Tail of a Kernel bug

Alexandros Sapranidis
Principal Site Reliability Engineer @ Elastic

@alexsapran

Assumption

- We are running on Linux
- We know what system calls are
- What is a container and what it does
- What is strace and how to do it inside a container
- What's the difference between filesystem VS container storage driver
- How to read the linux tree and git your way around it

Application Stack

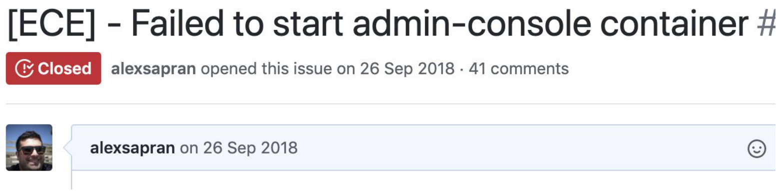
Application (Normally you app, Java/Scala/Kotlin/Rust/Go/html)

User space (Low-level system components, written in C, *open(), exec(), sbrk(), socket(), fopen(), calloc(), ….*)

Kernel (The linux kernel, *Process scheduling, IPC, Memory management, Network*)

Let begin with the story

Someone comes and tells you that the application failed to start, something like ...

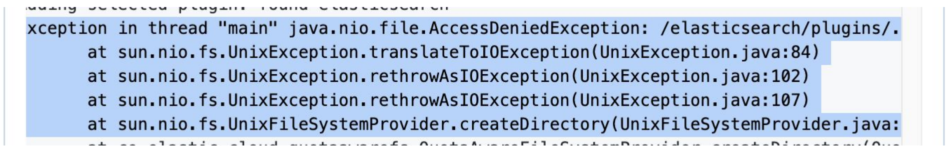


[ECE] - Failed to start admin-console container #

Closed alexsapran opened this issue on 26 Sep 2018 · 41 comments

alexsapran on 26 Sep 2018

.......deducted useless information from that guy^^ .....

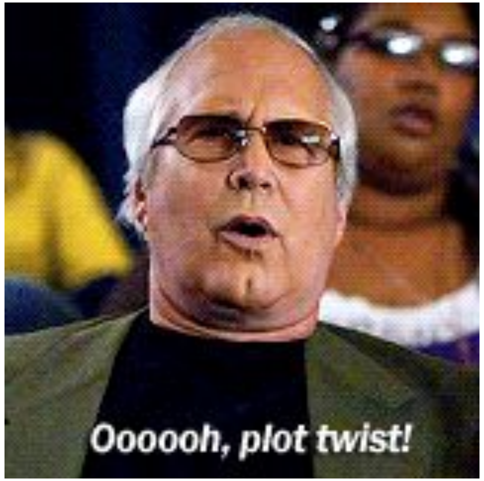


```
xception in thread "main" java.nio.file.AccessDeniedException: /elasticsearch/plugins/.
        at sun.nio.fs.UnixException.translateToIOException(UnixException.java:84)
        at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:102)
        at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:107)
        at sun.nio.fs.UnixFileSystemProvider.createDirectory(UnixFileSystemProvider.java:
```

java.nio.file.AccessDeniedException

The plot begins

- Q: Why the filesystem had wrong permissions, if that was the case, why everywhere else worked fine, except this one container?
- Q: What OS/Docker the customer was using?
- Q: How did we miss this during testing



A clue!!!

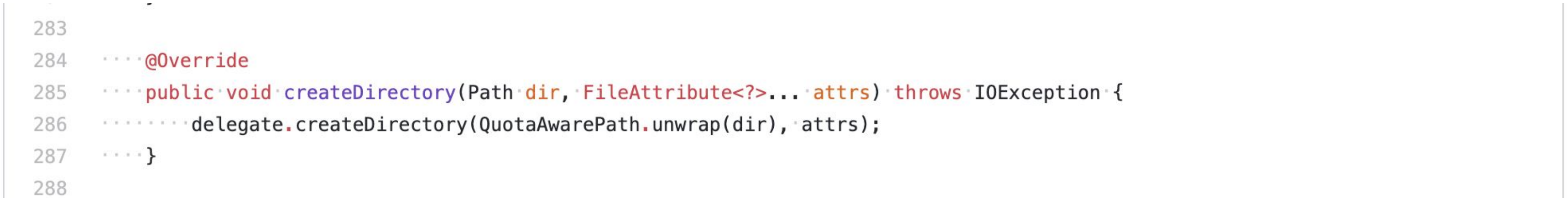
Kernel 4.4.121 worked which was the Kernel we were testing updating to 4.4.157 we managed to reproduce the issue !!

Finished right?

…..but still have not understood what happened here……

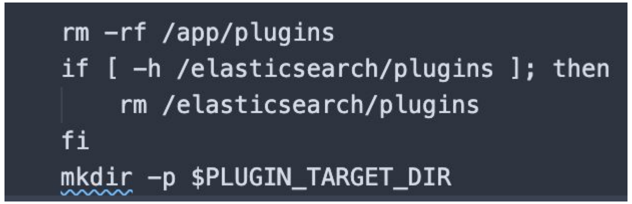
The code!

Reviewing the code, we immediately see that there are not much to do here.



```java
    @Override
    public void createDirectory(Path dir, FileAttribute<?>... attrs) throws IOException {
        delegate.createDirectory(QuotaAwarePath.unwrap(dir), attrs);
    }
```

Let's find who created that directory. Next in line is the container init script



```sh
rm -rf /app/plugins
if [ -h /elasticsearch/plugins ]; then
    rm /elasticsearch/plugins
fi
mkdir -p $PLUGIN_TARGET_DIR
```

Test the filesystem permissions inside docker

Create some directories

```
> elastic@4ee8e33470de:/elasticsearch$ mkdir -p plugins foobar
```

List the directories

```
> elastic@4ee8e33470de:/elasticsearch$ ls -lahZn

drwxr-xr-x   2 1000 1000 ?                    6 Sep 28 10:55 foobar

drwxr-xr-x   2    0    0 ?                    6 Sep 28 10:55 plugins
```

WHAT???????

Strange permissions right?

Lets strace it!

```
>elastic@4ee8e33470de:/elasticsearch$ strace mkdir -p plugins foobar
………. Alot of stuff happening trust me……….

81   umask(0)                  = 022

81   mkdir("plugins", 0755)         = 0

81   close(1)                  = 0
```

It looks ok!, but still the permission are wrong!

Start looking at Kernels changes

Start from the one we know it works and see what changed to the one it does not work

```
> sudo rpm -q --changelog kernel-lt-4.4.121-1.el7.elrepo.x86_64 |cat >
kernel-lt-4.4.121-1.el7.elrepo.x86_64

>sudo rpm -q --changelog kernel-lt-4.4.157-1.el7.elrepo.x86_64 |cat >
kernel-lt-4.4.157-1.el7.elrepo.x86_64
```

STUDY time

Eureka moment

https://github.com/torvalds/linux/commit/121b09d30d48a59a0ae621b130f3b4e42e724e68

ovl: override creds with the ones from the superblock mounter

Actual footage from me ^

The fix

Searching later versions we find a patch

https://github.com/torvalds/linux/commit/d0e13f5bbe4be7c8f27736fc40503dcec04b7de0

ovl: fix uid/gid when creating over whiteout

This was fixed and backported to 4.7 Kernel leaving the 4.4LTS affected thanks folks.

What is a whiteout you ask?

....

In order to support rm and rmdir without changing the lower filesystem, an overlay filesystem needs to record in the upper filesystem that files have been removed. This is done using whiteouts and opaque directories (non-directories are always opaque)......

Recap

1. We are running inside a container (we will see why this is important)
2. Our application get a file system permission error
3. It worked on older version of the kernel
4. This was an OverlayFS storage driver issue on a given Kernel version
5. This means that everyone that runs 4.4 TLS kernel are affected by this issue.

Lesson learned

- Overlay does crazy stuff when creates new files, mainly because of the layered filesystem
- The container filesystem is one more abstraction on top of the systems filesystem, so we need to debug that as well
- User space and Kernel are not scary places to hang out, if you know how to find what you are looking for
- Understanding your application environment helps a lot, in this case a patch kernel version broke our application.

QA

Thank you folks that's all for today!