Příloha B

Manuál k programu

Nástroj se používá k vyvolávání chyb v systémových voláních v operačním systému Linux. Nástroj obsahuje konzolovou aplikaci i aplikaci s grafickým uživatelským rozhraním. Obě verze programu je potřeba spouštět v terminálu jako uživatel root.

B.1 Požadavky

- operační systém Linux
- nástroj Systemtap
- Python verze 2
- knihovna PyGTK (pouze aplikace s GUI)
- manuálové stránky (alespoň sekce 2)

B.2 Konzolová aplikace

Konzolová aplikace se spouští následovně:

```
./fi-cmd.py -i INPUT -c COMMAND
```

kde `INPUT` je vstupní soubor s pravidly a `COMMAND` je aplikace, která se bude testovat (pokud obsahuje argumenty je třeba psát do uvozovek).

Příklad:

```
./fi-cmd.py -i vstup.txt -c "mkdir new_dir"
```

B.2.1 Struktura vstupního souboru

Vstupní soubor předaný pomocí přepínače -i se skládá z pravidel pro jednotlivá systémová volání. Pro jedno konkrétní systémové volání vypadá pravidlo následovně:

```
SYSCALL_NAME (ARG1, ARG2, ..., ARG6) = RETURN_CODE
```

kde

- `SYSCALL_NAME` je jméno systémového volání.

- `ARG1 .. ARG6` jsou argumenty systémového volání, argumenty nemusí být zadány. Pokud chceme zadat např. pouze třetí argument místo předchozích argumentů použijeme znak "-", pokud chceme zadat např. pouze první, tak následující argumenty nemusí být zadány (není potřeba ani znak "-").
- `RETURN_CODE` je návratová hodnota, kterou chceme vnutit systémovému volání např. `ENOENT`, `EPERM`, `EACCES`.

Příklad:

- `open(-,O_RDONLY) = EACCES` vnutí všem systémovým voláním `open`, které budou otvírat soubor pro čteni, návratovou hodnotu `EACCES` (přístup odmítnut).
- `mkdir = EEXIST` vnutí všem systémovým voláním `mkdir` návratovou hodnotu `EEXIST` (soubor již existuje).

V jednom souboru může být samozřejmě více pravidel pro různá systémová volání.

B.3 Aplikace s grafickým uživatelským rozhraním

Aplikace s grafickým uživatelským rozhraním se spouští příkazem:

```
./fi-gui.py
```

První spuštění trvá déle, protože aplikace hledá existující systémová volání a jejich návratové hodnoty.

Aplikace pracuje ve dvou módech. První mód tzv. Základní (Basic) má předchystané chyby, které lze rovnou vkládat do testovaného programu. Zatímco v druhém módu tzv. Pokročilém (Advanced) si musíme první vybrat, do kterých systémových volání budeme vkládat chyby a jaké budou tyto chyby. Přepínaní mezi módy se provádí pomocí položky Mode v menu aplikace.

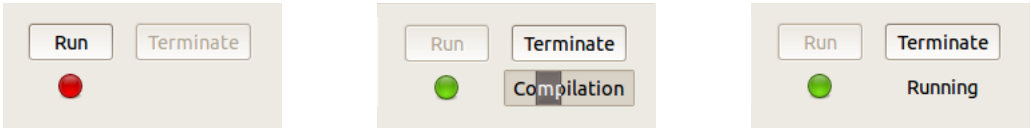
B.3.1 Společná čast pro oba módy

V horní části okna aplikace máme společný panel pro ovládání testované aplikace. V levé části horního panelu máme dvě textová pole, horní je určeno pro cestu k testované aplikaci a spodní je určeno pro případné argumenty předávané testované aplikaci. Pokud testujeme aplikace, jejíž cesta je určena v systémové proměnné `PATH`, nemusíme zadávat kompletní cestu, ale stačí zadat pouze její název. Aplikace umožňuje testovat pouze jeden proces. Pokud testovaná aplikace vytvoří další podprocesy, nebudou se vkládané chyby vztahovat na tyto podprocesy.

V pravé části horního panelu máme dva tlačítka. Tlačítko Run pro spuštění testované aplikace a tlačítku Terminate, které ukončí běh testované aplikace. Po spuštění testování se musí nejprve předpřipravit chyby uvnitř programu, což trvá nějakou dobu. O tom, že se něco děje nás informuje progressbar pod tlačítky.

Pod tlačítky máme barevný puntík (viz obrazek B.1), který nás informuje o stavu testování. Pokud je puntík červený testování nezačalo nebo již skončilo. Pokud je puntík zelený testování je spuštěno. Kromě toho máme vedle ještě informační text, který nám oznamuje, že testování běží (Running) nebo bylo ukončeno (Stopped). Při překládání systemtapového skriptu je text nahrazen progressbarem.

Ve spodní části okna můžeme vidět Log, kam se vypisují informace o právě vyvolané chybě. Formát vypisované informace je následující:



Obrázek B.1: Informace o stavu testování: Zastaveno - Překlad skriptu - Spuštěno

```
SYSCALL () = OLD_RET_VAL OLD_RET_NAME -> NEW_RET_VAL NEW_RET_NAME
```

kde

- `SYSCALL` je jméno systémového volání, v kterém byla vyvolána chyba
- `OLD_RET_VAL` je původní návratová hodnota
- `OLD_RET_NAME` je jméno konstanty původní návratové hodnoty (pokud návratová hodnota není chyba, tak tato hodnota chybí)
- `NEW_RET_VAL` je nová návratová hodnota
- `NEW_RET_NAME` je jméno konstanty nové návratové hodnoty

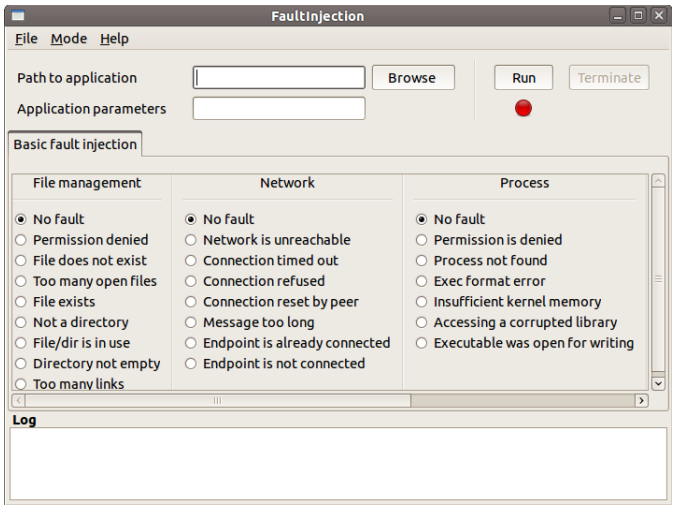
Například:

```
open () = 5 -> -13 EACCES
```

což znamená, že systémové volání `open` vrátilo původně hodnotu 5, tato hodnota byla ale nahrazena hodnotou -13, která značí chybu `EACCES` (Přístup zamítnut).

Informace o vyvolaných chybách se vždy při novém spuštění testované aplikace vymažou.

B.3.2 Základní mód



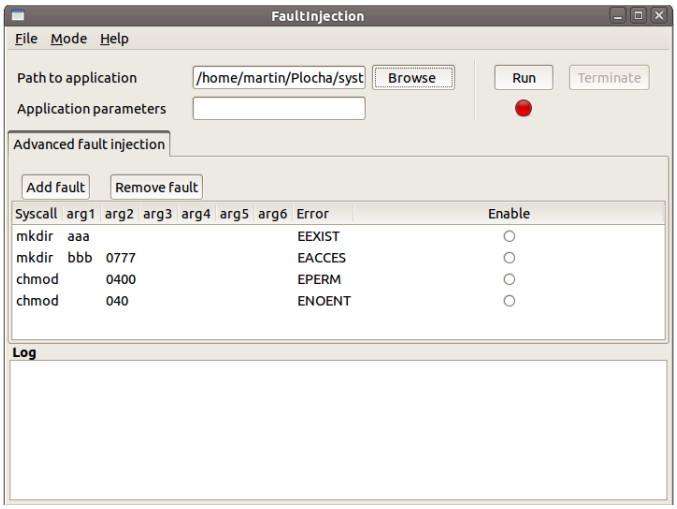
Obrázek B.2: Aplikace v základním módu

V základním módu máme předchystané chyby, které jsou rozděleny do následujících kategorií:

- File management - správa souborů
- Network - práce se sítí
- Process - práce s procesy
- Memory - práce s pamětí
- Read/Write - operace čtení a zápisu

Předchystané chyby se při spuštění zatím nikam nevkládají. Ke vložení odpovídající chyby dojde až zatrhneme odpovídající RadioButton u chyby. Chyby můžeme vybrat před spuštěním testované aplikace, ale i za běhu aplikace. Z každé kategorie může být najednou vybrána (vložena) maximálně jedna chyba. Do Logu se nám vypisují informace o aktuálně vyvolané chybě (více viz výše).

B.3.3 Pokročilý mód



Obrázek B.3: Aplikace v pokročilém módu

Do pokročilého módu se dostaneme přes menu Mode - Advanced. V pokročilém módu si musíme nejprve vybrat systémová volání a návratové hodnoty, které jim budou vnuceny. Systémové volání se přidá pomocí tlačítka Add fault. Po stisku tohoto tlačítka se nám zobrazí nový dialog. Pomocí comboboxu si vybereme systémové volání, do kterého budeme chtít vložit chybu. Poté se nám vedle v tabulce zobrazí možné návratové hodnoty, které v tomto volání lze vyvolat. Pomocí checkboxu v prvním sloupci se vybereme návratové hodnoty, která budeme chtít tomuto volání vnutit.

Pod vybraným jménem systémového volání máme šest textových polí, do nichž můžeme zadat argumenty, které chceme ,aby mělo testované volání. Pokud je nezadáme budou chyby vloženy do všech systémových volání tohoto jména.

Pokud máme zadány všechny požadované parametry, můžeme kliknout na tlačítko Add. V hlavním okně v tabulce se nám zobrazí všechny chyby (návratové hodnoty), které jsme vybrali. Pomocí tlačítka Add fault můžeme přidat obdobně další volání a návratové hodnoty, které v nich budeme vyvolávat. Můžeme zvolit znovu i systémové volání, které jsme už

jednou vybrali, ale nemůžeme u něho vybrat návratovou hodnotu, kterou jsme u něho ji6 jednou vybrali. Pomocí tlačítka Remove fault můžeme vybranou chybu odstranit.

A jako u Základního módu, aby se chyba v testovaném programu vyvolala musíme ji nejprve vybrat. To můžeme udělat pomocí RadioButton ve sloupci Enable. U každého systémového volání můžeme najednou vložit pouze jednu chybu. Opět se nám do Logu vypisují informace o aktuálně vyvolané chybě.

Vybraná systémová volání a chyby, které do nich budeme vkládat si můžeme uložit do souboru. Jednoduše v menu zvolíme File - Save as (nebo Save) a zadáme jméno souboru. Příště pak můžeme uloženou konfiguraci načíst pomocí menu File - Open. Načtenou konfiguraci můžeme změnit a opět uložit pomocí menu File - Save.

B.4 Použití testovacích příkladů

K aplikaci je vytvořeno několik jednoduchých programů na testování, které umožní uživateli seznámit se s aplikací. Testovací programy většinou opakují nějakou činnost v cyklu. Mezi jednotlivými opakováními cyklu je většinou časová prodleva 3 s, aby se mohla vložit/změnit chyba.

B.4.1 Testovací aplikace open

Aplikace je určena k ukázce jak testovat systémové volání `open`. V cyklu se snaží otevřít soubor pro čtení, pokud se ho podaří otevřít, tak ho uzavře a snaží se ho otevřít znovu.

Použití

V nástroji pro testování zadáme cestu k programu open. Do pole pro argumenty zadáme název souboru, který se bude otevírat. Tento soubor nemusí třeba ani existovat. Poté spustíme testování pomocí tlačítka Run. Jakmile je skript pro vkládání chyb přeložen (zmizí progressbar), můžeme vkládat požadovanou chybu z kategorie File management. V logu můžeme sledovat vkládané chyby a v terminálu pozorovat chybová hlášení, které testovaný program vypisuje. Program můžeme ukončit tlačítkem Terminate.

B.4.2 Testovací aplikace write

Aplikace je určena k ukázce testování systémového volání `write`. V cyklu aplikace vypisuje na standardní výstup čísla (od 0 do 9).

Použití

V nástroji pro testování zadáme cestu k programu write. Poté spustíme testování pomocí tlačítka Run. Jakmile je skript pro vkládání chyb přeložen (zmizí progressbar), můžeme vkládat požadovanou chybu z kategorie Read/Write. V logu můžeme sledovat vkládané chyby a v terminálu pozorovat chybová hlášení, které testovaný program vypisuje. Program můžeme ukončit tlačítkem Terminate.

B.4.3 Testovací aplikace memory

Aplikace je určena k ukázce testování systémového volání `mmap2`. V cyklu (5krát) aplikace alokuje paměť. Začíná na 128 kB a postupně velikost zvětšuje. Pokud se paměť podaří naalokovat, tak ji zase uvolní.

Použití

V nástroji pro testování zadáme cestu k programu memory. Poté spustíme testování pomocí tlačítka Run. Jakmile je skript pro vkládání chyb přeložen (zmizí progressbar), můžeme vkládat požadovanou chybu z kategorie Memory. V logu můžeme sledovat vkládané chyby a v terminálu pozorovat chybová hlášení, které testovaný program vypisuje. Program můžeme ukončit tlačítkem Terminate.

B.4.4 Testovací aplikace net

Aplikace je určena k ukázce testování systémového volání `sendto` a `recvfrom`. Skládá se ze dvou částí: klient a server. Klient zasílá serveru malá písmena abecedy a server zasílá klientovi velká písmena abecedy. Obě aplikace cyklí donekonečna. Na standardní výstup aplikace vypisují přijatá a odeslaná data.

Použití

V nástroji pro testování zadáme cestu k programu klient nebo server. Klient očekává dva parametry, adresu serveru (localhost) a port, server očekává pouze port. První spustíme server a pak klienta (jeden v konzoli a druhý v nástroji pro testování). Jakmile je skript pro vkládání chyb přeložen (zmizí progressbar), můžeme vkládat požadovanou chybu z kategorie Network. V logu můžeme sledovat vkládané chyby a v terminálu pozorovat chybová hlášení, které testovaný program vypisuje. Program můžeme ukončit tlačítkem Terminate. Druhý program v terminálu ukončíme pomocí kláves `Ctrl + C`.

B.4.5 Testovací aplikace dir

Aplikace je určena k ukázce testování systémového volání `mkdir` a `chmod`. Aplikace byla navržena pro demonstraci testování v pokročilém režimu, kde se zadávají i parametry jednotlivých systémových volání. Aplikace vytvoří adresář se jménem `aaa`, kterému pak postupně mění práva na čtené pro vlastníka, čtení pro skupinu a čtení pro ostatní. Nakonec tento adresář smaže a celá operace se opakuje, akorát se pracuje s adresářem `bbb`.

Použití

Přepneme se do pokročilého režimu pomocí menu Mode - Advanced. Pak zadáme cestu k programu dir a pomocí menu File - Open otevřeme soubor `dir.fi`, který obsahuje předchystané chyby. Poté spustíme testování pomocí tlačítka Run. Jakmile je skript pro vkládání chyb přeložen (zmizí progressbar), můžeme vkládat požadovanou chybu. Z předchystaných chyb můžeme např. zkusit vložit chybu pouze do systémového volání `mkdir` pro adresář `aaa`, nebo do systémového volání `chmod` pouze při změně práv na čtení pro vlastníka.

Pozn. Adresáře se vytváří s právy 0777, práva pro čtení vlastníka jsou 0400, pro čtení skupiny 040 a pro čtení ostatních 04.

B.4.6 Testovací aplikace cmd_mkdir

V adresáři `cmd_mkdir` se nenachází přímo testovací aplikace, ale je zde soubor `mkdir.fi`, který slouží k ukázce použití konzolové aplikace Při spuštění aplikace s tímto vstupem vložíme chybu `EACCES` do systémového volání `mkdir`, které bude vytvářet adresář se jménem `bbb`.

Použití

Konzolovou aplikaci spustíme následovně:

```
./fi-cmd.py -i mkdir.fi -c "mkdir bbb"
```

a v terminálu můžeme sledovat vyvolanou chybu. Pokud zkusíme vytvořit adresář s jiným jménem, tak se chyba do systémového volání nevloží.