

Forum:[Hack Forums](#)

Thread: [PicoCTF 2014 decryption problem](#)

Saegon - February 16, 2015, 4:18 am

The goal is to reverse engineer the program and find out what password is.

What's interesting is that to decrypt some string you simply encrypt it again. I tried to print out each value of key in `encrypt(input_data, password)`, but the compiler complains about indentation. Also, how can I input password as `arg[4]` when I don't know what it is. The problem included some file that I cannot upload.

It's true that I could probably look up the answer, but I'd rather solve as much of the problem that I can by myself.

Here is the code:

Code `#!/usr/bin/python`

```
import sys
```

```
.....
```

Daedalus Corporation encryption script.

```
.....
```

```
def xor(input_data, key):
```

```
    result = ""
```

```
    for ch in input_data:
```

```
        result += chr(ord(ch) ^ key)
```

```
    return result
```

```
def encrypt(input_data, password):
```

```
    key = 0
```

```
    for ch in password:
```

```
        key ^= ((2 * ord(ch) + 3) & 0xff)
```

```
    return xor(input_data, key)
```

```
def decrypt(input_data, password):
```

```
    return encrypt(input_data, password)
```

```
def usage():
```

```
    print("Usage: %s [encrypt/decrypt] [in_file] [out_file] [password]" % sys.argv[0])
```

```
    exit()
```

```
def main():
```

```
    if len(sys.argv) < 5:
```

```
        usage()
```

```
    input_data = open(sys.argv[2], 'r').read()
```

```
    result_data = ""
```

```
if sys.argv[1] == "encrypt":
    result_data = encrypt(input_data, sys.argv[4])
elif sys.argv[1] == "decrypt":
    result_data = decrypt(input_data, sys.argv[4])
else:
    usage()
```

```
out_file = open(sys.argv[3], 'w')
out_file.write(result_data)
out_file.close()
```

```
main()
```

Shiv - February 16, 2015, 5:07 am

(Feb 16, 2015 05:18 AM)Saegon Wrote: [\[CITE\]](#) Code def encrypt(input_data, password):

```
key = 0
for ch in password:
    key ^= ((2 * ord(ch) + 3) & 0xff)
```

```
    return xor(input_data, key)
```

key will always be between 0-255, you just have to skip the function encrypt and test all 256 possibilities for key directly in the xor function