

Branch: **master** ▼

WgmyFortnight / **keygenme** /

Fetching latest commit...

..

output

README.md

WGMY_Fortnight_0402.zip

keygen.py

keygenME.exe

solve.py

source.c

README.md

Challenge Description:

0402 Keygenme

We need the key badly but too bad, the server is no longer up. Can you help us
happier of you can give us a keygen.

Download the file at - <https://drive.google.com/open?id=1Rw2duLmmHTfimIeIIfsJX>

Inside the zip file got a PE32 file `keygenME.exe` :

```
# file keygenME.exe
```

```
keygenME.exe: PE32 executable (console) Intel 80386, for MS Windows
```

Try run it with `wine` in Linux:

```
# wine keygenME.exe 2>/dev/null
```

```
keygenme - wgmy2uni  
serial:
```

Looks like it require a serial key

Try with `strings` command:

```

...
...
...
...
keygenme - wgmy2uni
serial:
congratz!
nope!
...
...
...
IHDR
*<IDATx
...
IEND

```

Saw some key words for PNG image (IHDR,IDAT,IEND), so I run `foremost` on the PNG file got WGMY logo, I guess is the application icon:



Next, I decompiled it using Ghidra and I save the [source code](#)

Note: I changed the decompiled code to more readable, is not the original code

```
void main(){
    printf("
    printf("
    printf("\ \ \ \ \ \ \ \ / \ \ \ / < | | / \ \ / | \ \ \ \ | \n");
    printf(" \ \ / / / > Y Y \ \ \ | / \ \ | | / | \ \ | \n");
    printf(" \ \ \ \ \ \ \ \ / | \ | | / \ \ \ \ \ \ \ \ \ \ / | \ | / \ | \n");
    printf(" / \ \ \ / \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \n");
    printf("keygenme - wgmy2uni\n");
    printf("serial: ");
    fgets(&input,0x1e,stdin);
    sVar2 = strcspn(&input,"\n"); // Calculate the number of character before \n
    if (sVar2 < 0x1e) { // The input must be less than 30 characters
        (&input)[sVar2] = 0; // The \n character become null
        result = FUN_00401000(&input); // past the input to FUN_00401000 and return
        output = "congratz!\n";
        if (result != 1) { // If the result is not equal one then will
            output = "nope!\n"; // Meaning result must be equal one for output
        }
        printf(output);
        FUN_00401270();
        return;
    }
```

```
uint __fastcall FUN_00401000(char *input)
{
    char *containDash;
    char *token;
```

```

char *input2;
int valid = 1;
int index;
int valid_letters[26] = [1,0,0,0,4,0,0xffffffff7c,0,0,0,0,0,0xffffffff9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
char cVar1;
char temp;
int value;

input2 = input;
do {
    cVar1 = *input2;
    input2 = input2 + 1;
} while (cVar1 != 0);
containDash = strchr(input, '-');
if ((input2 + -(input + 1) == 0x13) && // Checks the length of input is 19
    containDash != NULL) {           // Check is it contain dash '-'
    do {
        containDash = strchr(containDash + 1, '-'); // Calculate number of dash in input
        valid = valid + 1;
    } while (containDash != NULL);
    if (valid == 4) {                // Needs 3 dashes (valid is initialize with 1)
        token = strtok(input, "-"); // Splits the input with dashes
        do {                         // Ex: ASD-ASD -> ASD,ASD
            if (token == NULL) {
                return (valid == 8); // Our target is let valid equal to 8
            }                        // So this function will return True
            index = 0;
            do {
                if (isalnum(token[index]) == 0) { // Checks the input is alphanumeric
                    return 0;
                }
                temp = token[index];
                if (('`' < temp) && (temp < '{')) { // Checks the input is not lower case
                    return 0;                    // Refer to ASCII table (Between 96-122)
                }
                if (('/' < temp) && (temp < ':')) { // Checks the input is not number
                    return 0;                    // Refer to ASCII table (Between 48-57)
                }                                // That means only Upper case is allowed
            } while (temp != 0);
            value = valid_letters[temp-65]; // 'A' is 65 in ASCII then the index is 0
                                            // 'B' is 1, 'C' is 2 etc.
                                            // Total 26 numbers in array is matched with 26
                                            // total 26 alphabet
            valid_letters[temp-65] = value + 1; // Each alphabet is set to 1
                                                // So each alphabet can only use once
                                                // Because of the condition checked above
            if (value != 0) { // Checks the value is 0, meaning only 0 from the array
                return 0;
            }
        }
    }
}

```

```

        index = index + 1;
    } while (index < 4); // Each separate loop 4 times and 3 dashes meaning
                        // should look like: ASDF-ASDF-ASDF-ASDF (19 characters)
    token = strtok(NULL, "-");
    valid = valid + 1;
} while( true );
}
}
return 0;
}

```

Condition of the serial key:

1. Length of 19
2. Contain 3 dashes
3. Without letter case letter and numbers
4. 4 character between dashes
5. Each character can only use once

So the serial key should be something like `ASDF-ASDF-ASDF-ASDF`

After I finished analyse the source code, then I using [python script](#) to filter out the

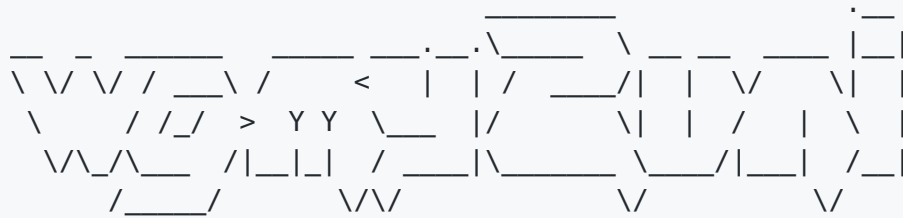
```

text = [1,0,0,0,4,0,0xffffffff7c,0,0,0,0,0,0xffffffff9,0,0,0,0,0x22c4,3,0,0,0,0,0xfffffffff]
possible_character = ''
for i,t in enumerate(text):
    if not t:
        possible_character += chr(i+65)
print possible_character

```

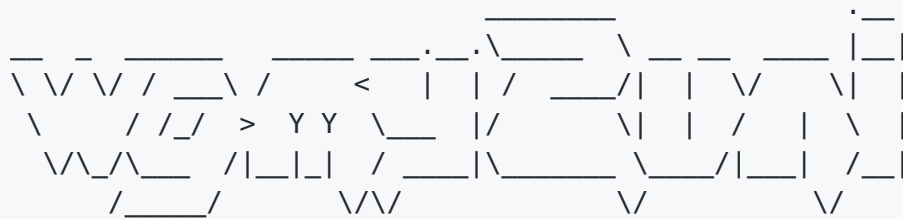
Result: `BCDFHIJKLNOPQTUVXZ`

Try the key in the program:



```
keygenme - wgmy2uni  
serial: BCDF-HIJK-LNOP-QTUV  
congratz!
```

Works even in different order:



```
keygenme - wgmy2uni  
serial: DFHI-JKLN-OPQT-UVXZ  
congratz!
```

Finally, I wrote a [keygen](#) that will randomly generate valid serial key:

```
key = ''  
for i in range(16):  
    temp = random.choice(valid_character)  
    key += temp  
    valid_character.remove(temp)  
    if (i+1) % 4 == 0:  
        key += '-'  
print key[:-1]
```

Conclusion

Valid serial key for this program is combination of these letters `BCDFHIJKLNOPQTUVXZ`
format of `XXXX-XXXX-XXXX-XXXX`