

# Analysing Remcos RAT's executable

krabsonsecurity.com/2018/03/02/analysing-remcos-rats-executable/

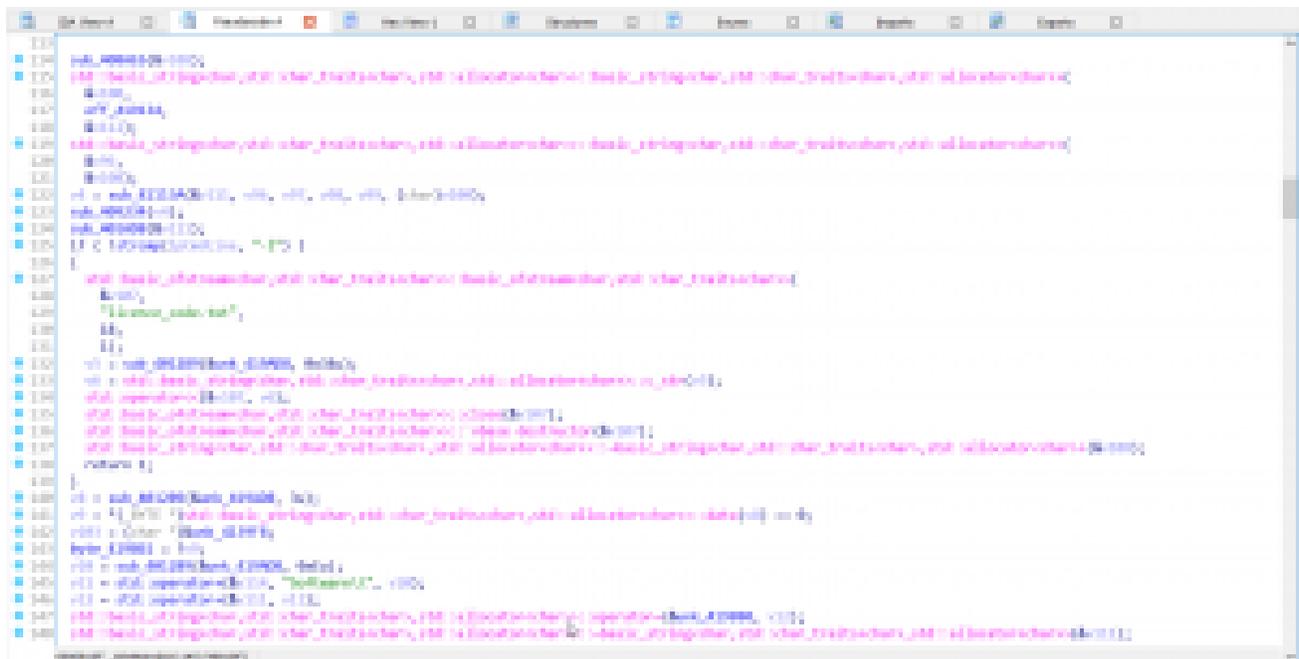
Posted on March 2, 2018

Remcos is a native RAT sold on the forums HackForums.net. It is an interesting piece of RAT (and the only one that is developed in a native language other than Netwire) and is heavily used by malware actors. Coded by the author, Viotto, it is self proclaimed to be a legal administration tool. Whether that is true or not is to be understood by people who have seen Remcos malware campaigns and the fact that the author also sells a crypter.

First thing we notice when looking at Remcos RAT is that it uses C++ and the CRT quite heavily. This leads to the output file being rather large (though still small by some standards) at 120kb.

```
std::basic_string<char, std::char_traits<char>, std::allocator<char> > basic_string<char, std::char_traits<char>, std::allocator<char> >
{
    off_418114,
    off_418114,
    std::basic_string<char, std::char_traits<char>, std::allocator<char> > basic_string<char, std::char_traits<char>, std::allocator<char> >
    {
        off_418114,
        off_418114,
        std::basic_string<char, std::char_traits<char>, std::allocator<char> > basic_string<char, std::char_traits<char>, std::allocator<char> >
    }
};
```

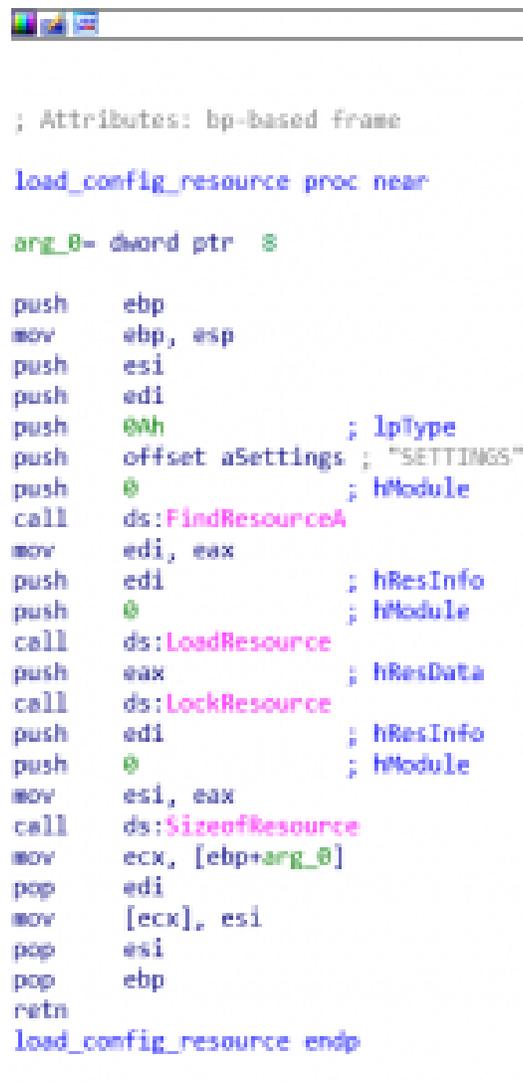
Another interesting thing is that Remcos allows you to extract the license (which is most likely based on the HWID) of the individual who created the stub easily by executing the file with the -l switch. If the sample is packed by a crypter that does not pass the command line parameter, you'll have to unpack it first.



Static Mutex (compared across multiple bins), runtime detection and hooking as well as memory signature anyone?

```
173 v17 = (CHAR *)OpenMutexA(0x100000u, 0, "Remcos_Mutex_Inj");
174 lpCmdLine = v17;
175 if ( v17 )
176 {
177     WaitForSingleObject(v17, 0x5A60u);
178     CloseHandle(lpCmdLine);
179 }
180 v184 = (char *)Data;
181 v183 = "Inj";
```

Simple resource loading using kernel32 APIs



```
; Attributes: bp-based frame
load_config_resource proc near
arg_0= dword ptr 8

push    ebp
mov     ebp, esp
push    esi
push    edi
push    0Ah           ; lpType
push    offset aSettings ; "SETTINGS"
push    0             ; hModule
call    ds:FindResourceA
mov     edi, eax
push    edi           ; hResInfo
push    0             ; hModule
call    ds:LoadResource
push    eax           ; hResData
call    ds:LockResource
push    edi           ; hResInfo
push    0             ; hModule
mov     esi, eax
call    ds:SizeofResource
mov     ecx, [ebp+arg_0]
pop     edi
mov     [ecx], esi
pop     esi
pop     ebp
retn
load_config_resource endp
```

Fragments from usage of the public password recovery tools developed by NirSoft (/stext is a command line switch used to dump passwords to a text file, [reference](#)):





```

20  GlobalData(0), 0, 0);
21  GetModuleHandle(0), 0, 0);
22  v1 = v1 & 0x00000000;
23  v1 = v1 & 0x00000000;
24  v1 = v1 & 0x00000000;
25  v1 = v1 & 0x00000000;
26  v1 = v1 & 0x00000000;
27  v1 = v1 & 0x00000000;
28  v1 = v1 & 0x00000000;
29  v1 = v1 & 0x00000000;
30  v1 = v1 & 0x00000000;
31  v1 = v1 & 0x00000000;
32  v1 = v1 & 0x00000000;
33  v1 = v1 & 0x00000000;
34  v1 = v1 & 0x00000000;
35  v1 = v1 & 0x00000000;
36  v1 = v1 & 0x00000000;
37  v1 = v1 & 0x00000000;
38  v1 = v1 & 0x00000000;
39  v1 = v1 & 0x00000000;
40  v1 = v1 & 0x00000000;
41  v1 = v1 & 0x00000000;
42  v1 = v1 & 0x00000000;
43  v1 = v1 & 0x00000000;
44  v1 = v1 & 0x00000000;
45  v1 = v1 & 0x00000000;
46  v1 = v1 & 0x00000000;
47  v1 = v1 & 0x00000000;
48  v1 = v1 & 0x00000000;
49  v1 = v1 & 0x00000000;
50  v1 = v1 & 0x00000000;
51  v1 = v1 & 0x00000000;
52  v1 = v1 & 0x00000000;
53  v1 = v1 & 0x00000000;
54  v1 = v1 & 0x00000000;
55  v1 = v1 & 0x00000000;
56  v1 = v1 & 0x00000000;
57  v1 = v1 & 0x00000000;
58  v1 = v1 & 0x00000000;
59  v1 = v1 & 0x00000000;
60  v1 = v1 & 0x00000000;
61  v1 = v1 & 0x00000000;
62  v1 = v1 & 0x00000000;
63  v1 = v1 & 0x00000000;
64  v1 = v1 & 0x00000000;
65  v1 = v1 & 0x00000000;
66  v1 = v1 & 0x00000000;
67  v1 = v1 & 0x00000000;
68  v1 = v1 & 0x00000000;
69  v1 = v1 & 0x00000000;
70  v1 = v1 & 0x00000000;
71  v1 = v1 & 0x00000000;
72  v1 = v1 & 0x00000000;
73  v1 = v1 & 0x00000000;
74  v1 = v1 & 0x00000000;
75  v1 = v1 & 0x00000000;
76  v1 = v1 & 0x00000000;
77  v1 = v1 & 0x00000000;
78  v1 = v1 & 0x00000000;
79  v1 = v1 & 0x00000000;
80  v1 = v1 & 0x00000000;
81  v1 = v1 & 0x00000000;
82  v1 = v1 & 0x00000000;
83  v1 = v1 & 0x00000000;
84  v1 = v1 & 0x00000000;
85  v1 = v1 & 0x00000000;
86  v1 = v1 & 0x00000000;
87  v1 = v1 & 0x00000000;
88  v1 = v1 & 0x00000000;
89  v1 = v1 & 0x00000000;
90  v1 = v1 & 0x00000000;
91  v1 = v1 & 0x00000000;
92  v1 = v1 & 0x00000000;
93  v1 = v1 & 0x00000000;
94  v1 = v1 & 0x00000000;
95  v1 = v1 & 0x00000000;
96  v1 = v1 & 0x00000000;
97  v1 = v1 & 0x00000000;
98  v1 = v1 & 0x00000000;
99  v1 = v1 & 0x00000000;
100 v1 = v1 & 0x00000000;

```

Starting the debug console

```

1 int __cdecl start_debug_console(char a1)
2 {
3     HWND v1; // eax
4     char Format; // [esp+Ch] [ebp-3E8h]
5     char v4; // [esp+Dh] [ebp-3E7h]
6     __int16 v5; // [esp+3F1h] [ebp-3h]
7     char v6; // [esp+3F3h] [ebp-1h]
8
9     AllocConsole();
10    v1 = (HWND)0;
11    hWnd = v1;
12    if ( !a1 )
13        ShowWindow(v1, 0);
14    freopen("CONOUT$", "a", &iob[1]);
15    memset(&v4, 0, 0x3E4u);
16    v5 = 0;
17    v6 = 0;
18    strcpy(&Format, " * REMCOS v");
19    strcat(&Format, "2.0.2 Pro");
20    strcat(&Format, "\n * Breaking-Security.Net\n\n");
21    return printf(&Format);
22 }

```

Very simple (and runtime detected) keylogger using SetWindowsHookExA (13 is the constant for WH\_KEYBOARD\_LL)

```

1 int __thiscall keylogger_hook(HWND *this)
2 {
3     HWND *v1; // esi
4     struct tagMSG Msg; // [esp+0h] [ebp-10h]
5
6     v1 = this;
7     dword_4107E8 = (int)this;
8     if ( *this )
9         goto LABEL_4;
10    *this = SetWindowsHookEx(13, fn, 0, 0);
11    while ( *v1 )
12    {
13    LABEL_4:
14        if ( !GetMessageA(&Msg, 0, 0, 0) )
15            return 0;
16        TranslateMessage(&Msg);
17        DispatchMessageA(&Msg);
18    }
19    return 0;
20 }

```

```

21 int __thiscall keylogger_hook(HWND *this)
22 {
23     v1 = this;
24     if ( !MSG_FROM_SYSTEM )
25         return 0;
26     if ( !keylogger_hook )
27     {
28         v1 = 0;
29         char *basic_string; // [esp+0h] [ebp+0h]
30         char *other_frame; // [esp+4h] [ebp+4h]
31         void *callback; // [esp+8h] [ebp+8h]
32         int basic_string; // [esp+10h] [ebp+10h]
33         int other_frame; // [esp+14h] [ebp+14h]
34         int callback; // [esp+18h] [ebp+18h]
35         int basic_string; // [esp+1Ch] [ebp+1Ch]
36         int other_frame; // [esp+20h] [ebp+20h]
37         int callback; // [esp+24h] [ebp+24h]
38         int basic_string; // [esp+28h] [ebp+28h]
39         int other_frame; // [esp+34h] [ebp+34h]
40         int callback; // [esp+38h] [ebp+38h]
41         int basic_string; // [esp+3Ch] [ebp+3Ch]
42         int other_frame; // [esp+40h] [ebp+40h]
43         int callback; // [esp+44h] [ebp+44h]
44         int basic_string; // [esp+48h] [ebp+48h]
45         int other_frame; // [esp+54h] [ebp+54h]
46         int callback; // [esp+58h] [ebp+58h]
47         int basic_string; // [esp+5Ch] [ebp+5Ch]
48         int other_frame; // [esp+60h] [ebp+60h]
49         int callback; // [esp+64h] [ebp+64h]
50         int basic_string; // [esp+68h] [ebp+68h]
51         int other_frame; // [esp+74h] [ebp+74h]
52         int callback; // [esp+78h] [ebp+78h]
53         int basic_string; // [esp+7Ch] [ebp+7Ch]
54         int other_frame; // [esp+80h] [ebp+80h]
55         int callback; // [esp+84h] [ebp+84h]
56         int basic_string; // [esp+88h] [ebp+88h]
57         int other_frame; // [esp+94h] [ebp+94h]
58         int callback; // [esp+98h] [ebp+98h]
59         int basic_string; // [esp+9Ch] [ebp+9Ch]
60         int other_frame; // [esp+A0h] [ebp+A0h]
61         int callback; // [esp+A4h] [ebp+A4h]
62         int basic_string; // [esp+A8h] [ebp+A8h]
63         int other_frame; // [esp+B4h] [ebp+B4h]
64         int callback; // [esp+B8h] [ebp+B8h]
65         int basic_string; // [esp+BCh] [ebp+BCh]
66         int other_frame; // [esp+C0h] [ebp+C0h]
67         int callback; // [esp+C4h] [ebp+C4h]
68         int basic_string; // [esp+C8h] [ebp+C8h]
69         int other_frame; // [esp+D4h] [ebp+D4h]
70         int callback; // [esp+D8h] [ebp+D8h]
71         int basic_string; // [esp+DCh] [ebp+DCh]
72         int other_frame; // [esp+E0h] [ebp+E0h]
73         int callback; // [esp+E4h] [ebp+E4h]
74         int basic_string; // [esp+E8h] [ebp+E8h]
75         int other_frame; // [esp+F4h] [ebp+F4h]
76         int callback; // [esp+F8h] [ebp+F8h]
77         int basic_string; // [esp+FC] [ebp+FC]
78         int other_frame; // [esp+100h] [ebp+100h]
79         int callback; // [esp+104h] [ebp+104h]
80         int basic_string; // [esp+108h] [ebp+108h]
81         int other_frame; // [esp+114h] [ebp+114h]
82         int callback; // [esp+118h] [ebp+118h]
83         int basic_string; // [esp+11Ch] [ebp+11Ch]
84         int other_frame; // [esp+120h] [ebp+120h]
85         int callback; // [esp+124h] [ebp+124h]
86         int basic_string; // [esp+128h] [ebp+128h]
87         int other_frame; // [esp+134h] [ebp+134h]
88         int callback; // [esp+138h] [ebp+138h]
89         int basic_string; // [esp+13Ch] [ebp+13Ch]
90         int other_frame; // [esp+140h] [ebp+140h]
91         int callback; // [esp+144h] [ebp+144h]
92         int basic_string; // [esp+148h] [ebp+148h]
93         int other_frame; // [esp+154h] [ebp+154h]
94         int callback; // [esp+158h] [ebp+158h]
95         int basic_string; // [esp+15Ch] [ebp+15Ch]
96         int other_frame; // [esp+160h] [ebp+160h]
97         int callback; // [esp+164h] [ebp+164h]
98         int basic_string; // [esp+168h] [ebp+168h]
99         int other_frame; // [esp+174h] [ebp+174h]
100        int callback; // [esp+178h] [ebp+178h]
101        int basic_string; // [esp+17Ch] [ebp+17Ch]
102        int other_frame; // [esp+180h] [ebp+180h]
103        int callback; // [esp+184h] [ebp+184h]
104        int basic_string; // [esp+188h] [ebp+188h]
105        int other_frame; // [esp+194h] [ebp+194h]
106        int callback; // [esp+198h] [ebp+198h]
107        int basic_string; // [esp+19Ch] [ebp+19Ch]
108        int other_frame; // [esp+200h] [ebp+200h]
109        int callback; // [esp+204h] [ebp+204h]
110        int basic_string; // [esp+208h] [ebp+208h]
111        int other_frame; // [esp+214h] [ebp+214h]
112        int callback; // [esp+218h] [ebp+218h]
113        int basic_string; // [esp+21Ch] [ebp+21Ch]
114        int other_frame; // [esp+220h] [ebp+220h]
115        int callback; // [esp+224h] [ebp+224h]
116        int basic_string; // [esp+228h] [ebp+228h]
117        int other_frame; // [esp+234h] [ebp+234h]
118        int callback; // [esp+238h] [ebp+238h]
119        int basic_string; // [esp+23Ch] [ebp+23Ch]
120        int other_frame; // [esp+240h] [ebp+240h]
121        int callback; // [esp+244h] [ebp+244h]
122        int basic_string; // [esp+248h] [ebp+248h]
123        int other_frame; // [esp+254h] [ebp+254h]
124        int callback; // [esp+258h] [ebp+258h]
125        int basic_string; // [esp+25Ch] [ebp+25Ch]
126        int other_frame; // [esp+260h] [ebp+260h]
127        int callback; // [esp+264h] [ebp+264h]
128        int basic_string; // [esp+268h] [ebp+268h]
129        int other_frame; // [esp+274h] [ebp+274h]
130        int callback; // [esp+278h] [ebp+278h]
131        int basic_string; // [esp+27Ch] [ebp+27Ch]
132        int other_frame; // [esp+280h] [ebp+280h]
133        int callback; // [esp+284h] [ebp+284h]
134        int basic_string; // [esp+288h] [ebp+288h]
135        int other_frame; // [esp+294h] [ebp+294h]
136        int callback; // [esp+298h] [ebp+298h]
137        int basic_string; // [esp+29Ch] [ebp+29Ch]
138        int other_frame; // [esp+300h] [ebp+300h]
139        int callback; // [esp+304h] [ebp+304h]
140        int basic_string; // [esp+308h] [ebp+308h]
141        int other_frame; // [esp+314h] [ebp+314h]
142        int callback; // [esp+318h] [ebp+318h]
143        int basic_string; // [esp+31Ch] [ebp+31Ch]
144        int other_frame; // [esp+320h] [ebp+320h]
145        int callback; // [esp+324h] [ebp+324h]
146        int basic_string; // [esp+328h] [ebp+328h]
147        int other_frame; // [esp+334h] [ebp+334h]
148        int callback; // [esp+338h] [ebp+338h]
149        int basic_string; // [esp+33Ch] [ebp+33Ch]
150        int other_frame; // [esp+340h] [ebp+340h]
151        int callback; // [esp+344h] [ebp+344h]
152        int basic_string; // [esp+348h] [ebp+348h]
153        int other_frame; // [esp+354h] [ebp+354h]
154        int callback; // [esp+358h] [ebp+358h]
155        int basic_string; // [esp+35Ch] [ebp+35Ch]
156        int other_frame; // [esp+360h] [ebp+360h]
157        int callback; // [esp+364h] [ebp+364h]
158        int basic_string; // [esp+368h] [ebp+368h]
159        int other_frame; // [esp+374h] [ebp+374h]
160        int callback; // [esp+378h] [ebp+378h]
161        int basic_string; // [esp+37Ch] [ebp+37Ch]
162        int other_frame; // [esp+380h] [ebp+380h]
163        int callback; // [esp+384h] [ebp+384h]
164        int basic_string; // [esp+388h] [ebp+388h]
165        int other_frame; // [esp+394h] [ebp+394h]
166        int callback; // [esp+398h] [ebp+398h]
167        int basic_string; // [esp+39Ch] [ebp+39Ch]
168        int other_frame; // [esp+400h] [ebp+400h]
169        int callback; // [esp+404h] [ebp+404h]
170        int basic_string; // [esp+408h] [ebp+408h]
171        int other_frame; // [esp+414h] [ebp+414h]
172        int callback; // [esp+418h] [ebp+418h]
173        int basic_string; // [esp+41Ch] [ebp+41Ch]
174        int other_frame; // [esp+420h] [ebp+420h]
175        int callback; // [esp+424h] [ebp+424h]
176        int basic_string; // [esp+428h] [ebp+428h]
177        int other_frame; // [esp+434h] [ebp+434h]
178        int callback; // [esp+438h] [ebp+438h]
179        int basic_string; // [esp+43Ch] [ebp+43Ch]
180        int other_frame; // [esp+440h] [ebp+440h]
181        int callback; // [esp+444h] [ebp+444h]
182        int basic_string; // [esp+448h] [ebp+448h]
183        int other_frame; // [esp+454h] [ebp+454h]
184        int callback; // [esp+458h] [ebp+458h]
185        int basic_string; // [esp+45Ch] [ebp+45Ch]
186        int other_frame; // [esp+460h] [ebp+460h]
187        int callback; // [esp+464h] [ebp+464h]
188        int basic_string; // [esp+468h] [ebp+468h]
189        int other_frame; // [esp+474h] [ebp+474h]
190        int callback; // [esp+478h] [ebp+478h]
191        int basic_string; // [esp+47Ch] [ebp+47Ch]
192        int other_frame; // [esp+480h] [ebp+480h]
193        int callback; // [esp+484h] [ebp+484h]
194        int basic_string; // [esp+488h] [ebp+488h]
195        int other_frame; // [esp+494h] [ebp+494h]
196        int callback; // [esp+498h] [ebp+498h]
197        int basic_string; // [esp+49Ch] [ebp+49Ch]
198        int other_frame; // [esp+500h] [ebp+500h]
199        int callback; // [esp+504h] [ebp+504h]
200        int basic_string; // [esp+508h] [ebp+508h]
201        int other_frame; // [esp+514h] [ebp+514h]
202        int callback; // [esp+518h] [ebp+518h]
203        int basic_string; // [esp+51Ch] [ebp+51Ch]
204        int other_frame; // [esp+520h] [ebp+520h]
205        int callback; // [esp+524h] [ebp+524h]
206        int basic_string; // [esp+528h] [ebp+528h]
207        int other_frame; // [esp+534h] [ebp+534h]
208        int callback; // [esp+538h] [ebp+538h]
209        int basic_string; // [esp+53Ch] [ebp+53Ch]
210        int other_frame; // [esp+540h] [ebp+540h]
211        int callback; // [esp+544h] [ebp+544h]
212        int basic_string; // [esp+548h] [ebp+548h]
213        int other_frame; // [esp+554h] [ebp+554h]
214        int callback; // [esp+558h] [ebp+558h]
215        int basic_string; // [esp+55Ch] [ebp+55Ch]
216        int other_frame; // [esp+560h] [ebp+560h]
217        int callback; // [esp+564h] [ebp+564h]
218        int basic_string; // [esp+568h] [ebp+568h]
219        int other_frame; // [esp+574h] [ebp+574h]
220        int callback; // [esp+578h] [ebp+578h]
221        int basic_string; // [esp+57Ch] [ebp+57Ch]
222        int other_frame; // [esp+580h] [ebp+580h]
223        int callback; // [esp+584h] [ebp+584h]
224        int basic_string; // [esp+588h] [ebp+588h]
225        int other_frame; // [esp+594h] [ebp+594h]
226        int callback; // [esp+598h] [ebp+598h]
227        int basic_string; // [esp+59Ch] [ebp+59Ch]
228        int other_frame; // [esp+600h] [ebp+600h]
229        int callback; // [esp+604h] [ebp+604h]
230        int basic_string; // [esp+608h] [ebp+608h]
231        int other_frame; // [esp+614h] [ebp+614h]
232        int callback; // [esp+618h] [ebp+618h]
233        int basic_string; // [esp+61Ch] [ebp+61Ch]
234        int other_frame; // [esp+620h] [ebp+620h]
235        int callback; // [esp+624h] [ebp+624h]
236        int basic_string; // [esp+628h] [ebp+628h]
237        int other_frame; // [esp+634h] [ebp+634h]
238        int callback; // [esp+638h] [ebp+638h]
239        int basic_string; // [esp+63Ch] [ebp+63Ch]
240        int other_frame; // [esp+640h] [ebp+640h]
241        int callback; // [esp+644h] [ebp+644h]
242        int basic_string; // [esp+648h] [ebp+648h]
243        int other_frame; // [esp+654h] [ebp+654h]
244        int callback; // [esp+658h] [ebp+658h]
245        int basic_string; // [esp+65Ch] [ebp+65Ch]
246        int other_frame; // [esp+660h] [ebp+660h]
247        int callback; // [esp+664h] [ebp+664h]
248        int basic_string; // [esp+668h] [ebp+668h]
249        int other_frame; // [esp+674h] [ebp+674h]
250        int callback; // [esp+678h] [ebp+678h]
251        int basic_string; // [esp+67Ch] [ebp+67Ch]
252        int other_frame; // [esp+680h] [ebp+680h]
253        int callback; // [esp+684h] [ebp+684h]
254        int basic_string; // [esp+688h] [ebp+688h]
255        int other_frame; // [esp+694h] [ebp+694h]
256        int callback; // [esp+698h] [ebp+698h]
257        int basic_string; // [esp+69Ch] [ebp+69Ch]
258        int other_frame; // [esp+700h] [ebp+700h]
259        int callback; // [esp+704h] [ebp+704h]
260        int basic_string; // [esp+708h] [ebp+708h]
261        int other_frame; // [esp+714h] [ebp+714h]
262        int callback; // [esp+718h] [ebp+718h]
263        int basic_string; // [esp+71Ch] [ebp+71Ch]
264        int other_frame; // [esp+720h] [ebp+720h]
265        int callback; // [esp+724h] [ebp+724h]
266        int basic_string; // [esp+728h] [ebp+728h]
267        int other_frame; // [esp+734h] [ebp+734h]
268        int callback; // [esp+738h] [ebp+738h]
269        int basic_string; // [esp+73Ch] [ebp+73Ch]
270        int other_frame; // [esp+740h] [ebp+740h]
271        int callback; // [esp+744h] [ebp+744h]
272        int basic_string; // [esp+748h] [ebp+748h]
273        int other_frame; // [esp+754h] [ebp+754h]
274        int callback; // [esp+758h] [ebp+758h]
275        int basic_string; // [esp+75Ch] [ebp+75Ch]
276        int other_frame; // [esp+760h] [ebp+760h]
277        int callback; // [esp+764h] [ebp+764h]
278        int basic_string; // [esp+768h] [ebp+768h]
279        int other_frame; // [esp+774h] [ebp+774h]
280        int callback; // [esp+778h] [ebp+778h]
281        int basic_string; // [esp+77Ch] [ebp+77Ch]
282        int other_frame; // [esp+780h] [ebp+780h]
283        int callback; // [esp+784h] [ebp+784h]
284        int basic_string; // [esp+788h] [ebp+788h]
285        int other_frame; // [esp+794h] [ebp+794h]
286        int callback; // [esp+798h] [ebp+798h]
287        int basic_string; // [esp+79Ch] [ebp+79Ch]
288        int other_frame; // [esp+800h] [ebp+800h]
289        int callback; // [esp+804h] [ebp+804h]
290        int basic_string; // [esp+808h] [ebp+808h]
291        int other_frame; // [esp+814h] [ebp+814h]
292        int callback; // [esp+818h] [ebp+818h]
293        int basic_string; // [esp+81Ch] [ebp+81Ch]
294        int other_frame; // [esp+820h] [ebp+820h]
295        int callback; // [esp+824h] [ebp+824h]
296        int basic_string; // [esp+828h] [ebp+828h]
297        int other_frame; // [esp+834h] [ebp+834h]
298        int callback; // [esp+838h] [ebp+838h]
299        int basic_string; // [esp+83Ch] [ebp+83Ch]
300        int other_frame; // [esp+840h] [ebp+840h]
301        int callback; // [esp+844h] [ebp+844h]
302        int basic_string; // [esp+848h] [ebp+848h]
303        int other_frame; // [esp+854h] [ebp+854h]
304        int callback; // [esp+858h] [ebp+858h]
305        int basic_string; // [esp+85Ch] [ebp+85Ch]
306        int other_frame; // [esp+860h] [ebp+860h]
307        int callback; // [esp+864h] [ebp+864h]
308        int basic_string; // [esp+868h] [ebp+868h]
309        int other_frame; // [esp+874h] [ebp+874h]
310        int callback; // [esp+878h] [ebp+878h]
311        int basic_string; // [esp+87Ch] [ebp+87Ch]
312        int other_frame; // [esp+880h] [ebp+880h]
313        int callback; // [esp+884h] [ebp+884h]
314        int basic_string; // [esp+888h] [ebp+888h]
315        int other_frame; // [esp+894h] [ebp+894h]
316        int callback; // [esp+898h] [ebp+898h]
317        int basic_string; // [esp+89Ch] [ebp+89Ch]
318        int other_frame; // [esp+900h] [ebp+900h]
319        int callback; // [esp+904h] [ebp+904h]
320        int basic_string; // [esp+908h] [ebp+908h]
321        int other_frame; // [esp+914h] [ebp+914h]
322        int callback; // [esp+918h] [ebp+918h]
323        int basic_string; // [esp+91Ch] [ebp+91Ch]
324        int other_frame; // [esp+920h] [ebp+920h]
325        int callback; // [esp+924h] [ebp+924h]
326        int basic_string; // [esp+928h] [ebp+928h]
327        int other_frame; // [esp+934h] [ebp+934h]
328        int callback; // [esp+938h] [ebp+938h]
329        int basic_string; // [esp+93Ch] [ebp+93Ch]
330        int other_frame; // [esp+940h] [ebp+940h]
331        int callback; // [esp+944h] [ebp+944h]
332        int basic_string; // [esp+948h] [ebp+948h]
333        int other_frame; // [esp+954h] [ebp+954h]
334        int callback; // [esp+958h] [ebp+958h]
335        int basic_string; // [esp+95Ch] [ebp+95Ch]
336        int other_frame; // [esp+960h] [ebp+960h]
337        int callback; // [esp+964h] [ebp+964h]
338        int basic_string; // [esp+968h] [ebp+968h]
339        int other_frame; // [esp+974h] [ebp+974h]
340        int callback; // [esp+978h] [ebp+978h]
341        int basic_string; // [esp+97Ch] [ebp+97Ch]
342        int other_frame; // [esp+980h] [ebp+980h]
343        int callback; // [esp+984h] [ebp+984h]
344        int basic_string; // [esp+988h] [ebp+988h]
345        int other_frame; // [esp+994h] [ebp+994h]
346        int callback; // [esp+998h] [ebp+998h]
347        int basic_string; // [esp+99Ch] [ebp+99Ch]
348        int other_frame; // [esp+1000h] [ebp+1000h]
349        int callback; // [esp+1004h] [ebp+1004h]
350        int basic_string; // [esp+1008h] [ebp+1008h]
351        int other_frame; // [esp+1014h] [ebp+1014h]
352        int callback; // [esp+1018h] [ebp+1018h]
353        int basic_string; // [esp+101Ch] [ebp+101Ch]
354        int other_frame; // [esp+1020h] [ebp+1020h]
355        int callback; // [esp+1024h] [ebp+1024h]
356        int basic_string; // [esp+1028h] [ebp+1028h]
357        int other_frame; // [esp+1034h] [ebp+1034h]
358        int callback; // [esp+1038h] [ebp+1038h]
359        int basic_string; // [esp+103Ch] [ebp+103Ch]
360        int other_frame; // [esp+1040h] [ebp+1040h]
361        int callback; // [esp+1044h] [ebp+1044h]
362        int basic_string; // [esp+1048h] [ebp+1048h]
363        int other_frame; // [esp+1054h] [ebp+1054h]
364        int callback; // [esp+1058h] [ebp+1058h]
365        int basic_string; // [esp+105Ch] [ebp+105Ch]
366        int other_frame; // [esp+1060h] [ebp+1060h]
367        int callback; // [esp+1064h] [ebp+1064h]
368        int basic_string; // [esp+1068h] [ebp+1068h]
369        int other_frame; // [esp+1074h] [ebp+1074h]
370        int callback; // [esp+1078h] [ebp+1078h]
371        int basic_string; // [esp+107Ch] [ebp+107Ch]
372        int other_frame; // [esp+1080h] [ebp+1080h]
373        int callback; // [esp+1084h] [ebp+1084h]
374        int basic_string; // [esp+1088h] [ebp+1088h]
375        int other_frame; // [esp+1094h] [ebp+1094h]
376        int callback; // [esp+1098h] [ebp+1098h]
377        int basic_string; // [esp+109Ch] [ebp+109Ch]
378        int other_frame; // [esp+1100h] [ebp+1100h]
379        int callback; // [esp+1104h] [ebp+1104h]
380        int basic_string; // [esp+1108h] [ebp+1108h]
381        int other_frame; // [esp+1114h] [ebp+1114h]
382        int callback; // [esp+1118h] [ebp+1118h]
383        int basic_string; // [esp+111Ch] [ebp+111Ch]
384        int other_frame; // [esp+1120h] [ebp+1120h]
385        int callback; // [esp+1124h] [ebp+1124h]
386        int basic_string; // [esp+1128h] [ebp+1128h]
387        int other_frame; // [esp+1134h] [ebp+1134h]
388        int callback; // [esp+1138h] [ebp+1138h]
389        int basic_string; // [esp+113Ch] [ebp+113Ch]
390        int other_frame; // [esp+1140h] [ebp+1140h]
391        int callback; // [esp+1144h] [ebp+1144h]
392        int basic_string; // [esp+1148h] [ebp+1148h]
393        int other_frame; // [esp+1154h] [ebp+1154h]
394        int callback; // [esp+1158h] [ebp+1158h]
395        int basic_string; // [esp+115Ch] [ebp+115Ch]
396        int other_frame; // [esp+1160h] [ebp+1160h]
397        int callback; // [esp+1164h] [ebp+1164h]
398        int basic_string; // [esp+1168h] [ebp+1168h]
399        int other_frame; // [esp+1174h] [ebp+1174h]
400        int callback; // [esp+1178h] [ebp+1178h]
401        int basic_string; // [esp+117Ch] [ebp+117Ch]
402        int other_frame; // [esp+1180h] [ebp+1180h]
403        int callback; // [esp+1184h] [ebp+1184h]
404        int basic_string; // [esp+1188h] [ebp+1188h]
405        int other_frame; // [esp+1194h] [ebp+1194h]
406        int callback; // [esp+1198h] [ebp+1198h]
407        int basic_string; // [esp+119Ch] [ebp+119Ch]
408        int other_frame; // [esp+1200h] [ebp+1200h]
409        int callback; // [esp+1204h] [ebp+1204h]
410        int basic_string; // [esp+1208h] [ebp+1208h]
411        int other_frame; // [esp+1214h] [ebp+1214h]
412        int callback; // [esp+1218h] [ebp+1218h]
413        int basic_string; // [esp+121Ch] [ebp+121Ch]
414        int other_frame; // [esp+1220h] [ebp+1220h]
415        int callback; // [esp+1224h] [ebp+1224h]
416        int basic_string; // [esp+1228h] [ebp+1228h]
417        int other_frame; // [esp+1234h] [ebp+1234h]
418        int callback; // [esp+1238h] [ebp+1238h]
419        int basic_string; // [esp+123Ch] [ebp+123Ch]
420        int other_frame; // [esp+1240h] [ebp+1240h]
421        int callback; // [esp+1244h] [ebp+1244h]
422        int basic_string; // [esp+1248h] [ebp+1248h]
423        int other_frame; // [esp+1254h] [ebp+1254h]
424        int callback; // [esp+1258h] [ebp+1258h]
425        int basic_string; // [esp+125Ch] [ebp+125Ch]
426        int other_frame; // [esp+1260h] [ebp+1260h]
427        int callback; // [esp+1264h] [ebp+1264h]
428        int basic_string; // [esp+1268h] [ebp+1268h]
429        int other_frame; // [esp+1274h] [ebp+1274h]
430        int callback; // [esp+1278h] [ebp+1278h]
431        int basic_string; // [esp+127Ch] [ebp+127Ch]
432        int other_frame; // [esp+1280h] [ebp+1280h]
433        int callback; // [esp+1284h] [ebp+1284h]
434        int basic_string; // [esp+1288h] [ebp+1288h]
435        int other_frame; // [esp+1294h] [ebp+1294h]
436        int callback; // [esp+1298h] [ebp+1298h]
437        int basic_string; // [esp+129Ch] [ebp+129Ch]
438        int other_frame; // [esp+1300h] [ebp+1300h]
439        int callback; // [esp+1304h] [ebp+1304h]
440        int basic_string; // [esp+1308h] [ebp+1308h]
441        int other_frame; // [esp+1314h] [ebp+1314h]
442        int callback; // [esp+1318h] [ebp+1318h]
443        int basic_string; // [esp+131Ch] [ebp+131Ch]
444        int other_frame; // [esp+1320h] [ebp+1320h]
445        int callback; // [esp+1324h] [ebp+1324h]
446        int basic_string; // [esp+1328h] [ebp+1328h]
447        int other_frame; // [esp+1334h] [ebp+1334h]
448        int callback; // [esp+1338h] [ebp+1338h]
449        int basic_string; // [esp+133Ch] [ebp+133Ch]
450        int other_frame; // [esp+1340h] [ebp+1340h]
451        int callback; // [esp+1344h] [ebp+1344h]
452        int basic_string; // [esp+1348h] [ebp+1348h]
453        int other_frame; // [esp+1354h] [ebp+1354h]
454        int callback; // [esp+1358h] [ebp+1358h]
455        int basic_string; // [esp+135Ch] [ebp+135Ch]
456        int other_frame; // [esp+1360h] [ebp+1360h]
457        int callback; // [esp+1364h] [ebp+1364h]
458        int basic_string; // [esp+1368h] [ebp+1368h]
459        int other_frame; // [esp+1374h] [ebp+1374h]
460        int callback; // [esp+1378h] [ebp+1378h]
461        int basic_string; // [esp+137Ch] [ebp+137Ch]
462        int other_frame; // [esp+1380h] [ebp+1380h]
463        int callback; // [esp+1384h] [ebp+1384h]
464        int basic_string; // [esp+1388h] [ebp+1388h]
465        int other_frame; // [esp+1394h] [ebp+1394h]
466        int callback; // [esp+1398h] [ebp+1398h]
467        int basic_string; // [esp+139Ch] [ebp+139Ch]
468        int other_frame; // [esp+1400h] [ebp+1400h]
469        int callback; // [esp+1404h] [ebp+1404h]
470        int basic_string; // [esp+1408h] [ebp+1408h]
471        int other_frame; // [esp+1414h] [ebp+1414h]
472        int callback; // [esp+1418h] [ebp+1418h]
473        int basic_string; // [esp+141Ch] [ebp+141Ch]
474        int other_frame; // [esp+1420h] [ebp+1420h]
475        int callback; // [esp+1424h] [ebp+1424h]
476        int basic_string; // [esp+1428h] [ebp+1428h]
477        int other_frame; // [esp+1434h] [ebp+1434h]
478        int callback; // [esp+1438h] [ebp+1438h]
479        int basic_string; // [esp+143Ch] [ebp+143Ch]
480        int other_frame; // [esp+1440h] [ebp+1440h]
481        int callback; // [esp+1444h] [ebp+1444h]
482        int basic_string; // [esp+1448h] [ebp+1448h]
483        int other_frame; // [esp+1454h] [ebp+1454h]
484        int callback; // [esp+1458h] [ebp+1458h]
485        int basic_string; // [esp+145Ch] [ebp+145Ch]
486        int other_frame; // [esp+1460h] [ebp+1460h]
487        int callback; // [esp+1464h] [ebp+1464h]
488        int basic_string; // [esp+1468h] [ebp+1468h]
489        int other_frame; // [esp+1474h] [ebp+1474h]
490        int callback; // [esp+1478h] [ebp+1478h]
491        int basic_string; // [esp+147Ch] [ebp+147Ch]
492        int other_frame; // [esp+1480h] [ebp+1480h]
493        int callback; // [esp+1484h] [ebp+1484h]
494        int basic_string; // [esp+1488h] [ebp+1488h]
495        int other_frame; // [esp+1494h] [ebp+1494h]
496        int callback; // [esp+1498h] [ebp+1498h]
497        int basic_string; // [esp+149Ch] [ebp+149Ch]
498        int other_frame; // [esp+1500h] [ebp+1500h]
499        int callback; // [esp+1504h] [ebp+
```







Sample hash: 46E4CFF5DD847E0A9AB26F2F92E89AE9E1BB14ED (file is available via VirusBay)

IDA Database File: [Uploaded to GitHub](#)

## **Comments ( 10 )**

---

1. ViottoPosted on 4:10 pm March 4, 2018

Nice review Mr. Krabs, but being the author of this software, I want to correct some mistakes you made during your analysis. 1) Regarding the legality question: I don't know where you live, but here in Europe what I am developing and selling is perfectly legal. We don't allow malicious usage, and this is why, as yourself stated, the customer licence is stored in the Remcos agent, and can be easily retrieved using the -l command. Anytime we get a report of someone abusing our software and not using it for legal means, we can promptly check and make his copy unusable anytime. We have a dedicated email just for abuse reports: [abuse@breakingsecurity.net](mailto:abuse@breakingsecurity.net) 2) You said: "size is quite large". 120 kb is large for a program which allows total control of any part of the Windows OS? It is a tiny size especially for today standards, where most of the software is bloated. I don't think you'll find other remote control software with a better size/number of functions ratio. Check TeamViewer, for example, which has just a fraction of Remcos functions. 3) In the last part you stated: "Remcos is not advanced and a killer can be easily developed when needed". Then still you forget that our focus is on legal administration and surveillance, we don't aid cybercriminals as you state. Actually, what you did is a criminal offence by european laws, which is called defamation towards a company. If you need any other clarification, we are always available. :) Best regards, Viotto BreakingSecurity.net administrator and developer

*Mr. Krabs* Posted on 4:24 pm March 4, 2018

1) I said "Whether that is true or not is to be understood by people who have seen Remcos malware campaigns and the fact that the author also sells a crypter." for a reason. Quite a lot of malware campaigns that used Remcos so far (statistics easily obtainable from honeypots and malware databases), wouldn't you agree? 2) Well maybe you shouldn't claim that it is 90kb or 100kb on your sites then because I saw that thus me making the comment 3) Maybe you should work better on preventing abuse then because malware campaigns which relies on Remcos is not uncommon at all, further more you are selling a crypter yourself so please don't try to look innocent. :) 4) Well excuse me, I don't know on what other forums do you advertise your product but HF is definitely one of them

ViottoPosted on 6:00 pm March 4, 2018

1) Yes, it happened that Remcos has been used in malware campaigns, and anytime we got a report, we banned licence and immediately stopped the whole campaign. Also guns get used for a lot of bad things, but if a manufacturer produces a gun which works well, it is not his fault if it is used in a robbery, isn't it? :) As gun manufacturers do, we do also print a serial code on each one of the licenced software. We have got the majority of customers who use our software respecting our contract and terms, is it so strange? The majority of these attacks have been done using the cracked version of our tool, so we have been a victim of it as well in those cases. 2) The normal Remcos stub (if disabling compression), in v2.0.2 is 108kb. You analysed the relocation stub which is 120kb more. In the Free version, size is even smaller, around 70 kb. So, in the site I wrote: around 100 kb, because it depends on settings such as stub type and compression. In no case goes bigger then 120kb, which as you know is a tiny size even for floppy disks standards. 3) Yes, and as Remcos, Octopus Protector can be used for many different purposes. Same terms of services applies to that. I think we did all possible measures to prevent abuse, but if you have got some extra idea, I will hear ;) 4) We don't sell on HackForums, that is just one of the places where we do advertise this product. We do sell just on our own site. Then mostly we do cooperate with companies which don't use forums. Best regards, Viotto  
BreakingSecurity.net administrator and developer

*Avira Pueple Wormetherth* Posted on 9:44 am March 6, 2018

1: Why would a valid license be used in correlation with a distribution campaign? Judging from your need of an invite code, you would be expected to personally filter out any suspicious customers (assuming the invite request is for customer validation). 2(referring to your point 3): Sales of such tools (Octopus Protector) wouldn't be so aggravating if the toys associated with the tool wasn't so blatantly forged. Crypters always have, and always will be used for illicit purposes only; a legitimate network owner would never require such a tool to bypass any third party protection under his/her own command. If you wish to defend this claim with something along the lines of "this tool is not for protection of malware, rather protection of legitimate executables," then why has your official channel uploaded a blatant AV bypass video?(<https://www.youtube.com/watch?v=BVxQxSfNJXQ>) Something a legitimate developer would never need. Also referring to a line in your description: "An executable file gets totally encrypted and protected from human and antivirus analysis." preventing a human from tampering I would understand, but why would a legitimate executable require protection from any form of AV analysis procedures? Overall your attempt to claim your malware legal, was extremely cringe. Anyone who sells a R.A.T alongside a crypter, is clearly in it for blackhat money.

2. Viotto Posted on 8:43 pm April 10, 2018

Hello mr Avira, late reply from me here but I just noticed your comment. Regarding your points: 1: Yes, our customers can purchase just if registered (and registration requires an invite code), so we do filter out suspicious customers, or ban them if we notice malicious activity after they purchased. Many attacks have been carried out using an old, cracked Remcos version, the cracker removed abuse protection so unfortunately this wasn't under our control. We have been damaged by this as well, since malicious users abused our software. 2: Crypters have many uses, such as protecting files from analysis and cracking. Since I own a cybersecurity company which involves hacking as well (in an ethical way), my Octopus Protector can be used against many kinds of analysis, including AV analysis. This is however not illegal, unless you are using it against unaware users. The video is a demonstration that even stronger AV protections can be bypassed. If your point was true, then even Metasploit would be illegal, but it isn't. Best regards, Viotto

o AnaI Posted on 11:19 am April 14, 2018

Viotto, 1. You clearly say that "Also guns get used for a lot of bad things, but if a manufacturer produces a gun which works well, it is not his fault if it is used in a robbery, isn't it?" In software, you have a lot more control over your customers, guns are a physical piece of metal, which makes your argument invalid. In software, you can place for example, an alert to the person launching the file on another computer that they are executing a file that can take control of your computer? Also you state "Yes, it happened that Remcos has been used in malware campaigns, and anytime we got a report, we banned licence and immediately stopped the whole campaign." Instead of only getting reports from a third party, why aren't you taking any action yourself cracking on the suspicious/malicious users? I clearly gave you an idea how to crack down on ANY malicious campaign, but obviously you are not going to invent this, because all your customers are using the product in a malicious manner, thus you don't want to take any action against it other than the bad reports. 2. You state "my Octopus Protector can be used against many kinds of analysis, including AV analysis" Why does your product need to make the files "crypted" in the crypter, undetected from anti-viruses? Only thing they do is analyze the file, see if it can be used maliciously, if yes, they detect the file. Every update of your product makes the stub undetected, which isn't required in order to control another persons computer, since the anti-viruses can be disabled manually on the "targets" PC. You made a video of an anti-viruses scanning your file after you put it thru your "crypter". Why is this relevant? Why aren't any other native packers making their programs undetected from anti-viruses if they are not used in a malicious manner?

o *Avira Pueple Wormetherth* Posted on 12:55 am April 18, 2018

Why would any security product need to be bypassed in a controlled environment? Your "my Octopus Protector can be used against many kinds of analysis, including AV analysis" could be argued against with something such as: Why would you need to protect your customer's file from a legitimate researcher? No one employed by a security company would wish to distribute your file; they are checking whether it is a potential threat to their customer base. If you really wished to portay a legitimate image, you could follow in the footsteps of LuminosityLink by forcing a pre-installation watermark. Functions such as: RunPE & "ReplacePE" play a major role in most modern malware campaigns. How would these methods aid in preventing analysis of a legimitmate file? Forgive the 8 day delay, and don't see this reply as a form of attack. I am merely trying to get your side of things

### 3. *Max*Posted on 10:11 pm October 13, 2018

"Only thing they do is analyze the file, see if it can be used maliciously, if yes, they detect the file." Really? Are we all to believe that? Unfortunately I disagree, because the AV industry has for a very very long time not done its homework regarding false positives. Some of the assumptions made in AV heuristics are equally cringeworthy as the apologetic tone of Viotto above. Nice wording, by the way. Something "can be used maliciously" and AV vendors "detect the file". Sounds easy. But there isn't even an industry-wide standard on what that first term means, let alone a standard on how things get detected. Serious AV testing is just in its infancy, despite AV testers having been around for quite some time. Oh, and a lot of things "can be used maliciously", but kitchen knives are still sold, despite having been used in murders (the same thing goes for a lot of blunt tools from any workshop). Of course that's the very reason you decided to use "can be used maliciously" as opposed to "is malicious". Because the world is not black and white and you know it. But when it comes to false positives no one cares and the world is black and white after all. Sure, detecting a RAT, even as PUA, is perfectly (!) fine. But AV vendors' analysis capacities have long been overwhelmed. Which leads to copying of detections [1] within the industry, which is not only evidenced by that infamous Kaspersky experiment a few years ago. But once propagated throughout the industry, these detections rarely get corrected even if they turn out to be false positives. Detections propagate, their removal in case of false positives does not. That's a real problem. Both for small independent software creators and their customers. Even though the original AV vendor, detecting an item first, may correct the mistake, it takes considerable effort to remove such a detection from all those dozens different AV products. And - as I pointed out - detections (all types!) in the AV industry propagate. Now, while sample sharing is a good thing, copying detections is not. There is a disparity between how easy it is for an AV to detect a given executable file as malicious (and no, most users won't even bother reading the fingerprint, so: detected by AV == malicious) and for that detection to spread throughout the industry and how effing hard it is to ever get a wrongful detection (which often borders on the commercial equivalent of character assassination, given that smaller companies are already disadvantaged as it is) out again. While in order to end up in detection of just about every AV product out there all you need to do is that you use some allegedly "shady APIs" or end up with some bored AV analyst looking at your file and making the wrong decision within the short allotted time (assuming it's not some inherently stupid sandbox with some heuristics created by equally bored analysts mounted on top: AI ... automated imbecility), in order to get out again you'd have to contact every single effing AV vendor individually and then it typically takes a long time to receive a response, if any. Let alone the time it takes to get the detection removed - if one manages it at all - or to "argue" one's case. This is an uphill battle for any legitimate software vendor having his product(s) end up in detection. And no AV vendor or the industry overall sees that as a problem. Because they act like a sheriff in the wild west, embodying the law (or so they think) and therefore what they do must

be The Right Thing™. Even if it effectively amounts to slander and the commercial equivalent of character assassination. Seriously, if AV vendors are the good guys, then why the heck can't they even play by commonly accepted rules or by time-proven paradigms such as "in dubio pro reo"? Now what the makers of Remcos RAT do is shady. No doubt about it. And selling the crypter alongside the RAT doesn't make it look any better. But PLEASE take off your rose-colored glasses, AV people! Small independent software vendors have no legal department. Heck, open source developers don't even have a sales or marketing department. Being detected by an AV is the death knell to many small software creators and is very demotivating indeed for open source developers. Don't take adding something INTO detection so lightly, would ya? And get your act together to provide one single contact to report false positives instead of having small ISVs running the gauntlet with understaffed "support departments" trying to shield actual analysts from seeing false positive reports. [1] And by copying I am not implying that this is done verbatim or so. Alone the difference in how the various engines work would make that hard. However, the fact that files are shared between AV vendors and that VirusTotal also shares files with the vendors leads to the observation that one of the heuristics of whether a file ought to end up in detection is apparently a.) the trust vendor A places in vendor X detecting that file already and/or b.) a threshold value of how many AV vendors already have said file in detection, leading to a vendor also detecting it. Now you should be able to see the vicious circle this creates, not to mention the issue that the turnaround for detections being copied is relatively fast, whereas alone the attempt by a smaller independent software developer to have a detection of his/her software removed takes AGES in comparison. Not to mention that you need to contact each and every AV vendor individually. Great stuff!

4. *KrabMaster2* Posted on 7:59 pm April 26, 2019

Nice website. I especially like your Revcode posts.

5. *hwac121* Posted on 8:01 pm June 10, 2019

Who the hell wants to purchase a software that has a backdoor built-in purposely by the developer so they can deactivate your software whenever they like? As for the author of this article...ALL hacking tools are used by both white hat and black hat...the only way to be a good white hat hacker and security professional is to understand and be able to use the exact same tools the criminals do.

---

[View Comments \(10\)...](#)