HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA AN TOÀN THÔNG TIN

BÁO CÁO THỰC HÀNH SỐ 6

Môn học: PHÂN TÍCH MÃ ĐỘC

Giới thiệu công cụ STRACE

Tên sinh viên: Ninh Chí Hướng

Mã sinh viên: B20DCAT094

Nhóm lớp : 02

Giảng viên hướng dẫn: PGS.TS Đỗ Xuân Chợ

HÀ NỘI, THÁNG 9/2023

Table of Contents

I. Giới thiệu:

- Strace là một tiện ích CLI được sử dụng để kiểm tra các lỗi trong hệ thống từ cho phép giám sát các cuộc gọi hệ thống được sử dụng bởi một chương trình nhất định và tất cả các tín hiệu mà nó nhận được. Tiện ích cho phép người dùng giám sát và (kể từ phiên bản 4.15) can thiệp vào quá trình tương tác giữa chương trình và lõibao gồm các cuộc gọi hệ thống đang diễn ra, cửa sổ bật lên và các thay đổi trạng thái quy trình.
- Hoạt động của nó có thể thực hiện được nhờ một tính năng của nhân linux được gọi là ptrace. Nó tương tự như ứng dụng giàn có sẵn trên các hệ thống Unix khác. Chương trình Cygwin cung cấp một tiện ích tương tự. Cách sử dụng phổ biến nhất của nó là khởi động nó cùng với chương trình đang được truy tìm, chương trình này sẽ in ra danh sách các lệnh gọi của hệ thống mà nó thực thi.
- Trong số các đặc điểm nổi bật so với Strace, những đặc điểm sau nổi bật:
    - Có thể chỉ định một bộ lọc tên syscall được theo dõi (sử dụng tùy chọn -e trace =): theo tên, chẳng hạn như clone, fork, vfork; bằng cách sử dụng một trong các nhóm được xác định trước, chẳng hạn như% ipc hoặc% tệp; hoặc (kể từ phiên bản 4.17) sử dụng cú pháp biểu thức chính quy, như clock_ *.
    - Chỉ định danh sách các tuyến đường để theo dõi (-P /etc/ld.so.cache chẳng hạn).
    - Chỉ định danh sách các bộ mô tả tệp có I / O sẽ được kết xuất
    - Đếm thời gian thực hiện và đếm cuộc gọi tổng hợp
    - In dấu thời gian tương đối hoặc tuyệt đối
    - Sửa đổi mã trả về và mã lỗi của các lệnh gọi hệ thống đã chỉ định và đưa tín hiệu vào sau khi thực thi
    - Trích xuất thông tin trên bộ mô tả tệp (bao gồm cả ổ cắm).
    - In dấu vết ngăn xếp, bao gồm (kể từ phiên bản 4.21) ký hiệu nhu cầu (-k).
    - Lọc theo trạng thái trả về cuộc gọi syscall
    - strace hỗ trợ giải mã các đối số của một số lớp lệnh ioctl, chẳng hạn như BTRFS_ *, V4L2_ *, DM_ *, NSFS_ *, MEM *, EVIO *, KVM_ * và một số lớp khác.

II. Khái quát chung:

- Bài tập này giới thiệu việc sử dụng tiện ích strace trong Unix để ghi lại các lời gọi của hệ thống (system call) được thực hiện bởi một chương trình đang chạy.
- Mục tiêu:
    - Giúp sinh viên hiểu cách sử dụng strace.
    - Hiểu cách đầu ra của strace tương ứng với các system call.
    - Sử dụng strace để phân tích động phần mềm.
- Yêu cầu đối với sinh viên: Sinh viên có kiến thức về Linux/Unix, system call, giao thức UDP. Có thể sử dụng câu lệnh "man" để tìm hiểu, ví dụ: man udp; man strace

III. Các bài tập:

- Bài thực hành bao gồm 2 máy: client và server. Tuy nhiên chỉ có terminal của client.
- Địa chỉ IP của server là 10.10.0.2, có thể truy cập từ client. Trên server có một dịch vụ đang chạy có tên observer và nó cung cấp một loại giao tiếp mạng sử dụng UDP.
- Sinh viên có một bản sao của chương trình observer trên client. Nhiệm vụ của sinh viên là sử dụng công cụ strace để phân tích động chương trình observer và xác định cổng mà nó lắng nghe.

1.Khám phá:

- Sử dụng lệnh ping để xác nhận kết nối giữa client và server.

```
[ubuntu@the-client ~]$ ping 10.10.0.2
PING 10.10.0.2 (10.10.0.2) 56(84) bytes of data.
64 bytes from 10.10.0.2: icmp_seq=1 ttl=64 time=0.447 ms
64 bytes from 10.10.0.2: icmp_seq=2 ttl=64 time=0.415 ms
64 bytes from 10.10.0.2: icmp_seq=3 ttl=64 time=0.181 ms
64 bytes from 10.10.0.2: icmp_seq=4 ttl=64 time=0.178 ms
^C
--- 10.10.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3053ms
rtt min/avg/max/mdev = 0.178/0.305/0.447/0.126 ms
[ubuntu@the-client ~]$
```

- Sử dụng lệnh file trong Unix để tìm hiểu một chút về chương trình observer.

```
[ubuntu@the-client ~]$ file observer
observer: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared li
bs), for GNU/Linux 2.6.32, BuildID[sha1]=25c1d79fb76a5cdf9924a6ca08c2c4fb3fedff98, not stripped
[ubuntu@the-client ~]$
```

- Thử chạy chương trình observer

```
[ubuntu@the-client ~]$ ./observer
[ubuntu@the-client ~]$
```

2.Sử dụng strace:

- Sử dụng công cụ strace để quan sát các system call Linux được thực hiện bởi chương trình observer khi chạy: strace ./observer

```
[ubuntu@the-client ~]$ strace ./observer
execve("./observer", ["./observer"], 0x7ffcfa596960 /* 23 vars */) = 0
brk(NULL)                               = 0x1379000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0cdb577000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=22025, ...}) = 0
mmap(NULL, 22025, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f0cdb571000
close(3)                                = 0
open("/lib64/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0`&\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2156592, ...}) = 0
mmap(NULL, 3985920, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f0cdaf89000
mprotect(0x7f0cdb14d000, 2093056, PROT_NONE) = 0
mmap(0x7f0cdb34c000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c300
0) = 0x7f0cdb34c000
mmap(0x7f0cdb352000, 16896, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
x7f0cdb352000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0cdb570000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0cdb56e000
arch_prctl(ARCH_SET_FS, 0x7f0cdb56e740) = 0
access("/etc/sysconfig/strcasecmp-nonascii", F_OK) = -1 ENOENT (No such file or directory)
access("/etc/sysconfig/strcasecmp-nonascii", F_OK) = -1 ENOENT (No such file or directory)
mprotect(0x7f0cdb34c000, 16384, PROT_READ) = 0
mprotect(0x601000, 4096, PROT_READ)     = 0
mprotect(0x7f0cdb578000, 4096, PROT_READ) = 0
munmap(0x7f0cdb571000, 22025)           = 0
brk(NULL)                               = 0x1379000
brk(0x139a000)                          = 0x139a000
brk(NULL)                               = 0x139a000
open("/tmp/log.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
fstat(3, {st_mode=S_IFREG|0664, st_size=0, ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0cdb576000
write(3, "port is 10100\n", 14)         = 14
socket(AF_INET, SOCK_DGRAM, IPPROTO_IP) = 4
open("/var/run/config.txt", O_RDONLY)   = -1 ENOENT (No such file or directory)
write(3, "Error opening config file\n", 26) = 26
exit_group(1)                           = ?
+++ exited with 1 +++
[ubuntu@the-client ~]$
```

- Cổng đang mở là 10100

3.Kiểm tra cổng bằng sendudp.py:

- Kiểm tra xem file sendudp.py có nội dung gì? Làm nhiệm vụ gì?

```
#!/usr/bin/env python
import sys
import socket
host = '10.10.0.2'
if len(sys.argv) < 2:
    print('./sendudp.py <port>')
    exit(1)
try:
    port = int(sys.argv[1])
except:
    print('bad port number')
    exit(1)
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_addr = (host, port)
msg = str.encode("HI THERE")
sock.sendto(msg, server_addr)
sock.settimeout(5)
try:
    got = sock.recvfrom(1024)
    print(got[0].decode('utf-8'))
except:
    print("No reply from %d" % port)
```

- Chương trình sendudp.py là một chương trình viết bằng Python. Nhiệm vụ của chương trình này là gửi một gói tin UDP đến một máy chủ đích trên một cổng cụ thể, và sau đó chờ nhận phản hồi từ máy chủ đó.
- Cụ thể, chương trình này lấy địa chỉ IP của máy chủ đích là '10.10.0.2' từ biến host. Nếu không có cổng được cung cấp dưới dạng tham số dòng lệnh, chương trình sẽ in ra thông báo hướng dẫn và thoát với mã lỗi 1. Nếu cổng được cung cấp, chương trình sẽ thử chuyển đổi nó sang số nguyên. Nếu không thành công, nó sẽ in ra thông báo lỗi và thoát với mã lỗi 1.
- Sau đó, chương trình tạo một đối tượng socket UDP bằng cách sử dụng socket.socket(socket.AF_INET, socket.SOCK_DGRAM). Nó sau đó gửi một gói tin chứa chuỗi "HI THERE" đến địa chỉ máy chủ và cổng được chỉ định.
- Tiếp theo, chương trình đặt thời gian chờ nhận dữ liệu là 5 giây bằng cách sử dụng sock.settimeout(5). Sau đó, nó cố gắng nhận dữ liệu từ socket. Nếu nhận được dữ liệu, nó sẽ in ra nội dung của gói tin nhận được dưới dạng chuỗi. Nếu không nhận

được dữ liệu trong thời gian chờ, chương trình sẽ in ra thông báo "No reply from <port>", trong đó <port> là số cổng đã chỉ định.

- Sau khi tìm được cổng hay kiểm tra kết quả bằng cách sử dụng tập lệnh sendudp.py: ./sendudp.py <port>

```
[ubuntu@the-client ~]$ ./sendudp.py 10100
Yup, that is the port number!

[ubuntu@the-client ~]$
```

4.Phân tích động:

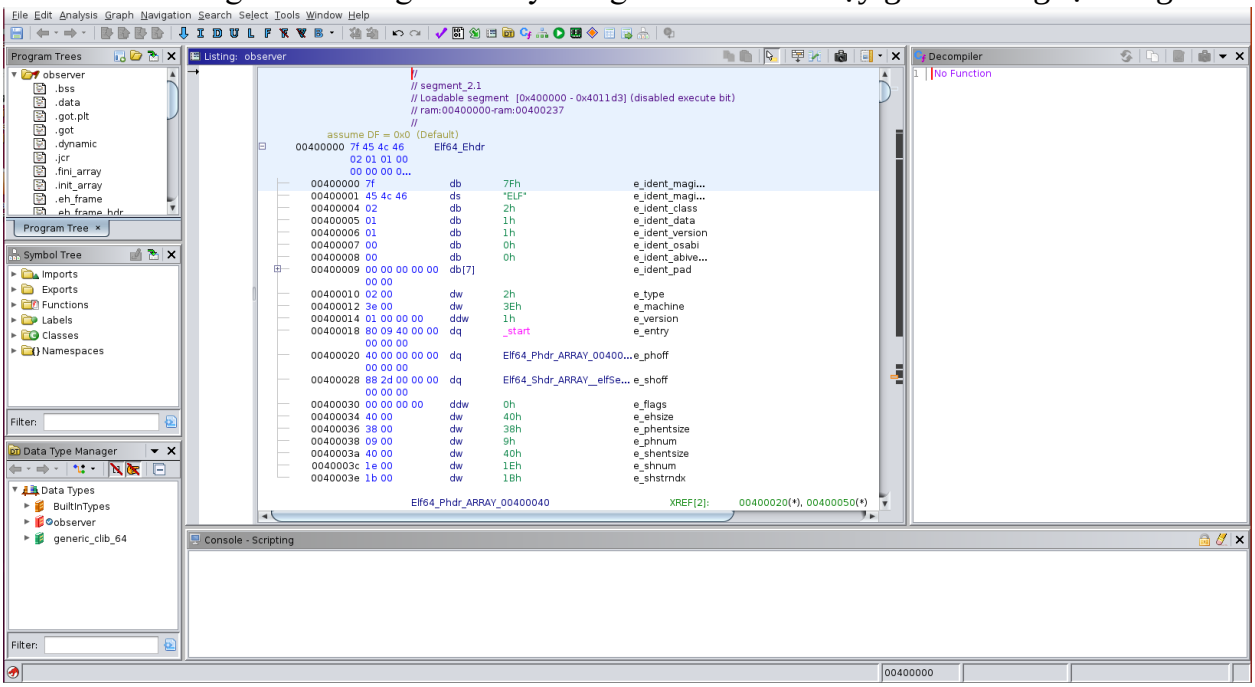
- Xem xét các tình huống mà kiểu phân tích động được cung cấp bởi strace có thể hữu ích do sự giới hạn của các chương trình khác. Ví dụ, các công cụ như nmap có thể gặp khó khăn trong việc xác định các cổng UDP mở. Ngay cả khi biết số cổng cần quét, nmap cũng có thể không cho kết quả tin cậy. Thử câu lệnh: sudo nmap -sU -p <port> 10.10.0.2

```
[ubuntu@the-client ~]$ sudo nmap -sU -p 10100 10.10.0.2

Starting Nmap 6.40 ( http://nmap.org ) at 2023-10-31 07:49 UTC
Nmap scan report for strace.the-server.student.lan (10.10.0.2)
Host is up (0.00010s latency).
PORT      STATE  SERVICE
10100/udp closed unknown
MAC Address: 02:42:0A:0A:00:02 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.38 seconds
[ubuntu@the-client ~]$
```

- Các công cụ phân tích tĩnh như Ghidra là một phương pháp thay thế khác để phân tích ngược các chức năng của chương trình. Tuy nhiên, một số chương trình có thể làm mờ các chức năng của chúng, làm phức tạp quá trình xác định các cổng. Sinh viên hãy phân tích chương trình observer sử dụng công cụ ghidra, sau đó mô tả lại các chức năng của chương trình này trong báo cáo. Để chạy ghidra dùng lệnh: ./ghidra



- Mở tệp /tmp/log.txt ở chế độc ghi

```
25
26    local_28 = fopen("/tmp/log.txt","w");
```

- Biết được port

```
28    iVar1 = rand();
29    local_2c = iVar1 % 2000 + 10000;
30    fprintf(local_28,"port is %d\n",(ulong)local_2c);
31    fflush(local_28);
```

- Mở tệp /var/run/config.txt ở chế độ đọc

```
local_40 = fopen("/var/run/config.txt","r");
```

- Mở tệp /etc/hostname

```
local_58 = fopen("/etc/hostname","r");
```

- Tạo 1 socket

```
local_1c = socket(2,2,0);
```

IV. Kết luận:

Strace là một công cụ mạnh mẽ được sử dụng trong các hệ điều hành dựa trên Unix (như Linux) để theo dõi và ghi lại các cuộc gọi hệ thống và hoạt động của quá trình. Strace cung cấp khả năng theo dõi các cuộc gọi hệ thống, bao gồm việc mở và đóng tệp, đọc và ghi dữ liệu, tạo tiến trình, giao tiếp qua socket, và nhiều hoạt động khác liên quan đến hệ thống. Công cụ này cho phép người dùng xem chi tiết về hoạt động của một chương trình, bao gồm các tham số đầu vào, giá trị trả về, lỗi xảy ra và các sự kiện liên quan khác. Strace có thể được sử dụng để gỡ lỗi và phân tích các vấn đề liên quan đến hệ thống, xác định các lỗi xảy ra trong quá trình thực thi chương trình, và nắm bắt thông tin chi tiết về các tệp và tài nguyên mà chương trình sử dụng. Strace là một công cụ mạnh mẽ nhưng có thể phức tạp đối với người mới sử dụng hoặc không quen thuộc với lĩnh vực hệ thống và lập trình dưới môi trường Unix.

```
student@ubuntu:~/labtainer/labtainer-student$ checkwork strace
Results stored in directory: /home/student/labtainer_xfer/strace
Labname strace

Student              |    strace_count |      found_port |
==================== | =============== | =============== |
B20DCAT094           |               1 |               Y |
What is automatically assessed for this lab:
```