

# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

## FACULTAD DE CIENCIAS

### ANÁLISIS DE SOFTWARE MALICIOSO

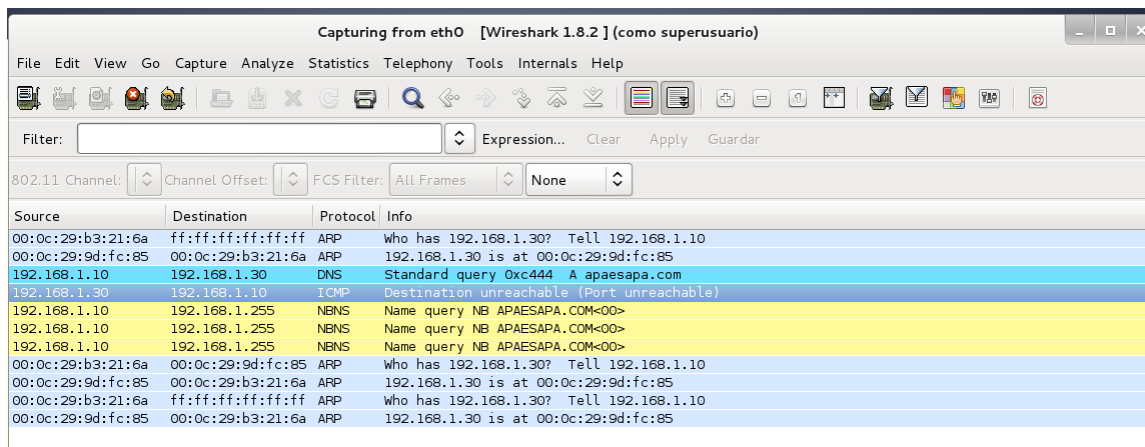
#### TAREA OPCIONAL: GETDOWN

Páes Alcalá Alma Rosa

23 de octubre de 2019

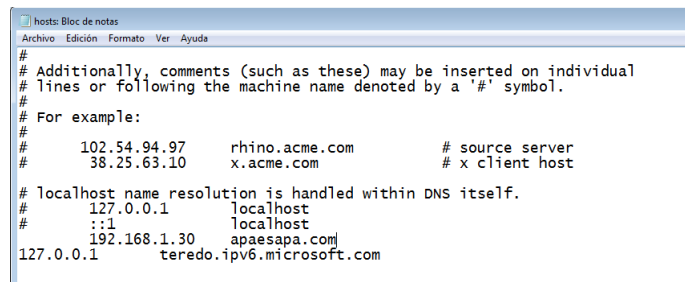
## Recursos solicitados por la pieza

Dada la pieza *aPaes.exe*, debemos verificar primero qué acciones realiza y qué recursos solicita cuando su ejecución es exitosa. Para eso, ejecutamos la pieza y leemos el tráfico generado en Wireshark -instalado en la máquina Debian-:



Source	Destination	Protocol	Info
00:0c:29:b3:21:6a	ff:ff:ff:ff:ff:ff	ARP	Who has 192.168.1.30? Tell 192.168.1.10
00:0c:29:9d:fc:85	00:0c:29:b3:21:6a	ARP	192.168.1.30 is at 00:0c:29:9d:fc:85
192.168.1.10	192.168.1.30	DNS	Standard query 0xc444 A apaesapa.com
192.168.1.30	192.168.1.10	ICMP	Destination unreachable (Port unreachable)
192.168.1.10	192.168.1.255	NBNS	Name query NB APAESAPA.COM<00>
192.168.1.10	192.168.1.255	NBNS	Name query NB APAESAPA.COM<00>
192.168.1.10	192.168.1.255	NBNS	Name query NB APAESAPA.COM<00>
00:0c:29:b3:21:6a	00:0c:29:9d:fc:85	ARP	Who has 192.168.1.30? Tell 192.168.1.10
00:0c:29:9d:fc:85	00:0c:29:b3:21:6a	ARP	192.168.1.30 is at 00:0c:29:9d:fc:85
00:0c:29:b3:21:6a	ff:ff:ff:ff:ff:ff	ARP	Who has 192.168.1.30? Tell 192.168.1.10
00:0c:29:9d:fc:85	00:0c:29:b3:21:6a	ARP	192.168.1.30 is at 00:0c:29:9d:fc:85

Figura 1: La pieza solicita conectarse a un dominio, pero no existe un registro de éste.



```
# hosts: Bloc de notas
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com   # source server
#       38.25.63.10      x.acme.com       # x client host
#
# localhost name resolution is handled within DNS itself.
#
#       127.0.0.1         localhost
#       ::1               localhost
#       192.168.1.30     apaesapa.com
#       127.0.0.1         teredo.ipv6.microsoft.com
```

Figura 2: Editamos el archivo *hosts* en Windows, indicando que la solicitud de conexión al dominio *apaesapa.com* debe redirigirse a la dirección IP de la máquina Debian

Source	Destination	Protocol	Info
192.168.1.10	192.168.1.30	TCP	1032 > 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
192.168.1.30	192.168.1.10	TCP	80 > 1032 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
192.168.1.10	192.168.1.30	TCP	1032 > 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
192.168.1.30	192.168.1.10	TCP	80 > 1032 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
192.168.1.10	192.168.1.30	TCP	1032 > 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
192.168.1.30	192.168.1.10	TCP	80 > 1032 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
192.168.1.30	192.168.1.10	DNS	Standard query 0x169e AAAA MalwareAnalysisLab.localdomain
00:0c:29:b3:21:6a	00:0c:29:9d:fc:85	ARP	who has 192.168.1.30? Tell 192.168.1.10
00:0c:29:9d:fc:85	00:0c:29:b3:21:6a	ARP	192.168.1.30 is at 00:0c:29:9d:fc:85
192.168.1.30	192.168.1.10	DNS	Standard query 0x169e AAAA MalwareAnalysisLab.localdomain
00:0c:29:9d:fc:85	00:0c:29:b3:21:6a	ARP	who has 192.168.1.10? Tell 192.168.1.30
192.168.1.30	192.168.1.10	DNS	Standard query 0xf5b3 AAAA MalwareAnalysisLab
00:0c:29:b3:21:6a	00:0c:29:9d:fc:85	ARP	192.168.1.10 is at 00:0c:29:b3:21:6a
192.168.1.30	192.168.1.10	DNS	Standard query 0xf5b3 AAAA MalwareAnalysisLab
00:0c:29:b3:21:6a	ff:ff:ff:ff:ff:ff	ARP	who has 192.168.1.30? Tell 192.168.1.10

Figura 3: Al volver a ejecutar, ya puede conectarse al dominio, pero solicita recursos al puerto 80, el cual está inhabilitado

```
malware@MalwareAnalysisLab: ~
Archivo Editar Ver Buscar Terminal Ayuda
root@MalwareAnalysisLab:/home/malware# /etc/init.d/apache2 start
[ ok ] Starting web server: apache2.
root@MalwareAnalysisLab:/home/malware# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp6      0      0 0:::80                  :::*                    LISTEN
tcp6      0      0 0:::443                 :::*                    LISTEN
root@MalwareAnalysisLab:/home/malware#
```

Figura 4: Inicializamos el servidor web *apache2*, y verificamos que haya conexión al puerto 80

Source	Destination	Protocol	Info
00:0c:29:b3:21:6a	ff:ff:ff:ff:ff:ff	ARP	who has 192.168.1.30? Tell 192.168.1.10
00:0c:29:9d:fc:85	00:0c:29:b3:21:6a	ARP	192.168.1.30 is at 00:0c:29:9d:fc:85
192.168.1.10	192.168.1.30	TCP	1033 > 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
192.168.1.30	192.168.1.10	TCP	80 > 1033 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=16
192.168.1.10	192.168.1.30	TCP	1033 > 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
192.168.1.10	192.168.1.30	HTTP	GET /pcfix.exe?affid=23456732-34459 HTTP/1.1
192.168.1.30	192.168.1.10	TCP	80 > 1033 [ACK] Seq=1 Ack=335 Win=15680 Len=0
192.168.1.30	192.168.1.10	HTTP	HTTP/1.1 404 Not Found (text/html)
192.168.1.10	192.168.1.30	TCP	1033 > 80 [RST, ACK] Seq=335 Ack=501 Win=0 Len=0
00:0c:29:9d:fc:85	00:0c:29:b3:21:6a	ARP	who has 192.168.1.10? Tell 192.168.1.30
00:0c:29:b3:21:6a	00:0c:29:9d:fc:85	ARP	192.168.1.10 is at 00:0c:29:b3:21:6a

Figura 5: La pieza ha logrado ejecutarse exitosamente, y vemos que ha descargado un recurso llamado *pcfix.exe*

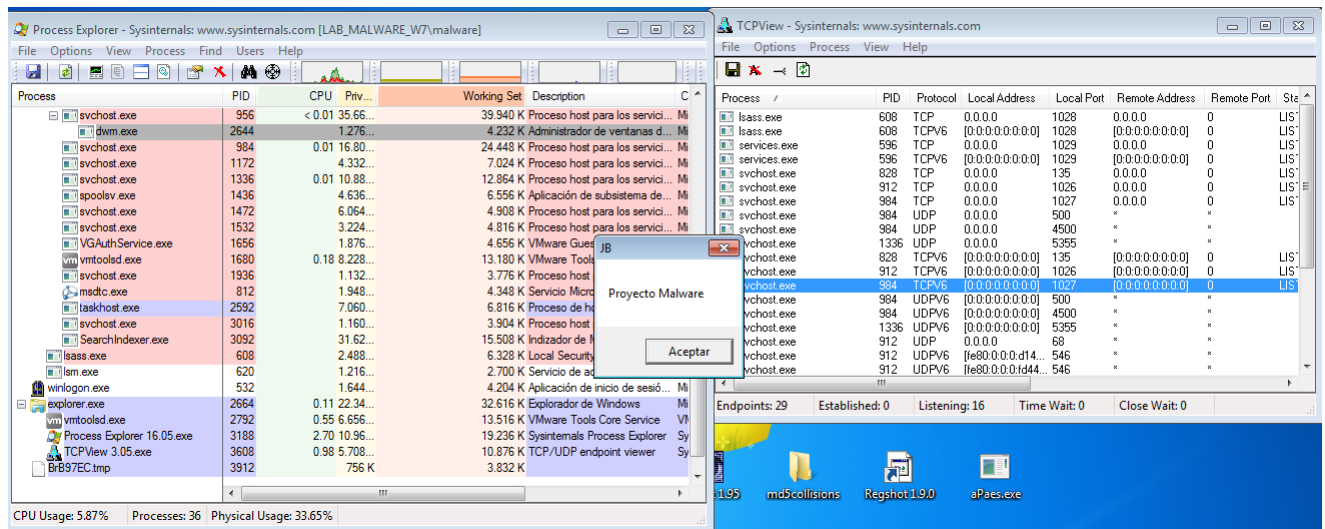


Figura 6: Al descargarse y ejecutarse, vemos esto como resultado. Previamente se había preparado el recurso solicitado en la ruta `/var/www/`, que no es más que una copia del ejecutable `HolaMundo.exe`, disponible en `/srv/ftp/`

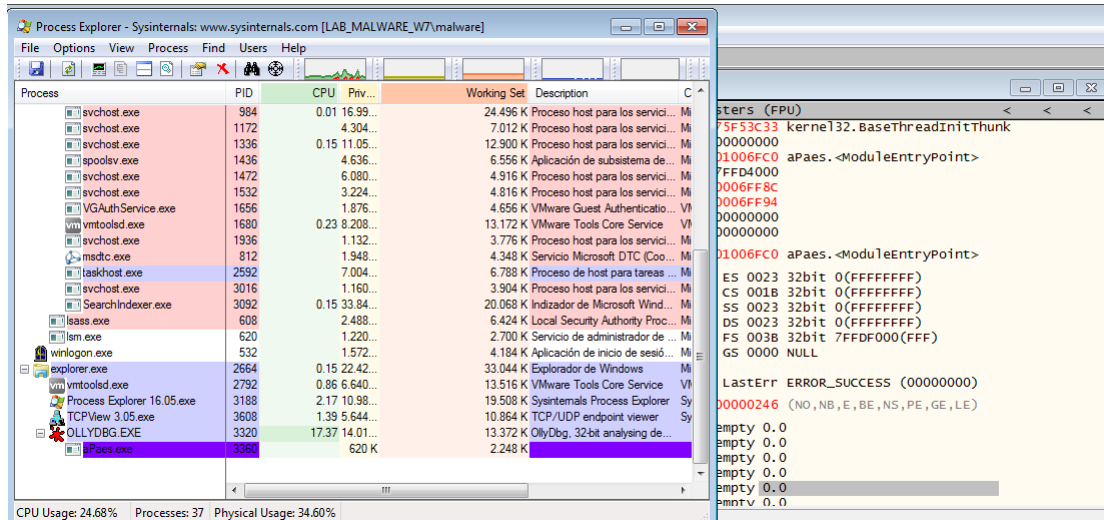


Figura 7: Ahora ejecutaremos la pieza bajo el debugger *OllyDbg*

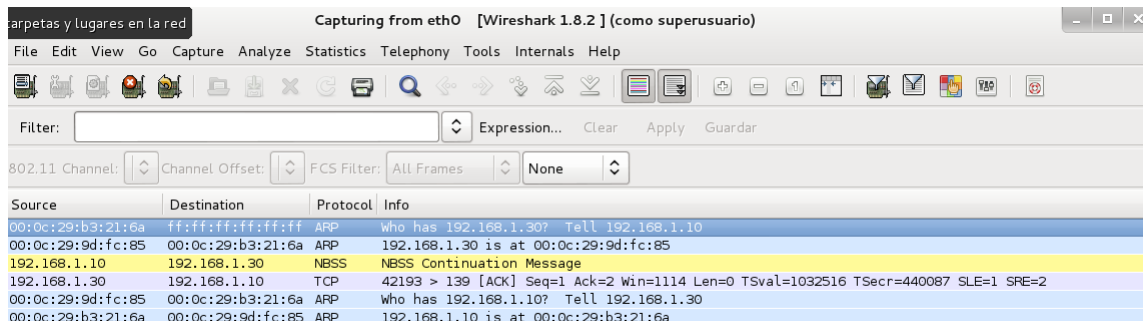


Figura 8: ...Y no realiza conexión alguna

## Modificación de la pieza

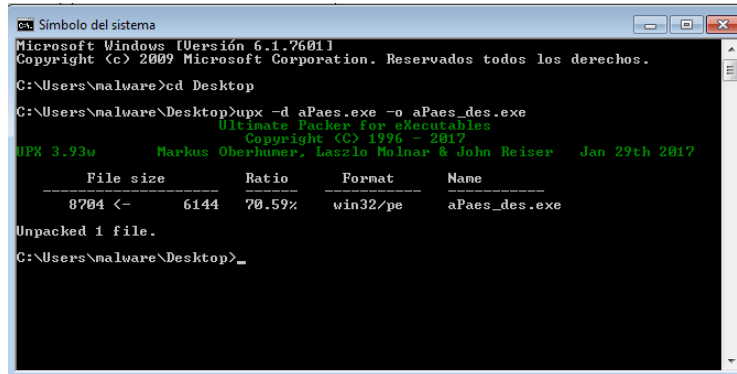


Figura 9: Para empezar a modificar la pieza, tal que se ejecute bajo un debugger o no, debemos desempaquetarla

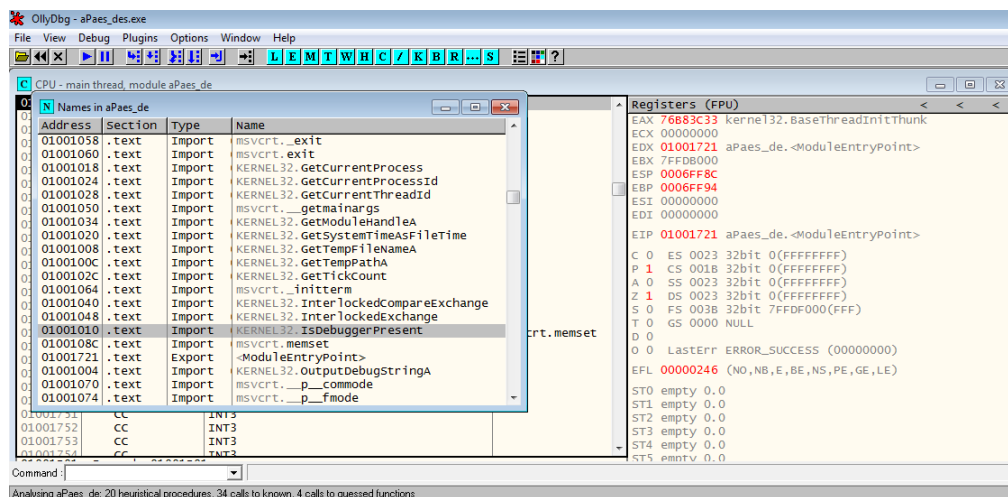


Figura 10: Abrimos la pieza desempaquetada con el debugger y localizamos la función *IsDebuggerPresent*

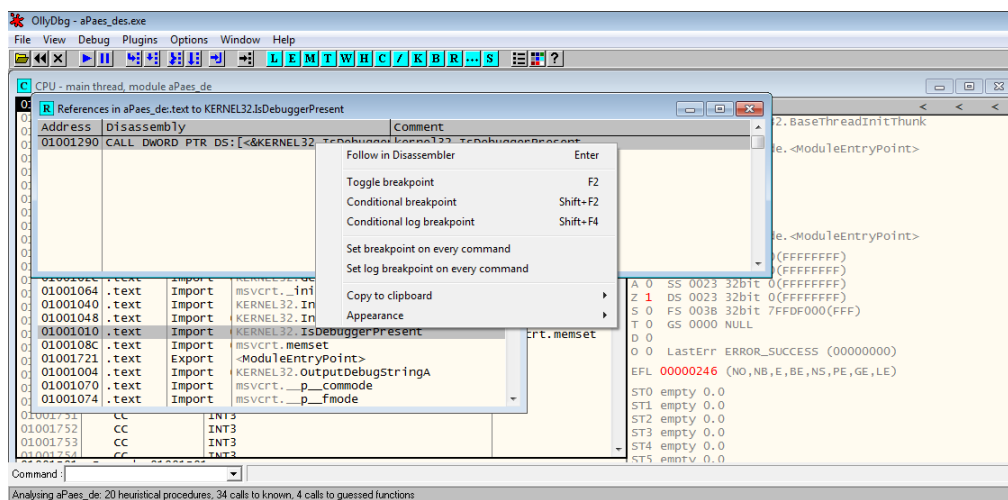


Figura 11: Entramos a la opción *Follow in Disassembler*

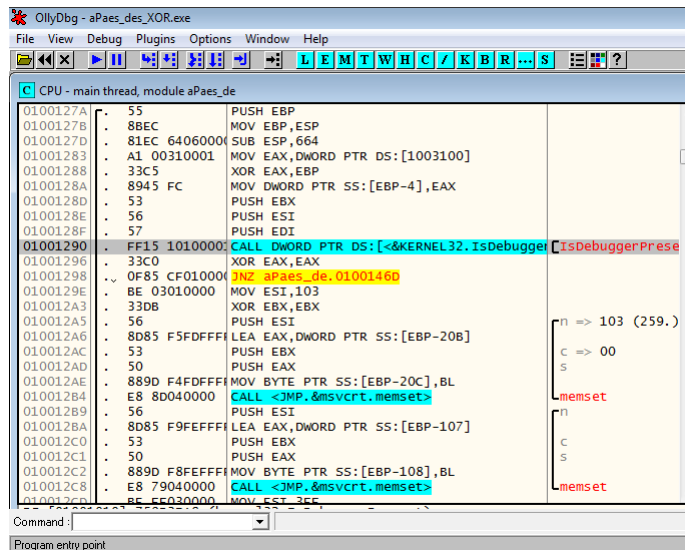


Figura 12: La instrucción 01001296 contenía un *TEST EAX,EAX*. Para lograr la correcta ejecución con o sin debugger, cambiamos la instrucción a *XOR EAX,EAX*

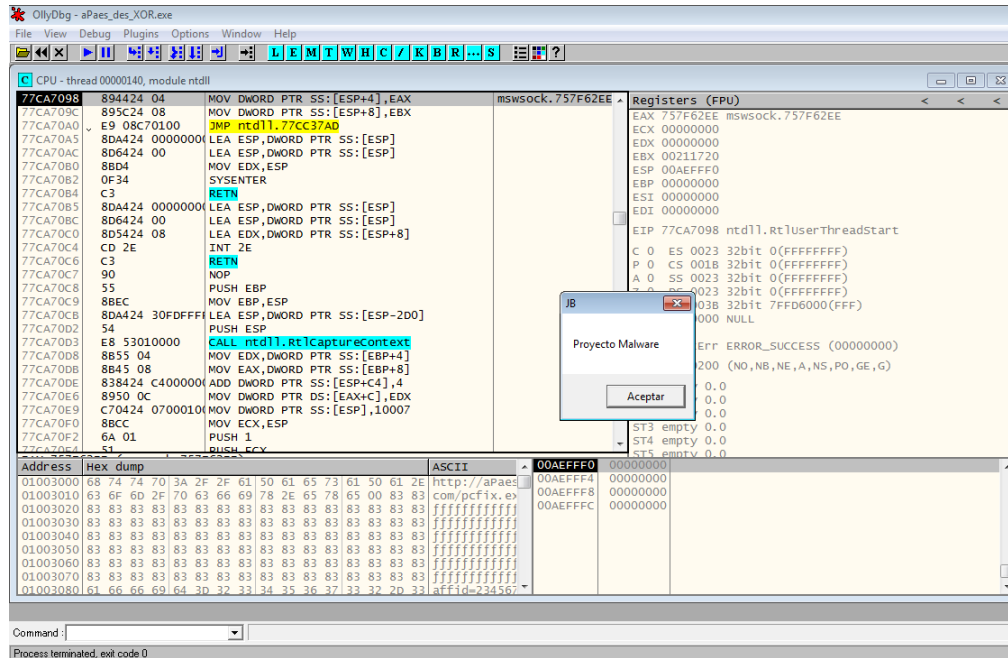


Figura 13: Y vemos que se ejecuta correctamente, aún bajo la acción de un debugger

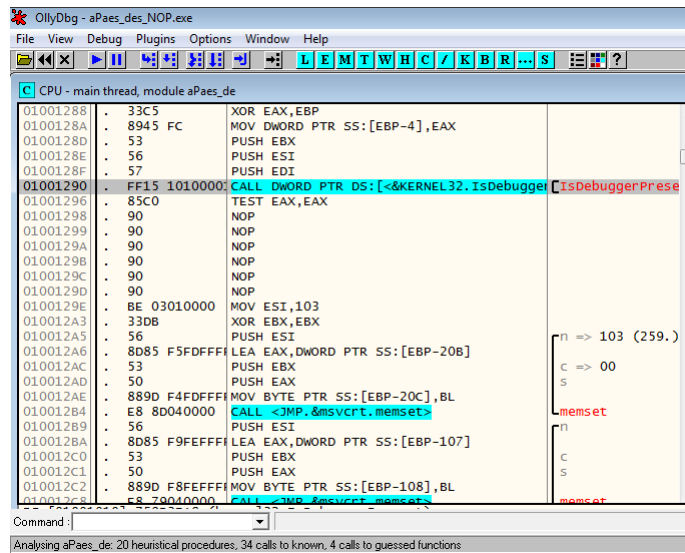


Figura 14: Para evitar que la ejecución sea nula bajo un debugger, rellenamos con *NOP*'s la instrucción que indicaba el salto hacia este conjunto de instrucciones (instrucción 01001298)

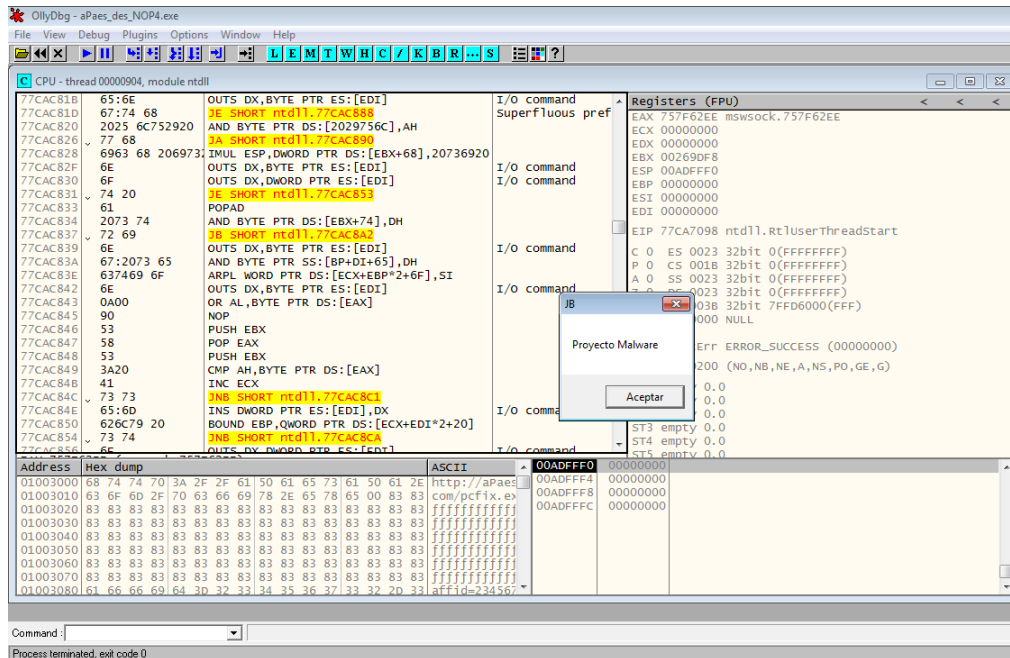


Figura 15: Y de igual manera que la otra modificación, funciona correctamente a pesar de estar bajo la acción de un debugger

## Conclusiones

La práctica de evitar la ejecución de una pieza de malware bajo la acción de un debugger es una manera sencilla y básica de prevenir su detección y eliminación; dado que existen otras maneras más especializadas de prevenir esto, tales como el empaquetado y cifrado de malware, modificación dinámica de código, bloqueo de acciones de los antivirus, etc.

Respecto a la realización de la tarea, posiblemente el objetivo de la modificación con *NOP*'s era rellenar con esta instrucción el bloque de código referente a las acciones realizadas bajo la acción de un debugger. Sin embargo, nada funcionó hasta que llené de *NOP*'s la instrucción de salto -¡funcionó!-.

## Fuentes

<https://www.kaspersky.com/resource-center/threats/combating-antivirus>