$ java -version  
java version "1.7.0_09"  
OpenJDK Runtime Environment (IcedTea7 2.3.3) (7u9-2.3.3-0ubuntu1~12.10.1)  
OpenJDK Client VM (build 23.2-b09, mixed mode, sharing)  
kdhee@kdhee:~$ █
```

<java 버전 확인>