Portal World Domination Introduces:
The 'My First Patch' Tutorial

Greetings PWD Members. Since it has been suggested that the next tutorial be one of how to make a patch file, here it is.

What should I write my patch in?

You can use just about any language to create a patch file (except Java and a few others), but not all languages are well suited for this task. For instance, VB 4 is easy to prototype programs in but do you really want to distribute a 800k DLL (If you use bare-bones, 4M minimum with any OLE OCX included in the project). Another windows answer, though not common, is Delphi. While Delphi has access to the windows API, the code it generates to use it is not 100% stable (Trust me.. this was a project for a friend of mine). More suitable languages are C or C++, and Assembler. The advantage to Assembler is that it creates the smallest possible executable ever. It is also easy to store the compiler and linker (their total zipped are less than 500k for MASM/LINK) However, if you should happen down this path, know that there are 2 syntax's out there: MASM (Microsoft's) and TASM (Borland's) and both have plusses and minuses. I like MASM cause it's small to handle (VERY small), and more strict in nature. TASM code on the other hand, is very forgiving and pliable, and makes it easier to write self-manipulating code. The disadvantages of using Assembler are: Assembler takes a while to learn. And is not intuitive to maintain (though if you do it right once, you'll almost never need to change the code much, only a few DB statements.) Another disadvantage of Assembler is that it is very tedious to make a windows program in it (Talk about re-inventing the wheel!). C and C++ on the other hand, create a small executable. And lack the disadvantages Assembler has (maintainability and platform independence.) and add nicer error handling (although you have to invoke it.) and offer almost as much control over the code as Assembler. There is also the additional advantage to being able to compile into Assembler from C and then compiling your own assembler (though this is another story.) As for Pascal: Although I've seen Pascal patches that are sometimes smaller. Error handling in Pascal sucks. (Runtime error 12 in 19de:1928. Yuck!)
Although PWD still makes a practice of mostly distributing DOS based patch files, we want to be ready for the day when there is no 16 bit OS available. Since C is easily portable to windows, for this quick intro. I'll cover the C version of our PWD patch.

The C/C++ standard PWD Patch.

Whoa? Did I say Standard? Does this mean this is the only PWD way to make a patch? No. What this does mean is that this patch method is the PWD supported method. In fact, most head members should know how to modify, configure and compile this file. If you DON'T want a technical explanation about how it works, skip the next section.

Technical Explanation

Before I cover what to modify to make your patches, let me explain a little bit about what it does (for non-C/C++ gurus). This design is actually DiM's, taken from the first Pascal patch source he gave me. I adopted it because I discovered it that it was easy to maintain and lends itself nicely to the output of a file compare (FC /B <ORIGINAL FILE> <PATCHED FILE>) as explained below. The program stores 2 arrays. The first array contains a list of file offsets in the following manor:

0x<32bit file offset>, … ← the ellipses (…) indicate additional entries.

The second contains the list of bytes to find and replace (if found) at the respective file offset, stored in the following manor:

<# of Bytes at Offset to replace>, 0x<Byte to find><Byte to replace it with if found>, …

The bytes are stored in this manor due to the way I make patches.

What do I do to configure the Patch file?

There should be a universal tool to make a patch. Although there are some (UCF uses protect 6.x for the majority of their patches), they are either harder to use, contain unknown code, or just don't work. I mention unknown code because it is a concern of mine. PWD should never leave itself open to accusations of harming a users computer through bad code or virii. We have yet to have someone (including myself) take the time to make a patch creator to ease this process although I do have designs for it if anyone has the time. Here is the PWD Patch File:

---------- >8 ---------- >8 ---------- CUT HERE ---------- >8---------- >8 ---------- >8

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BANNER                "!!!Portal World Domination!!!"
#define HEADER                "Ultra-Soft Downy"
#define AUTHOR                "StarFury 1996"
#define FILENAME              "ultra.exe"

#define OPENTYPE              "r+"
#define HI(x)                 (x >> 8)
#define LO(x)                 (x   & 0x00FF)
#define MAXBYTESPERCRACK 8
#define BYTE                  unsigned char
#define ULONG                 unsigned long


/* crkLen indicates the number of differant offsets to make changes to */
#define crkLen    2

/* I usually paste the diffs from a fc /b <Orig> <Patched> Here.
0000231B: 74 90   See.. this is the first location
0000231C: 03 90   2 bytes long...
00005977: 74 90   and this is the second location
00005978: 03 90   also 2 bytes long...
*/

/* the list of offsets to start patching the file at (count should match
crkLen )
```

```
*/
ULONG crkOfs[crkLen] =
      {
      0x0000231B,
      0x00005977
      };

/* crkTbl first states the bytes to change at the given location
** followed by a list of 0xBefore|After byte sets...
**     for instance.. there are 2 differant offsets (as shown in crkOfs,
above)
**     of which both have 2 bytes to change.. for the first location the
**     patch will look for 7403 and replace it with 9090....
*/

ULONG crkTbl[crkLen][MAXBYTESPERCRACK] =
{
      {2, 0x7490, 0x0390},
      {2, 0x7490, 0x0390}
};

main(int argc, char *argv[])
{
      BYTE inbuf[MAXBYTESPERCRACK], inchar, bSame, patchbyte;
      FILE *fp;
      int bts, Ix, Jx;
      ULONG fOffset;
      long io_num;
      int   result, numread, dupchars, alreadpatched;


      /*-----------------------------------------*/
      /* print out the Header and the Author lines */
      /*-----------------------------------------*/
      printf("%s\n%s\n%s\n", BANNER, HEADER, AUTHOR);

      /*-----------------------------------------*/
      /* Open the File                                        */
      /*-----------------------------------------*/
      if( (fp = fopen( FILENAME, OPENTYPE )) == NULL )
      {
            printf("Cannot open %s. Please move %s to the directory containing
%s.   Exiting.\n", FILENAME, argv[0], FILENAME);
            return(0);
      }

      /*-----------------------------------------*/
      /* Verify the patch data                              */
      /*-----------------------------------------*/
      bSame = 1;
      alreadpatched = 0;
      for (Ix = 0; Ix < crkLen; Ix++)
      {
            /* Go to the spot */
            fOffset = crkOfs[Ix];
            result = fseek( fp, fOffset, SEEK_SET);
            if( result )
```

```
      {
            printf( "Error Seeking in File.(s)\n" );
            fclose(fp);
            return(0);
      }

    /* Read the signature buffer */
    bts = (int) crkTbl[Ix][0];
      dupchars = 0;
      numread = fread( inbuf, sizeof( char ), bts, fp);
    for (Jx = 0; Jx < bts; Jx++)
    {
      inchar = inbuf[Jx];
      patchbyte = (BYTE) HI(crkTbl[Ix][Jx+1]);

      if (patchbyte != inchar)
            {
                  if (inchar == ((BYTE) LO(crkTbl[Ix][Jx+1])))
                        dupchars++;

                  bSame = 0;
            }
    }

      if (bSame == 0)
      {
            if (dupchars == bts)
            {
                  alreadpatched++;
            }
      }
}


/* Are they the same? */
if (!bSame)
{
      if (((alreadpatched+1) == crkLen) || (alreadpatched == crkLen))
           printf("Patch has already been applied.\n");
      else
           printf("Incorrect version\n");
      return(0);
}

/*-----------------------------------------*/
/* Patch the Data                          */
/*-----------------------------------------*/
for (Ix = 0; Ix < crkLen; Ix++)
{
      /* Go to the spot */
      fOffset = crkOfs[Ix];
      result = fseek( fp, fOffset, SEEK_SET     );
      if( result )
      {
            printf( "Error Seeking in File.(s)\n" );
            fclose(fp);
            return(0);
```

```
      }

    /* Read the signature buffer */
    bts = (int) crkTbl[Ix][0];
    for (Jx = 0; (Jx < bts); Jx++)
    {
            patchbyte = (BYTE) LO(crkTbl[Ix][Jx+1]);
      io_num = fputc(patchbyte, fp );
    }
}


/*-------------------------------------------*/
/* Verify the patch was successful data      */
/*-------------------------------------------*/
bSame = 1;
for (Ix = 0; Ix < crkLen; Ix++)
{
      /* Go to the spot */
      fOffset = crkOfs[Ix];
      result = fseek( fp, fOffset, SEEK_SET);
      if( result )
      {
            printf( "Error Seeking in File.(s)\n" );
            fclose(fp);
            return(0);
      }

    /* Read the patched buffer */
    bts = (int) crkTbl[Ix][0];
      numread = fread( inbuf, sizeof( char ), bts, fp);
    for (Jx = 0; Jx < bts; Jx++)
    {
      inchar = inbuf[Jx];
      patchbyte = (BYTE) LO(crkTbl[Ix][Jx+1]);

      if (patchbyte != inchar)
            {
                  bSame = 0;
            }
    }

}


/* Are they the same? */
if (!bSame)
{
      printf("File was not patched successfully (write protected?)\n");
      return(0);
}

/* Notify Success */
printf( "File was patched Successfully.\n" );


/* Close the file */
```

```
      fclose(fp);

      /* Now return positively */
      return (1);
}
```

---------- >8 ---------- >8 ---------- CUT HERE ---------- >8---------- >8 ---------- >8

As indicated in the actual C file, there are few things to change to actually make the patch file:

1) On the following Line, set the HEADER to be the program name (and version).

```
      #define HEADER                "Ultra-Soft Downy"
```

2) Now change the AUTHOR to your Nick/Name and Year you make the patch

```
      #define AUTHOR                "StarFury 1996"
```

3) Now change the FILENAME to indicate the file to be patched, whether it's an EXE, DLL, or whatever (ever a .P file ☺)

```
      #define FILENAME              "ultra.exe"
```

4) Now You have to set the crkLen constant to the number of individual offsets to patch within the file (NOT the number of bytes to patch… the number of places to patch)

```
      #define crkLen    2
```

5) Now set up the crkOfs array to list the offsets to patch in the file (This array must have the same number of entries as crkLen)

```
   ULONG crkOfs[crkLen] =
      {
      0x0000231B,             /* Note this is the First offset
                              ** Into the file to patch
                              */

      0x00005977               /* This is the Second offset, and because
                               ** It's the last offset, we don't put
                               ** a comma on the end.
                               */
      };
```

5) And Lastly, as described in the Technical section, fill in the crkTbl with the actual bytes to replace and the bytes to replace them with. The crkTbl is a 2 dimensional array containing a the Number of bytes at the current offset to change, and then a list of what the bytes to find/change are.

```
   ULONG crkTbl[crkLen][MAXBYTESPERCRACK] =
   {
      {2, 0x7490, 0x0390},
      {2, 0x7590, 0x0390}
   };
```

6) To help illustrate, the output from FC for the above code is this:

```
Comparing files ultra.org and ultra.exe
0000231B:  74 90
0000231C:  03 90
00005977:  75 90
00005978:  03 90
```

7) Now compile this c file with a compiler. I prefer to create an exe rather than a com using the large model but either work well.

How I make a patch.

When I crack a file, I usually make a backup copy of the executable I patch first (copying the file from XXXXXX.EXE to XXXXXX.ORG) and then crack or disassemble the program so that I may use a hex editor (I use HexWorks) on the copy and test it as I think I find the bytes to patch. In cracking a program with a large number of protection checks, this helps because I test each patch I make to the executable (with HexWorks) while I try to root out any other 'Bugs' as Voxel puts it. When I'm done, I usually end up with a totally cracked copy that works flawlessly. I COULD distribute just the cracked executable (though this is rarely desirable; No one wants to download extra unnecessary crap.) I take this cracked executable, and compare it to the original (Remember, I was working on a copy) by using DOS file compare (FC). Afterward, I take the output from FC and use it to create and configure a C patch file, compile it, and then test the patch. To test the patch, I copy the XXXXXX.EXE to XXXXXX.CRK and copy the XXXXXX.ORG to XXXXXX.EXE once again, apply the patch, and then use FC to compare the XXXXXX.EXE to the XXXXXX.CRK.
If there are no differences, then I know I created my patch file correctly. Here is an example of these steps:

Let's say we have a program that does something cool, and the main file where the protection is is called ULTRA.EXE.

1) Make a backup of the file to be patched.
   COPY ULTRA.EXE ULTRA.ORG
2) Now patch the file using Norton utilities or HexWorks. (Both have search ability)
3) After your cracked EXE is working, compare the newly cracked EXE to the original.
   FC /B ULTRA.ORG ULTRA.EXE > ULTRA.DIF
4) Now edit and compile the C Patch file (as explained above) into a patch executable
5) Now make a backup of the ULTRA.EXE file (cause it's working great!)
   REN ULTRA.EXE ULTRA.CRK
   COPY ULTRA.ORG ULTRA.EXE
6) Now run the created C Patch executable on the copy of the original.
   ULTRA_K.EXE (or ULTRA_K.COM)
7) Now compare the executable your patch created with the one you hand cracked.
   FC /B ULTRA.CRK ULTRA.EXE
8) Repeat steps 4-7 until the above comparison says there are no differences.
9) Package the patch with an PWD.NFO (and the original program if distribution is needed) and Up it to the PWD Site for release testing.

How to contact us!

We're always looking for new members who really want to learn! See any of the senior members or Head members for membership! We can help you learn. If your ever on Undernet stop by #PWD. Or look for our members scattered around the world. Here is a list:

First the two founding (Senior) Members and the best of our group:

- DiM
- StarFury

Then the Head Members, those of us that cracked something:

- _Lasher_ / Imajix
- B_Spline
- Kratz
- VoXeL

Finally those valuable Members that are there to help and learn:

- BigD_
- Cyah
- DrmWEaver
- EvilAngel
- Fouton
- Goa
- IcedFire
- Locote
- Maug
- |No_One|
- Quequog
- Slvrmoon
- TimbrWlf
- Tungsten
- Goa
- ^Arj

StarFury