Contents

Introduction

PESpin is a Windows executable files (EXE, DLL) protector, compressor coded in pure assembly language using MASM. It allows compression of the whole executable PE32+ (AMD64) - code, data and resources, leaving them executable and protects against patching and disassembling.

Supported x64 operating systems:

- Windows 7, Vista, XP

PESpin is released as Freeware.

History

v1.21a

- fixed GUI bug in PESpin_x64.exe on Win8 x64

v1.21

- fixed DLLs compatibility with IE9
- minor GUI changes

v1.2

- added password protection
- added overlays handling

v1.104

- minor improvements
- minor bugs fixed

v1.1

- added debug-blocker

v1.09

- fixed export table processing bug in specific applications
- improved on demand import resolving

v1.08

- added checksum protection option
- added remove OEP option

v1.0

- added new option import resolving on demand

v0.9

- added aPLib compression (optional)

v0.8

- added license key system

Program options

List of features

- Decreases size of most PE32+ files a lot while leaving them still functional
- Protects your files against patching/modifications and disassembling
- License key system
- PE32+ Optimization
- Sections/Objects-renaming
- Polymorph decryption routines
- Import table encryption
- Application integrity check (crc)
- Application password protection
- Resource, Relocations, Exports-handling/redirection
- Removing orginal entry point (OEP)
- Command line support
- Entire thing written in assembler

Options, Controls

[Section names]
'User name'- type maximum 8 chars or nothing then sections name will be cleared.
'Don't rename' - sections name will be the same as in original file.

[Compress resources]
Compress resource section. By default RT_MANIFEST, RT_ICON, RT_VERSION, RT_GROUP_ICON, "MUI" aren't compressed.
Disable this option if for some reason your resources that must remain uncompressed.

[Import resolving]

***via Windows loader* -** Windows loader will locate all the imported functions and make them available to the file being loaded. This type of import resolving will most likely take negative effect on compression ratio, since import section won't be processed.

Attention

Please note that if KERNEL32.dll is not present inside import table this type of import resolving cannot be used (for password protection also COMCTL32.dll is required). In such case PESpin will automatically switch import resolving to: via GetProcAddress.

v***ia GetProcAddress*** - Same as above only PESpin will do the job using kernel32!GetProcAddress function.

via Hash - More secure than resolving by GetProcAddress since functions names aren't exposed, plus files with large number of imported functions will achieve slightly better compression ratio.

On Demand - Import resolving on the fly, provides good defence against import rebuilder tools since import table is never resolved completely. Depending on how your program is set this can reduce performance slightly or significantly. Please test your application when applying this option.

[Use Compression]

Possibility to choose compression library, available:

- UCL compression
- aPLib compression
- None

[Remove OEP]
PESpin will remove physically few instructions after your Orginal Entry Point (OEP). Disable this option if your program returns the control to the OEP after being executed.

Attention

Cause of the way .DLL files entry-point function is constructed, this option will not be apply to .DLL files even if enabled.

[Checksum protection]
Checks the integrity of the file to detect intentional modifications such as code patching or unintentional modifications such as virus infection.

[Debug blocker]
This option only applies to .exe files, causes the program to create two instances whenever it is run - called "parent" and "child" processes. Some instructions of your program code will be patched over and converted into so called nanomites. Upon execution, PESpin loader will create two instances of your program called "parent" and "child" processes. Nanomites are going be executed in the "parent" process and the results will be passed on to the "child" process.

Tip

Advanced users can adjust number of nanomites (maximum size of nanomites database) from *settings.ini* file section *advanced_settings*, possible range from 0 to 5000. Please note that the higher number of nanomites will increased size of application and will significantly decrease performance.

Attention

Disable this option if your application won't run or runs significant slower. Don't use this option if your application uses self-modifying code.

[Password protection]
Protected file will be encrypted using entered password (a max of 127 characters).
You need to enter password every time that your program is going to be executed.

Attention

It's not possible to recover a password that has been lost!

[Auto run after file loading]
Automatically start of protection process.

[Exit when done]
Exit when file will be successfully protected.

[Create backup file]
Create backup file.

License key system

How does it work?

Selected parts of your application are “cut off” and encrypted using the best and most secure encryption algorithm Advanced Encryption Standard (AES / Rijndael) with key size 256 bits. Upon execution of protected file PESpin loader will try to open a license file (key.pes) if succeed digitall signature will be computed, and if valid protected code will be decrypted and placed in it's original position.

```
//Example: Bubble Sort in C (found somewhere on the Internet)

#include <stdio.h>
#include <iostream.h>


int License_code(unsigned __int64 a, unsigned __int64 b)
{
    return 0;
}


void bubbleSort(int *array,int length)//Bubble sort function
{
      int i,j;

    License_code(0xC0DE064011111111, 0xC0DE064022222222);
      // the area to protect starts here

      for(i=0;i<10;i++)
      {
            for(j=0;j<i;j++)
            {
                  if(array[i]>array[j])
                  {
                        int temp=array[i];
                        array[i]=array[j];
                        array[j]=temp;
                  }

            }

      }
      // end of area to protect
    License_code(0xC0DE064044444444, 0xC0DE064088888888);

}

void printElements(int *array,int length)
{
      int i=0;
      for(i=0;i<10;i++)
```

```
            printf("%d ", array[i]);
}


void main()
{

      int a[]={9,6,5,23,2,6,2,7,1,8};
      bubbleSort(a,10);
      printElements(a,10);
}
```

original output file not protected :

```
      23 9 8 7 6 6 5 2 2 1
```

file protected but license file is not present or corrupted :

```
      9 6 5 23 2 6 2 7 1 8
```

file protected with valid license file (key.pes) present in the directory with executable:

```
      23 9 8 7 6 6 5 2 2 1
```

```
void bubbleSort(int *array,int length)
{
       int i,j;

License_code(0xC0DE064011111111,0xC0DE064022222222
);

       for(i=0;i<10;i++)
       {
              for(j=0;j<i;j++)
              {
                     if(array[i]>array[j])
                     {
                            int temp=array[i];
                            array[i]=array[j];
                            array[j]=temp;
                     }

              }

       }

License_code(0xC0DE064044444444,0xC0DE064088888888
);

}
```

View of the function when valid license file is present

```
void bubbleSort(int *array,int length)
{
       int i,j;

License_code(0xC0DE064011111111,0xC0DE064022222222
);














License_code(0xC0DE064044444444,0xC0DE064088888888
);

}
```

View of the function without a valid license file

How to use?

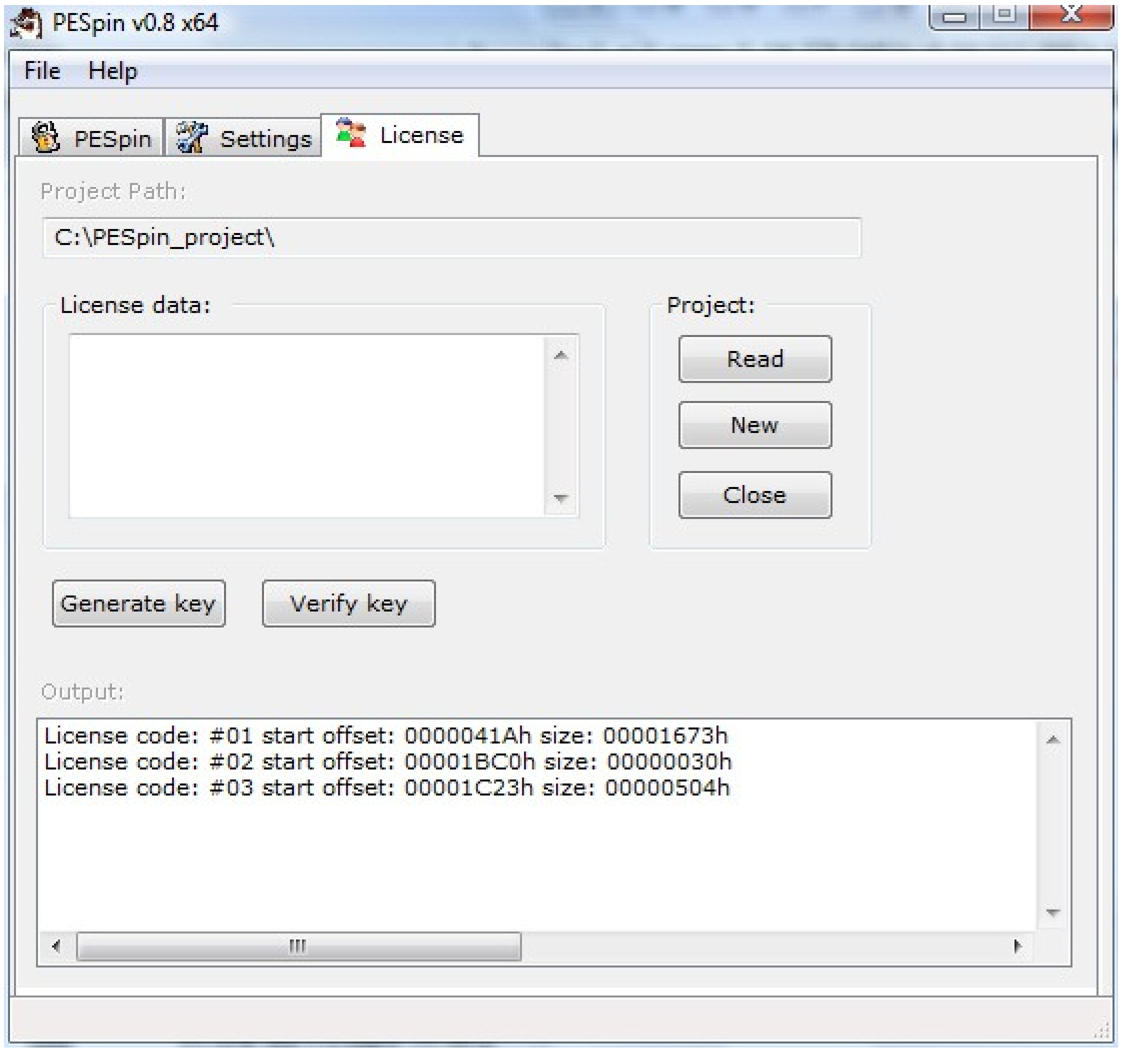
To use license key system you need to adjust your source code and recompile, so that PESpin will be able to recognize which fragments of code you wish to execute only if a valid license file is present. Recognition of area to protect is based on function parameters. Depending on your c/c++ optimization settings this can be sometimes a bit tricky since compiler can thwart recognition putting `License_code` function as inline (in that case mark function as `noinline`) or worse removing it completely as unused. If you encounter such problem try to use #pragma optimize ("[optimization-list]", {on | off}) for example:

```
#pragma optimize("", off)
int License_code(unsigned __int64 a, unsigned __int64 b)
{
  return 0;
}
#pragma optimize("", on)
```

More information on C/C++ optimize pragma can be found here.

Well, in worse possible scenario you will have to somehow connect `License_code` function with the rest of your code.

To activate licensing support inside an application code you must (after defining in your source code functions with parameters above) read or create new project in License tab. After protecting in License tab output you can find information about offset and size of protected code.



C:\PESpin_project\

```
License code: #01 start offset: 0000041Ah size: 00001673h
License code: #02 start offset: 00001BC0h size: 00000030h
License code: #03 start offset: 00001C23h size: 00000504h
```

Warranty

Credits

This product uses:

UCL compression library - http://www.oberhumer.com/opensource/ucl/

aPLib compression library - http://www.ibsensoftware.com/

CryptoAPI RSA Library version 1.0.0 (c) WiteG

LDE64 x64 Length Disassembler Engine by BeatriX - http://beatrix2004.free.fr

Thanks to:
TiGa - BeatriX - Stanley White - WiteG - ReWolf

Contact

In case of errors, problems, etc. feel free to contact me.

Regards,
cyberbob

pespin.64@gmail.com
pespin.w.interia.pl