Software is imperfect!
...so how to deal with it?

Krzysztof Opasiak
Samsung R&D Institute Poland
Agenda

Intro & Problem Statement

Audit Based Solution

rlimit-events PoC

Discussion
Intro & Problem Statement
George on holidays

• George is a clever guy
• ...but he can't count money
• He goes to holidays and takes a suitcase filled with...
• First he uses all the cash
• Then tries to pay with paper
• Is he going to go to jail?
George on holidays

• George is a clever guy
• ...but he can't count money
• He goes to holidays and takes a suitcase filled with…
• First he uses all the cash
• Then tries to pay with paper
• is he going to go to jail?
George on holidays - answer

• Yes
  • If he pays with toilet paper and get caught

• No
  • If he doesn't get caught
  • If he doesn't pay with toilet paper
George on holidays - solution

- Teach George how to count money
- Check George's suitcase every N min
- Install "IoT HW" in suitcase which notifies when there is $ X left
Linux resources

system-wide

• CPU time
• Memory
• I/O
• etc.
• Controlled using cgroups

process-specific

• CPU time
• Stack size
• Number of open files
• etc.
• Controlled using rlimits
```c
void store_data(char *path,
                char *data, int len)
{
    int fd;
    fd = open(path, O_RDWR);
    write(fd, data, len);
}

• Is this going to work?
• Yes, as long as:
  • path is valid and writable
  • limit of open fd has not been reached
• After that we are going to lose the data…
```
void store_data(char *path, char *data, int len)
{
    int fd;

    fd = open(path, O_RDWR);

    write(fd, data, len);
}

• Is this going to work?
• Yes, as long as:
  • path is valid and writable
  • limit of open fd has not been reached
• After that we are going to lose the data…
george.c - solution

- Check return value and handle -EMFILE correctly
  - What if we don't have access to source code?
george.c - solution

- Check return value and handle -EMFILE correctly
  - What if we don't have access to source code?
- Use some monitoring solution - polling (Nagios & friends)
  - Good for servers where power is unlimited
  - Increase energy consumption esp. on small IoT devices
  - No idea what happens between checks

Get notification when limit has been reached

Even better - get notification without losing the data
george.c - solution

- Check return value and handle -EMFILE correctly
  - What if we don't have access to source code?
- Use some monitoring solution - polling (Nagios & friends)
  - Good for servers where power is unlimited
  - Increase energy consumption esp. on small IoT devices
  - No idea what happens between checks
- Get notification when limit has been reached
- Even better - get notification without loosing the data
Audit Based Solution
Audit Solution

- Set rlimit to some reasonable value like
- Use audit syscall
- Get netlink notification when some syscall returned -EMFILE
Problems

• Post factum notification
• Cannot prevent data loss
• Error is returned to the service
• Not all resources are traceable (eg. CPU)
Problems

- Post factum notification
- Cannot prevent data loss
- Error is returned to the service
- Not all resources are traceable (e.g., CPU)
- Audit is very slow
- Measured overhead for open() syscall:
  - 33% for cold file
  - 45% for hot file
rlimit-events PoC
rlimit-events

- Notification sent by kernel when given process crosses certain resource usage level
- Resources are the same as for rlimits
- Notification via fd
- Auto-cleanup when fd is closed
- Monitoring level may be smaller than rlimit
- A little bit similar concept rejected due to tracepoints usage
API

- Use netlink socket to get watch fd
- Install watch levels via ioctl()
- Levels inherited by children
- Block in read() to get events
- Use poll() to handle multiple fds

```c
struct rlimit_noti_level w = {
    .subj = {
        .pid = PID,
        .resource = RLIMIT_NOFILE,
    },
    .value = 10,
};

noti_fd = get_noti_fd_via_netlink();
ioctl(noti_fd, &w);

read(noti_fd, noti, sizeof(noti));
print_noti(&noti);
close(noti_fd);
```
Design
Design
Benefits

- No need to filter nor route notifications in userspace
- Multiple monitors allowed
- Multiple notification levels allowed
- Notification possible before subject reaches the soft limit without returning error to service itself
- Can prevent data loss
- Can be used as Nagios Passive Check
There is no free lunch

• But this one is really cheap!
• No watchers -- less than 1% overhead
• Measured overhead for 40 000 open() calls:
  5.6% for hot files
  1.6% for cold files
TODO

- Cleanup
- Remove hacks
- Implement poll()
- Implement for other limits
- code @github.com/kopasiak
Discussion
• Is the idea fine?
• Is it acceptable upstream?
• Is the API fine? Maybe Netlink should be replace with sth other? But with what?
• What permissions should be necessary?
Thank you!

Krzysztof Opasiak
Samsung R&D Institute Poland

+48 605 125 174
k.opasiak@samsung.com
Images

- https://openclipart.org/detail/283205/empty-suitcase
- https://openclipart.org/detail/166284/snailescargotdecroissance