Security in QEMU
How Virtual Machines provide Isolation

Stefan Hajnoczi <stefanha@redhat.com>

KVM Forum 2018

About me

Reviewer of CVE fixes

Participant in vulnerability disclosure process

QEMU contributor since 2010

Member of Red Hat’s virtualization team

QEMU Security Process

Found a security bug or not sure if it’s a security bug?

https://wiki.qemu.org/SecurityProcess

Please follow this process so fixes can be rolled out with minimal risk to users.

Use Cases and their Security Requirements

Many QEMU use cases exist

They have different security requirements

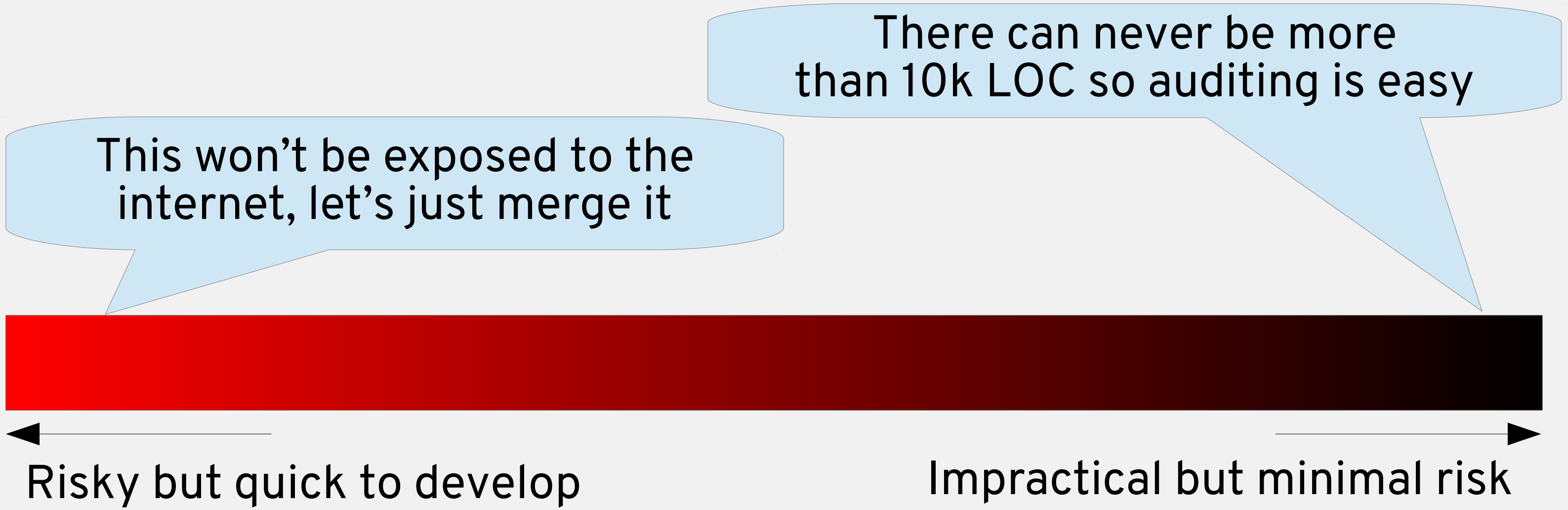
Toy kernel development

Guest is trusted
User is trusted
No internet access

Which security requirements does QEMU fulfill?

Are they a superset of my use case's requirements?

How do we agree on Security Requirements?


There can never be more than 10k LOC so auditing is easy
This won’t be exposed to the internet, let’s just merge it
Risky but quick to develop
Impractical but minimal risk

The community has a consensus that works for its participants

It evolves over time as people join or leave the project

QEMU’s Security Requirements

For **virtualization** use cases:

- Guest is untrusted
- User-facing interfaces are untrusted (e.g. remote desktop)
- Network protocols are untrusted
- User-supplied files are untrusted

Non-virtualization use cases are not backed by security claims

- TCG (just-in-time compiler) use cases rely on old unaudited code

(Check SecurityProcess wiki page for latest info if reading in future)

Fine-grained Security Support List needed?

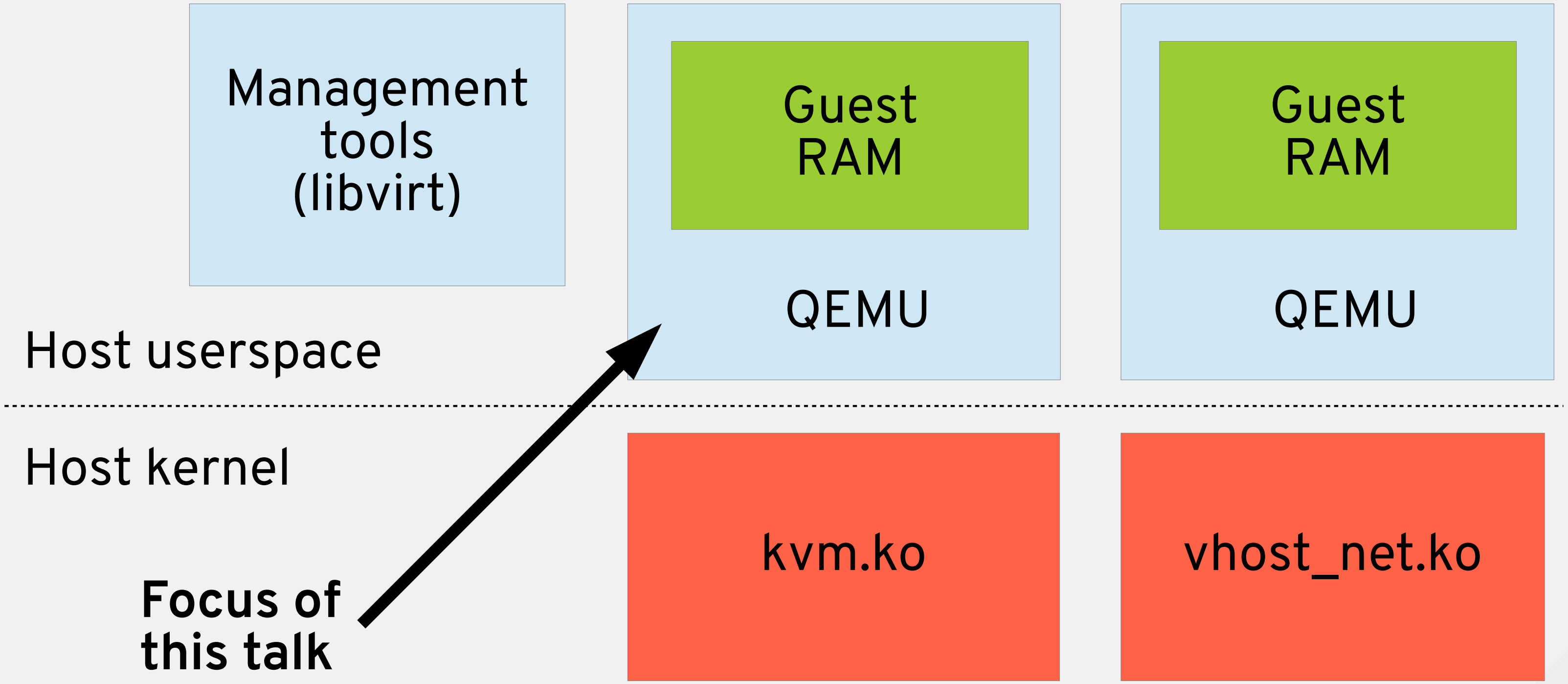
Not all QEMU features are hardened and production-quality

Downstreams support a subset of features

Safe features may not be apparent to upstream newcomers

Wish: Let’s create a fine-grained “safe features” list

Architecture (QEMU, KVM/TCG, libvirt)

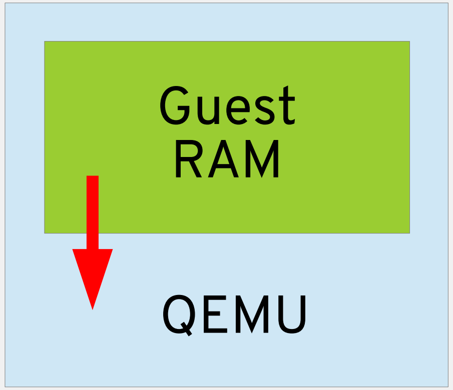


QEMU Guest Isolation (1)

1. The guest must not gain control of QEMU

Attack surfaces:

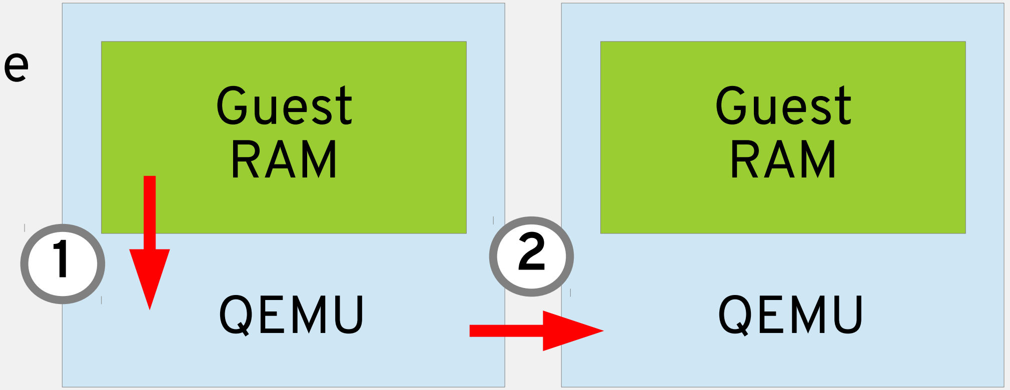
- Device emulation
- TCG (not covered in this presentation)



QEMU Guest Isolation (2)

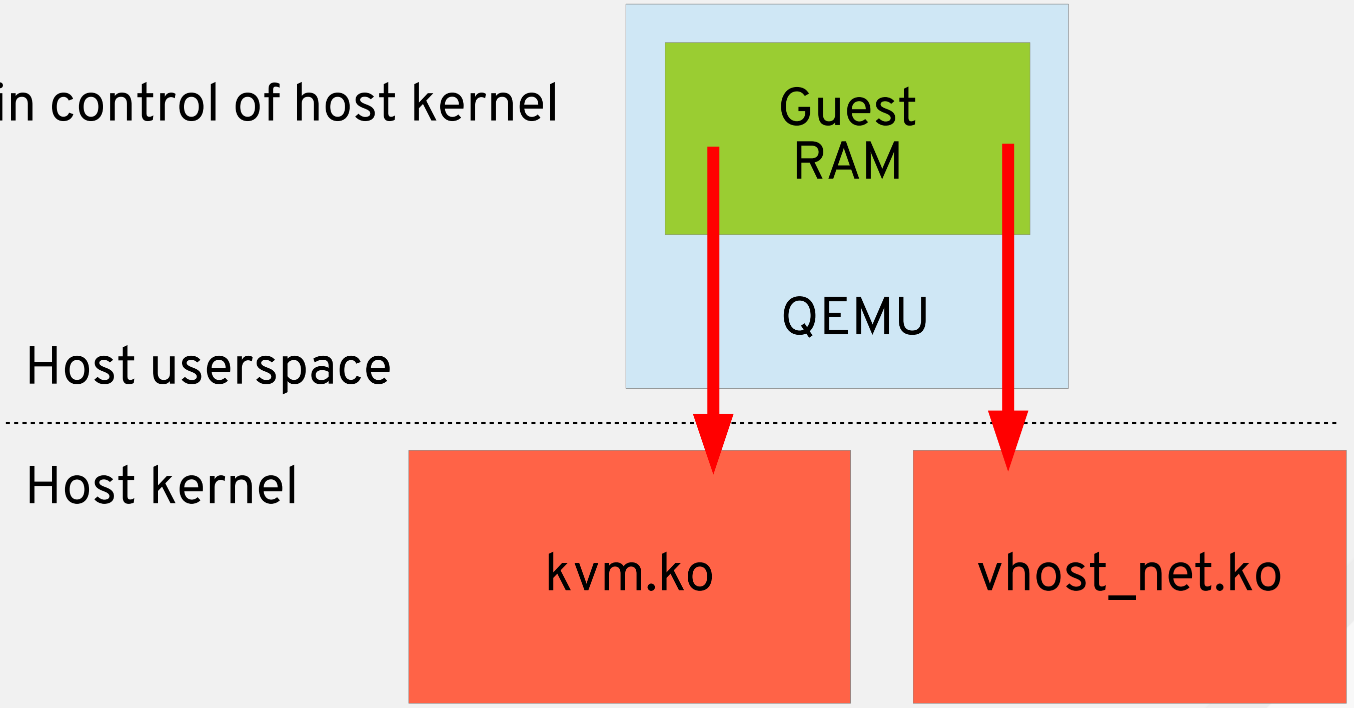
2. Must not gain access to other guests

Traditional attacks on other guests are possible over the network, but another vector exists if you gain control of QEMU



QEMU Guest Isolation (3)

3. Must not gain control of host kernel



Defense in Depth

The virtualization stack consists of layers

Compromising one layer must not compromise the entire system

Makes it more challenging for an attacker

Securing the QEMU Process

Management tools (e.g. libvirt) should:

- Run QEMU as an unprivileged user
- Restrict the QEMU process using SELinux to prevent access to other guests' disks or debugging them (ptrace)
- Configure resource controls on the QEMU process

Always check your management tool is doing this!

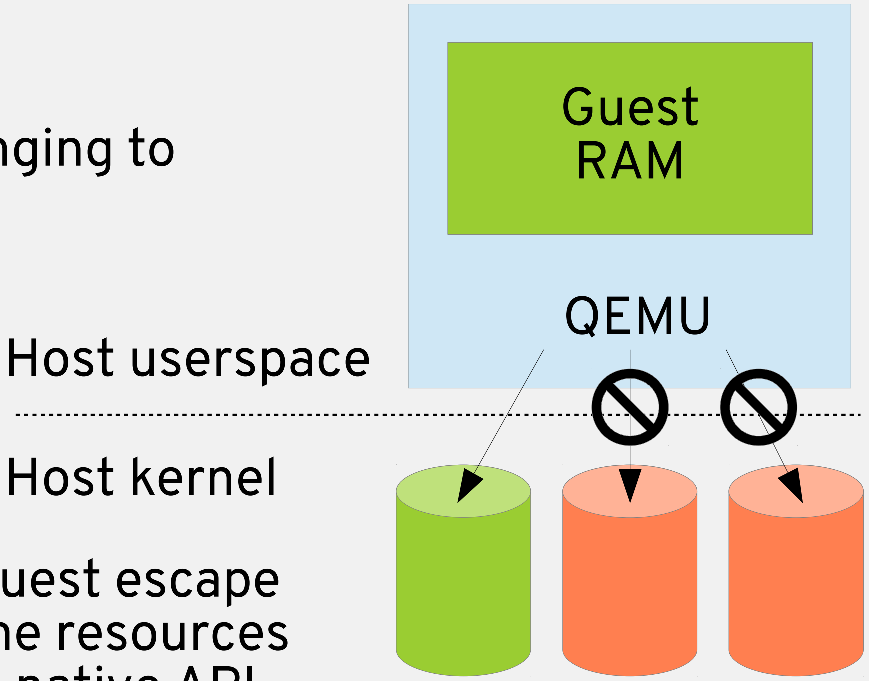
- If you run QEMU manually or with a custom tool, beware.

Theory: Principle of least Privilege

QEMU only has resources belonging to this specific guest

If guest escapes into QEMU it does not gain access to other resources!

When implemented perfectly, guest escape only provides access to the same resources as within the guest but with the native API



Practice: Principle of least Privilege

Escaping into QEMU exposes native APIs unavailable in the guest

SELinux and seccomp reduce the host userspace attack surface, but restricting everything is hard

- See Eduardo Otubo's QEMU Sandboxing for Dummies talk

Escaping into QEMU may give access to storage network

- Protect network disks with authentication (iSCSI, Ceph, etc)

Resource limits implemented by QEMU, like rate-limits, can be bypassed once QEMU is compromised

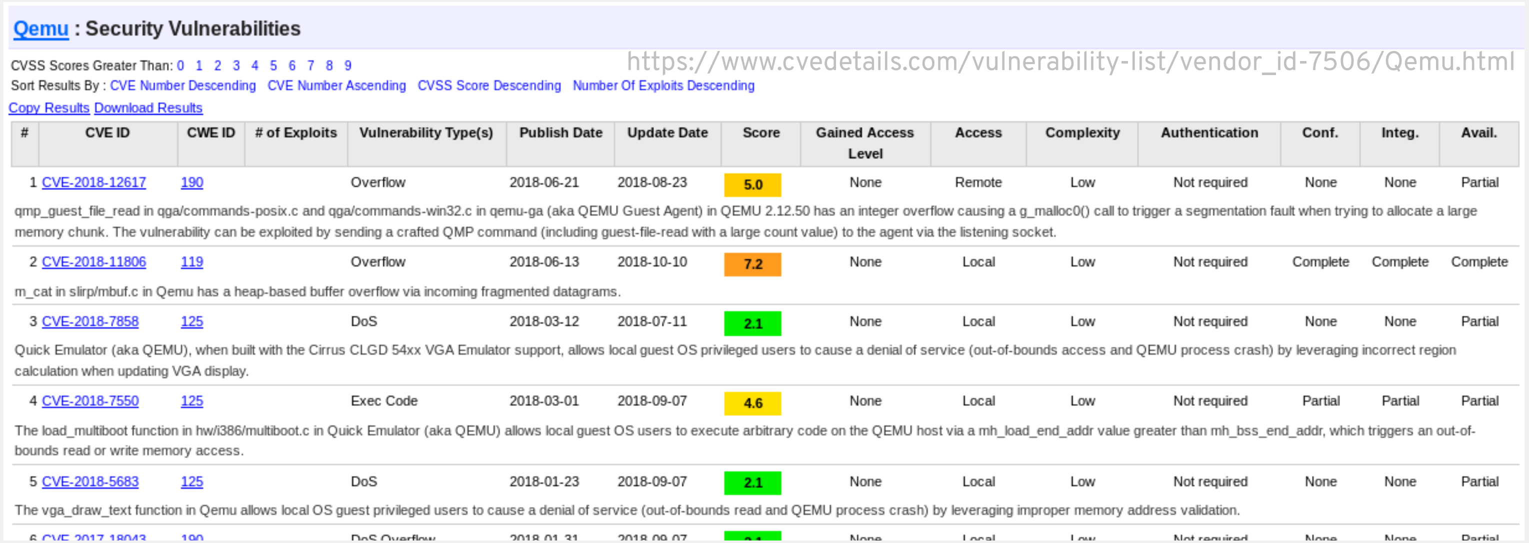
Recap: Principle of least Privilege

Design new features to only give QEMU access to resources belonging to the guest

Sometimes exceptions are necessary for practical reasons and this should be documented

Real-world QEMU Security Bugs

What do real bugs look like? How can they be prevented?



https://www.cvedetails.com/vulnerability-list/vendor_id-7506/Qemu.html

Qemu : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending
Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2018-12617	190		Overflow	2018-06-21	2018-08-23	5.0	None	Remote	Low	Not required	None	None	Partial
qmp_guest_file_read in qga/commands-posix.c and qga/commands-win32.c in qemu-ga (aka QEMU Guest Agent) in QEMU 2.12.50 has an integer overflow causing a g_malloc0() call to trigger a segmentation fault when trying to allocate a large memory chunk. The vulnerability can be exploited by sending a crafted QMP command (including guest-file-read with a large count value) to the agent via the listening socket.														
2	CVE-2018-11806	119		Overflow	2018-06-13	2018-10-10	7.2	None	Local	Low	Not required	Complete	Complete	Complete
m_cat in slirp/mbuf.c in Qemu has a heap-based buffer overflow via incoming fragmented datagrams.														
3	CVE-2018-7858	125		DoS	2018-03-12	2018-07-11	2.1	None	Local	Low	Not required	None	None	Partial
Quick Emulator (aka QEMU), when built with the Cirrus CLGD 54xx VGA Emulator support, allows local guest OS privileged users to cause a denial of service (out-of-bounds access and QEMU process crash) by leveraging incorrect region calculation when updating VGA display.														
4	CVE-2018-7550	125		Exec Code	2018-03-01	2018-09-07	4.6	None	Local	Low	Not required	Partial	Partial	Partial
The load_multiboot function in hw/i386/multiboot.c in Quick Emulator (aka QEMU) allows local guest OS users to execute arbitrary code on the QEMU host via a mh_load_end_addr value greater than mh_bss_end_addr, which triggers an out-of-bounds read or write memory access.														
5	CVE-2018-5683	125		DoS	2018-01-23	2018-09-07	2.1	None	Local	Low	Not required	None	None	Partial
The vga_draw_text function in Qemu allows local OS guest privileged users to cause a denial of service (out-of-bounds read and QEMU process crash) by leveraging improper memory address validation.														
6	CVE-2017-18043	190		DoS Overflow	2018-01-31	2018-09-07	2.1	None	Local	Low	Not required	None	None	Partial

CVE-2015-3456 – VENOM

Guest-triggerable buffer overflow in floppy disk controller code:

```
$ git show e9077462
@@ -2004,7 +2007,9 @@ static void fdctrl_write_data(...)
     FLOPPY_DPRINTF("%s: %02x\n", __func__, value);
-    fdctrl->fifo[fdctrl->data_pos++] = value;
+    pos = fdctrl->data_pos++;
+    pos %= FD_SECTOR_LEN;
+    fdctrl->fifo[pos] = value;
```

Device Emulation Security Checklist

1. C programming bugs (buffer overflows, use-after-free, etc)
2. Validate inputs from guest
3. Handle device accesses at unexpected moments or in an unusual order (e.g. submitting another request while one is pending)
4. Validate migration state upon load
5. Copy in guest memory (other vcpus race with your thread)

Other Attack Surfaces

User-facing interfaces are untrusted (VNC, SPICE)

Network protocols (WebSocket, NBD, etc)

User-supplied files (kernel images, disk images)

CVE-2017-14167 – multiboot loader

Kernel loader for multiboot files forgot to validate inputs

Heap buffer overflow triggered by malicious multiboot file

Values from
untrusted file

```
$ git show ed4f86e8
    mbs.mb_buf = g_malloc(mb_kernel_size);
    fseek(f, mb_kernel_text_offset, SEEK_SET);
    if (fread(mbs.mb_buf, 1, mb_load_size, f) !=
```

The HMP/QMP Monitor

Provides administrative access to guest

Same abilities as QEMU process to access files on host, etc

Do not expose the monitor directly to untrusted users!*

* Monitor white-list has been discussed as future solution

What can we learn from the bugs?

Many are Denial of Service and not memory corruption (good!)

Actually gaining access to other guests or host kernel requires additional steps → defense in depth

C coding bugs (integer overflows, buffer overflows, etc) are common

How do we improve QEMU Security? (1)

Finding bugs early

- Static analysis tools - already in use today
- Fuzzing attack surfaces – limited activity upstream
- Code audits – we have code review but no formal audit activity

(Remember bugs found in code for non-virtualization use cases may not be treated as security bugs)

How do we improve QEMU Security? (2)

Mitigating impact of bugs

- Sandboxing – what's next after SELinux and seccomp?
- Multi-process QEMU – smaller processes can be sandboxed more effectively
- Modules – only load features as needed to reduce code available to exploits relying on return-oriented programming
- Compiling out features – for example `./configure --disable-tcg` is now possible!

How do we improve QEMU Security? (3)

Eliminating sources of bugs

- Using a “safe” programming language – Rust has been discussed
- Restricting ourselves to safe APIs – bounds-checked FIFO instead of open-coded C array

Get Involved

A lot of activity underway to improve security

Participate in approaches that interest you

Discuss on the QEMU mailing list <qemu-devel@nongnu.org>

Thank you!

My blog: https://blog.vmsplice.net/

IRC: stefanha on #qemu irc.oftc.net

CVE-2016-9602 – virtfs root directory escape

O_NOFOLLOW still follows symlinks in dirname

Malicious guests can provide a path with a symlink

```
@@ -359,13 +378,9 @@ static int local_closedir(…
 static int local_open(FsContext *ctx,…
 {
-    fd = open(buffer, flags | O_NOFOLLOW);
+    fd = local_open_nofollow(ctx, fs_path->data, …
```