NIEYLANA, SSLEVIN

Unwrapping WildTangent Games

Version 1.1
January 2009

Table of Contents

Editor: Nieylana

Foreword

The WildTangent Wrapper is a Software Protection system developed by WildTangent Inc, which is based out of Redmond WA. The wrapper is used as a marketing technique for developers to release their software as a trial, and provide you the option of either buying the game outright or paying as you play.

Each game comes with 2 free trials, however these trials are only deducted if you have played the game for longer than 3 minutes. After you have used up both trials you must either buy the game or buy wildcoins to continue playing.

The WildTangent Wrapper also offers developers the ability to give a pre-determined about of trial play time instead of trial sessions, but the majority if not all use the trial sessions option.

All WildTangent games come with 2 executable files. The executable ending in '-WT.exe' is called the launcher, and the other executable is called the shell.

By wrapping an executable with WildTangent, the code from the application is stripped away and replaced with 00s. This stripped code is then added to the launcher executable as an encrypted overlay. If the launcher determines the user is allowed to play the game it will then rebuild the executable in memory and run it.

The methods used to store wildcoins and trial sessions use encryption and randomization very heavily so we will not be pursuing this option in the paper.

The goal of this paper is to show a couple things the developers of the wrapper overlooked, and exploit these, ultimately resulting in obtaining the original executable prior to being wrapped.

The techniques described in this paper are the result of work between myself and SSlEvIN of ARTeam, we started this project more or less just for fun, but the unique challenges provided by this wrapper quickly made it more and more of a challenge. Initially I started to pursue the possibility of resetting the trial sessions but due to the heavy use of cryptography I abandoned that route, and went for other ways. After about a week, I managed a way to defeat it, this technique is described in Target 4.

Enjoy,
Nieylana, SSLEVIN

Disclaimers

All code included with this tutorial is free to use and modify; we only ask that you mention where you found it. This tutorial is also free to distribute in its current unaltered form, with all the included supplements.

All the commercial programs used within this tutorial have been used only for the purpose of demonstrating the theories and methods described. No distribution of patched applications has been done under any media or host. The applications used were most of the times already been patched by other fellows, and cracked versions were available since a lot of time. ARTeam or the authors of the papers shouldn't be considered responsible for damages to the companies holding rights on those programs. The scope of this document as well as any other ARTeam tutorial is of sharing knowledge and teaching how to patch applications, how to bypass protections and generally speaking how to improve the RCE art. We are not releasing any cracked application.

Verification

ARTeam.esfv can be opened in the ARTeamESFVChecker to verify all files have been released by ARTeam and are unaltered. The ARTeamESFVChecker can be obtained in the release section of the ARTeam site: http://releases.accessroot.com

Table of Contents

1. Unwrapping WildTangent Games, Nieylana/SSIEvIN

1.1. Abstract

This tutorial will cover the basics on unwrapping 4 targets wrapped with the WildTangent Wrapper, each target looked at in this tutorial must be handled a different way. We will cover a standard game, 2 games with Flash Overlays (FWS/CWS, and 10JP), and an expired game.

Tools used in this tutorial include:

1. OllyDbg (Latest Version)
2. LordPE (Latest Version)
3. HexEditor (I Use 010 Editor)

1.2. Targets

The games are available for download at:

- Aces of the Galaxy: http://hp.wildgames.com
- Diego's Safari Adventure: http://hp.wildgames.com
- Dora's Carnival2: http://dell.wildgames.com
- Mahjong Quest3: http://hp.wildgames.com

1.3. Aces of the Galaxy

1.3.1 Preparation

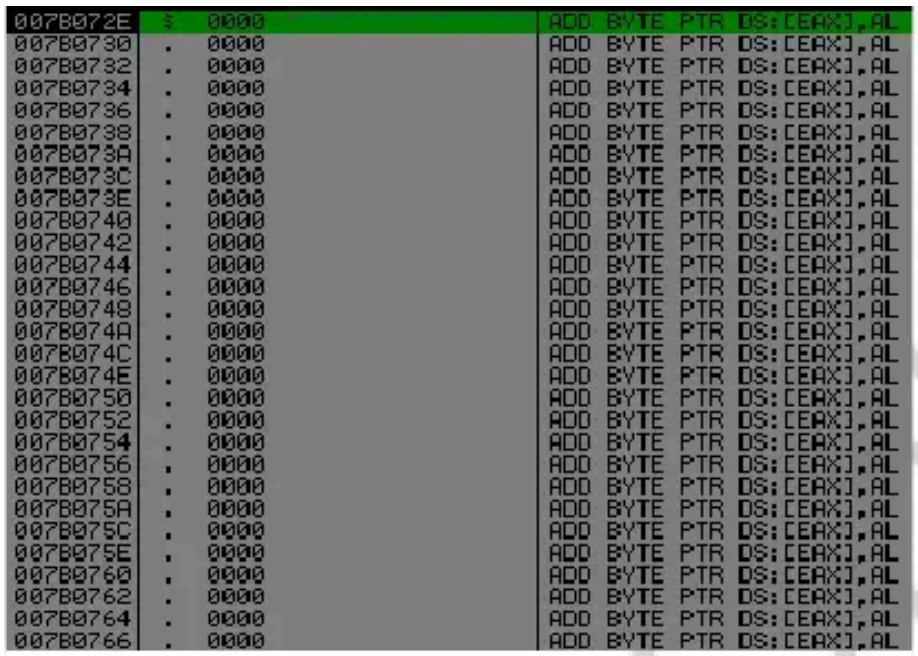
If you scan the –WT.exe executable in the target's directory with PEiD you will see that the wrapper is written if Microsoft Visual C++.

1.3.2 Checking out the target

Open up the installation directory : C:\Program Files\HP Games\Aces of the Galaxy. You will notice there is AcesOfTheGalaxy.exe and AcesOfTheGalaxy-WT.exe

If you try to run AcesOfTheGalaxy.exe, it crashes. Let's look at it in Olly to figure out why.

This is our OEP:



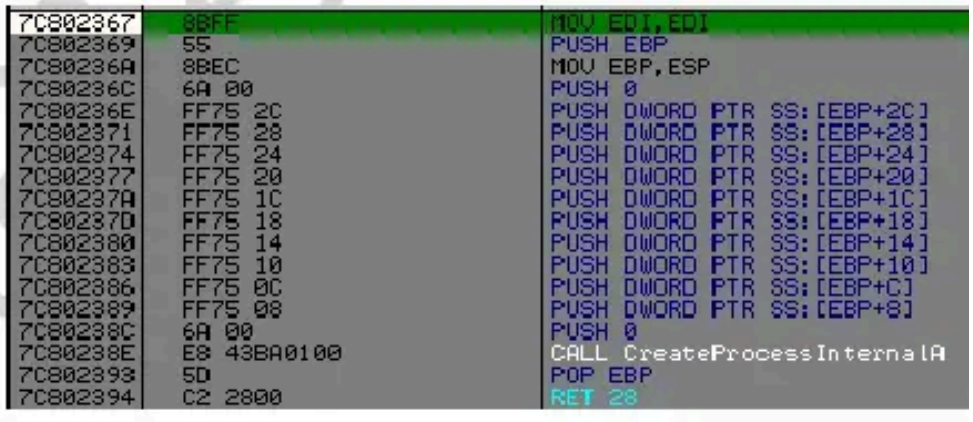
It should be clear as to why it crashes.... There's NO CODE?!.

Now if you run the AcesOfTheGalaxy-WT.exe (now called the Launcher), the application runs and you have to use tokens to play the games.

The launcher uses CreateProcessA to start AcesOfTheGalaxy.exe (The ShellEXE). Remember ShellEXE contains no code, it's has place holders where the code used to be.

By starting the process with CREATE_SUSPEND the Launcher is able to write the code into the place holders before continuing execution.

So let's open the Launcher EXE in OllyDbg, Press Ctrl+G and type CreateProcessA. Set a SWBP on this API so we know when the launcher is trying to start the shellEXE.



Press F9 to run the Application.

After the WildTangent Launcher window shows up click on Play. OllyDbg should break on the CreateProcessA API. Step with F8 until the RET28, and then step F8 one more time to return to user code.

You may have to analyze the code (Ctrl+A). You should see this:

```
0048DC8C  .  8985 24FFFFFF     MOV DWORD PTR SS:[EBP-DC],EAX
0048DC92  .  C645 FC 03        MOV BYTE PTR SS:[EBP-4],3
0048DC96  .  8D8D 08FFFFFF     LEA ECX,DWORD PTR SS:[EBP-F8]
0048DC9C  .  E8 2F3BF7FF       CALL 004017D0
0048DCA1  .  83BD 24FFFFFF 00  CMP DWORD PTR SS:[EBP-DC],0
0048DCA8  .v 0F84 96000000     JE 0048DD44
0048DCAE  .  A1 181F4B00       MOV EAX,DWORD PTR DS:[4B1F18]
0048DCB3  .  83E0 01           AND EAX,1
0048DCB6  .  F7D8              NEG EAX
0048DCB8  .  1BC0              SBB EAX,EAX
0048DCBA  .  F7D8              NEG EAX
0048DCBC  .  48                DEC EAX
0048DCBD  .  0985 48FFFFFF     MOV DWORD PTR SS:[EBP-B8],EAX
0048DCC3  .  8B8D 48FFFFFF     MOV ECX,DWORD PTR SS:[EBP-B8]
0048DCC9  .  51                PUSH ECX                          ;Arg2
0048DCCA  .  8B55 08           MOV EDX,DWORD PTR SS:[EBP+8]
0048DCCD  .  8B02              MOV EAX,DWORD PTR DS:[EDX]
0048DCCF  .  50                PUSH EAX                          ;Arg1
0048DCD0  .  8B8D 00FFFFFF     MOV ECX,DWORD PTR SS:[EBP-100]
0048DCD6  .  E8 C5000000       CALL 0048DDA0                     ;AcesOfTh.0048DDA0
0048DCDB  .  0FB6C8            MOVZX ECX,AL
0048DCDE  .  85C9              TEST ECX,ECX
0048DCE0  .v 74 2B             JE SHORT 0048DD0D
0048DCE2  .  68 60264B00       PUSH 004B2660
0048DCE7  .  E8 E1D4FFFF       CALL 0048B1CD
0048DCEC  .  83C4 04           ADD ESP,4
0048DCEF  .  8985 44FFFFFF     MOV DWORD PTR SS:[EBP-BC],EAX
0048DCF5  .  8B55 08           MOV EDX,DWORD PTR SS:[EBP+8]
0048DCF8  .  8B42 04           MOV EAX,DWORD PTR DS:[EDX+4]
0048DCFB  .  50                PUSH EAX                          ;hThread
0048DCFC  .  FF15 A8F24800     CALL DWORD PTR DS:[<&KERNEL32.ResumeThread>]  ;ResumeThread
```

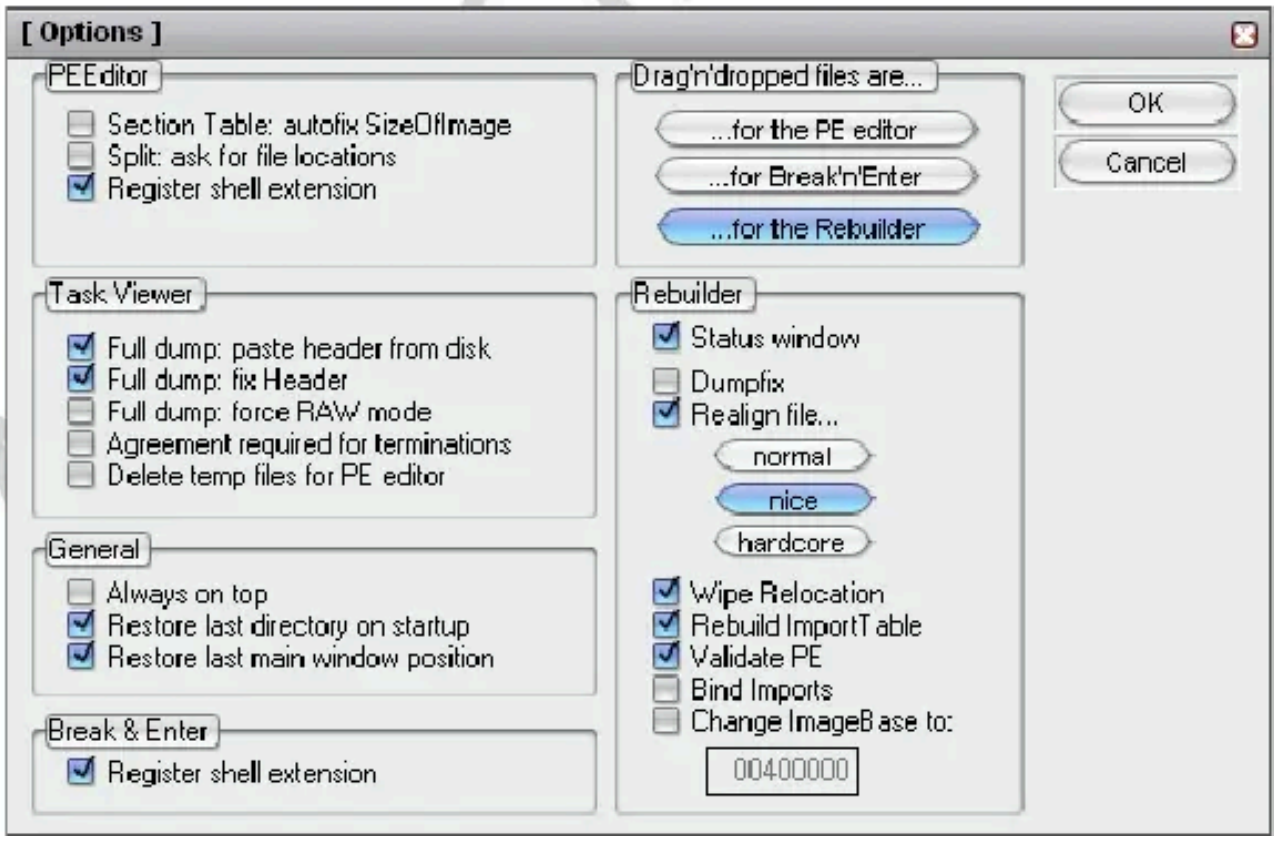
Notice the call to ResumeThread, between where we are right now, and the ResumeThread, the launcher must write all the code into the shell EXEs memory space. This occurs at line 0048DCD6.

Step all the way until the call to ResumeThread. DO NOT step over the call.

At this point we have the shell executable loaded into memory and all the code written to the process's memory space. So effectively we have the original executable in memory.

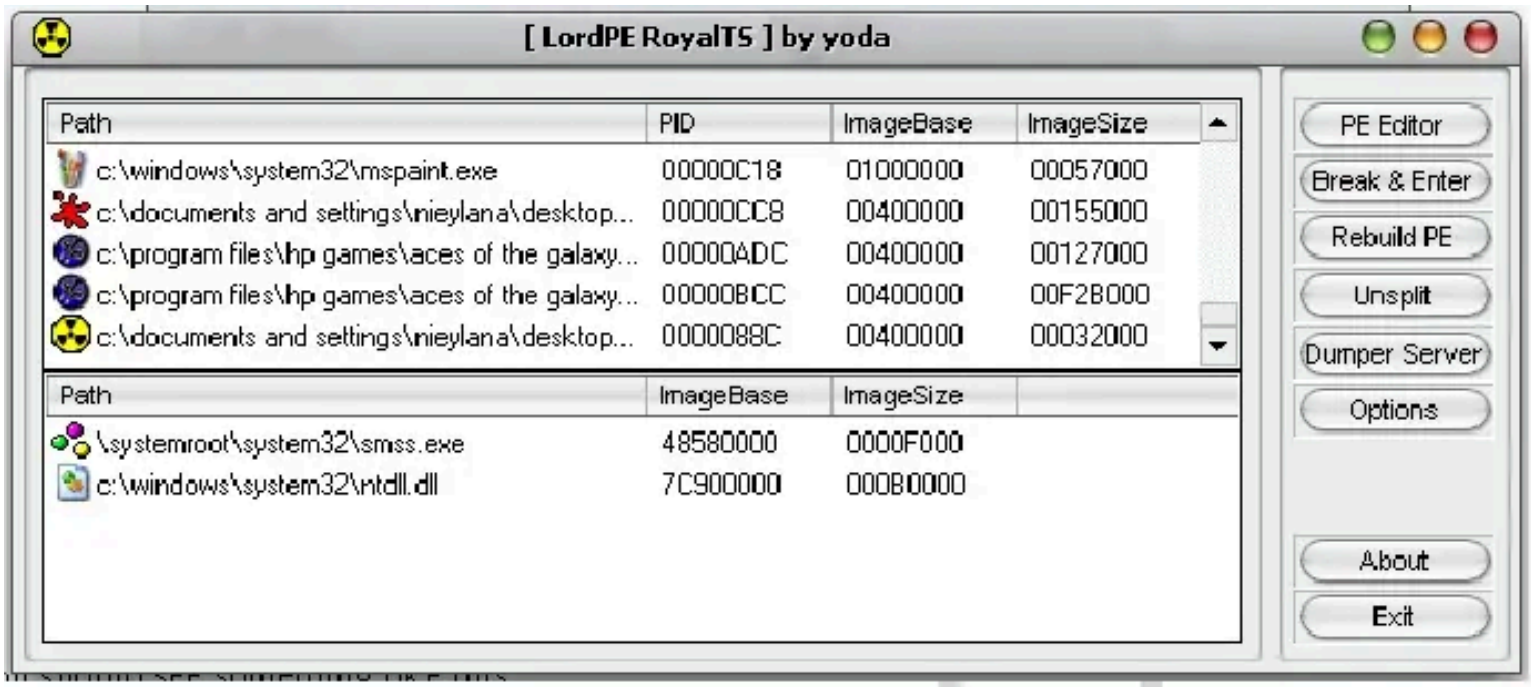
Minimize OllyDbg, we're done with it. DO NOT exit though

Open up LordPE and make sure your settings are as follows:



After your options match mine, click ok. Then scroll down to the bottom of the list of processes,

You should see something like this:



Notice the 2 Aces of the Galaxy processes, the top one is the Launcher, the bottom is the Shell.

Right click on the bottom Aces Of the Galaxy Process (AcesOfTheGalaxy.exe) and select Dump Full.

Save the dumped file to the installation directory of the program. You have successfully dumped the game. The dumped.exe will run as the full version. This is the generic method of unwrapping the games. Prolems only occure when the game is Flash based because you must re-append the Flash Overlay to the dumped executable. The next target will show how to re-append the flash overlay to the dumped executable.

1.4. Diego's Safari Adventure (FWS/CWS Overlay)

1.4.1 Preparation

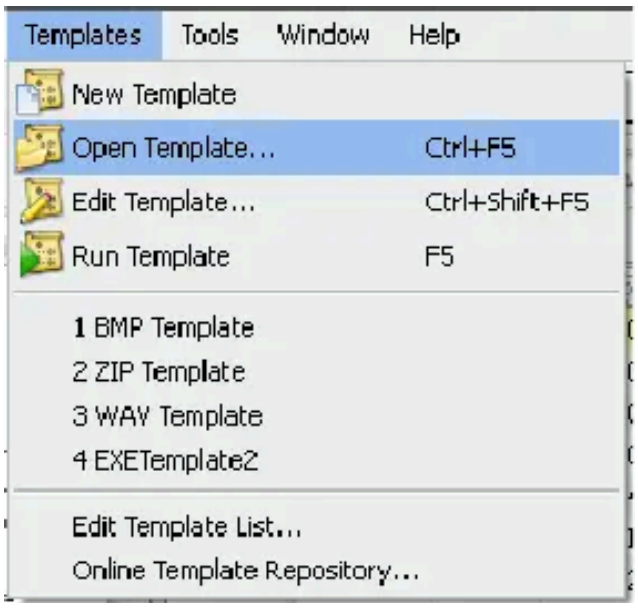
If you scan this target's launcher with PEiD you will again see that it was written in Microsoft Visual C++, but you will also notice that it has an overlay. We will find out later that the overlay it has is in fact an FWS Flash Overlay.

1.4.2 Checking Out The Target

First you need to create a dumped file like explained above, name it dumped.exe.

Re-appending the flash overlay is going to rely heavily on a hex editor. I recommend using 010 Editor (cracks available for it) because it has a template functionality which will help you easily locate the Overlay and it's size and other things. This tutorial is going to assume you have 010 Editor.

Open up the shellexe (Diego.exe) in 010 Editor, then click on Open Template



Select the EXE Template (I use EXETemplate2, available at www.sweetscape.com/010editor/templates/files/EXETemplate2.bt)

After the Template is open Press F5 to run the template on Diego.exe. You should get this:

struct IMAGE_DOS_HEADER dos_he...		0h	40h	Fg:	Bg:
UCHAR doscode[64]		40h	40h	Fg:	Bg:
DWORD MSlinkerSignatureRich[26]		80h	68h	Fg:	Bg:
struct IMAGE_NT_HEADERS nt_head...		F8h	F8h	Fg:	Bg:
struct IMAGE_SECTION_HEADER sec...		1F0h	A0h	Fg:	Bg:
BYTE textsection[2052096]		1000h	1F5000h	Fg:	Bg:
BYTE rdatasection[331776]		1F6000h	51000h	Fg:	Bg:
BYTE datasection[233472]		247000h	39000h	Fg:	Bg:
BYTE rsrcsection[180224]		280000h	2C000h	Fg:	Bg:
BYTE Overlay[1204]		2AC000h	4B4h	Fg:	Bg:

Click on the BYTE Overlay[1204]

```
2A:C000h: 46 57 53 09 AC 04 00 00 78 00 07 D0 00 00 17 70  FWS.¬...x..Ð...p
2A:C010h: 00 00 18 01 00 44 11 18 00 00 00 7F 13 CB 01 00  .....D.......Ë..
2A:C020h: 00 3C 72 64 66 3A 52 44 46 20 78 6D 6C 6E 73 3A  .<rdf:RDF xmlns:
2A:C030h: 72 64 66 3D 27 68 74 74 70 3A 2F 2F 77 77 77 2E  rdf='http://www.
2A:C040h: 77 33 2E 6F 72 67 2F 31 39 39 39 2F 30 32 2F 32  w3.org/1999/02/2
2A:C050h: 32 2D 72 64 66 2D 73 79 6E 74 61 78 2D 6E 73 23  2-rdf-syntax-ns#
2A:C060h: 27 3E 3C 72 64 66 3A 44 65 73 63 72 69 70 74 69  '><rdf:Descripti
2A:C070h: 6F 6E 20 72 64 66 3A 61 62 6F 75 74 3D 27 27 20  on rdf:about=''
2A:C080h: 78 6D 6C 6E 73 3A 64 63 3D 27 68 74 74 70 3A 2F  xmlns:dc='http:/
2A:C090h: 2F 70 75 72 6C 2E 6F 72 67 2F 64 63 2F 65 6C 65  /purl.org/dc/ele
```

This appears to be a FWS Overlay (FWS reversed is SWF or Shockwave Flash). If on other files it shows as CWS that's fine. To find out what version of Flash this overlay is look at the 4th byte. In this example it's running Flash 9.

Next click on Edit->Copy As->Copy As Hex Text.

Now open up your dumped.exe in 010 Editor. Run the EXE template on this file as well (Should just have to press F5). Click on the Overlay [4032]. Then right click on the selected HEX and select Delete. Re-run the template and assure no overlay appears.

Now scroll to the end of the file and select Edit->Paste From->Paste From Hex Text, save the file dumped.exe and exit 010 Editor. Your dumped file should now run like full version.

Note: This can also be done without 010 Editor, you can use ANY decent hex editor. Just open up the shell exe and search for FWS or CWS, once found select from there all the way to the end of the file. That's your overlay; just re-attach it to the end of the dumped.exe.

1.5. Dora's Carnival 2 (10JP Overlay)

1.5.1 Preparation

This target when scanned looks the same as the FWS Overlay target, the only difference is the type of overlay it has and how to re-attach it to the dumped file.

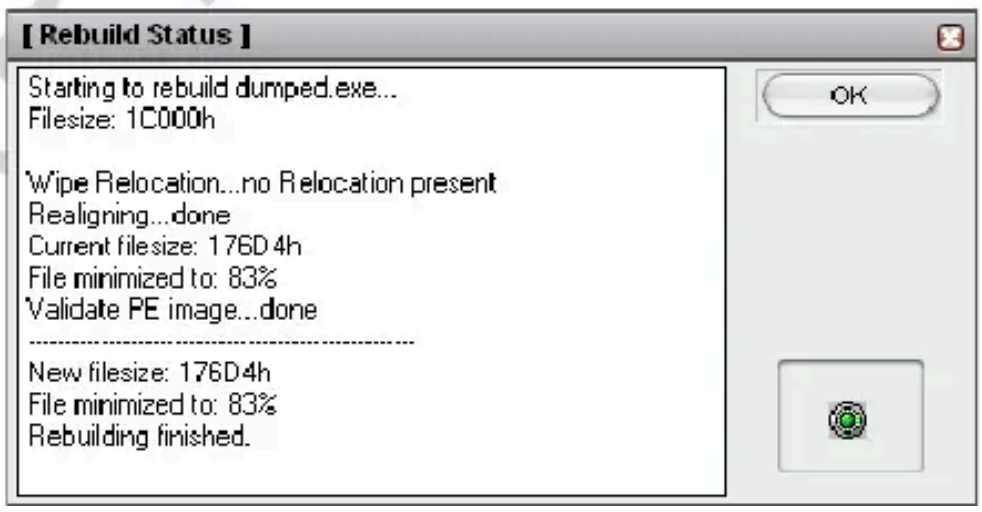
1.5.2 Checking Out The Target

Again we start with a dumped exe file as described at the beginning of the tutorial.

With this target we must also re-append the Flash Overlay, but it's a different type of overlay. As explained by Ghandi in his tutorial covering flash overlays with Armadillo, there's 3 types of Flash Overlays. The SWF type we discussed with Target 2, and Director type overlays. We're dealing with Director Type A, I haven't found a Director Type B target to test with.

Ok, first things first, open up Dora CarnivalII.exe in a hex editor and scroll to the very bottom and look at the last 4 bytes (00 A0 01 00) flip these around and you get 0001A000, this is where the overlay needs to begin in the dumped file.

Open up the Dumped file in your hex editor, scroll to the bottom. You'll notice the file is 1C000 bytes in size. Well if you recall we need the overlay to start at 1A000, the file is too big. Let's try rebuilding the file with LordPE:



New File size is 176D4, which is smaller than 1A000.

NOTE: If rebuilding the PE File doesn't make it small enough you can compress it with UPX (make sure overlay hasn't been added yet) then you can pad to the correct address and append overlay

Now that we've got the file smaller, we'll need to pad it with zeros so that the overlay can start at 1A000, file will end at 19FFF before overlay is added.

We'll need to put 292Ch bytes at the end of the file (19FFF – 176D4 = 292C)

After you've padded the file to the correct size, open up Dora Carnival II.exe in your hex editor. Recall that the last DWORD of the file tells us where the Overlay starts (in this example at 01A000) go to this address, it should start with 10JP (31 30 4A 50) select from here to the end of the file, and copy to the clipboard. Open up your paddeded Dumped file. And add the overlay to the end of it (should start at 01A000 in this example)

After you have added the overlay save the file your dump.exe should run like the original executable.

1.6. Mahjong Quest 3 (Expired)

1.6.1 Preparation

The target is the same type of target as the very first one we discussed (no overlay). The difference is for this target we will be discussing how to unwrap an executable after you have used up all the trial sessions.

1.6.2 Checking Out The Target

This target is a standard target (no overlay), but it's also run out of trial sessions, I'm going to show you how to get the application to run so that you can dump it as previously explained. Note that with the walkthrough for this target we will stop at the ResumeThread API where you would dump the file.

1.6.2.1 What is expired status really?:

The HTML Pages call the ShowDiv and HideDiv functions located in the Javascript code to show and hide certain options. When the application has determined that you have run out of trial sessions, it hides the Play button that allows Trial Session Playing and replaces it with a Play button that tries to deduct the amount of coins it costs from your online/offline WildCoins bank. When they can't get the coins it displays the error that you are out of trial sessions. So what we need to do first is edit the javascript code so that we get a play button that doesn't try to use wildcoins.

1.6.2.2 Editing the JavaScript

Open up the programs installation directory, within that folder locate the folder matching your locale information (ex, mine is EN-US, if you computer is in french, then your folder would be FR). Located in this folder is all the pages the app can display, open up the Local_Assets folder, and finally inside there open up the JS folder.

Right Click on the Start.js and select edit or open with notepad, scroll down until you see this code :

```
// if owned, hide token options, allow users to play directly
        if ( _isOwned ) {
        window.frames.content_right_iframe.HideAllContent();
        HideDiv( "playTokensDiv");
        HideDiv( "playTrial");
        HideDiv( "playSponsored");
        HideDiv( "sessionCost");
        HideDiv( "sessionCostShadow" );
        HideDiv( "sessionCost2");
        HideDiv( "sessionsRemaining");
        HideDiv( "quickPlay");
        HideDiv( "quickPlayPh");
        ShowDiv( "exit_button");
        HideDiv( "config_banner");
        ShowDiv( "offline_banner");
        ShowDiv( "playOwned");
        CenterInterface();
        }
```

This code is going to hide everything except the stuff that is viewable by somebody that has bought the application. We need to get this code to execute so that we have a valid Play Button. Change the "if(_isOwned)" to "if(!_isOwned)". Now anytime you run the application it will show you the Play button as if you owned the program. But there are more tricks we need to bypass. Go ahead and try to play the game, you'll quickly find out we still get the same message.

1.6.2.3 Analyzing the WTExecutable

If you look at the memory map of the executable, you'll notice it has 2 sections that are not standard, the '.pecode' and the '.pccode' sections. Let's look at them from inside Olly.

.pecode Starting Address: 0047F000

```
CWDE
AND DWORD PTR DS:[EAX-3C],28
AND ESI,DWORD PTR DS:[ESI+56]
AND DWORD PTR DS:[EDI],FFFFFFF1
ADC EAX,8380DAB5
XCHG EAX,ECX
AAD 0CC
RETF
XLAT BYTE PTR DS:[EBX+AL]
XCHG DWORD PTR DS:[EDX],EBX
IRETD
CMP BL,BYTE PTR SS:[ESP+ESI*2]
POP DS
FSTP QWORD PTR DS:[ECX+1B]
XOR BYTE PTR SS:[EBP+44BE0B37],DL
SUB DWORD PTR DS:[EDX-37],EAX
LEAVE
INT1
FSTP ESI
POPAD
PREFIX REP:
LOOPDE SHORT 0047EFDE
RCL DWORD PTR DS:[ESI],0BF
```

.pccode Starting Address: 0048D000

```
ADD DL,BYTE PTR DS:[ESI+A5837
PUSH EDX
SBB EDI,ECX
POP SS
ADC BL,83
AAD 4E
SCAS BYTE PTR ES:[EDI]
ADC BYTE PTR DS:[9D4D0D0D],DL
IN AL,7
RETF
ENTER 0EDFA,0DC
MOV AH,19
BOUND EAX,QWORD PTR DS:[EAX]
DEC EBP
LOOPDE SHORT 0048D0A4
POP EDX
STC
RETF
OUT 3B,AL
MOV CL,0DC
DEC ESP
???
```

This code doesn't seem to be legit, they must do some on-the-fly decrypting. Sections are encrypted using AES (Rijndael). Go ahead and run the program from within Olly (be sure Olly is invisible to IsDebuggerPresent).

After running the application and getting the nag screen look at the code in these sections again.

.pecode Starting Address: 0047F000

```
PUSH EBP
MOV EBP,ESP
PUSH -1
PUSH 00478461
MOV EAX,DWORD PTR FS:[0]
PUSH EAX
MOV DWORD PTR FS:[0],ESP
SUB ESP,0EC
MOV EAX,DWORD PTR DS:[4B0B90]
XOR EAX,EBP
MOV DWORD PTR SS:[EBP-18],EAX
PUSH ESI
MOV BYTE PTR SS:[EBP-A5],0
LEA ECX,DWORD PTR SS:[EBP-54]
CALL 00401760
MOV DWORD PTR SS:[EBP-4],0
PUSH 5C
LEA ECX,DWORD PTR SS:[EBP-54]
CALL 00402630
```

.pccode Starting Address: 0048D000

```
PUSH EBP
MOV EBP,ESP
PUSH -1
PUSH 0047877E
MOV EAX,DWORD PTR FS:[0]
PUSH EAX
MOV DWORD PTR FS:[0],ESP
SUB ESP,0EC
MOV EAX,DWORD PTR DS:[4B0B90]
XOR EAX,EBP
MOV DWORD PTR SS:[EBP-18],EAX
PUSH ESI
MOV BYTE PTR SS:[EBP-A5],0
LEA ECX,DWORD PTR SS:[EBP-54]
CALL 00401760
MOV DWORD PTR SS:[EBP-4],0
PUSH 5C
LEA ECX,DWORD PTR SS:[EBP-54]
CALL 00402630
```

Seems as though the code is being decrypted at runtime. Let's restart and place a BP at the WriteProcessMemory API

Also, it's worth noting that the .pecode section handles the code responsible for making the actual game's process and loading it with the correct code. The .pccode section is responsible for all protection related functions.

After placing BP on WriteProcessMemory, run the application. It should break with the Process parameter being -1 (writing to itself), and the address being 47F000 (the .pecode section). Press F9 again, should break again but this time writing to 48D000 (the .pccode section). Press F9 one more time, and the program should stay running with the nag screen visible.

Now we need to find when the application starts to run code out of the .pccode section. DO NOT set any kind of BPs in the .pccode section, doing this will cause the application to either crash or terminate itself.

Instead go ahead and click play on the nag screen. Olly should break again on the WriteProcessMemory API, again it's writing to itself, and again to the .pccode section. This is done to overwrite any patches a reverser may have applied to the section. Now that the application is done playing around with the .pccode section (and no longer monitoring for BPs) set a Memory Break-Point On Access to the .pccode section.

Go ahead and run the application again by pressing F9, after a little while you should break here: (Address: 48D3C0). Now remove the Memory breakpoint

```
PUSH EBP                                    |
MOV EBP,ESP                                 |
PUSH -1                                     |
PUSH 00478798                               | SE handler installation
MOV EAX,DWORD PTR FS:[0]                    |
PUSH EAX                                    |
MOV DWORD PTR FS:[0],ESP                    |
SUB ESP,58                                  |
PUSH ESI                                    |
MOV DWORD PTR SS:[EBP-58],ECX               |
PUSH 0048FCD8                               | Arg1 = 0048FCD8
MOV ECX,DWORD PTR SS:[EBP-58]               |
ADD ECX,0B0                                 | Mahjong0.00408EA0
CALL 00408EA0                               |
CALL 0048DA70                               |
MOVZX EAX,AL                                |
TEST EAX,EAX                                |
JE SHORT 0048D418                           |
PUSH 0048FCD4                               | Arg1 = 0048FCD4 ASCII "debugger"
MOV ECX,DWORD PTR SS:[EBP-58]               |
ADD ECX,0B0                                 | Mahjong0.00408EA0
CALL 00408EA0                               |
XOR AL,AL                                   |
JNZ 0048D73E                                |
MOV ECX,DWORD PTR DS:[4B25B0]               |
CALL 0040D270                               |
MOVZX ECX,AL                                |
TEST ECX,ECX                                |
JNZ SHORT 0048D444                          |
PUSH 0048FCE0                               | Arg1 = 0048FCE0 ASCII "notallowed"
MOV ECX,DWORD PTR SS:[EBP-58]               |
ADD ECX,0B0                                 | Mahjong0.00408EA0
CALL 00408EA0                               |
```

Scroll down a little ways, you'll see the string “outoftrialsession” (Address: 48D50B). Right above that you will see a JNZ SHORT, set a SWBP there. And run until the BP

Once you are at this address, check to see if the application is going to jump here. An application whose trial sessions haven't expired will jump here. So the exact opposite is also true No Trials equals No Jump simply flip the Zero flag from it's current state to cause the application to jump here, this jump immediately leads to a call that calls the code to create the game'sexe. The game will now run.

You may experience some problems with the game actually displaying properly, or it could just be my virtual machine. So after you flip the flag, set a BP on the CreateProcess API and run the application, when you break on this API, run until return, then step out of the function, back in user code you will see a call to ResumeThread, set a BP there and run the app again until this BP. Now proceed to dump the program as detailed in previous targets.

2. Greetings

Greetings fly out to my many friends at ARTeam, especially SSIEvIN, Shub-Niggurath, and Nacho_DJ. There are many others out there that I owe a lot to, not only for helping with ideas on WildTangent, but for helping me learn RE in general. As far as teams go, ARTeam, SnD, and members of the former TeamICU.

3. Document History

- Version 1.0 first public release
- Version 1.1 formatted the tutorial to ARTeam standards, included my Unwrapper/Loader, my OllyDbg WildTangent Unwrapper Script, my ASM Search Routine, and a video tutorial of unwrapping a WildTangent Game.

3.1. Conclusions

So, to conclude the paper, we have discussed the methods of unwrapping WildTangent Games. For standard non-expired games, this can simply be done by putting a BP on the CreateProcessA API, and running until user code, then placing a BP on ResumeThread. Then dumping shell exe from memory using LordPE. We then discussed the methods of re-appending different types of overlays to the unwrapped executables to allow proper execution. Lastly we discussed a method of unwrapping expired games with a 1-byte patch in layer 3 code.

I hope you all have enjoyed this paper as much as I liked making it and exploring the world of WildTangent. Any questions or suggestions can be directed to me at the ARTeam Forums.