

Forum:[Hack Forums](#)

Thread: [Tool For Checking Password Lists](#)

Sertroph - July 7, 2013, 8:12 pm

I have been developing a fully fledged WPA/WPA password list manager, ([a remake of a previous tool](#)), and one of it's features is checking a password list before adding it to the database.

I wanted to hype up my password list manager, and I also wanted to test out the verification feature. This is the result.

What does this do? It checks a text file for invalid WPA/WPA2 password rules. Here is a screenshot of it in action.

Here is the text file it checked.

Code Good Password

good password

Good Password 1234567890

good password 1234567890

GoodPassword

goodpassword

BadPass

badpass

badpa55

badpas™

BadPas™

B3dPas™

This is a bad password because the character length is way above the maximum limit of 63 characters, and WPA won't allow such horrible things to exist in the first place anyways.

Here is the source code, so you can leech it and not give credit to me. (It's coded in C++)

Code #include <iostream>

#include <fstream>

#include <string>

```
int help(char *argv[]) {
```

```
    std::cout << "Usage: " << argv[0] << " <PasswordFile> [Options]" << std::endl;
```

```
    std::cout << " Options:" << std::endl;
```

```
    std::cout << " -wpa Verify password list using WPA Rules (WPA uses same rules as WPA2)"
```

```
<< std::endl;
```

```
    //std::cout << " -clean Automatically delete invalid passwords from file" << std::endl;
```

```
    std::cout << " -goodlist Print all valid passwords" << std::endl;
```

```
    std::cout << " -badlist Print all invalid passwords" << std::endl;
```

```
    std::cout << " Example:" << std::endl;
```

```
    std::cout << " " << argv[0] << " passwordlist.txt -wpa -badlist" << std::endl;
```

```

    return(1);
}

int does_exist(char *filetouse) {
    //first check to see if the cfg file exists
    FILE * pFile;
    if(pFile = fopen (filetouse,"r")) {
        //the file exists
        return(1);
    } else {
        //the file doesnt exist
        return(0);
    }
}

int verify(int argc, char *argv[]) {
    //variables for later usage
    //booleans for user options
    bool wpa = false;
    //bool clean = false;
    bool goodlist = false;
    bool badlist = false;
    //strings for file reading
    std::string password;
    std::ifstream infile;
    //strings for password statistics
    int numberoflines = 0;
    int numberoftolong = 0;
    int numberoftoshort = 0;
    int numberofbadchar = 0;
    int numberofgood = 0;
    int numberofempty = 0;

    const char *filetouse = argv[1];

    //check all the options
    for (int i = 0; i < argc; i++) {
        //options are -wpa -clean -goodlist -badlist
        std::string options[4] = {"-wpa", "-clean", "-goodlist", "-badlist"};
        if(argv[i] == options[0]) {
            wpa = true;
        }
        /*if(argv[i] == options[1]) {
            clean = true;
        }*/
        if(argv[i] == options[2]) {
            goodlist = true;
        }
        if(argv[i] == options[3]) {
            badlist = true;
        }
    }
}

```

```

//read the file line by line and check to see if the passwords are right
infile.open(filetouse); //open the user file
while(getline(infile, password)) { //while not the end of the file
//getline(infile,password); //read the current line to std::string password

//count the lines in the password field
numberoflines += 1;

//check password
if(wpa == true) { //if checking WPA Rules
    if(password == "") { //empty line
        //skip empty lines
        numberofempty += 1;
    } else if(password.size() > 63) {
        //password is to big to be used
        if(badlist == true) {
            std::cout << password << std::endl;
        }
        numberoftolong += 1;
    } else if(password.size() < 8) {
        //password is to small to be used
        if(badlist == true) {
            std::cout << password << std::endl;
        }
        numberoftoshort += 1;
    } else if
(password.find_first_not_of("!\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNopqrstuvwxyz
XYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~ ") != std::string::npos) {
        //password contains a bad character
        if(badlist == true) {
            std::cout << password << std::endl;
        }
        numberofbadchar += 1;
    } else {
        if(goodlist == true) {
            std::cout << password << std::endl;
        }
        numberofgood += 1;
    }
}
}
infile.close();

std::cout << "Checked " << numberoflines << " passwords." << std::endl;
std::cout << " " << numberofempty << " lines contained no text" << std::endl;
std::cout << " " << numberoftoshort << " lines were to short" << std::endl;
std::cout << " " << numberoftolong << " lines were to long" << std::endl;
std::cout << " " << numberofbadchar << " lines contained illegal characters" << std::endl;
std::cout << " " << numberofgood << " lines contained good passwords" << std::endl;

return(0);
}

```

```

int main(int argc, char *argv[]) {
    if(argc <= 1){ //not enough options so help is shown
        if(help(argv)) {
            std::cout << "Fatal error. Quitting program." << std::endl;
            return(1);
        } //else program is working fine
    } else { //plenty of options, better check to see if they are the right ones
        if(argv[1] == std::string("help")) { //display help
            if(!help(argv)) {
                std::cout << "Fatal error. Quitting program." << std::endl;
                return(1);
            } //else program is working fine
        } else {
            if(does_exist(argv[1])){
                verify(argc, argv);
            } else {
                std::cout << "Invalid file" << std::endl;
            }
        }
    }
}

```

Here is the windows executable.

[wpachecker.exe - 963 KB](#)

Here are the [VirusTotal results](#):

Detection Ratio is 3 / 47. I'm honestly not sure why the program is being detected as a virus. If you compile the code yourself, you will get the same thing as the pre-compiled version and as you can plainly see, there is no malicious code in the source.

TheGrayAviator - July 8, 2013, 12:40 am

If you need someone to compile that for Linux, you need to provide the source code.

yacked2 - July 8, 2013, 2:35 am

How can you said that password is too long ? WPA support 63 char password

Sertroph - July 8, 2013, 7:46 am

(Jul 8, 2013 02:35 AM)yacked2 Wrote: [\[CITE\]](#) How can you said that password is too long ? WPA support 63 char password

If you count the characters on that last line you'll notice that the number is over 120.

(Jul 8, 2013 12:40 AM)Greyhat.Infiltrator Wrote: [\[CITE\]](#) If you need someone to compile that for Linux, you need to provide the source code.

The source code is on this thread. It doesn't have multiple files so I just copied/pasted it here. I dunno how you missed it.

PinkPanther - July 8, 2013, 8:56 am

Hey, what toolchain did you use for this?

Sertroph - July 8, 2013, 9:03 am

(Jul 8, 2013 08:56 AM)PinkPanther Wrote: [\[CITE\]](#) Hey, what toolchain did you use for this?

I used Code::Blocks as my IDE, and the in-built MingW/Cygwin setup. It's nothing special.

t.vanhuong - August 26, 2013, 6:06 pm

File not found,please re-upload!
Thank

abdo123 - *August 27, 2013, 6:17 am*

Link broken .

better upload it here if you have some privileges !

Sertroph - *September 22, 2013, 6:23 pm*

I updated the download link. I didn't know that my chosen host does not keep files forever. Sorry about that.

ankitchauhan - *December 26, 2014, 12:06 pm*

GREAT WORK DONE BY U...THANK FOR POSTING THIS POST