//**AAAARGHHH!**//

After a pretty nasty working expansion of the raid system of our production Zimbra server, fsck went berserk on our ext3 filesystem and reported missing inode links in various places (say „15367“ inodes). „Thankfully“ only our hsm-storage was affected. HSM is Zimbra’s way of automatically moving old messages to a cheaper and slower storage. So, mostly, this went under the user’s radar.

But selecting certain old messages resulted in a „Missing blob item“ error message displayed to the user.

What now? Let’s recap Zimbras database structure, which is technically based on MySQL. Zimbra assigns each user a mailstore-local mailbox id. This ID is put into a group of IDs running through exactly 100 groups. So the first 100 mailboxes go in their same-numbered group, mailbox #101 goes again into mailboxgroup1 and so on. This partitioning is made because of performance issues.

Additionally, a „zimbra“ database exists, that holds the information on what user goes with what mailbox. The „mboxgroup“-databases hold the meta information for the mails of a specific mailbox and is mainly used for searching and displaying purposes. The real content of the mail is stored inside the filesystem in a somewhat crude calculated path.

As I said, only our HSM-store was corrupted. The HSM-store is a so-called „zimbra-volume“, that is administered using the Zimbra Admin UI or the „zmvolume“ command. Let’s have a look at our volumes:

```
(as user zimbra)
# zmvolume -l
Volume id: 1
name: message1
type: primaryMessage
path: /opt/zimbra/store
compressed: false
current: true
```

Volume id: 2
name: index1
type: index
path: /opt/zimbra/index

compressed: false
current: true

Volume id: 3
name: message2
type: secondaryMessage
path: /var/opt/zimbra/hsm
compressed: false
current: true

As you can see, our hsm store is in „/var/opt/zimbra/hsm“ and has the volume id 3. The table „mail_item“ in a „mboxgroup“-database in turn has a „locator“ field, that corresponds with this volume id. So to find all mails, that reside in the HSM-store, you'll have to select those with the „locator“-field set to 3.

Another nice thing about the „mail_item“-table is, that it holds a base64-encoded sha-digest of the blob file.

So the plan was:

* Create the appropriate sha-digest for all missing inode-links (currently residing in /var/opt/zimbra/lost+found) in a script-readable form
* Search all mailboxes for missing blob items and check, if the digest of the missing blob item is matched against the missing inodes
* If so, copy (to be sure) the missing inode back to the real place.

To create the SHA-Digests you can just do the following:

```
#!/bin/bash

cd /var/opt/zimbra/lost+found

for FILE in `ls .`;
do
echo -n "$FILE:" >> /tmp/lostfound;
cat $FILE | openssl dgst -binary -sha1 | openssl base64 | sed -re "s/\//,/gi" >> /tmp/lostfound;
done
```

This builds up a file with two columns, separated by the „:“. The left side is the name of the

inode-file, the right side is the digest.

The digest itself is calculated using openssl: a sha1-digest, which is then base64-encoded. In that string, each „/“ is replaced with a „,“. That is the filesystem-safe digest-generation, that Zimbra uses (you’ll have to dig through the source code to find this).

To finally roam through the databases, I created the following python script:

```
#!/usr/bin/python

path = "/var/opt/zimbra/hsm/0"
locator = 3
lostfoundpath = "/var/opt/zimbra/lost+found"
lostfounddb = "/tmp/lostfound"
db_socket = "/opt/zimbra/db/mysql.sock"
db_user = "zimbra"
db_password = "VERYSECRET"

import glob
from os import path as ospath
import MySQLdb
import hashlib
import base64
import sys
import shutil

tmp = open(lostfounddb)

lostfounddb = {}

for line in tmp:

line = line.rstrip()
(filename, digest) = line.split(":")
lostfounddb[digest] = filename

dirs = glob.glob("%s/*" % path)

for dir in dirs:
mboxid = int(ospath.basename(dir))
```

```
print "Mailbox-ID: %s" % mboxid
mboxgroup = mboxid % 100
if mboxgroup == 0:
mboxgroup = 100
print "Mailboxgroup: %s" % mboxgroup

db = MySQLdb.connect(
unix_socket=db_socket,
user=db_user,
passwd=db_password,
db="mboxgroup%s" % mboxgroup
)

print "Selecting all mail items"

c = db.cursor()
c.execute("select id, blob_digest, mod_content from mail_item where mailbox_id=%s and
locator=%d" % (mboxid, locator))
mails = c.fetchall()

for mail in mails:
print "Checking Mail %s" % mail[0] parent_folder = int(mail[0]) >> 12

mailpath = "%s/%s/msg/%s/%s-%s.msg" % (
path,
mboxid,
parent_folder,
mail[0],
mail[2] )

if not ospath.exists(mailpath):
sys.stderr.write("Mailbox %s/Mail %s: Blob file %s does not exist!\n" % (mboxid, mail[0],
mailpath))

if mail[1] in lostfounddb:
sys.stderr.write("Lost+Found %s has the same digest! Copying it.\n" % lostfounddb[mail[1]])
shutil.copy("%s/%s" % (lostfoundpath, lostfounddb[mail[1]]), mailpath)

else:
```

```
blob_file = open(mailpath)
m = hashlib.sha1()
for line in blob_file:
m.update(line)
blob_file.close()
blob_hash = base64.b64encode(m.digest())
blob_hash = blob_hash.replace("/", ",")

if blob_hash != mail[1]:
sys.stderr.write("Mailbox %s/Mail %s/File %s: Blob hash %s doesn't match the one I generated: %s\n" % (
mboxid,
mail[0],
mailpath,
mail[1],
blob_hash
))
```

Be sure to match the configuration variables at the beginning of the script according to your environment. The mysql user and password are within the server's localconfig and can be retrieved as such:

```
(as user zimbra)
zmlocalconfig -s zimbra_mysql_user
zmlocalconfig -s zimbra_mysql_password
```

If you run this script, it will check all mailboxes, that also exist inside the hsm-directory (so newer mailboxes aren't checked). In these mailboxes all mails with the specified locator are checked:

* Does the blob file exist?
* If not: Does it exist as a missing inode?
* If yes: Copy it to the right place.
* Does the digest match the digest of the local blob file?((this sometimes doesn't work as Zimbra apparently used another digest-generator in earlier versions. However, I could not seem to generate the right digest for those mails))

Yes, I know, that matching a missing file because of its digest can lead to possible wrong

results. But this was the only possibility I had, because the blobs really were the raw mail content without a back reference to a mail id or something. So beat me.

I hope, this helps some people, who have the same horrible situation. The same script should also work for non-hsm-volumes (like the production default /opt/zimbra/store/0).