DLL Injection - rychlý a špinavý úvod

Zrovna jsem si hrál s tsocks a torify a tak mě napadlo "jak to asi funguje?". Věděl jsem, že to nějak zrádně podstrčí programu upravenou funkci místo jiné. Tak jsem si řekl, že se tomu podívám na zoubek... Po 5 min jsem měl kupodivu velice snadný postup, jak to celé provést - i když třeba ne příliš čistě...

Celý trik spočívá v nastavení proměnné prostředí LD_PRELOAD na cestu ke knihovně, která obsahuje funkce se stejným prototypem (název, typ návratové hodnoty, typy argumentů), jako funkce, které chceme nahradit. To zajistí příkaz

```
export LD_PRELOAD="/cesta/ke/knihovne";
```

Od této chvíle bude dynamický linker, který je součástí jádra linuxu (nebo resp jde o knihovnu, která se jmenuje `ld-linux.so` nebo `ld.so` a je linkovaná do všech programů, aby potom umožnila linkování dalších knihoven...) zpřístupňovat kód naší knihovny každému programu, který bude v tomto prostředí spuštěn a funkce z ní budou upřednostněny před funkcemi z ostatních knihoven...

A nyní praktická ukázka:

Mám zdrojový soubor victim.c, kterému budu podstrkovat "zlou" knihovnu:

victim.c

```
#include <stdio.h>

int main() {
    puts("this text is output");
}
```

Zkompiluji ho jako běžnou binárku:

```
[harvie@harvie-ntb shared]$ gcc -o victim victim.c
```

Dále mám zdrojový soubor podvodné knihovny test.c:

test.c

```
#include <stdio.h>

int puts(const char *text) {
  return printf("puts(\"%s\");\n", text);
}
```

Povšimněte si, že v příkladu jsem vytvořil dvojče funkce puts(), což je nejjednodušší funkce, která umožňuje vypsání textu na monitor. Moje verze ale kromě vypsání zadaného textu ještě navíc vypisuje informaci, že se jedná o funkci puts() tím, že okolo textu vypíše závorky a svoje jméno, jak

uvidíte ve výpisu dole.

Dalším problémem je, že nemohu k vypsání textu již použít funkci puts(), protože jsem ji vlastně zneškodnil. Já jsem proto pro výpis textu použil trochu odlišnou funkci printf(), která má více možností, ale zvládne to samé jako puts(). Také by bylo možné někde uchovat pointer na pravou funkci puts() a potom ji pomocí něj z podvržené funkce zavolat. Jiným asi častějším řešením je funkci napsat znovu pomocí jiného kódu, to ovšem nechám na vás. Funkce samozřejmě původní činnost vůbec vykonávat nemusí a může třeba jen uživateli oznámit "program XY se právě pokusil přečíst /etc/shadow", atd...

Tento soubor zkompiluji jako sdílenou (dynamicky linkovanou) knihovnu (parametr -shared):

```
[harvie@harvie-ntb shared]$ gcc -shared -fPIC test.c -o test.so
```

pozn.: při mém testu injektáž fungovala i bez parametru -fPIC, který má zajistit, že kód půjde spustit nezávisle na umístění v paměti, ale běžně se doporučuje parametr použít u všech sdílených knihoven.

A tady již můžete vidět, jak to celé funguje:

```
[harvie@harvie-ntb shared]$ export LD_PRELOAD=""
[harvie@harvie-ntb shared]$ ./victim
this text is output
[harvie@harvie-ntb shared]$ export LD_PRELOAD="./test.so"
[harvie@harvie-ntb shared]$ ./victim
puts("this text is output");
```

Asi ani nemusím říkat, že dávat do LD_PRELOAD relativní cestu je prasárna nejvyššího kalibru, ale funguje to. ;o) Každopádně pro seriózní použití je lepší zadat cestu celou. Pokud potřebujete preloadnout víc knihoven, oddělte je mezerami (to se stalo například, když jsem testoval spuštění programu s torify a aoss wraperama najednou...)

Také by pro upřesnění bylo dobré zmínit, že funkce, které jsou přímo zakompilované do programu nelze pomocí DLL Injektáže ovlivnit. Pokud tedy v souboru victim.c vytvoříme ještě jednu funkci puts(), tak "přepíše/přebije" tu standartní i preloadnutou.

V mém jednoduchém příkladu jsem dosáhl toho, že mohu například pro ladící účely zjistit, kdy libovolný zvolený program používá funkci puts(). To je samozřejmě jen drobná demonstrace toho, co se s DLL injekcemi dá skutečně dělat. Nebudu tu nijak rozebírat, k jakým druhům kladné či záporné činnosti toto lze použít, protože to záleží jen na vaší fantazii.

Pokud chcete zjistit, jak to funguje na windows, nebo sháníte podrobnější informace, tak wikipedie je celkem vyčerpávající.