# SUSE Manager / Uyuni - Salt SSH Push: Thin package corrupted


| | |
|---|---|
| **Date:** | 2021-03-02 |
| **modified:** | 2021-03-02 |
| **tags:** | Uyuni, SUSE Manager, SSH, Salt, Exception |
| **description:** | SUSE Manager / Uyuni corrupted thin cause "ImportError: No module named 'salt.exceptions'" |
| **category:** | Linux |
| **slug:** | suse-manager-uyuni-salt-ssh-push-thin-package-corrupted |
| **Author:** | Dominik Wombacher |
| **lang:** | en |
| **transid:** | suse-manager-uyuni-salt-ssh-push-thin-package-corrupted |
| **Status:** | published |


At work we had issues with almost all Salt-SSH / SSH-Push connected Clients for months due to an unhandled exception `ImportError: No module named 'salt.exceptions'` that occurred regularly. It took me a lot of time to track the issue down together with SUSE so I want to share my Experience.

## Error

```
[...]
"stderr": "Traceback (most recent call last):\n File
\"/var/tmp/.user_a567dc_salt/salt-call\", line 26, in \n from
salt.scripts import salt_call\n File
\"/var/tmp/.user_a567dc_salt/pyall/salt/scripts.py\", line 21, in \n
from salt.exceptions import SaltSystemExit, SaltClientError,
SaltReqTimeoutError\nImportError: No module named 'salt.exceptions'"
[...]
```

## Background

Normally Salt is using an Agent (Salt-Minion) but also supports an Agentless approach called Salt-SSH. A package (*"thin.tgz"*), which contains all python code and files that Salt need to run Jobs on a server, will be uploaded and extracted to **/var/tmp/.<user>_<id>_salt/** on the target Server. The "*id*" is related to the used Salt Version, the "*user*" is the one configured on the Salt Master - in our Case SUSE Manager - which has full sudo permissions and is allowed to login via SSH on the target.

Everytime Salt connect to run a Job, some checks will performed to verify that state of the thin package:

- Does the Folder for the specified user with the matching id exist in */var/tmp/*
- Is the file *"code-checksum"* available and does the containing checksum matches with the thin.tgz package on the Salt Master

As long both checks are successful, the Salt Master will _not_ upload a new *thin.tgz* - that's important to understand.

# Troubleshooting

First it was totally unclear were the issue was coming from, it could occur with every job type, on different server at a different time. The thin package was always corrupted at some point what caused the unhandled exception. The Workaround was to manually delete the whole folder of the Salt thin package on the target Server so the Salt Master uploaded a fresh copy next time.

SUSE helped to go through tons of logs and to visualize a timeline were I could see that the issue seem to occur every 7 days.

Further investigation confirmed, in this case the problem was self-made. Someone from the Team configured systemd to automatically cleanup the tmp folder on a lot of server and neither tested nor documented it well. Removing the thin package folder wouldn't be a problem, it would be just newly uploaded, as long it's done right but it wasn't.

```
/etc/tmpfiles.d/clean_tmp.conf

D /tmp 1777 root root 7d
D /var/tmp 1777 root root 7d
```

The tmpfiles config removed only files and folder owned by user *root*, but a few files, like "*code-checksum*", of the thin package folder are owned by salt. So every 7 days almost all files were removed and therefore corrupted Salt thin, but the Salt Master found the folder and the "*code-checksum*" file with the correct hash and thought the thin state is healthy.

# Solution

At the end I just removed the custom tmpfiles config and once again the thin folder manually on all affected server.

For us there wasn't much benefit in deleting the thin package weekly and generate high load and consume bandwith to re-upload it. In case you prefer the regular cleanup of your tmp folders, just ensure that the whole thin folder and all files are covered and either removed ore kept untouched by your ruleset.

Even though it was a human error and not directly related to SUSE Manager / Uyuni or Salt it demonstrated how fragile such components can be. From a technical point of view I understand why there are just a few checks and validation of the hash in a file. Doing that for the whole folder content on every job run would horrible slow and increase the load. But as we can see, it comes with a price and can be affected by something you wouldn't expect.