Ghandi [ARTEAM]

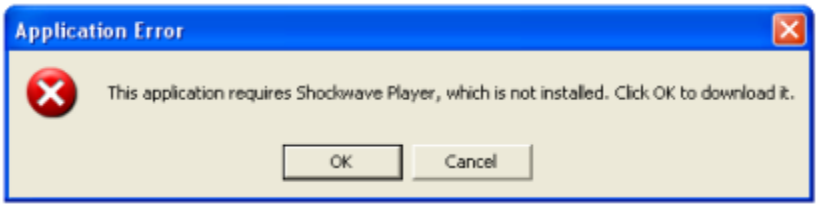
Shockwave Flash & Director Overlays, the Armadillo Aspect

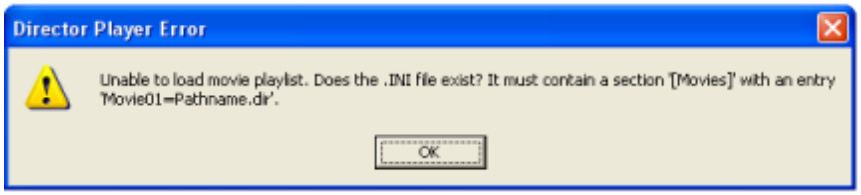
Version 1.0
Last Rev. July 2007

1. Introduction

Let me start by saying that this is not going to be a tutorial on unpacking Armadillo protected applications, it is more intended to give a little understanding to why some games don't work **after** unpacking them.

If you have unpacked a game and been greeted with one of the following messages, then you have encountered these before and are most probably wondering what went wrong, even confused:





There are also ones which, when unpacked, will result in a SWF player that loads showing a blank screen with a file menu on the top toolbar. These allow you to manually load Shockwave Flash files, both locally and from the internet, but are missing the original movie that was integrated into the original file.

Without further ado, lets see what is happening...

Happy Reversing,
Ghandi

Editor: Shub-Nigurrath

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 document 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, and cracked versions were available since a lot of time. ARTeam or the authors of the paper cannot be considered responsible damages the companies holding rights on those programs. The scope of this tutorial 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. Abstract

In a nutshell, we're going to see how this mechanism works and after we understand it, we will see what is needed to be done to correct the issues that arise when unpacking these kinds of files. This isn't really a hard thing to do, not after you know what is going wrong, so put on your thinking caps and read on...

2. Tools used, skills required.

OllyDbg

LordPE

Any good hex editor

Basic Armadillo unpacking experience

3. Targets

Because there are different types of overlays, i have chosen 3 targets for this tutorial.

Director Type A – Hidden Expedition: Everest (BigFish Games version) *Note: The installer is silent, no dialogs!*
http://downloads.bigfishgames.com/downloads/F2002T1L1.exe

Director Type B – Carrie the Caregiver (Gamenext/Oberon version) *Note: The installer is in German!*
http://www.gamenext.de/exe/Carrie_the_Caregiver-setup.exe

Shockwave Flash – Solitaire Pop (PlayFirst version)
http://www.playfirst.com/game/solitairepop/download/windows

4. Understanding the overlay mechanism

I haven't figured out a way to tell whether a file is a Director or Shockwave prior to unpacking, unless you can take the file information displayed by Windows as an indicator.

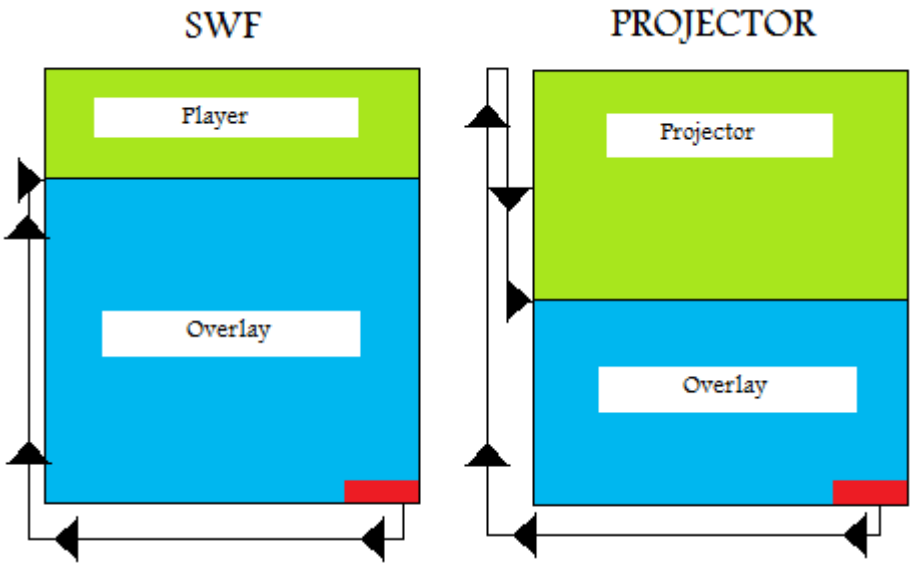
The example Shockwave game will have **Macromedia Flash Player v8.0**, whereas the Director examples merely have **Macromedia Projector**. There is a big difference between the two overlay types, Flash are relocatable and can be attached to the end of any size file, whereas Director files need to be treated differently if you want to modify them.

Both types, however, have as their primary index *(my terminology, it will suffice for now.)* a DWORD value that is stored as the last 4 bytes of the file. When the player/projector is run, the process will open its file with read-access, and set the file pointer to the last 4 bytes of the file. Then the 4 bytes are read and used to set the file pointer again and another 4 bytes are read, then compared with the overlay signature(s) it is capable of playing.

If there is not a recognized signature found, it will then take appropriate action, whether that be launching the player in standalone mode (Flash) or attempting to load the .INI file (Director). In the case of Director files, if there is no .INI file found, the projector will display the appropriate error messages before exiting.

Here's where they branch away from each other: Director projectors use the DWORD value read from the end of the file to set the file pointer from the start of the file (FILE_BEGIN), Flash players use it to set the file pointer from the end of the file (FILE_END). Director overlays closely follow the RIFF/RIFX format, they are a collection of files and directories, with each directory containing more raw offsets to the files contained within. Changing the location of the overlay would mean that ALL the offsets would need to be adjusted to reflect this.

Here's a visual of that, just in case you are confused, sorry for the graphics, I'm not much of an artist:



4.1. Armadillo and these overlays

Armadillo doesn't leave the overlay in the same location for various reasons, such as:

1. The actual size of the overlay isn't a constant, nor is the end size of a projector due to icons and other resources, so leaving it at the correct location for Director files would be out of the question. Armadillo adds sections (file bloat) too, which would change the location of the overlay. Armadillo by default inserts its sections (basically merging two executable files) between the .rsrc section and the section preceding it. If this were changed, then it would be possible to add it's sections AFTER the overlay, as long as the all important DWORD value was copied to the last 4 bytes of the file.

2. Leaving the overlay unprotected would defeat the whole purpose of protecting the file. If this were the case, we could just extract the overlay and attach it to another player/projector (as Deroko's tutorial shows.) thus bypassing Armadillo without the need to unpack it at all.

I know you're asking now: "*But if the overlay isn't where it needs to be, then how does the protected file still access it at all?*"
Well, the Armadillo authors have taken this into account, to allow people to use their product with Macromedia applications.
The overlay is stripped from the executable, stored in another location (an Armadillo section) and memory is allocated at runtime, then the overlay is inserted into this memory. Armadillo hooks ***_lopen***, ***_llseek***, ***_lread*** and ***_lclose*** so that when the projector needs to access itself on disk to read this data, it can simply return the pointer to this memory and the projector will function happily, oblivious to the fact it has been altered. (*SWF players use other file i/o API, the principle is the same.)*

Now, lets see how we can then restore the unpacked files so they can run as they were originally intended, once again!
Because the portals require the protection to be compatible with as many programs as possible (read: profit), they can't really use protection options that are too severe, so this makes the files relatively easy to unpack...

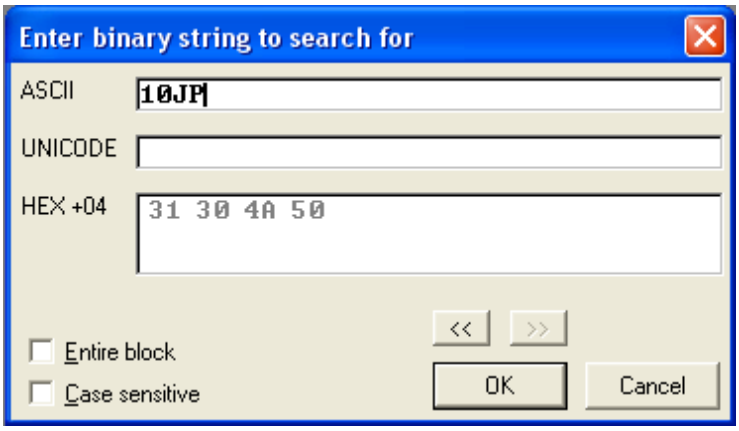
5. Restoring the overlay to an unpacked file.

As i stated in the introduction, I'm not showing how to unpack Armadillo. If you have read the tutorial this far and don't know how to unpack Armadillo protected files, i suggest that you put this paper down and go study some tutorials that cover that part. ARTeam has some excellent tutorials for this, plus Teddy Roger's site 'Tuts4You' has a good range that can show you the required skills so you can do this part. Unpack the targets and use LordPE to wipe the Armadillo section headers, then rebuild the file so it is a valid executable once again, see the chapter at the end of this article if you are unsure how to do so.

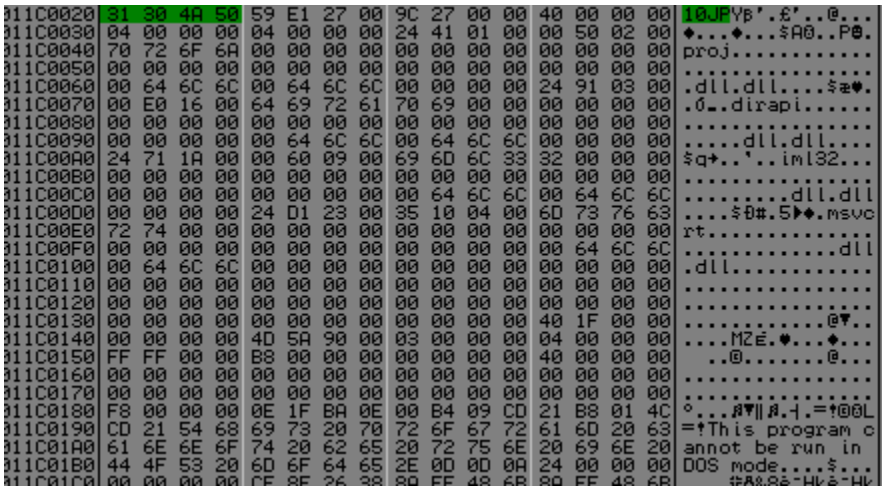
We'll tackle Hidden Expedition: Everest first, for no particular reason other than i wanted to. ;)

5.1. Director Type A

Open the protected file with OllyDbg and bypass Debug-Blocker (OpenMutexA trick) so this remains a single process. After this, breakpoint CreateThread and run the target until it hits your breakpoint, go to the memory view and do a binary search for the ASCII string **10JP**:



This is the signature for Director v10 overlays, it has been pointed out to me by Nacho_DJ that Director v6 overlays have the signature bytes **59JP**, so a wildcard search would be ??JP at +20h bytes. You should get a hit:



You can see the names of the dll's contained in the overlay, plus the beginning of the first dll's file header, the MZ signature and the DOS stub. Go back to the memory view and right click the region, select Dump Memory-Area to dump that region of memory to a temporary file, we'll need to clean it a little before attaching it to the unpacked projector. Once you have dumped the region, close OllyDbg and open our overlay in the hex editor, then delete the first 20h bytes:

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 50 00 9A 00 50 00 9A 00 00 00 00 00 00 00 00 00 ; P.š.P.š.........
00000010h: 00 F0 CB 00 00 F0 CB 00 36 0C 00 00 00 0B 00 00 ; .ðË..ðË.6.......
00000020h: 31 30 4A 50 59 E1 27 00 9C 27 00 00 40 00 00 00 ; 10JPYá'.œ'..@...
00000030h: 04 00 00 00 04 00 00 00 24 41 01 00 00 50 02 00 ; ........$A...P..
00000040h: 70 72 6F 6A 00 00 00 00 00 00 00 00 00 00 00 00 ; proj............
00000050h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000060h: 00 64 6C 6C 00 64 6C 6C 00 00 00 00 24 91 03 00 ; .dll.dll....$`..
00000070h: 00 E0 16 00 64 69 72 61 70 69 00 00 00 00 00 00 ; .à..dirapi......
00000080h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
```

They were only in use when this was a memory region, but now they're no longer needed. Now, the overlay file begins at the signature bytes **10JP**:

```
          0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00000000h: 31 30 4A 50 59 E1 27 00 9C 27 00 00 40 00 00 00 ; 10JPYá'.œ'..@...
00000010h: 04 00 00 00 04 00 00 00 24 41 01 00 00 50 02 00 ; ........$A...P..
00000020h: 70 72 6F 6A 00 00 00 00 00 00 00 00 00 00 00 00 ; proj............
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000040h: 00 64 6C 6C 00 64 6C 6C 00 00 00 00 24 91 03 00 ; .dll.dll....$`..
00000050h: 00 E0 16 00 64 69 72 61 70 69 00 00 00 00 00 00 ; .à..dirapi......
00000060h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000070h: 00 00 00 00 00 64 6C 6C 00 64 6C 6C 00 00 00 00 ; .....dll.dll....
00000080h: 24 71 1A 00 00 60 09 00 69 6D 6C 33 32 00 00 00 ; $q...`..iml32...
00000090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000a0h: 00 00 00 00 00 00 00 00 00 64 6C 6C 00 64 6C 6C ; .........dll.dll
```

Go to the end of the file and scroll upwards until you reach the beginning of the NULL bytes:

```
00cbe380h: 42 4A 4A 42 4A 4A 42 4A 41 46 4C 41 36 00 00 00 ; BJJBJJBJAFLA6...
00cbe390h: F9 00 F9 00 F9 00 F9 00 F9 00 F9 00 F9 00 F9 00 ; ù.ù.ù.ù.ù.ù.ù.ù.
00cbe3a0h: F9 00 F9 00 F9 00 F9 00 F9 00 F9 00 F9 00 F9 00 ; ù.ù.ù.ù.ù.ù.ù.ù.
00cbe3b0h: F9 00 F9 00 F9 00 F9 00 F9 00 F9 00 F9 00 F9 00 ; ù.ù.ù.ù.ù.ù.ù.ù.
00cbe3c0h: F9 00 F9 00 F9 00 00 40 01 00 00 00 00 00 00 00 ; ù.ù.ù..@........
00cbe3d0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00cbe3e0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00cbe3f0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00cbe400h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
```

We need to allow a single NULL byte, because a projector file isn't going to be 01400000h bytes in size, so selecting from the 2nd NULL byte **after** the 01, delete the rest of the file. Now your overlay ends on the 1st NULL byte, like so: