

Forum:[Hack Forums](#)

Thread: [C | I can't figure out how to correctly get the resources out of my stub](#)

?Onyx? - October 15, 2019, 7:28 am

Hey all,

I don't do much C programming so I'm not sure on how good the code is but I'm stuck right now. And am asking if anyone here might be able to help.

I'm trying to hide a file in my stub like so:

PHP Code

```
<?php
#include <stdio.h>
#include <stdlib.h>
#include <Windows.h>

// Read the bytes of the path specified.
char *read_file_bytes(char *path, int *out_filelen)
{
    FILE *fileptr;
    char *buffer;
    int filelen = 0;

    fileptr = fopen(path, "rb");
    if (fileptr)
    {
        fseek(fileptr, 0, SEEK_END);
        filelen = ftell(fileptr);
        rewind(fileptr);

        buffer = (char *)malloc((filelen + 1) * sizeof(char));
        if (buffer != NULL)
        {
            fread(buffer, filelen, 1, fileptr);
            fclose(fileptr);
            *out_filelen = filelen;
            return buffer;
        }
    }
    return NULL;
}

// XOR every byte with a part of the key.
char *encrypt_file_bytes(char *file_bytes, int file_length, char *key, int key_length)
{
    char *encrypted_file_bytes = (char *)malloc(file_length * sizeof(char));

    for (int i = 0; i < file_length; i++)
```

```

    {
        encrypted_file_bytes[i] = file_bytes[i] ^ key[i % key_length];
    }

    return encrypted_file_bytes;
}

int modify_resources(LPCSTR szTargetPE, int id, char *lpBytes, DWORD dwSize)
{
    // Handle to the resources section of our stub
    HANDLE hFile = BeginUpdateResourceA(szTargetPE, FALSE);
    // Attempt to update the resources and store the result in a bool
    int update_status = UpdateResourceA(hFile, MAKEINTRESOURCEA(10),
    MAKEINTRESOURCEA(69), MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), lpBytes,
    dwSize);
    // Close the handle / end updating the resources so we can use them again.
    EndUpdateResourceA(hFile, FALSE);
    return update_status;
}

int main(int argc, char *argv[])
{
    // Check given arguments
    if (argc == 1)
    {
        printf("Pass the file to be encrypted as an argument to this executable \r\n");
        return 1;
    }

    // Read our file
    int target_file_length = 0;
    char *target_file_bytes = read_file_bytes(argv[1], &target_file_length);

    if (target_file_length == 0)
    {
        printf("Unable to read the file specified.");
        return 1;
    }

    // Encrypt file
    char *encryption_key = "abcdefghijklmnop";
    int encryption_key_length = sizeof(encryption_key);
    char *encrypted_file_bytes = encrypt_file_bytes(target_file_bytes, target_file_length,
    encryption_key, encryption_key_length);

    // Hide the image into our stub
    if (modify_resources("stub.exe", 10, encrypted_file_bytes, target_file_length))
    {
        printf("Succes!");
    }
    else
    {
        printf("Failure!");
    }
}

```

```
}  
}
```

And I load it from my stub like this:

PHP Code

```
<?php  
int get_resource_data(char *out_resources)  
{  
    HMODULE hModule = GetModuleHandleA(NULL);  
    HRSRC hResource = FindResourceA(hModule, MAKEINTRESOURCEA(69),  
MAKEINTRESOURCEA(10));  
    HGLOBAL temp = LoadResource(NULL, hResource);  
    int resource_size = SizeofResource(NULL, hResource);  
    // Why continue if we can't load anything?  
    if (resource_size == 0)  
        return 0;  
    out_resources = (char *)malloc(resource_size * sizeof(char));  
    memcpy((void *)out_resources, temp, resource_size);  
    return resource_size;  
}  
  
// XOR every byte with a part of the key.  
char *decrypt_file_bytes(char *file_bytes, int file_length, char *key, int key_length)  
{  
    char *decrypted_bytes = (char *)malloc(file_length * sizeof(char));  
  
    for (int i = 0; i < file_length; i++)  
    {  
        decrypted_bytes[i] = file_bytes[i] ^ key[i % key_length];  
    }  
  
    return decrypted_bytes;  
}  
  
int main()  
{  
    char *resource_data;  
    int resource_size = get_resource_data(resource_data);  
  
    if (resource_size == 0)  
    {  
        printf("No resources found! \r\n");  
        return 1;  
    }  
  
    char *encryption_key = "abcdefghijklmnop";  
    int encryption_key_length = sizeof(encryption_key);  
    char* decrypted_file_data = decrypt_file_bytes(resource_data, resource_size, encryption_key,  
encryption_key_length );  
  
    // Write out file to disk  
    FILE *write_ptr;
```

```

write_ptr = fopen("decrypted_file.exe", "wb");
if (write_ptr)
{
    fwrite(decrypted_file_data, sizeof(char), resource_size, write_ptr);
    fclose(write_ptr);

    return 1;
}

return 0;
}

```

But when I hide the builder in the stub and have the stub write its resources out to disk again the file is completely different. So I made another small program which loads the stub in a buffer, encrypts it and then decrypts it and writes the first 15 bytes to the console which has the correct output. So I believe the error is in loading the resources. What did I do wrong there and how can I improve?

Thank you for your time.

HFDone - October 15, 2019, 9:17 am

start with changing char pointer of the file data to unsigned char pointer

?Onyx? - October 15, 2019, 10:18 am

(Oct 15, 2019 09:17 AM)HFDone Wrote: [\[CITE\]](#) start with changing char pointer of the file data to unsigned char pointer

Thank you, will do!

Is this because the unsigned char goes from 0 - 255 so it can represent the bytes better?

(Oct 15, 2019 09:17 AM)HFDone Wrote: [\[CITE\]](#) start with changing char pointer of the file data to unsigned char pointer

Hey I've tried editing the code as you said. Using unsigned char* to store the binary data.

Weird thing is that when I get the data from resources and write it out to a file this is the data I get from a PE dumper:

PHP Code

```

<?php
#include <stdio.h>
#include <stdlib.h>
#include <Windows.h>

int get_resource_data(unsigned char *out_resources)
{
    HMODULE hModule = GetModuleHandleA(NULL);
    HRSRC hResource = FindResourceA(hModule, MAKEINTRESOURCEA(69),
MAKEINTRESOURCEA(10));
    HGLOBAL temp_data = LoadResource(hModule, hResource);
    int resource_size = SizeofResource(hModule, hResource);

    if(resource_size == 0)
        return 0;
}

```

```

    out_resources = (unsigned char*)malloc(resource_size * sizeof(char));
    memcpy((void*)out_resources, temp_data, resource_size);

    return resource_size;
}

int write_file(unsigned char* file_buffer, char* file_path, int file_length)
{
    FILE *fileptr = fopen(file_path, "wb");
    if(!fileptr)
    {
        printf("Unable to write file");
        return 0;
    }

    fwrite(file_buffer, 1, file_length, fileptr);
    fclose(fileptr);
    return 1;
}

int main()
{
    unsigned char* resource_buffer;
    int filelength = get_resource_data(resource_buffer);
    printf("%d", filelength);
    write_file(resource_buffer, "stub_exe.exe", filelength);
}

```

Either I'm very very blind or the solution is a bit too hard for my skills in C but I still can not seem to find the problem and the solution on it.

HFDone - October 15, 2019, 11:08 am

```
printf("%d", filelength);
```

is the length is the same as the file size

?Onyx? - October 15, 2019, 11:20 am

(Oct 15, 2019 11:08 AM)HFDone Wrote: [\[CITE\]](#) printf("%d", filelength);
 is the length is the same as the file size
 Yes, in both the builder and stub the output is 3072, which is the filesize.

HFDone - October 15, 2019, 11:28 am

did you change to unsigned in your read code xor and resource too (read_file_bytes
 encrypt_file_bytes modify_resources)

?Onyx? - October 15, 2019, 12:01 pm

(Oct 15, 2019 11:28 AM)HFDone Wrote: [\[CITE\]](#) did you change to unsigned in your read code xor
 and resource too (read_file_bytes encrypt_file_bytes modify_resources)
 Yes, I tried to create the smallest program possible to still have the load / unload functionality which
 is as follows:

Builder:
 PHP Code
 <?php
 #include <stdio.h>

```
#include <stdlib.h>
#include <Windows.h>
```

```
unsigned char* read_file(char* file_path, int *out_filelength)
{
    FILE *fileptr;
    unsigned char *buffer;
    int filelen = 0;

    fileptr = fopen(file_path, "rb");
    if (fileptr)
    {
        fseek(fileptr, 0, SEEK_END);
        filelen = ftell(fileptr);
        rewind(fileptr);

        buffer = (unsigned char *)malloc((filelen) * sizeof(unsigned char));
        if (buffer != NULL)
        {
            fread(buffer, filelen, 1, fileptr);
            fclose(fileptr);
            *out_filelength = filelen;
            return buffer;
        }
    }
    return NULL;
}
```

```
int write_file(unsigned char* file_buffer, char* file_path, int file_length)
{
    FILE *fileptr = fopen(file_path, "wb");
    if(!fileptr)
    {
        printf("Unable to write file");
        return 0;
    }

    fwrite(file_buffer, 1, file_length, fileptr);
    fclose(fileptr);
    return 1;
}
```

```
int modify_resources(LPCSTR szTargetPE, int id, unsigned char *lpBytes, DWORD dwSize)
{
    // Handle to the resources section of our stub
    HANDLE hFile = BeginUpdateResourceA(szTargetPE, FALSE);
    // Attempt to update the resources and store the result in a bool
    int update_status = UpdateResourceA(hFile, MAKEINTRESOURCEA(10),
    MAKEINTRESOURCEA(69), MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), lpBytes,
    dwSize);
    // Close the handle / end updating the resources so we can use them again.
    EndUpdateResourceA(hFile, FALSE);
    return update_status;
}
```

```
}
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    // // Check given arguments
```

```
    // if (argc == 1)
```

```
    // {
```

```
    //     printf("Pass the file to be encrypted as an argument to this executable \r\n");
```

```
    //     return 1;
```

```
    // }
```

```
    int file_length = 0;
```

```
    unsigned char* file_bytes;
```

```
    file_bytes = read_file(argv[0], &file_length);
```

```
    printf("%d \r\n", file_length);
```

```
    if(file_bytes == 0)
```

```
    {
```

```
        printf("Unable to read the file specified!");
```

```
        return 1;
```

```
    }
```

```
    // write_file(file_bytes, "testread.exe", file_length);
```

```
    if (modify_resources("stub.exe", 10, file_bytes, file_length))
```

```
    {
```

```
        printf("Success! Written %d bytes", file_length);
```

```
    }
```

```
    else
```

```
    {
```

```
        printf("Failure!");
```

```
    }
```

```
    return 0;
```

```
}
```

```
Stub:
```

```
PHP Code
```

```
<?php
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <Windows.h>
```

```
int get_resource_data(unsigned char *out_resources)
```

```
{
```

```
    HMODULE hModule = GetModuleHandleA(NULL);
```

```
    HRSRC hResource = FindResourceA(hModule, MAKEINTRESOURCEA(69),
```

```
MAKEINTRESOURCEA(10));
```

```
    HGLOBAL temp_data = LoadResource(hModule, hResource);
```

```
    int resource_size = SizeofResource(hModule, hResource);
```

```
    if(resource_size == 0)
```

```
        return 0;
```

```
    out_resources = (unsigned char*)malloc(resource_size * sizeof(char));
```

```

memcpy((void*)out_resources, temp_data, resource_size);

return resource_size;
}

int write_file(unsigned char* file_buffer, char* file_path, int file_length)
{
    FILE *fileptr = fopen(file_path, "wb");
    if(!fileptr)
    {
        printf("Unable to write file");
        return 0;
    }

    fwrite(file_buffer, 1, file_length, fileptr);
    fclose(fileptr);
    return 1;
}

int main()
{
    unsigned char* resource_buffer;
    int filelength = get_resource_data(resource_buffer);
    printf("%d", filelength);
    write_file(resource_buffer, "stub_exe.exe", filelength);
}

```

And does still not produce the desired output

HFDone - October 15, 2019, 12:31 pm

probably the problem is with the resource functions

are using clean stub and try to open the stub after writing with resource editor and check if you can see the right file data

Edit - the problem is with the read resource function pointer param

put this in your main stub instead

```

PHP Code
<?php
HMODULE hModule = GetModuleHandleA(NULL);
HRSRC hResource = FindResourceA(hModule, MAKEINTRESOURCEA(69),
MAKEINTRESOURCEA(10));
HGLOBAL temp_data = LoadResource(hModule, hResource);
int resource_size = SizeofResource(hModule, hResource);

if(resource_size == 0)
    return 0;

unsigned char* out_resources = (unsigned char*)malloc(resource_size * sizeof(char));
memcpy((void*)out_resources, temp_data, resource_size);
printf("%d", resource_size);
write_file(out_resources, "stub_exe.exe", resource_size);

```

?Onyx? - October 15, 2019, 2:50 pm

(Oct 15, 2019 12:31 PM)HFDone Wrote: [\[CITE\]](#) probably the problem is with the resource functions are using clean stub and try to open the stub after writing with resource editor and check if you can see the right file data

Edit - the problem is with the read resource function pointer param

put this in your main stub instead

PHP Code

```
<?php
```

```
HMODULE hModule = GetModuleHandleA(NULL);
HRSRC hResource = FindResourceA(hModule, MAKEINTRESOURCEA(69),
MAKEINTRESOURCEA(10));
HGLOBAL temp_data = LoadResource(hModule, hResource);
int resource_size = SizeofResource(hModule, hResource);

if(resource_size == 0)
    return 0;

unsigned char* out_resources = (unsigned char*)malloc(resource_size * sizeof(char));
memcpy((void*)out_resources, temp_data, resource_size);
printf("%d", resource_size);
write_file(out_resources, "stub_exe.exe", resource_size);
```

Yes that works! Thanks a lot :)

SKIDROOTKIT - October 16, 2019, 1:05 pm

you can easily just study pe format and remove it directly instead of using any sort of api

?Onyx? - October 18, 2019, 1:35 pm

(Oct 16, 2019 01:05 PM)SKIDROOTKIT Wrote: [\[CITE\]](#) you can easily just study pe format and remove it directly instead of using any sort of api

You mean as in copy the data from the PE without the FindResourceA and LoadResource functions?

Thanks for the tip. I'm still learning a lot so that might be a goal for later.

SKIDROOTKIT - October 18, 2019, 4:48 pm

(Oct 18, 2019 01:35 PM)?Onyx? Wrote: [\[CITE\]](#) You mean as in copy the data from the PE without the FindResourceA and LoadResource functions?

Thanks for the tip. I'm still learning a lot so that might be a goal for later.
pretty much yes