GitHub - monoxgas/sRDI: Shellcode implementation of Reflective DLL Injection. Convert DLLs to position independent shellcode

By monoxgas

Archived: 2026-04-05 15:40:45 UTC

sRDI allows for the conversion of DLL files to position independent shellcode. It attempts to be a fully functional PE loader supporting proper section permissions, TLS callbacks, and sanity checks. It can be thought of as a shellcode PE loader strapped to a packed DLL.

Functionality is accomplished via two components:

- C project which compiles a PE loader implementation (RDI) to shellcode
- Conversion code which attaches the DLL, RDI, and user data together with a bootstrap

This project is comprised of the following elements:

- **ShellcodeRDI:** Compiles shellcode for the DLL loader
- **NativeLoader:** Converts DLL to shellcode if neccesarry, then injects into memory
- **DotNetLoader:** C# implementation of NativeLoader
- **Python\ConvertToShellcode.py:** Convert DLL to shellcode in place
- **Python\EncodeBlobs.py:** Encodes compiled sRDI blobs for static embedding
- **PowerShell\ConvertTo-Shellcode.ps1:** Convert DLL to shellcode in place
- **FunctionTest:** Imports sRDI C function for debug testing
- **TestDLL:** Example DLL that includes two exported functions for call on Load and after

The DLL does not need to be compiled with RDI, however the technique is cross compatiable.

Use Cases / Examples

Before use, I recommend you become familiar with Reflective DLL Injection and it's purpose.

Convert DLL to shellcode using python

```
from ShellcodeRDI import *

dll = open("TestDLL_x86.dll", 'rb').read()
shellcode = ConvertToShellcode(dll)
```

Load DLL into memory using C# loader

```
DotNetLoader.exe TestDLL_x64.dll
```

Convert DLL with python script and load with Native EXE

```
python ConvertToShellcode.py TestDLL_x64.dll
NativeLoader.exe TestDLL_x64.bin
```

Convert DLL with powershell and load with Invoke-Shellcode

```
Import-Module .\Invoke-Shellcode.ps1
Import-Module .\ConvertTo-Shellcode.ps1
Invoke-Shellcode -Shellcode (ConvertTo-Shellcode -File TestDLL_x64.dll)
```

Flags

The PE loader code uses `flags` argument to control the various options of loading logic:

- `SRDI_CLEARHEADER` [0x1]: The DOS Header and DOS Stub for the target DLL are completley wiped with null bytes on load (Except for e_lfanew). This might cause issues with stock windows APIs when supplying the base address as a psuedo `HMODULE`.
- `SRDI_CLEARMEMORY` [0x2]: After calling functions in the loaded module (`DllMain` and any exports), the DLL data will be cleared from memory. This is dangerous if you expect to continue executing code out of the module (Threads / `GetProcAddressR`).
- `SRDI_OBFUSCATEIMPORTS` [0x4]: The order of imports in the module will be randomized before starting IAT patching. Additionally, the high 16 bits of the flag can be used to store the number of seconds to pause before processing the next import. For example, `flags | (3 << 16)` will pause 3 seconds between every import.
- `SRDI_PASS_SHELLCODE_BASE` [0x8]: As opposed to passing supplied user data to the exported function, sRDI will instead pass the base address of the currently executing shellcode block. This can be useful for self-cleanup inside more advanced modules.

Building

This project is built using Visual Studio 2019 (v142) and Windows SDK 10. The python script is written using Python 3.

The Python and Powershell scripts are located at:

- `Python\ConvertToShellcode.py`
- `PowerShell\ConvertTo-Shellcode.ps1`

After building the project, the other binaries will be located at:

- `bin\NativeLoader.exe`
- `bin\DotNetLoader.exe`
- `bin\TestDLL_<arch>.dll`
- `bin\ShellcodeRDI_<arch>.bin`

If you would like to update the static blobs inside any of the tools:

```
 > python .\lib\Python\EncodeBlobs.py -h
usage: EncodeBlobs.py [-h] solution_dir

sRDI Blob Encoder

positional arguments:
  solution_dir  Solution Directory

optional arguments:
  -h, --help    show this help message and exit

> python lib\Python\EncodeBlobs.py C:\code\srdi

[+] Updated C:\code\srdi\Native/Loader.cpp
[+] Updated C:\code\srdi\DotNet/Program.cs
[+] Updated C:\code\srdi\Python/ShellcodeRDI.py
[+] Updated C:\code\srdi\PowerShell/ConvertTo-Shellcode.ps1
```

Alternatives

If you find my code disgusting, or just looking for an alternative memory-PE loader project, check out some of these:

- https://github.com/fancycode/MemoryModule - Probably one of the cleanest PE loaders out there, great reference.
- https://github.com/TheWover/donut - Want to convert .NET assemblies? Or how about JScript?
- https://github.com/hasherezade/pe_to_shellcode - Generates a polymorphic PE+shellcode hybrids.
- https://github.com/DarthTon/Blackbone - Large library with many memory hacking/hooking primitives.

Credits

The basis of this project is derived from "Improved Reflective DLL Injection" from Dan Staples which itself is derived from the original project by Stephen Fewer.

The project framework for compiling C code as shellcode is taken from Mathew Graeber's reasearch "PIC_BindShell"

Source: https://github.com/monoxgas/sRDI