Some kata-runtime annotations can execute arbitrary code

Bug #1878234 reported by Christophe de Dinechin on 2020-05-12

This bug affects 1 person

270

Affects	Status	Importance	Assigned to	Milestone
Kata Containers	Fix Released	Critical	Unassigned	

Bug Description

```
================================
This issue is being treated as a potential security risk under embargo.
Please do not make any public mention of embargoed (private) security
vulnerabilities before their coordinated publication by the Kata
Containers Vulnerability Management Team in the form of an official
Kata Containers Security Advisory. This includes discussion of the bug
or associated fixes in public forums such as mailing lists, code review
systems and bug trackers. Please also avoid private disclosure to other
individuals not already approved for access to this information, and
provide this same reminder to those who are made aware of the issue
prior to publication. All discussion should remain confined to this
private bug report, and any proposed fixes should be added to the bug
as attachments.
================================

A few of the kata-runtime annotations can be used to execute arbitrary
pre-existing binaries on the host.

For example, "virtio_fs_daemon" in combination with "virtio_fs_extra_args"
makes it possible to invoke a host binary with arbitrary args.

The hypervisor.path and hypervisor.jailer_path annotations could also be
used the same way.

Suggestion for fix: add valid annotation values to the configuration file
that lists the acceptable values for such annotations, with a suitable
default value of "empty".
```

See original description

CVE References

2020-27151

Christophe de Dinechin (i-christophe) wrote on 2020-05-13: #1

```
Proof of concept demonstrated with Kubernetes and current master using the
following yaml file:

apiVersion: v1
kind: Pod
metadata:
  name: annotation
  labels:
    app: annotation
  annotations:
    io.katacontainers.config.hypervisor.virtio_fs_daemon: "/usr/local/
bin/hello"
spec:
  runtimeClassName: kata-qemu-virtiofs
  containers:
    - name: jenkins
      image: jenkins/jenkins
      command: [ "bash" ]
      args: [ "-c", "ulimit -n 5000; ulimit -a; /usr/local/bin/jenkins.sh"
]

(The ulimit is to address another issue)

% cat /usr/local/bin/hello
#!/bin/bash

echo "Hello was invoked with args $@" >> /tmp/hello.log
/opt/kata/bin/virtiofsd "$@"

% cat /tmp/hello.log:

Hello was invoked with args --fd=3 -o source=/run/kata-containers/shared/
sandboxes/e42fde60bf44cf2ed68562d8640c2cadb09f8cec0c5fdf1852a156a2c91a23c7
-o cache=always --syslog -o no_posix_lock -d
Hello was invoked with args --fd=3 -o source=/run/kata-containers/shared/
sandboxes/701037911fc2803374a584df686575644602487efde8d7ffbe42f4f05bbab9cd
-o cache=always --syslog -o no_posix_lock -d
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-13: #2

```
Of note, if /usr/local/bin/hello is not executable, then the following
happens:

Failed to create pod sandbox: rpc error: code = Unknown desc = container
create failed: panic: runtime error: invalid memory address or nil pointer
dereference
[signal SIGSEGV: segmentation violation code=0x1 addr=0x0
pc=0x562180e63a48]
goroutine 7 [running]:
os.(*Process).wait(0x0, 0xc000000004, 0xc00035db50, 0x1)
  /usr/lib/golang/src/os/exec_unix.go:17 +0x28
os.(*Process).Wait(...)
  /usr/lib/golang/src/os/exec.go:125
github.com/kata-containers/runtime/virtcontainers.(*qemu).setupVirtiofsd
.func1(0x562181bea460, 0xc00058a040, 0xc0001ca500, 0xc0002dc000)
  /home/ddd/go/src/github.com/kata-containers/runtime/virtcontainers/
qemu.go:689 +0x4a5
```

Report a bug

This report contains **Public Security** information

Everyone can see this security related information.

You are not directly subscribed to this bug's notifications.

Edit bug mail

Other bug subscribers

Subscribe someone else

Notified of all changes

- Archana Shinde
- Christophe de Din...
- Dr. David Alan Gi...
- Julio Montes
- Karen Noel
- Peng Tao
- Wang Xu
- fidencio

May be notified

- Salvador Fuentes ...
- Wei Zhang

Patches

- Patches that only addresses the VirtIOFS case - It looks like the hypervisor is not
- Patch arbitrary host execution vulnerability in kata runtime
- Add patch

Bug attachments

- draft-KCSA
- Series of patches to control annotations
- Series of patches to control annotations
- Add attachment

```
created by github.com/kata-containers/runtime/virtcontainers.(*qemu)
.setupVirtiofsd
           /home/ddd/go/src/github.com/kata-containers/runtime/
virtcontainers/qemu.go:682 +0x2d8
  Warning FailedCreatePodSandBox 53s (x17 over 4m38s) kubelet, muse
(combined from similar events): Failed to create pod sandbox: rpc error:
code = Unknown desc = container create failed: panic: runtime error:
invalid memory address or nil
 pointer dereference
[signal SIGSEGV: segmentation violation code=0x1 addr=0x0
pc=0x55fdd99cea48]
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-14: #3

Patch under way.

Here are the annotations I identified so far with exec capabilities:

```
hypervisor.path
hypervisor.jailer_path
hypervisor.ctlpath
hypervisor.virtio_fs_daemon
```

Not confirmed yet, but likely a risk:
```
proxy.path
shim.path
netmon.path
```

Additionally, no exec but system file overwrite capabilities:

```
hypervisor.vhost_user_store_path
hypervisor.file_mem_backend
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-14: #4

Patches that only addresses the VirtIOFS case - It looks like the hypervisor is not (3.0 KiB, application/x-tar)

First pass at patch attached. It looks like the annotation for hypervisor path is not implemented.

Errors shown below

```
  Warning FailedCreatePodSandBox 2m10s (x16 over 5m57s) kubelet, muse
Failed to create pod sandbox: rpc error: code = Unknown desc = container
create failed: virtiofs daemon /usr/local/bin/hello required from
annotation is not valid
  Warning FailedCreatePodSandBox 47s (x6 over 116s) kubelet, muse Failed
to create pod sandbox: rpc error: code = Unknown desc = container create
failed: virtiofs daemon /usr/local/bin/hello required from annotation is
not valid (e
xpect one of [])
  Warning FailedCreatePodSandBox 32s (x2 over 4m21s) kubelet, muse Failed
to create pod sandbox: rpc error: code = Unknown desc = container create
failed: /opt/kata/bin/kata-qemu-virtiofs: line 2: /opt/kata/bin/kata-
runtime: Text fi
le busy
  Warning FailedCreatePodSandBox 6s (x2 over 19s) kubelet, muse Failed to
create pod sandbox: rpc error: code = Unknown desc = container create
failed: virtiofs daemon /usr/local/bin/hello returned with error:
fork/exec /usr/loca
l/bin/hello: permission denied
```

Peng Tao (bergwolf) wrote on 2020-05-15: #5

What about adding a global config option to disable the config-via-annotation property completely (and enable it by default)? Instead of trying to identify and remove each risky annotation configs, we keep the ability and warn about their usage.

In general, config-via-annotation is not a capability we would like to expose to end users. They are mostly targeting service providers who wrap their own services (e.g., container or kerbenetes as a server) over infrastructural Kubernetes and have a proper validation on every field user can input at a higher level.

Also while the executable options are more dangerous, other options might be used for an expolit as well. That's why I think we should completely disable such config-via-annotation for end users.

Christophe de Dinechin (i-christophe) wrote on 2020-05-15: #6

Sent reply this morning through email, but apparently the system is not smart enough to update the bug!!!

On 15 May 2020, at 08:20, Christophe de Dinechin <email address hidden> wrote:

> Le 15 mai 2020 à 05:35, Peng Tao <email address hidden> a écrit :

> What about adding a global config option to disable the config-via-
> annotation property completely (and enable it by default)? Instead of
> trying to identify and remove each risky annotation configs, we keep the
> ability and warn about their usage.

I think this is a good idea, but as an addition to the proposed fix, not a replacement.

If you decide to enable it, you should still not be allowed to execute host binaries.

I will add the option in a next iteration of the patch series.

> In general, config-via-annotation is not a capability we would like to
> expose to end users. They are mostly targeting service providers who
> wrap their own services (e.g., container or kerbenetes as a server) over
> infrastructural Kubernetes and have a proper validation on every field
> user can input at a higher level.

Actually, the reason I started looking into annotations is because they solved a problem we had, specifically about passing kernel boot options on a per-workload basis. So I conclude there are valid use cases where you would want them and still want the system to be secure.

```
> Also while the executable options are more dangerous, other options
> might be used for an expolit as well. That's why I think we should
> completely disable such config-via-annotation for end users.
```

```
I plan to review other options as well. This one is urgent and comes
first.
Other options may lead to exploits on a case-by-case basis, and indeed,
they need to be carefully scrutinized.
```

Peng Tao (bergwolf) wrote on 2020-05-15: **Re: [Bug 1878234] Re: Some kata-runtime annotations can execute arbitrary code** #7

```
On Fri, May 15, 2020 at 3:40 PM Christophe de Dinechin
<email address hidden> wrote:
[...]
Those options also have their valid use cases. For example, they would
allow easy rolling upgrade and fallback for each of these binaries.
Some additional validation/protection is OK but I don't think we
should completely disable them.

[...]

>
> > Also while the executable options are more dangerous, other options
> > might be used for an expolit as well. That's why I think we should
> > completely disable such config-via-annotation for end users.
>
> I plan to review other options as well. This one is urgent and comes
first.
> Other options may lead to exploits on a case-by-case basis, and indeed,
> they need to be carefully scrutinized.
And as you identify each secure option, it can be put into the
whitelist instead of being moving out of the blacklist.

Cheers,
Tao
--
Into Sth. Rich & Strange
```

Peng Tao (bergwolf) wrote on 2020-05-15: #8

```
So to summary what I have in mind:

Let's add a global config option about how config annotation is handled:
   -. disabled (the default)
   -. whitelist: only config options that we have identified to be secure
are allowed
   -. free-for-all: all config options are allowed to be altered via
annotations, not for end users, upper layer should have proper validation
on user's input

What do you think?
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-15: #9

```
While having a global "enable/disable" flag makes a lot of sense,
and is easy to implement and test (it's basically one "if" in
addAnnotations),
I don't like the "free-for-all/whitelist" distinction, because it would
add one more test for every single annotation.

There is no really good way to factor the annotation validation code,
what you need is on a per-annotation basis. So I can't have a single
function distinguishing between whitelist and free-for-all at one spot.
Distributing this test all over the place is fragile and error prone.

Does not mean it cannot be done, but it seems complicated and would
certainly delay the fix. So in order to understand why we need it,
could you give me some use cases where it would be useful and not
unreasonably dangerous to select free-for-all?
```

Peng Tao (bergwolf) wrote on 2020-05-15: #10

```
For `free-for-all`, it is the infrastructure's responsibility to make sure
no such thing as a kata config annotation is passed by users. This is
common for a cloud provider that would not really allow users to define
any kata related pod annotations, while the service provider can still
make use of kata's annotations to do a lot of customization. IOW, like
I've been repeated several times, IMO the config-via-annotation
functionality is not for end users. It is more targeting services
providers who wrap their own services on top of kata and kubernetes.

It is ok to just introduce a simple `enable`/`disable` flag at a quick
fix. But I don't think we should disable part of the functionality w/o
providing a way to enable them.
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-15: #11

```
> It is ok to just introduce a simple `enable`/`disable` flag at a quick
fix.
> But I don't think we should disable part of the functionality w/o
providing
> a way to enable them.

That makes me wonder if you have missed the mechanism I proposed just for
that in my patch.

Basically, if you want, say, to allow either /opt/kata/bin/virtiofsd and
/usr/local/bin/virtiofsd, all you need in your config is:

virtio_fs_daemon_list = [ "/opt/kata/bin/virtiofsd", "/usr/local/
bin/virtiofsd" ]

Then the mechanism works as before. You are just restricted as to which
binaries are accepted in the annotation.

Right now, this is an exact match. It could be a glob, i.e. we could maybe
accept /opt/kata/bin/virtiofsd* as a valid entry in the list.
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-15: #12

```
I believe the point I am making is that the policy of "free-for-all"
should probably be per-annotation, and that it's frankly easy enough to
implement it using pattern matching, which would give better control and
require less code (a single regexp test as opposed to testing different
conditions at different places).
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-15: #13

```
On top of the already submitted series, using regexp matching would look
like:

Author: Christophe de Dinechin <email address hidden>
Date: Fri May 15 15:42:08 2020 +0200

    config: Match valid virtiofsd using regular expressions

    This increases the configuration flexibiliy by allowing free-for-all
    annotations with arbitrary restrictions. For example, you can accept
    all variants of `virtiofsd` that are somewhere under `/usr`/ as well
    as one specifically installed under /opt/kata/bin/virtiofsd by adding
    this to the configuration:

    virtio_fs_daemon_list = [ "/opt/kata/bin/virtiofsd",
"/usr/.*/virtiofsd" ]

    Suggested-by: Peng Tao <email address hidden>
    Signed-off-by: Christophe de Dinechin <email address hidden>

diff --git a/virtcontainers/pkg/oci/utils.go b/virtcontainers/pkg/oci/
utils.go
index b47583ec..533f5593 100644
--- a/virtcontainers/pkg/oci/utils.go
+++ b/virtcontainers/pkg/oci/utils.go
@@ -11,6 +11,7 @@ import (
  "fmt"
  "path/filepath"
  goruntime "runtime"
+ "regexp"
  "strconv"
  "strings"
  "syscall"
@@ -194,6 +195,15 @@ func contains(s []string, e string) bool {
  return false
 }

+func regexpContains(s []string, e string) bool {
+ for _, a := range s {
+ if matched, _ := regexp.MatchString(a, e); matched {
+ return true
+ }
+ }
+ return false
+}
+
 func newLinuxDeviceInfo(d specs.LinuxDevice) (*config.DeviceInfo, error)
{
  allowedDeviceTypes := []string{"c", "b", "u", "p"}

@@ -663,7 +673,7 @@ func addHypervisporVirtioFsOverrides(ocispec
specs.Spec, sbConfig *vc.SandboxCon
  }

  if value, ok := ocispec.Annotations[vcAnnotations.VirtioFSDaemon]; ok {
- if !contains(runtime.HypervisorConfig.VirtioFSDaemonList, value) {
+ if !regexpContains(runtime.HypervisorConfig.VirtioFSDaemonList, value) {
    return fmt.Errorf("virtiofs daemon %v required from annotation is not
valid", value)
   }
   sbConfig.HypervisorConfig.VirtioFSDaemon = value
```

Dr. David Alan Gilbert (dgilbert-h) wrote on 2020-05-15: #14

```
My problem with that is what to do with 'virtio_fs_extra_args' - on our C
implementation I don't think there's anything too bad; but on the rust
version you're allowed to turn off sandboxing; and while I say 'not too
bad' - I owuldn't always trust adding arbitrary extra args is safe.
```

Peng Tao (bergwolf) wrote on 2020-05-15: #15

```
Yeah, I do understand your per-annotation whitelist approach. The problem
is that users would have to predefine a list of acceptable values for
every config option that is considered dangerous. Would it be too much for
a user to configure? Other than the executables (shim,proxy,
qemu,virtiofsd), we have other options that is equally important (like
kernel path, image path, initrd path, default vCPU number, default memory
size, guest kernel parameters etc.). We just might have too many options
for users to whitelist.

And to ease users pain of configuration, in my proposal, the `white-list`
option actually means that we only allow a predefined (by us!) set of
options that can accept annotation configuration.
```

Peng Tao (bergwolf) wrote on 2020-05-15: #16

```
Yes, Dave, I think none of the kata related configurations are for end
users (I'm repeating myself once again ;-). That's why I am proposing
three level of annotation configurations:

-. disabled for the good

-. white-list: only allow part of the options to be changed via
annotations, we are responsible for deciding which is safe enough to be
exposed to end users

-. free-for-all: upper layer of the infrastructure have done enough
validation, so we just provide all basic functionality and let upper layer
decide. Usually this is the case none of the kata related options are
allowed from end users by the infrastructure, but the infra itself can
still use such functionality
```

If we think it is too complex to implement, we can just remove the middle one, and simply have a global disable/enable button for the annotation configurations.

Christophe de Dinechin (i-christophe) wrote on 2020-05-15: #17

Exploring the code a little more, I see that

a) While there is an annotation for hypervisor.path, it's not implemented
b) Implementing does not work at least for qemu because qemuPath() goes straight to asserts.
c) This led me to discover that there is another way to override paths.

So now I need to make sure there isn't an issue with assets as well.

Christophe de Dinechin (i-christophe) wrote on 2020-05-15: #18

> The problem is that users would have to predefine a list of acceptable
> values for every config option that is considered dangerous.

Well, the way I'm currently doing it is to just have a comment-out list in the template configuration files. In other words, something like this:

```
# List of valid annotations values for the virtiofs daemon (default:
empty)
# virtio_fs_daemon_list = [ "/opt/kata/bin/virtiofsd", "/usr/.*/virtiofsd"
]
```

So we get to choose what we see as a valid default. If the user wants to enable our list, it's simply one commenting-out away. Not that difficult.

> If we think it is too complex to implement, we can just remove the
> middle one, and simply have a global disable/enable button for the
> annotation configurations.

The middle one is the one that is easy to implement, and the way I did it with regexps now, it also trivially allows whitelisting.

If you want a global whitelisting free-for-all, it then becomes nothing more than us providing a reasonable configuration-ffoa.toml.in (where ffoa stands for "free for all") with reasonably set ".*" values at the right spots (sometimes, "reasonable" might be something like allowing something named virtiofsd anywhere under /opt or /usr, or allowing anything that has vmlinu in it anywhere, etc.

In short, I believe I have implemented exactly what you want, except that it's not an option called "enable_annotations = free-for-all" with hard-coded values in the code, it's a default configuration file that exposes the mechanism and lets the user tweak it. To me, this is more functional, somewhat safer, and more importantly, way easier to implement that way (as in it's already ready to go for hypervisor and virtiofsd paths).

Peng Tao (bergwolf) wrote on 2020-05-15: #19

Yeah, it works from functionality point of view. My only concern is that it makes it harder for users to actually change the configuration file. Kata configuration would become more complex than it already is, because we would have to list all possible combinations (well, at least most of them) for most config options. Yeah, regex helps, but it still would be pretty complex for a configuration file. Ease of use has been one of our targets for a long time. I'm not sure if ease of configuration is part of it though. Anyway, it is just my feeling. Others might think it is just fine. I guess you can send a PR and let the community decide.

Christophe de Dinechin (i-christophe) wrote on 2020-05-15: #20

Who is responsible for filing a CVE for Kata? Should one be created here for people using this in production?

Christophe de Dinechin (i-christophe) wrote on 2020-05-15: #21

Patch arbitrary host execution vulnerability in kata runtime (36.6 KiB, text/plain)

A more complete patch is attached. This addresses the vulnerabilities I could find related to path handling. It also implements two missing annotations to make sure that this is done correctly and consistently.

The common approach in all cases is that for each dangerous "path" configuration that can be set through annotations, a matching "patch_list" configuration has been added that filters which annotations are accepted for "path". This lets system administrators indicate what kind of annotation they are willing to accept on their system. For example:

```
[hypervisor]
path = /usr/bin/qemu
path_list = [ "/usr/bin/qemu", "/opt/kata/bin/qemu" ]
```

At Peng Tao's suggestion, I added the possibility of "wildcard" filtering using regular expressions. So you can for example configure as follows:

```
[hypervisor]
path = /usr/bin/qemu
path_list = [ "/usr/.*/qemu.*", "/opt/.*/qemu.*" ]
```

While code-wise, this is extremely simple (one-line difference between text match and regexp match), some of my colleagues are a bit concerned that this presents a misconfiguration risk. One of the possible misconfigurations is explained in the configuration examples. Please read the configuration-qemu.toml.in and variants for an explanation.

I'm personally in favor of trusting sysadmins, in particular if we present them with relatively solid defaults and documents that lists of paths are safer than regexps. But I know of at least one colleague who thinks otherwise.

fidencio (fidencio) wrote on 2020-05-16: #22

So, before going through the patches, I'd like to have an agreement on what will be a reasonably okay solution.

```
The problem is real, and it is CVE material as annotations can just be
added by any user.

I see two different levels where a solution should be applied:
* The use of a mandatory validating web-hook, which cannot be enforced by
the kata project as it sits in an upper layer in the stack;

* The use of an allowed list of binaries, which can easily be enforced by
distributions and easily extended by administrators;

I personally see the former as desired and the latter as required to solve
the raised by Christophe.

Some thoughts about the requirements for the implementation of having an
allowed list of binaries:
* It must be (also) configurable via build time, then distributions have
an easy way to enforce this;

* It must not have support for regular expressions;
  - It is easy to get something wrong with regular expressions;
  - In a real world use-case, I cannot see the number of allowed binaries
reaching two digits per a single option, meaning that explicitly adding
one or two more paths to the list is not costly at all;
  - Christophe mentions "trusting sysadmins", and it has absolutely
nothing to do with my point;
  - It opens another holes in the security, as shown by the need of adding
the following warning: "SECURITY WARNING: If you use regular expressions,
be mindful that an attacker could craft an annotation that uses .. to
escape the paths you gave."
    - It just feels the same as adding a security warning on the
annotations usage itself;

* Do not add ways to allow a security issue to exist or do not add ways to
disable a behaviour added by a security fix;
  - It will make one's life harder by configuring a few more paths? It'll,
but I fail to see this as an excuse for leaving a security hole open;

I would really like to hear opinions about the suggestions given on how to
address the issue, to learn about points I probably missed, and to have an
agreement on the solution to be implement at the first place.

Does this make sense to everyone involved?
```

Peng Tao (bergwolf) wrote on 2020-05-16: #23

```
First of all, IMO the idea of exposing config-via-annotation functionality
to end users is dangerous itself.

Other than the binary paths, there are a lot of other config options that
also need to be filtered if they come from a random end user. For example,
the default vcpu and memory size, the ability to preallocate guest memory,
the ability to disable swap for guest memory, they can also cause serious
problems for the infrastructure. And there are less-dangerous options like
whether 9pfs or virtiofs is used, or which networking model is used, or if
any experimental features are enabled, -- they can also cause unexpected
chaos for the system.

So the white-list solution proposed by Christophe cannot be a complete
solution until every option is properly reviewed and white-listed. And it
throws another question, how do you white-list a boolean option?

So instead of jumping to a solution, I'd like to ask get a consensus on
what want to provide first.

1. We *do not* want end users to customize kata containers via
annotations.
2. We *want* to allow infrastructure to customize kata containers via
annotations.

Can we agree on the above two points?
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-16: #24

```
My opinion on the issues raised by Fiencio and Peng Tao fits in one
sentence: when you have a big hole in your ship, you first plug it, and
only then discuss how to repair the rest of the ship.

This bug is extremely serious. It *presently* allows any remote user to
reboot any host in a cluster, install malware, erase its filesystem, or
worse.

So now is NOT the time to discuss the sex of angels, certainly not on a
private bug. I will simply not engage in such discussion at the moment. We
should do that after plugging the hole, and we should do that in public.

The question right now is whether my patch plugs the hole or not in
systems that either install the default or existing configuration files.
That's the only relevant question. If it does, then just get the thing in,
warn existing users via a CVE, and get production systems patched in the
wild as quickly as possible.

Then, AFTER THIS IS DONE, we can discuss later, on the mailing list, if we
want to remove regexps, if there is any serious security risk for other
options.

That being said, I will quickly address Peng Tao's and Fidencio's
objections.

- Peng Tao: I disagree with your point 1. The reason I discovered the hole
is precisely because I was about to construct a similar mechanism, only to
be told it existed already. We can debate why this is necessary on the
mailing list.

- Fidencio: We can also debate on regexp. It is a very secondary part of
the patch, specifically the test in the "regexpContains" function. So easy
to change later if there is a consensus. Right now, I'd argue that it
absolutely does not matter, because the default is an empty list, so
whether you use the default config or some existing config, you don't
allow these annotations. Annotations being off by default, the hole is
plugged unless a sysadmin ignores the warning and explicitly adds a
somewhat hard to discover option. In short, I made the hosts safe by
default, and right now, this is what matters.
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-16: #25

```
Here is what I believe are objective criteria for a fix:
```

1. Must address the issue immediately
2. Must not remove or break existing functionality
3. If there is a conflict between 1 and 2, provide an option to restore functionality.
4. Be easy to use, well documented, idiot-proof, good looking and planet-friendly.

Point 1 is the reason for my comments about this being an emergency, and rejecting the characterisation of "jumping to a solution".

Point 2 is the reason why I disagree with Tao. The "fix" to a problem like this is not to unilaterally and without public discussion remove a feature that actual production systems may rely on. Irrespective of whether this feature was reasonable to start with or not (we may debate on this later), the undisputed fact is that this feature is there right now, so a fix that would just disable all annotations is actually a risk more than a solution (notably, the risk that folks will not upgrade because we broke their use case).

Point 3 is the reason why I disagree with Fabiano. Until someone can prove to me that there is no QE system somewhere dropping nightly builds of qemu in a known location and testing them through an annotation, I'm reluctant to not being able to offer some wildcarding mechanism. This is the reason I immediately agreed with Tao's initial wildcarding suggestion, not some dark wish to over-engineer things ;-) Like for point 2, failing to provide a fall-back mechanism is a risk, because it may prevent some people from accepting the fix.

I fully appreciate and understand the points made by both Tao and Fabianno, and I want to have this discussion. However, IMHO, that discussion must happen in public and only once the hole has been plugged. For now, I'd rather have eyes on the proposed fix, not some bike shedding.

Therefore, I submit to the court that the only relevant question right now is whether my fix makes systems secure again *by default*. We can debate later how to make it easier to configure or how to protect lower-risk annotations, or even debate about the whole annotation thing, RBAC, etc.

Also, nobody answered my question about CVEs ;-)

fidencio (fidencio) wrote on 2020-05-16:	#26

Christophe.

We're discussing the solution and we're doing it with the right audience.

I understand we disagree, that's fine, but the decision should not be taken between what you, Tao, and I think. There's a reason a committee is present to deal with such issues.

Also, it's a Saturday night. You raised your question one day ago.

Please, give people time read the whole thread, to digest your questions, and then they'll give their answers.

Christophe de Dinechin (i-christophe) wrote on 2020-05-16:	#27

> We're discussing the solution

Well, part of my concern is that I feel we are not discussing the solution I proposed as much as spending a lot of time imagining alternatives. That's what I referred to as bike shedding.

Should we spend time on

a) fact: does the patch prevents remote execution by default?

or

b) opinion: can we trust sysadmins with regexps?

c) opinion: should we allow annotations?

I say please focus on (a) for now. Discussions (b) and (c) are important, but they could happen in public once (a) is done and some patch is widely available. I don't mind at all if we patch over my patch to do all kinds of other things. It's just not urgent.

> Also, it's a Saturday night. You raised your question one day ago.

I assume you are talking about the CVE question? I remember asking it on Tuesday, which is how I learned about the process to follow. I asked again one day ago, because I want to be super-extra-duper sure that I'm not the one supposed to write the KCSA described in https://github.com/kata-containers/community/blob/master/VMT/VMT.md. According to the doc, the VMT is supposed to do it, and I'm not a VMT member. But maybe everybody assumes that since I reported it, I'm the one who would at least draft the KCSA?

Anyway, Kata is about "security". I'm not saying it, the project is claiming that. Top title on the web page: "the security of VMs". So you'd expect that a security hole that opens a root-access backdoor for remote execution would be a kind of "red alert, all hands on deck" situation where "Saturday" would not count as a valid counter argument ;-)

The project's credibility on the security front depends on how it reacts to such problems.

Anyway, it's Sunday for me now ;-)

Peng Tao (bergwolf) wrote on 2020-05-17:	#28

Christophe, for a quick fix, a single enable/disable flag is much easier to implement and maintain than a white-list. The latter one is more complex than you think. All the options I listed in my previous reply are not supposed to be altered by a random user. My fear is that you might end up with filtering just the binary paths. I would like to ask what is the long term plan going your direction? I would prefer a simple boolean flag (which is also a quick fix) if there is no long term plan on how the other options are handled.

To be specific, I was asking:
1. will there be a classification of user-modifiable options and non-user-modifiable options?
2. will sysadmin be able to modify the classification?

If I understand your messages correctly, the answers are two YES. Then I agree with your solution. But please confirm if we are on the same page.

And w.r.t. filing new CVEs, anyone can file a CVE. VMT will be responsible for triaging the security issue and coordinate the progressive disclosure of a vulnerability. We've handled several CVEs before and they all come from outside reporters.

fidencio (fidencio) wrote on 2020-05-18: #29

Christophe,

I'm sorry you feel an easy way for distros to start using your changes right away is discussing the sex of angels. It's not, having the options being set during build time would just ease the adoption of the fix by the distros.

I'd like to ask you, please, be humble enough to understand people who more experience in the project may take their time to review your comments, to make suggestions, and that the suggestions my be different than what you proposed (or not).

If it happens to be the case, please, be humble enough to listen to the ideas, try to learn from them, instead of forcing yours.

Anyways, it's been made clear my suggestions are not valid nor welcome here.

fidencio (fidencio) wrote on 2020-05-18: #30

Sorry for the typos, not enough coffee in the morning yet.

* people who have more experience ...
* suggestions may be different ...

Peng Tao (bergwolf) on 2020-05-18

description: updated
description: updated
Changed in katacontainers.io:
importance: Undecided → Critical

Peng Tao (bergwolf) wrote on 2020-05-18: #31

draft-KCSA (1.3 KiB, text/plain)

I'm attaching a KCSA draft here so that once we have a CVE number and a fix in-place, we can fill in the form and send it out.

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #32

Download full text (5.5 KiB)
Peng Tao,

> a single enable/disable flag is much easier to implement and maintain than a white-list.

1) It's not really simpler. It's a flag check instead of a function call. The three-state variant you suggested in #16 is clearly more complicated, which is when I noticed that regexps were addressing your concerns while keeping the patch simple.

2) It's not a long term solution, but a "quick fix" that does not address the criteria I outlined in #25. This implies that we would then need to replace it with something else. Do we want to offer an option to user, then walk it back and offer another one later? Why would you prefer that?

3) It's also not simpler in the sense that there is no patch for it. Again, I really invite you to focus on the existing patch, and check if it solves the problem and if it can be amended later (i.e. if it would not require a walk-back).

> The latter one is more complex than you think.

Why do you think so? What is missing in my patch? Have you reviewed my patch?

> All the options I listed in my previous reply are not
> supposed to be altered by a random user.

But they can be altered by random users today.

This is another issue with the simple flag: it only gives back functionality to those who need it by re-exposing the bug entirely. This is what I called a "risk" in my response to you in comment #25.

> My fear is that you might end up with filtering just the binary paths.

Have you reviewed my patch?

It does filter only paths, because right now this is where the bug is. You cannot remote execute with other annotations. The dangerous bug today is remote execution.

I am not saying there is no risk on other annotations. However, I have reviewed them, and so far, my evaluation of the risk for all other annotations is that it is negligible *relative to* the ability to execute arbitrary code.

Specifically:

- Annotations that configure the guest kernel, including boot options, kernel image, guest hook paths, etc, are normally contained by the VM layer. It may make administrative sense to provide similar filtering for them, but it's not a security issue for the host unless there is a flaw in the hypervisor.

- Annotations that configure the guest, e.g. memory size, do require scrutiny. For example, memory could be use for denial-of-service attacks. We need to have a discussion on the mailing list on how to fix it. While there is a security risk, it is somewhat second-order relative to the ability to copy all data from the host to some remote machine or wipe its content clean.

> I would like to ask what is the long term plan going your direction?

I will split your question in three:

a) Is there a long term plan for the path filtering options in the patch I submitted: I tried to design them to address your earlier comment #5 and #8. I explained why a *global* option does not work, but I modified the *per-path* option to address your concern about making "free-for-all" possible (although the regexp approach has the additional benefit that you can restrict the free-for all, e.g. allow only /usr/bin/qemu.* and not any path).

b) Is there a long term plan for other annotations: Yes, they need to be reviewed car...

Read more...

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #33

Fabiano,

> I'm sorry you feel an easy way for distros to start using your
> changes right away is discussing the sex of angels.

No, I was *complaining* that we were discussing the sex of angels.
In other words, instead of (a) in comment #27, we have so far only
discussed (b) and (c) which are IMO secondary.

> It's not, having the options being set during build time would
> just ease the adoption of the fix by the distros.

My patch allows that, unless I am mistaken.

> I'd like to ask you, please, be humble enough to understand
> people who more experience in the project may take their time
> to review your comments, to make suggestions, and that the
> suggestions my be different than what you proposed (or not).

My humility or lack therefore is even more off-topic than whether
sysadmins with root access can be trusted with regexps, and veers even
more into "personal opinion" territory.

> If it happens to be the case, please, be humble enough to listen
> to the ideas, try to learn from them, instead of forcing yours.

The regexp idea that you don't like was in response to Peng Tao's
comments. While I did not immediately implement his suggestion as is (more
specifically, I tried, it became very complicated, so I looked for another
way), I recognized that he had pointed out a real issue, namely that my
initial proposal would disable a feature in use cases which I consider
legitimate. In other words, I learned from Peng Tao, and that's why the
second iteration of the patch uses regexps and not string matching.

> Anyways, it's been made clear my suggestions are not valid nor
> welcome here.

If this is how you feel, then it's a wild misinterpretation of my response
to your comments. You don't like regexps, fine. I share your concerns
regarding the risk, so if there was no reason for them, I would not have
added them.

I've explained the reason in comment #25, right after you brought the
issue up in #24. Let me repeat here the core objection I have to removing
regexps:

Until someone can prove to me that there is no QE system somewhere
dropping nightly builds of qemu in a known location and testing them
through an annotation, I'm reluctant to not being able to offer some
wildcarding mechanism.

In other words, it's not that I don't hear you or don't learn from you.
It's that I see a problem with your proposal, which you have not
addressed. If you respond to my question, i.e. if you show me how a string
list can be used to address the use case above, or if you show that the
use case is not valid, then you will change my mind. Writing to my manager
will not.

fidencio (fidencio) wrote on 2020-05-18: #34

Christophe,

>> It's not, having the options being set during build time would
>> just ease the adoption of the fix by the distros.
>
> My patch allows that, unless I am mistaken.

You're completely mistaken. None of your patches touch the Makefile itself
and I fail to understand how you'd do that without adding the options
there.

>> If it happens to be the case, please, be humble enough to listen
>> to the ideas, try to learn from them, instead of forcing yours.
>
> The regexp idea that you don't like was in response to Peng Tao's
comments. While I did not
> immediately implement his suggestion as is (more specifically, I tried,
it became very
> complicated, so I looked for another way), I recognized that he had
pointed out a real issue,
> namely that my initial proposal would disable a feature in use cases
which I consider
> legitimate. In other words, I learned from Peng Tao, and that's why the
second iteration of the patch uses regexps and not string matching.

I'm not forcing my idea to not using reg-exp. I've just tried to mention
that leaving the chance for the admin to insert a security issue by
mistake may not be the best way to solve a security issue.

Yet, I'd like to hear the opinion of other developers.

>> Anyways, it's been made clear my suggestions are not valid nor
>> welcome here.

> If this is how you feel, then it's a wild misinterpretation of my
response to your comments.

Oh, sorry for my misinterpretation then, sincerely.

I really got it wrong that you completely ignored the comment about having
it configured during build time.

> You don't like regexps, fine. I share your concerns regarding the risk,
so if there was no reason for them, I would not have added them.

>
> I've explained the reason in comment #25, right after you brought the issue up in #24.
> Let me repeat here the core objection I have to removing regexps:
>
> Until someone can prove to me that there is no QE system somewhere dropping nightly builds of
> qemu in a known location and testing them through an annotation, I'm reluctant to not being
> able to offer some wildcarding mechanism.

And here's the part that I'd like to hear from other developers about the weight of having regex which may lead to a security issue, as stated in the comment in the patch ... and having an admin to tweak the code which will be used by the QE, before sending the code to the QE.

> In other words, it's not that I don't hear you or don't learn from you. It's that I see a
> problem with your proposal, which you have not addressed. If you respond to my question, i.e.
> if you show me how a string list can be used to address the use case above, or if you show that
> the use case is not valid, then you will change my mind. Writing to my manager will not.

I also see a problem with your proposal, commented on that, and learned that I misinterpreted what you said, even when the issue is still there and you were not able to check that.

Dr. David Alan Gilbert (dgilbert-h) wrote on 2020-05-18: #35

The regexp's seem reasonable to me - although it does end up a lot more complex than it was before.
I think you'd have to have everything initially blacklisted and then allow the admin to add the entries for things they wanted to allow the user to change; which is going to be a bit painful for them.

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #37

> although it does end up a lot more complex than it was before.

The second patch is larger than the first one because
1. It adds all the documentation, modification to config files, etc.
2. It does all the dangerous annotations, not just hypervisor.path.

The actual difference, everything else being equal, is shown in comment #13. I don't see that as "a lot more complex".

Peng Tao (bergwolf) wrote on 2020-05-18: #39

Download full text (7.3 KiB)
On Mon, May 18, 2020 at 3:50 PM Christophe de Dinechin
<email address hidden> wrote:
[...]
We do not have to take it back if the initial option is
`config_via_annotation = disabled`. We first accept `enabled` and
`disabled`. Later extend it with an config option whitelist, which
lists exactly which option can be changed via annotations.

[...]

[...]

> Specifically:
>
> - Annotations that configure the guest kernel, including boot options,
> kernel image, guest hook paths, etc, are normally contained by the VM
> layer. It may make administrative sense to provide similar filtering for
> them, but it's not a security issue for the host unless there is a flaw
> in the hypervisor.
>
> - Annotations that configure the guest, e.g. memory size, do require
> scrutiny. For example, memory could be use for denial-of-service
> attacks. We need to have a discussion on th...

Read more...

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #38

Peng Tao, I reviewed the KCSA, it looks good to me. Thanks!

Minor suggestions:
- in the title, I would add "arbitrary code *on the host*" like in the description
- You put v1.9.0 and above. I did not check, but I assume this is when annotations were added?
- Red Hat instead of RedHat
- would actually reboot -> will reboot ?

Congratulations for spelling my name right!

Peng Tao (bergwolf) wrote on 2020-05-18: #40

On Mon, May 18, 2020 at 5:50 PM Dr. David Alan Gilbert
<email address hidden> wrote:
>
> The regexp's seem reasonable to me - although it does end up a lot more complex than it was before.
> I think you'd have to have everything initially blacklisted and then allow the admin to add the entries for things they wanted to allow the user to change; which is going to be a bit painful for them.
>
Yes, indeed. It is going to be a lot painful for sysadmins. But I like the idea of blacklisting everything first,

Cheers,
Tao
--
Into Sth. Rich & Strange

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #41

```
> You're completely mistaken. None of your patches touch
> the Makefile itself and I fail to understand how you'd
> do that without adding the options there.

You simply add the path_list you want in the
configuration-qemu.toml for your distro:

[hypervisor]
path_list = ["/usr/bin/qemu-kvm", "/usr/bin/qemu-kata" ]

Why would that not be a distro build time solution to patch the
configuration?

Hmmm… I _guess_ what you mean with your Makefile remark is that I should
add an @ variable in the configuration.in files that you could pass at
build time like for other paths? If that is what you mean, that makes
perfect sense. Will do that.

> Oh, sorry for my misinterpretation then, sincerely.

Based on the sentence below, I think this may have been sarcastic.

> I really got it wrong that you completely ignored the comment
> about having it configured during build time.

I replied "My patch allows that, unless I am mistaken" So I did not
"completely ignore" the comment, stating that my patch was doing what you
were asking, and leaving open the possibility that I had misinterpreted
with "unless I'm mistaken".

So yes, I'm afraid you really got it wrong, but by trying really hard to
guess what you meant, I believe you mean I should add @ variables in the
templates to make it easier for a distro to add some default value as
"make VAR=value". Correct?
> And here's the part that I'd like to hear from other
> developers about the weight of having regex which may
> lead to a security issue, as stated in the comment in
> the patch ... and having an admin to tweak the code which
> will be used by the QE, before sending the code to the QE.

First, I only gave this QE example as an proof by existence that such
scenarios exist. I have little doubt that this is not the only one.

Second, you seem to argue on one hand that sysadmins are too incompetent
to use regexps correctly, but at the same time, competent enough to patch
go code.

Third, as I've argued earlier, we can probably fine-tune the configuration
later, as long as the default is reasonably safe, which I believe it is.
My concern here is that by getting lost into second-order details, we
delay fixing the first-order problem. It is not that second-order details
should not be addressed, only that they should be addressed on the mailing
list once it's reasonable to do so because the base flaw is fixed.

Let me put it another way. If there are any users who actually need
/build/qemu[a-z0-9-]* as a pattern for their in-production system, they
are not watching this bug, but they are watching kata-dev. So we can start
with the regexp approach, then ask "we feel this feature is risky, here is
why. Does anybody need it. If not, we remove it in one week". Then if
nobody needs it, we change one line, and the filter only accepts string
lists and not regexp lists.

Similarly, this is why for the moment, I only address in my patch the
super-dangerous things like hypervisor.path, and not for example memory
allocation. The more general discussion about how we want the memory
allocation to be constrained can safely happen on kata-dev once we patched
the major flaw that any such discussion would inevitably expose.
```

fidencio (fidencio) wrote on 2020-05-18: #42

```
> Hmmm… I _guess_ what you mean with your Makefile remark is that I should
add an @ variable in the configuration.in files that you could pass at
build time like for other paths? If that is what you mean, that makes
perfect sense. Will do that.

Cool, thanks!
I believe it's cleaner and easier for distros to adapt that if it's just a
make parameter (please, check @QEMUVIRTIOFSPATH@ for reference).

> Second, you seem to argue on one hand that sysadmins are too incompetent
to use regexps correctly, but at the same time, competent enough to patch
go code.

No, I don't. I consider myself incompetent to use regexps, and I think
that adding two or three paths manually to the configuration.toml may be a
good trade-off, when considering possible holes regexps could lead to (as
you pointed out)

My concern about regexps is that it's hard to move back once it's in
production.

Let's say we add the fix with the regexps now and release a version of
kata with the fix. Once it's done there's no way to move back to block
regexps without breaking compatibility with a previous release version.
However, if we don't take this approach, we can add it later without any
issue.

Btw, I'd love to see a release happening as soon as your patches get
merged, as it'd allow distros to update their packages in no time, helping
the end consumers.
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #43

Download full text (3.5 KiB)

```
> My concern about regexps is that it's hard to move back
> once it's in production.

> Let's say we add the fix with the regexps now and release
> a version of kata with the fix. Once it's done there's no
> way to move back to block regexps without breaking
> compatibility with a previous release version. However,
> if we don't take this approach, we can add it later without
> any issue.

The concern is legitimate. However, let's consider two worlds.
```

In world A, someone actually uses annotations in production today with unpredictable values, e.g. /build/qemu-$DATE for hypervisor.path.

In world B, people use annotations with a small set of values that can be predicted ahead of time, e.g. /usr/bin/qemu and /opt/kata/bin/qemu

In world C, nobody uses annotations

My worldview right now is we don't know if we are in world A, B or C. Maybe someone else does know, that would easily change my mind on what the correct fix is.

Now, we submit a fix. We have four approaches so far, that I will call P, P', F and D:
P : We whitelist annotations with a flag that whitelists everything
P': We whitelist annotations names
F : We whitelist annotations values with strings
D : We whitelist annotations values with regexps

I hope that I represented our respective positions correctly.

Now, my question is: what do we recommend to users once the fix is deployed. How do we write the release note?

P only works for world C. If we are in world A and B, your release note reads like: "If you are using annotations today, then set enable_annotations to true, but please be mindful that this will allow any user to remotely execute any pre-existing binary on your host." To me, it's a no-brainer that we would have to walk back this one pretty quickly if we are in world

P' is only marginally better. In world A and B, the release note now reads like: "If you are using annotations today, then set enable_annotations = ["hypervisor.path"], but please be mindful that this will allow any user to remotely execute any pre-existing binary on your host.". Again, it's a no-brainer that we will have to walk this one back quickly if we happen to be in world A or B.

F works for world B and C. However, if there is a chance we are in world A, then the release note must read: "If you are in case C, this feature has been disabled, sorry". It's not nearly as bad as cases P and P', but it's still a loss of functionality that I don't know how to evaluate.

D works for world A, B and C. The release note reads: "If you need annotations in case B, then whitelist the binaries with [hypervisor] path_list = ["/path/a", "/path/b"]. If you need annotations in case A, then whitelist the binaries with [hypervisor] path_list = ["/most-restrictive-path-regexp"] but please read the SECURITY WARNING section carefully as you may open your host to binary execution attack with an improperly chosen pattern."

Now, if after discussing with the list, we realize that we are in world B, then walking back the fix is not a problem. If, however, it would be difficult to walk it back, then it means that we are actually in world A, in which case walking it back would be a net loss in functionality.

I understand...

Read more...

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #44

Three worlds obviously, after I integrated Peng Tao's suggested approach for completeness...

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #45

Reading back, I realize that there is an

F': We whitelist annotation values with strings, but with an option for a string to represent a directory.

Initially, Fabiano suggested to me privately that we whitelist only directories. As I replied in a chat message, that is risky, because if someone whitelists /usr/bin/ in order to grant access to /usr/bin/qemu-kvm, then it also gives access to all other /usr/bin/ programs.

Maybe, however, we can whitelist names but a name that ends with / is interpreted as a directory. In my example, you could then whitelist ["/usr/bin/qemu-kvm", "/build/"]

I'm pretty sure you can reasonably argue that F' covers enough of world A that it would be good enough.

However, while it looks more restrictive, it forces me to whitelist entire directories, and this in itself looks like a risk, e.g. it would be so tempting to open /usr/bin/. So that would need a different SECURITY WARNING label, but you would still need one.

fidencio (fidencio) wrote on 2020-05-18: #46

Christophe,

I didn't suggest that. I was brainstorming checking whether we could trust only the directory and, obviously, that's not the case as I even mentioned a few minutes later in that thread.

That's what, sincerely, I was also trying to do when I raised the bullets I did in the comment https://bugs.launchpad.net/katacontainers.io/+bug/1878234/comments/22.

You found the issue, you have a few days ahead of us thinking how to solve it. :-)

In any case, your point is clear, I guess my point on https://bugs.launchpad.net/katacontainers.io/+bug/1878234/comments/22 is also clear, and I guess Peng Tao's comment is also clear.

For now, I'd like to hear the input from other folks as well.

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #47

Since it turns out all the whitelisting so far is on paths, maybe filepath.Glob() is more appropriate than regexp.

Julio Montes (devimc) wrote on 2020-05-18: #48

Folks, just to be clear (I lost the context after the regexp stuff), is
this a vulnerability bug? (yes/no)

If it is, then we should warn other projects like podman and cri-o since
something similar can be done with their annotations.

see https://github.com/containers/libpod/blob/bfcec3203ea517f408ad90d42
789fb6eb62e7d81/pkg/annotations/annotations.go#L65

```
// PrivilegedRuntime is the annotation for the privileged runtime path
PrivilegedRuntime = "io.kubernetes.cri-o.PrivilegedRuntime"
```

From my point of view, this is not a bug, by default no user has access to
the cluster, the admin must grant him/her access:

```
To start using your cluster, you need to run the following as a regular
user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #49

> Folks, just to be clear (I lost the context after
> the regexp stuff), is this a vulnerability bug? (yes/no)

Yes.

> If it is, then we should warn other projects like podman
> and cri-o since something similar can be done with their
> annotations.

But don't you need to have root access on the machine in these cases?

> From my point of view, this is not a bug, by default no user has
> access to the cluster, the admin must grant him/her access:

Understood, but that access is intended to run a workload in a container,
and this gives access to what is outside the containers. So you can reboot
the host, copy all the data to some other nodes, etc.

In other words, does cluster admin granting the access rights above
expects the user to be able to shutdown worker nodes at will from a remote
machine? What was given was a key to the cluster, not a ssh key to the
root user.

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #50

> and this gives access to what is outside the containers.

Sorry, phrasing unclear. I meant "this bug gives you access".

Julio Montes (devimc) wrote on 2020-05-18: #51

> But don't you need to have root access on the machine in these cases?

No, same as this bug, you only need access to the annotations

> Understood, but that access is intended to run a workload in a
container, and this gives access
> to what is outside the containers. So you can reboot the host, copy all
the data to some other
> nodes, etc.

incorrect, if you have the keys to run containers, you have the keys of
the kingdom (see bellow)

> In other words, does cluster admin granting the access rights above
expects the user to be able to shutdown worker nodes at will from a remote
machine?

yes, admin must grant access *only* to trustworthy users, otherwise the
system can be easily pwned (see bellow)

> What was given was a key to the cluster, not a ssh key to the root user.

Create me an user account (without admin priviledges, no sudo) in your
cluster and let me run the following container, let's see what I can do
after running it ;)

```
apiVersion: v1
kind: Pod
metadata:
  name: hack-pod
spec:
  containers:
  - image: busybox
    name: hack-container
    command: [ "sh", "-c", "echo -e 'pwned.123\npwned.123' | passwd root"
]
    tty: true
    stdin: true
    volumeMounts:
    - mountPath: /etc/
      name: hack-volume
  restartPolicy: OnFailure
  volumes:
  - name: hack-volume
    hostPath:
      path: /etc/
      type: Directory
```

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #52

Also, after discussion with other people, I realized that "sex of angels"
is probably totally obscure and may have been misinterpreted. Sorry.

It's a literal translation from a French expression. In French, "discuter du sexe des anges", literally meaning "talk about the sex of angels" means having a discussion about secondary topics when there is a more important topic to deal with. Something like "putting the cart before the horses" but for talk.

The origin of the expression is the Constantinople battle of May 29, 1453, where the Turks allegedly won because the Byzantine forces were too busy arguing the deep theological question of whether angels were male or female.

So now this bug became an opportunity for a totally pointless cultural exchange ;-)

Sorry about the confusion this expression may have caused. For some reason, I believed it was also common in English.

Julio Montes (devimc) wrote on 2020-05-18: #53

Apologies if my comment was not clear enough or misunderstood, going back to the bug,

this is a bug if and *only if* the user can set/add annotations, right?

What does a user need to set/add annotations?

Christophe de Dinechin (i-christophe) wrote on 2020-05-18: #54

>> What was given was a key to the cluster, not a ssh key to the root user.

> Create me an user account (without admin priviledges, no sudo)
> in your cluster and let me run the following container, let's
> see what I can do after running it ;)

Ouch.

So I guess that what you are saying is that not restricting access to annotations at all is consistent with the way the rest of the system works. If so, I agree this may not be a kata-runtime bug, although I must admit I find it very counter-intuitive. I am probably not the only one.

Going to sleep on this one.

Archana Shinde (amshinde) wrote on 2020-05-19: #55

Kubernetes defaults are not really secure by default__ One can pretty much bring the cluster down or gain elevated privileges by using hostpath as seen in the example posted by devimc.
The responsibility to secure the cluster lies with the cluster admin using admission controller or Pod Security Policies to limit what an user can do with the pod. Pod Security policies allow for a fine grained control where pods that use hostpath or hostpid or run as privileged are rejected.

We have taken a similar approach so far by allowing annotations with Kata and placing the responsibility of validating fields on the service provider by having proper admission controls in place.
That being said, I do see the value of kata annotations being used by the end-users and not just the service provider, such as being able to use different kernel or boot options for a particular workload.
So it makes sense to have validation for the annotations in Kata.
I am leaning towards Tao's suggestion, add an ability to completely disable/enable annotations and extend this with a whitelist, with us identifying annotations that are safe for a user to configure.
We would need to systematically go through the list of annotations and identify those that can be whitelisted or need additional validations. (We can potentially identify those that can be removed altogether in 2.0)
Specifically for the the exec binaries, instead of having regular expressions, we can have a list of system paths such as ["/usr/bin", "/usr/local/bin", "/opt/kata/bin"].
That way it is ok to use an executable as long as the base directory of the executable belongs to these paths. Would like to see what others think.

Christophe de Dinechin (i-christophe) wrote on 2020-05-19: #56

Julio,

Yes; the bug only exists if the user can set annotations.

I tried to think more about why my mental model broke down on your scenario. I tested it, and indeed, it seems to work. But I'm confused *why* it does work. My understanding was that the container root was in some namespace and therefore would have the access of a regular user to the host filesystem.

If I try to mount a filesystem using podman, here is what I get (which is what I expected):

```
# mkdir /tmp/somedir
# podman run -it -v /tmp/somedir:/somedir busybox sh
/ # ls /somedir/
ls: can't open '/somedir/': Permission denied
```

If I try to do the same experience suggested by your pod with podman, here is what I get:

```
# podman run -it -v /etc:/etc busybox sh
/ # passwd
passwd: unknown uid 0
```

This is what I expected. So why is it different with your example?

Christophe de Dinechin (i-christophe) wrote on 2020-05-19: #57

Archana,

> Specifically for the the exec binaries, instead of having
> regular expressions, we can have a list of system paths
> such as ["/usr/bin", "/usr/local/bin", "/opt/kata/bin"]

This option is mentioned in #45. What are the scenarios where you think it would help?

Christophe de Dinechin (i-christophe) wrote on 2020-05-19: #58

> The responsibility to secure the cluster lies with
> the cluster admin using admission controller or Pod
> Security Policies to limit what an user can do with
> the pod. Pod Security policies allow for a fine grained
> control where pods that use hostpath or hostpid or run
> as privileged are rejected.

Does that mean we should connect the annotation whitelists to pod security
policies instead of config file? Or do you see a benefit to having both a
config-based control of annotations as well as pod security policies?

Christophe de Dinechin (i-christophe) wrote on 2020-05-19: #59

Yesterday, I replaced regexps with globs as I initially intended (see
comment #11 and then #47). The function to check is a little bit more
complicated, but not that much:

```
+func checkPathIsInGlobList(list []string, path string) bool {
+ for _, glob := range list {
+ filenames, _ := filepath.Glob(glob)
+ for _, a := range filenames {
+ if path == a {
+ return true
+ }
+ }
+ }
+ return false
+}
+
```

I believe that globs are more natural to express families of paths, and
they don't suffer from the same risks for that use case as regexps, i.e.
if you have /opt/kata/bin/*, it won't accept /opt/kata/bin/../
../../whatever as input.

Experimenting, one thing that surprised me is that the Go filepath.Glob()
does not seem to have GLOB_BRACE, i.e. you can't have /foo/{abc,def}/bar.
That's not a big deal IMO.

So for paths, that would be my recommendation.

Christophe de Dinechin (i-christophe) wrote on 2020-05-19: #60

I thought if I left the + at beginning of lines, it would not destroy
indentation.

Is there a way to put code in a comment on this web site?

ERNST Eric (ericisonic) wrote on 2020-05-19: #61

Bonjour,

Euh, je ne sais pas pourquoi j'ai atteris dans ce sujet de discution.....

Le mar. 19 mai 2020 à 10:11, Christophe de Dinechin <email address hidden>
a écrit :

[...]

Christophe de Dinechin (i-christophe) wrote on 2020-05-19: #62

Series of patches to control annotations (110.0 KiB, application/x-tar)

Attached is a revised series of patches.

0012-config-Add-makefile-variables-for-path-lists.patch is my attempt at
addressing Fabiano's comments on distro build-time configuration. I am not
entirely sure this is the right way to do it for arrays.

0014-config-Use-glob-instead-of-regexp-to-match-paths-in-.patch is the
replacement of regexps with glob for paths. With that change, I believe
that the security concerns are no longer there, so I removed the SECURITY
WARNING.

0015-config-Whitelist-hypervisor-annotations-by-name.patch is my attempt
at implementing Peng Tao's whitelisting idea. I have only implemented it
for the hypervisor. Agent and runtime may follow, but I have several
questions about this one, including:

1. If an annotation is not whitelisted, should we fail to start, or just
ignore it? It's a bit harder to ignore it with the existing code
structure, so for now it fails.

2. Should each section have its own "enable_annotations" configuration? Or
does it make sense to add a global [anotations] section to configure all
the annotation-related aspects?

But then, the top-level question is whether this a bug. I still believe it
is better to not allow arbitrary execution so easily, so I still see that
as a bug, although following Julio's comments, it may well fit in the
"normal" security model.

Christophe de Dinechin (i-christophe) wrote on 2020-05-19: #63

Series of patches to control annotations (110.0 KiB, application/x-tar)

Attached is a revised series of patches.

0012-config-Add-makefile-variables-for-path-lists.patch is my attempt at
addressing Fabiano's comments on distro build-time configuration. I am not
entirely sure this is the right way to do it for arrays.

0014-config-Use-glob-instead-of-regexp-to-match-paths-in-.patch is the
replacement of regexps with glob for paths. With that change, I believe
that the security concerns are no longer there, so I removed the SECURITY
WARNING.

0015-config-Whitelist-hypervisor-annotations-by-name.patch is my attempt
at implementing Peng Tao's whitelisting idea. I have only implemented it

for the hypervisor. Agent and runtime may follow, but I have several questions about this one, including:

1. If an annotation is not whitelisted, should we fail to start, or just ignore it? It's a bit harder to ignore it with the existing code structure, so for now it fails.

2. Should each section have its own "enable_annotations" configuration? Or does it make sense to add a global [anotations] section to configure all the annotation-related aspects?

But then, the top-level question is whether this a bug. I still believe it is better to not allow arbitrary execution so easily, so I still see that as a bug, although following Julio's comments, it may well fit in the "normal" security model.

Archana Shinde (amshinde) wrote on 2020-05-19: #64

> If I try to mount a filesystem using podman, here is what I get (which is what I expected):

```
> # mkdir /tmp/somedir
> # podman run -it -v /tmp/somedir:/somedir busybox sh
> / # ls /somedir/
> ls: can't open '/somedir/': Permission denied
```

> If I try to do the same experience suggested by your pod with podman, here is what I get:

```
> # podman run -it -v /etc:/etc busybox sh
> / # passwd
> passwd: unknown uid 0
```

> This is what I expected. So why is it different with your example?

Christophe, looks like you are running a rootless container there with podman(I'll let you confirm), which explains the permissions issue, whereas containers launched with k8s are launched as root.

> Specifically for the the exec binaries, instead of having
> regular expressions, we can have a list of system paths
> such as ["/usr/bin", "/usr/local/bin", "/opt/kata/bin"]

I mentioned this as a way to get rid of regular expressions in the configuration. You may simply get the base dir of the path provided for the executable using golang path.Dir (https://golang.org/pkg/path/#Dir) and check if the directory is contained within the system paths configured(using maybe go strings.Prefix)

Christophe de Dinechin (i-christophe) wrote on 2020-05-20: #65

> Christophe, looks like you are running a rootless
> container there with podman(I'll let you confirm),
> which explains the permissions issue, whereas
> containers launched with k8s are launched as root.

I'm running podman as root, if that's what you are asking. However, even in that case, the container root user is not host uid 0, but uses namespaces to map that to some other user. This is the reason of the failure above.

Apparently, the behavior reported for k8s does not happen with OpenShift either. OpenShift, like podman, blocks this particular form of attack. I suspect it would be possible to achieve a similar result with a privileged container, however.

> I mentioned this as a way to get rid of regular
> expressions in the configuration.

In comment #62, I shared a new iteration of the patch series, which uses glob instead of regexp. I see no security issue with glob, since a pattern like /opt/kata/bin/../../../usr/bin/ls is not a valid match for glob pattern /opt/kata/bin/*. As mentioned in comment #11, this was my first choice, but for (very invalid) reasons I first shared an implementation using regexps.

Do you object to using glob patterns? They are pretty familiar to any Unix sysadmin IMO.

Christophe de Dinechin (i-christophe) wrote on 2020-10-15: #66

Known as CVE-2020-27151.

Karen Noel (knoel-9) wrote on 2020-10-15: #67

Cool. We don't ship Kata yet, but we have a component in Bugzilla. Will
the
Red Hat ProdSec team create a BZ to make sure it gets fixed downstream?

Karen

On Thu, Oct 15, 2020, 6:05 AM Christophe de Dinechin <
<email address hidden>> wrote:

[...]

Archana Shinde (amshinde) wrote on 2020-11-20: #68

Christophe, now that we have the fix pushed in a release, we should make the CVE public. I dont see it has been published yet.

Changed in katacontainers.io:
status:New → Confirmed
status:Confirmed → Fix Committed

Archana Shinde (amshinde) on 2020-12-03

Changed in katacontainers.io:
status:Fix Committed → Fix Released

Archana Shinde (amshinde) on 2020-12-03

information type:Private Security → Public Security

Christophe de Dinechin (i-christophe) wrote on 2020-12-09: #69

```
> Christophe, now that we have the fix pushed in a release, we should make
the CVE public. I dont see it has been published yet.

It's public now, see https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-
2020-27151.
```

See full activity log

To post a comment you must log in.