# Unintended user data deletion from dCache

Robert Illingworth
FIFE
3 Feb 2023

# Short version

**Upgrade to ifdhc_config v2_6_16 !**

# More details

Around the end of October (ifdhc_config v2_6_10) we introduced a bug to file transfers where doing

```
ifdh cp file /pnfs/path/to/dir
```

would ***remove*** the target directory, and its contents. This is clearly a major issue. Data removed from (non-tape-backed) dCache this way is not recoverable.

(Note that this would not have worked before either, as copying to a directory requires the `-D` option, but it wouldn't have overwritten anything).

For reasons we don't entirely understand, this week we suddenly received multiple reports of overwritten directories. As far as we have been able to figure out, nothing changed recently to make it more likely to occur

# More details II

We have identified from logs a (possibly incomplete) list of around 16 users who have done this. The scope of the damage likely varies a lot. I know of at least one case who overwrote `/pnfs/minerva/persistent/users/<username>`, which is obviously very bad

We expect that in most cases this happened during interactive use, as it's easy to forget the `-D`. Batch jobs tend to use "official" scripts that have been tested, and already had the correct options, but there was nothing to stop this happening from a batch job too

ifdhc_config v2_6_16 changes the copy options so the overwrites will no longer happen even if the `-D` option is omitted. (You should still use it, as what does happen depends on the underlying transfer protocol)

# Even more details (for the experts)

```
ifdh cp file /pnfs/path/to/dir
```

ended up as something along the lines of

```
gfal-copy -f --just-copy file https://.../path/to/dir
```

--just-copy means skip pre-copy checks and -f is force copy. The unfortunate result of this is that it would try writing to the target URL, and if that failed it would issue a remove and try again. And it turned out that even without the force option, depending on the response code from the server, it could remove the target anyway.

The solution has been to remove the just-copy option, at the cost of an extra check for any existing file before the transfer. The reason why it was done this way was that with some protocols checking in advance is a very expensive operation. However, with WebDAV the check is relatively cheap, and the consequences of not doing it very serious

# Parting thoughts (lessons learned?)

Token authorization provides a way to do finer grained permissions. It's not hard to imagine a similar problem that involved mass deletion of data through leveraging the scalability of thousands of grid jobs.

Processes should run with the minimum permissions to do the task required. If grid jobs only have the power to upload files, and not delete them, there's no way for jobs run amok to remove all your experiment production data – something that is possible with our current proxy-based model.

We can't eliminate human error entirely, and with the complexities of our systems we can't be sure that we can avoid similar bugs in the future, but we can try to mitigate possible damage done by using features such as token capabilities