

Secrets of Opera's Magic Wand

Opera stores all these login passwords in encrypted format in the 'Magic Wand File' called 'Wand.dat' within the profile folder. This profile path is different for different versions of Opera. Here is the Wand file (wand.dat) location for some of the recent versions of Opera.

For Opera Version 10 and above

```
[Windows NT/2K/2k3/XP]
C:\Documents and Settings\\Application Data\Opera\Opera\wand.dat

[Windows Vista/Windows 7]
C:\users\\AppData\Roaming\Opera\Opera\wand.dat
```

For Opera Version less than 10

```
[Windows NT/2K/2k3/XP]
C:\Documents and Settings\\Application Data\Opera\Opera\profile\wand.dat

[Windows Vista/Windows 7]
C:\users\\AppData\Roaming\Opera\Opera\profile\wand.dat
```

Opera stores following information in the password Wand file for each of the stored entry in the following order,

- Login URL of website
- Main URL of website
- Username field ID
- Username
- Password field ID
- Password

All these information are stored in the encrypted format in wand.dat file. Opera uses Triple-DES algorithm along with static salt data to encrypt these secret information.

Decrypting Passwords from Opera Wand File

Most of the information present here is based on the original work of sna@reteam.org. I am putting it in more simple detailed explanation for everyone to understand and decipher the art of decrypting Opera's Magic Wand.

Opera Wand file consists of multiple encrypted blocks for each of the stored password entries. Each such encrypted block mainly consist of following fields,

- Size of encrypted block (4 bytes)
- DES Key Length (1 byte)
- DES Key (8 bytes)

Size of encrypted data (4 bytes)
Encrypted Data

Opera uses the following static salt data to encrypt each of the secret information.

```
0x83, 0x7D, 0xFC, 0x0F, 0x8E, 0xB3, 0xE8, 0x69, 0x73, 0xAF, 0xFF
```

Here are the step by step instructions for decrypting the above encrypted data from each block

Retrieve the DES key (8 bytes) for the current encrypted block. Then perform the byte stream with the above opera salt and the retrieved DES key.

Now compute the MD5 checksum of this byte stream. The checksum will be stored in the variable md5hash1.

```
char buffer[256];  
DWORD dwBufSize=11+8;  
GetMD5Hash((char*)buffer, dwBufSize, (char*)md5hash1, MD5_DIGEST_LENGTH);
```

Next perform another byte stream with md5hash1, opera salt and DES key in that order and compute its MD5 hash. Final checksum will be stored in md5hash2 variable.

```
char buffer[256];  
DWORD dwBufSize=MD5_DIGEST_LENGTH + 11 + 8;  
GetMD5Hash((char*) buffer, dwBufSize, (char*)md5hash2, MD5_DIGEST_LENGTH);
```

Now use both the MD5 hashes to create schedule keys for decryption as shown below,

```
DES_key_schedule key1, key2, key3;  
  
DES_set_key_unchecked((const_DES_cblock *)&md5hash1[0], &key1);  
  
DES_set_key_unchecked((const_DES_cblock *)&md5hash1[8], &key2);  
  
DES_set_key_unchecked((const_DES_cblock *)&md5hash2[0], &key3);
```

Create DES vector component required for decryption using the second MD5 hash,

```
DES_cblock iVector;  
memcpy(iVector, &md5hash2[8], sizeof(DES_cblock));
```

Finally decrypt the encrypted data using Triple DES decryption function

```
char decryptData[512];  
DES_ede3_cbc_decrypt(encryptData, decryptData, dataLength, &key1, &key2, &key3,  
&iVector, DES_DECRYPT);
```

On successful execution, the decrypted data will be copied to the decryptData variable in unicode format. You can use WideCharToMultiByte function to convert it back to ascii format.

The decryption related structures and functions mentioned here are part of OpenSSL Crypto library .