UP23 HW1

Due Date: 2023-04-17

Secured API Call

The homework aims to practice library injection, API hijacking, and GOT rewriting. You have to implement a `sandbox.so` by following the specification given below. Please read it carefully before you implement your homework.

Specification

Program Launcher

> You don't have to implement the launcher. Please use our launcher to invoke commands and load the required shared object.
> You can download our launcher from here (https://up23.zoolab.org/up23/hw1/launcher).

We use a `launcher` program to execute a command and load your `sandbox.so` using `LD_PRELOAD`. The launcher executes the command and passes the required environment variables to an invoked process. The environment variables include:

- `SANDBOX_CONFIG`: The path of the configuration file for `sandbox.so`.
- `LOGGER_FD`: the file descriptor (fd) for logging messages.

The usage of the `launcher` program is as follows

```
Usage: ./launcher sandbox.so config.txt command arg1 arg2 ...
```

Sandbox

You have to implement the `sandbox.so` that supports the following features.

- Implement a `__libc_start_main` to hijack the process's entry point.
- In the `__libc_start_main`, you should perform the necessary initializations and then call the real `__libc_start_main`.
- Your `sandbox.so` cannot have a function with the same function name listed in the API function list. To hijack an API function, you must perform GOT hijacking in the `__libc_start_main` of `sandbox.so`. The functions you have to hijack are listed in the API Function List section.

API Function List

All functions listed below should be logged to the file descriptor (fd) passed by the environment variable `LOGGER_FD` (check the examples for the detailed format).

1. `open`
   Allow a user to set the file access blacklist so that files listed in the blacklist cannot be opened. If a file is in the blacklist, return -1 and set `errno` to EACCES. Note that for handling symbolic linked files, your implementation has to follow the links before performing the checks.

2. `read`
   Your implementation must log all content into a file. The log file should be named in the format `{pid}-{fd}-read.log` and be created at `read` function on this fd be called first time. (If an fd is used more than one time in a process, keep logging the content into the same log file.)

   Furthermore, your implementation has to allow a user to filter the read content based on a keyword blacklist. The filter should be active for all read operations. If the filter detects a matched keyword in a read content, close the fd and return -1 with an `errno` setting to EIO. Do not log the content if it is filtered.

Suppose the blacklist contains the keyword `S3CR3T`. The following cases should be detected by the filter.

- Reading gets `abcd`, `def`, `S3CR3T`
- Reading gets `abcd`, `S3C`, `R3T` (should be detected on read of the `R3T`)

3. `write`
   Your implementation must log all content into a file. The log file should be named in the format `{pid}-{fd}-write.log` and be created at `write` function on this fd be called first time. (If an fd is used more than one time in a process, keep logging the content into the same log file.)

4. `connect`
   Allow a user to block connection setup to specific IP addresses and PORT numbers. If the IP and PORT is blocked, return -1 and set `errno` to ECONNREFUSED.

5. `getaddrinfo`
   Allow a user to block specific host name resolution requests. If a host is blocked, return EAI_NONAME.

6. `system`
   Commands invoked by `system` function should also be hijacked and monitored by your sandbox. Note that you cannot invoke the `launcher` again in your implementation. The correct and incorrect relationship for the invoked process should look like the example below.

   Correct example: `process1` calls `system("/bin/sh")`

```
- launcher
    |- process1
         |- /bin/sh
```

   Incorrect example: `process1` calls `system("/bin/sh")`

```
- launcher
    |- process1
         |- launcher
              |- /bin/sh
```

Configuration File Format

The configuration file is a text file containing blocked content for each API function. For each API, the general form is as follows.

```
BEGIN <API>-blacklist
rule1
rule2
...
END <API>-blacklist
```

A sample configuration is given below.

```
BEGIN open-blacklist
/etc/passwd
/etc/shadow
END open-blacklist

BEGIN read-blacklist
-----BEGIN CERTIFICATE-----
END read-blacklist

BEGIN connect-blacklist
www.nycu.edu.tw:4433
google.com:80
END connect-blacklist

BEGIN getaddrinfo-blacklist
www.ym.edu.tw
www.nctu.edu.tw
END getaddrinfo-blacklist
```

Note that there are multiple lines between `BEGIN` and `END` tags except the `read-blacklist`.

Examples

Here we provide some running examples. Please notice that the results presented here could be different from your runtime environment. You may simply ensure that the behavior is expected and the output format is correct. All the running examples use the same configuration given below.

```
BEGIN open-blacklist
/etc/passwd
/etc/group
END open-blacklist

BEGIN read-blacklist
-----BEGIN CERTIFICATE-----
END read-blacklist

BEGIN connect-blacklist
www.nycu.edu.tw:443
google.com:80
END connect-blacklist

BEGIN getaddrinfo-blacklist
www.ym.edu.tw
www.nctu.edu.tw
google.com
END getaddrinfo-blacklist
```

Example1

- command: `./launcher ./sandbox.so config.txt cat /etc/passwd`
- output:

```
[logger] open("/etc/passwd", 0, 0) = -1
cat: /etc/passwd: Permission denied
```

Example2

- command: `./launcher ./sandbox.so config.txt cat /etc/hosts`
- output:

```
[logger] open("/etc/hosts", 0, 0) = 5
[logger] read(5, 0x7fb7b2db2000, 131072) = 177
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.208.2   00fd90b988c7
[logger] write(1, 0x7fb7b2db2000, 177) = 177
[logger] read(5, 0x7fb7b2db2000, 131072) = 0
```

Example3

- command: `./launcher ./sandbox.so config.txt cat /etc/ssl/certs/Amazon_Root_CA_1.pem`
- output:

```
[logger] open("/usr/share/ca-certificates/mozilla/Amazon_Root_CA_1.crt", 0, 0) = 5
[logger] read(5, 0x7f6a9c486000, 131072) = -1
cat: /etc/ssl/certs/Amazon_Root_CA_1.pem: Input/output error
cat: /etc/ssl/certs/Amazon_Root_CA_1.pem: Bad file descriptor
```

Example4

> Deleted because duplicated to Example1.

Example5

- command: `./launcher ./sandbox.so config.txt wget http://google.com -t 1`
- output:

```
--2023-03-29 15:07:09--  http://google.com/
Resolving google.com (google.com)... [logger] getaddrinfo("google.com","(null)",0x7f
failed: Name or service not known.
wget: unable to resolve host address 'google.com'
```

Example6

- command: `./launcher ./sandbox.so config.txt wget https://www.nycu.edu.tw -t 1`
- output:

```
--2023-03-29 15:08:12--  https://www.nycu.edu.tw/
Resolving www.nycu.edu.tw (www.nycu.edu.tw)... [logger] getaddrinfo("www.nycu.edu.tw
203.66.32.225, 203.66.32.227, 203.66.32.231, ...
Connecting to www.nycu.edu.tw (www.nycu.edu.tw)|203.66.32.225|:443... [logger] conne
failed: Connection refused.
Connecting to www.nycu.edu.tw (www.nycu.edu.tw)|203.66.32.227|:443... [logger] conne
failed: Connection refused.
Connecting to www.nycu.edu.tw (www.nycu.edu.tw)|203.66.32.231|:443... [logger] conne
failed: Connection refused.
Connecting to www.nycu.edu.tw (www.nycu.edu.tw)|203.66.32.233|:443... [logger] conne
failed: Connection refused.
Connecting to www.nycu.edu.tw (www.nycu.edu.tw)|203.66.32.226|:443... [logger] conne
failed: Connection refused.
Connecting to www.nycu.edu.tw (www.nycu.edu.tw)|203.66.32.234|:443... [logger] conne
failed: Connection refused.
Connecting to www.nycu.edu.tw (www.nycu.edu.tw)|203.66.32.228|:443... [logger] conne
failed: Connection refused.
Connecting to www.nycu.edu.tw (www.nycu.edu.tw)|203.66.32.229|:443... [logger] conne
failed: Connection refused.
```

Example7

- command: `./launcher ./sandbox.so config.txt wget http://www.google.com -q -t 1`
- output:

```
[logger] getaddrinfo("www.google.com","(null)",0x7ffd88b905f0,0x7ffd88b905b8) = 0
[logger] connect(5, "142.251.43.4", 16) = 0
[logger] write(5, 0x56126a865840, 129) = 129
[logger] read(5, 0x56126a8658d0, 511) = 511
[logger] read(5, 0x56126a865acf, 512) = 512
[logger] read(5, 0x56126a865ccf, 129) = 129
[logger] read(5, 0x56126a865840, 6) = 6
[logger] read(5, 0x56126a866610, 8192) = 8192
[logger] read(5, 0x56126a866610, 6408) = 4650
[logger] read(5, 0x56126a866610, 1758) = 1758
[logger] read(5, 0x56126a865840, 2) = 2
[logger] read(5, 0x56126a865840, 3) = 3
[logger] read(5, 0x56126a865840, 2) = 2
```

- Note:
  - You should find an `index.html` that contains the HTML content. And you should also find the same content in the log file.

Example8

- command: `./launcher ./sandbox.so config.txt python3 -c 'import os;os.system("wget http://www.google.com -q -t 1")'`
- output:

```
[logger] read(5, 0x560ce341c5e0, 3908) = 3907
[logger] read(5, 0x560ce341d523, 1) = 0
[logger] read(5, 0x560ce341cad0, 33180) = 33179
[logger] read(5, 0x560ce3424c6b, 1) = 0
[logger] read(5, 0x560ce3428530, 10922) = 10921
[logger] read(5, 0x560ce342afd9, 1) = 0
[logger] read(5, 0x560ce3429540, 1598) = 1597
[logger] read(5, 0x560ce3429b7d, 1) = 0
[logger] read(5, 0x560ce342b990, 3664) = 3663
[logger] read(5, 0x560ce342c7df, 1) = 0
[logger] read(5, 0x560ce342fd60, 6752) = 6751
[logger] read(5, 0x560ce34317bf, 1) = 0
[logger] read(5, 0x560ce3433b00, 17923) = 17922
[logger] read(5, 0x560ce3438102, 1) = 0
[logger] read(5, 0x560ce3434b10, 31557) = 31556
[logger] read(5, 0x560ce343c654, 1) = 0
[logger] read(5, 0x560ce3435af0, 4274) = 4273
[logger] read(5, 0x560ce3436ba1, 1) = 0
[logger] read(5, 0x560ce3404330, 32838) = 32837
[logger] read(5, 0x560ce340c375, 1) = 0
[logger] read(5, 0x560ce340c3f0, 10516) = 10515
[logger] read(5, 0x560ce340ed03, 1) = 0
[logger] read(5, 0x560ce3408d20, 3908) = 3907
[logger] read(5, 0x560ce3409c63, 1) = 0
[logger] read(5, 0x560ce340bf40, 3548) = 3547
[logger] read(5, 0x560ce340cd1b, 1) = 0
[logger] read(5, 0x7f7bafc41150, 226) = 225
[logger] read(5, 0x7f7bafc41231, 1) = 0
[logger] system("wget http://www.google.com -q -t 1")
[logger] getaddrinfo("www.google.com","(null)",0x7ffd704a8120,0x7ffd704a80e8) = 0
[logger] connect(9, "142.251.43.4", 16) = 0
[logger] write(9, 0x55f9bb917110, 129) = 129
[logger] read(9, 0x55f9bb9171a0, 511) = 511
[logger] read(9, 0x55f9bb91739f, 512) = 512
[logger] read(9, 0x55f9bb91759f, 129) = 129
[logger] read(9, 0x55f9bb917110, 6) = 6
[logger] read(9, 0x55f9bb925990, 8192) = 8192
[logger] read(9, 0x55f9bb925990, 6383) = 4650
[logger] read(9, 0x55f9bb925990, 1733) = 1733
[logger] read(9, 0x55f9bb917110, 2) = 2
[logger] read(9, 0x55f9bb917110, 3) = 3
[logger] read(9, 0x55f9bb917110, 2) = 2
```

Homework Submission

Please provide a Makefile to compile your source code. Make sure your code can be compiled by `make`. We use the following command to test your program.

```
make
./launcher {your sandbox.so} {config file name} command arg1 arg2 ...
```

Please pack your files into a single `tar.gz` archive and submit your homework via the E3 system.

Grading

1. [30%] Your program has the correct output for all test cases listed in the examples section.
2. [60%] We use `N` additional test cases to evaluate your implementation. You get $\frac{60}{N}$ points for each correct test case.
3. [10%] If you can perform GOT hijacking without an external program, you get 10% * {ratio of the score you got from the above two items}.

> You may leverage external libraries that can be installed from the official Ubuntu package repository using the `apt` command. In this case, please list your library dependencies as comments in the `Makefile` and ensure you use the correct parameters to link against the required libraries.

Demo Steps

- https://md.zoolab.org/s/gLXd81Myi (https://md.zoolab.org/s/gLXd81Myi)

Hints

- You don't need to consider the non-PIE binary. We will use PIE binary to test your program only.
- You can use `readelf -r` to find the GOT offset of the binary.