Bringing the Power of eBPF to QEMU

Chen Zhang <chen.zhang@intel.com>

Intel Virtualization Enabling Team

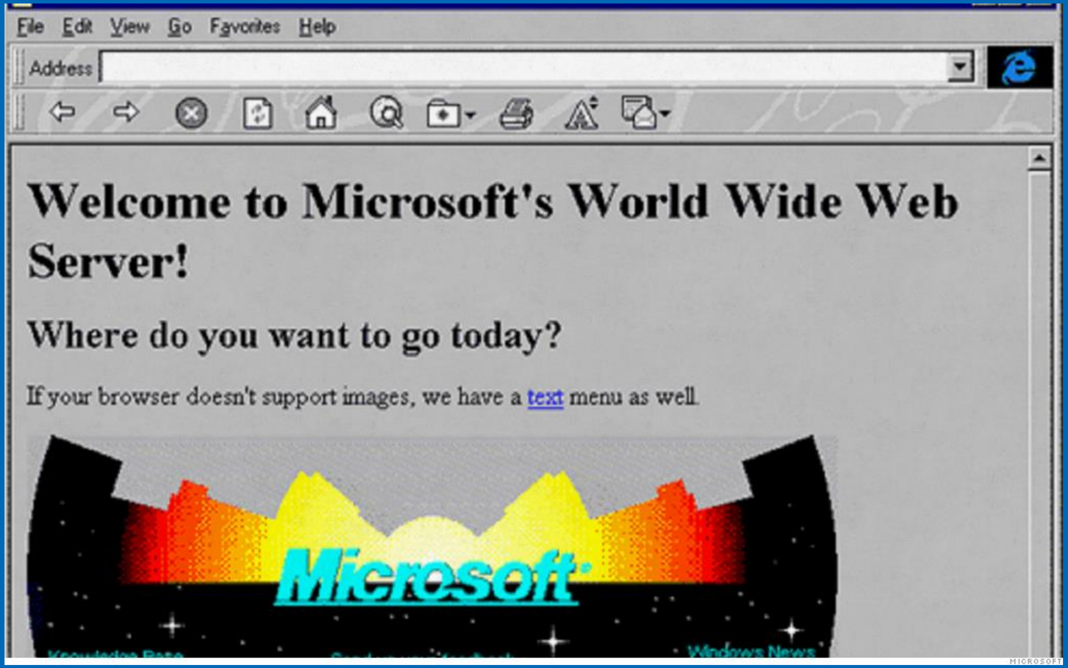
Agenda

- Generic eBPF Technology
- Userspace eBPF Technology
- QEMU userspace eBPF

What is eBPF?

eBPF (which is no longer an acronym for anything) is a revolutionary technology with origins in the Linux kernel that can run sandboxed programs in a privileged context such as the operating system kernel. It is used to safely and efficiently extend the capabilities of the kernel without requiring to change kernel source code or load kernel modules.

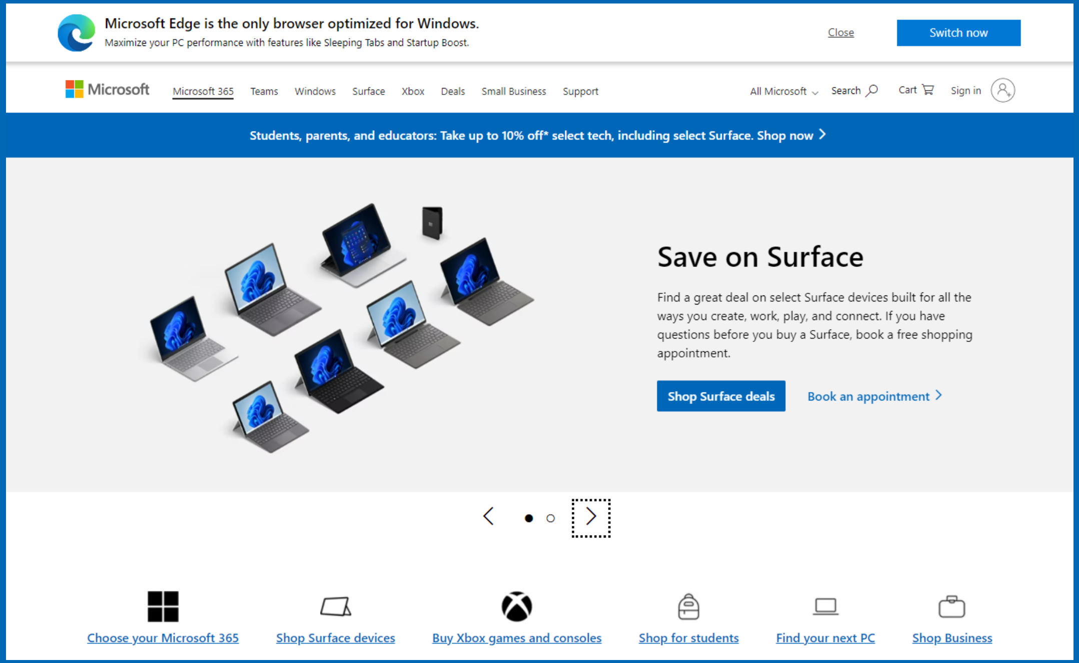
eBPF is to the Kernel/QEMU what JavaScript is to the browser.



Welcome to Microsoft's World Wide Web Server!

Where do you want to go today?

If your browser doesn't support images, we have a text menu as well.



Microsoft Edge is the only browser optimized for Windows.

Maximize your PC performance with features like Sleeping Tabs and Startup Boost.

Students, parents, and educators: Take up to 10% off* select tech, including select Surface. Shop now >

Save on Surface

Find a great deal on select Surface devices built for all the ways you create, work, play, and connect. If you have questions before you buy a Surface, book a free shopping appointment.

Not against Microsoft... :)

Kernel eBPF Based Project

Katran
High-performance L4 Loadbalancer

facebookincubator/katran

Cilium
Networking, security and load-balancing for k8s

cilium/cilium

NETFLIX et al.

bcc, bpftrace
Performance troubleshooting & profiling

iovisor/bcc

Android & Security
kernel runtime security instrumentation (KRSI), Android BPF loader, eBPF traffic monitor

CLOUDFLARE

Traffic Optimization
DDoS mitigation, QoS, traffic optimization, load balancer

cloudflare/bpftools

Falco
Container runtime security, behavior analysis

falcosecurity/falco

Agenda

- Generic eBPF Technology
- Userspace eBPF Technology
- QEMU userspace eBPF

Why is having an user space version of eBPF?

Programmability Essentials

- eBPF is famous for permits user-defined instrumentation on a functioning kernel context.
- The real reason is kernel eBPF provide programmability, extensibility, and agility. Rather than a way to cross user space/kernel space.

Userspace eBPF Based Project

- Oko
  - An extension of Open vSwitch-DPDK that provides runtime extension with BPF programs.
  - Contributed by Orange. URL: https://github.com/Orange-OpenSource/Oko
- Solana
  - An open source project implementing a new, high-performance, permissionless blockchain.
  - Contributed by The Solana Foundation. URL: https://github.com/solana-labs/solana
- DPDK eBPF support
  - A set of libraries and drivers for fast packet processing.
  - Contributed by Intel. URL: https://doc.dpdk.org/guides/prog_guide/bpf_lib.html
- eBPF for Windows
  - A work-in-progress project that allows existing eBPF toolchains and APIs familiar in the Linux ecosystem to be used on top of Windows.
  - Contributed by Microsoft. URL: https://github.com/microsoft/ebpf-for-windows

Programmability Essentials

- Safety
  - eBPF sandboxing.
- Continuous Delivery
  - Deploy anytime with seamless upgrades.
- Performance
  - Native Execution (JIT compiler).

Agenda

- Generic eBPF Technology
- Userspace eBPF Technology
- QEMU userspace eBPF

QEMU Development

- Option 1: Upstream Support
  - Change QEMU source code
  - Submit the patch to community
  - Wait years of time for your users to upgrade
- Cons:
  - No one can spend so much time waiting.

- Option 2: Downstream Support
  - Change QEMU source code
  - Maintain internal code base.
  - Rebase new upstream patches.
- Cons:
  - Need take efforts maintain internal version.

intel

Userspace eBPF for QEMU

- Flexibility
  - Ability to extend certain types of functionality without limited to exposed API.
- Agility
  - Admin can generate custom solutions without the delay, even no need reboot the VM.
- Performance
  - Native Execution (x86-64 and Arm64 JIT compiler).

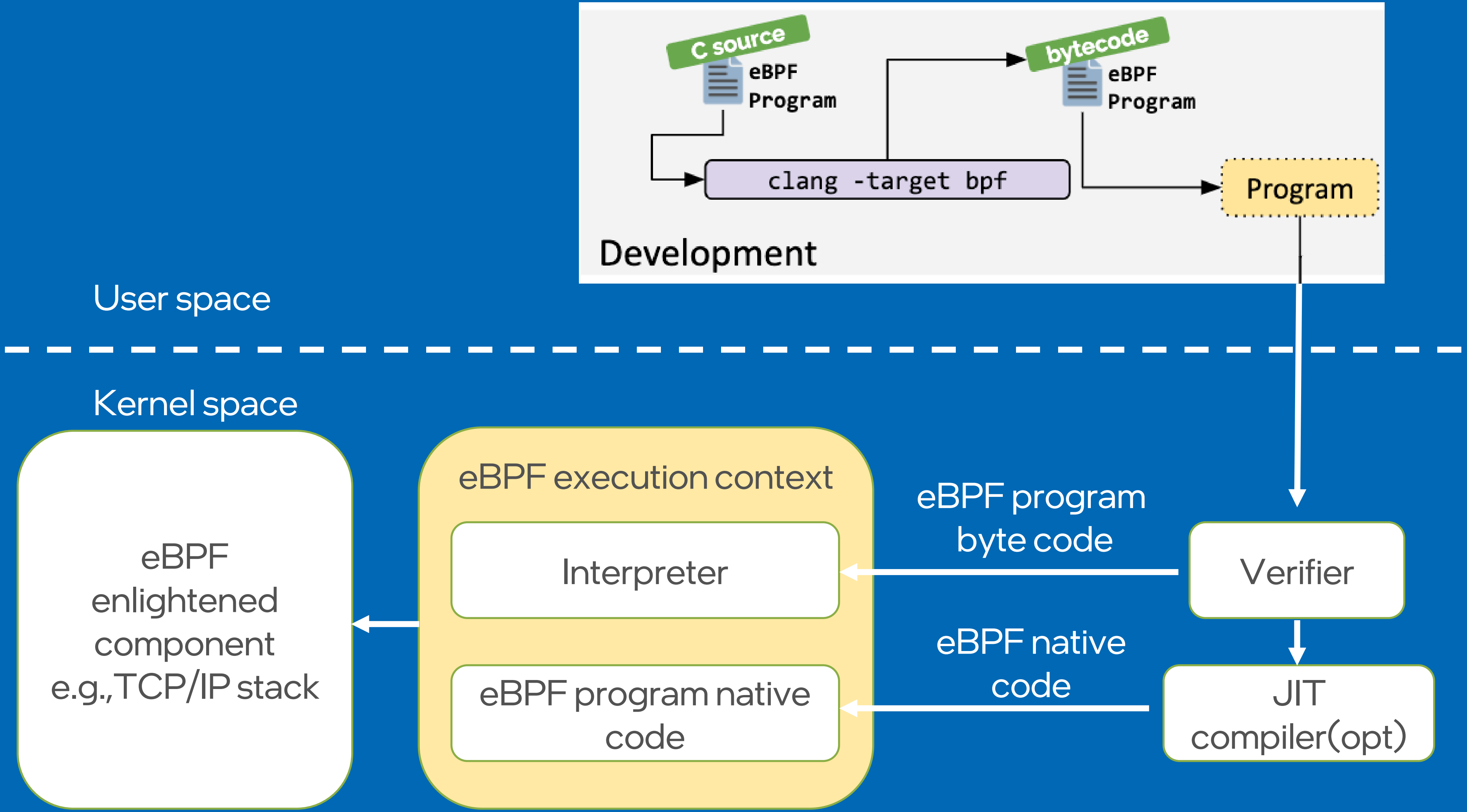
uBPF project

- uBPF project includes an eBPF assembler, disassembler, interpreter (for all platforms), and JIT compiler (for x86-64 and Arm64 targets). Used by windows eBPF support and other projects.
- Maintained by iovisor:
  - https://github.com/iovisor/ubpf/
- As a shared library loaded by QEMU.

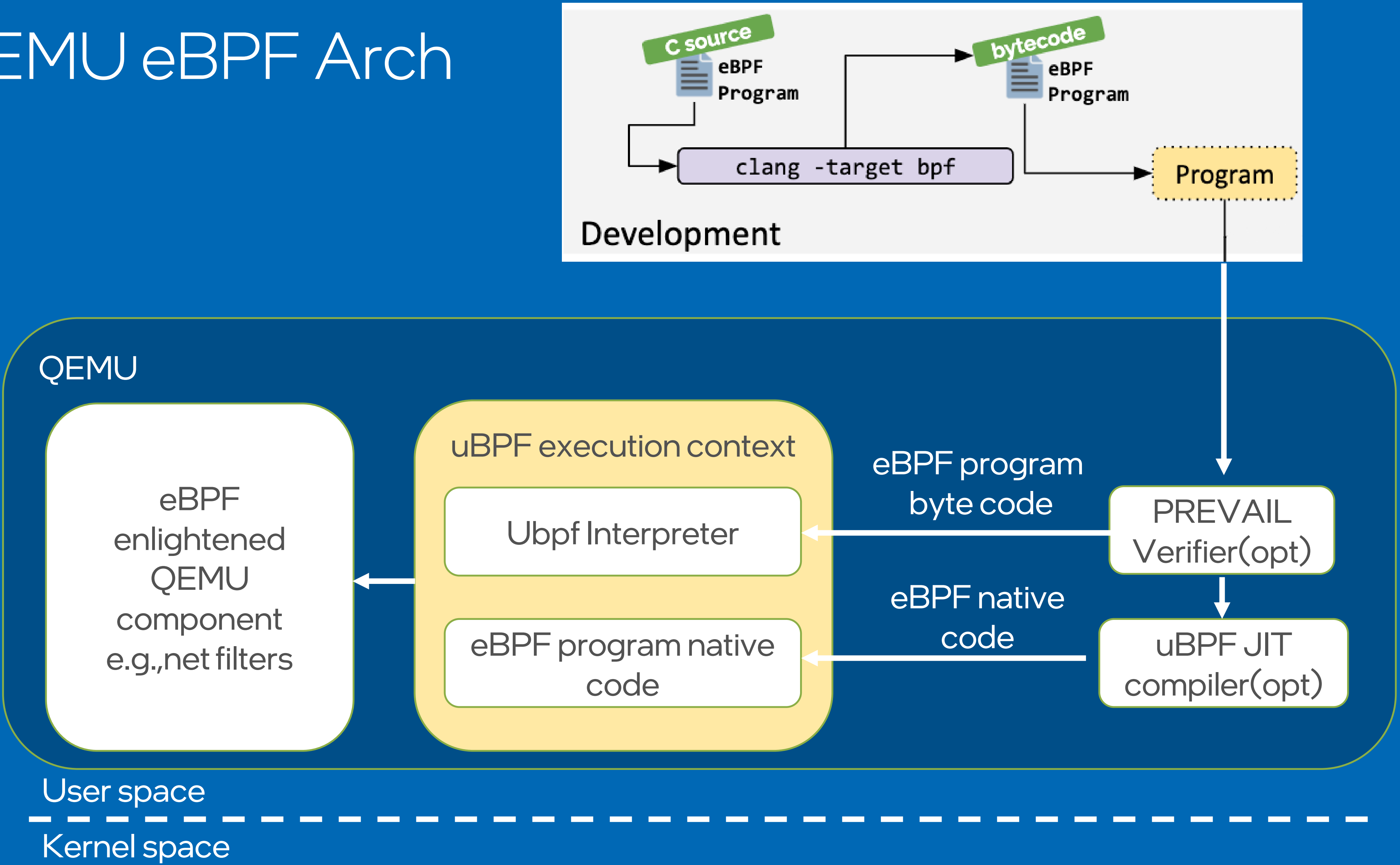
PREVAIL project

- A Polynomial-Runtime EBPF Verifier using an Abstract Interpretation Layer.
- Contributed by Microsoft:
  - https://github.com/vbpf/ebpf-verifier
- Maybe QEMU eBPF just need parts of it. (eBPF program running in sendbox and can't call QEMU's data/function without helpers. If someone already has admin permission on host, he has many ways to bypass it).

Kernel eBPF Arch

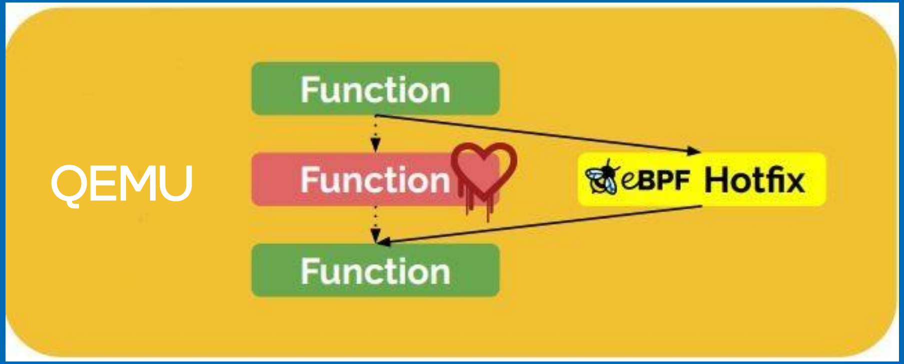


QEMU eBPF Arch



What QEMU Userspace eBPF Can Do?

- Packet filtering/tracing (aka tcpdump), classification and statistics collection.
- New additions can evolve at a rapid pace. Much quicker than normal QEMU Development. Especially for 0-day issue.
- Enable the QEMU hot patching we always dreamed about.
- Enable P4 program language to Qemu network filter.
- 100% modular and composable. Inspire new innovations based on eBPF.

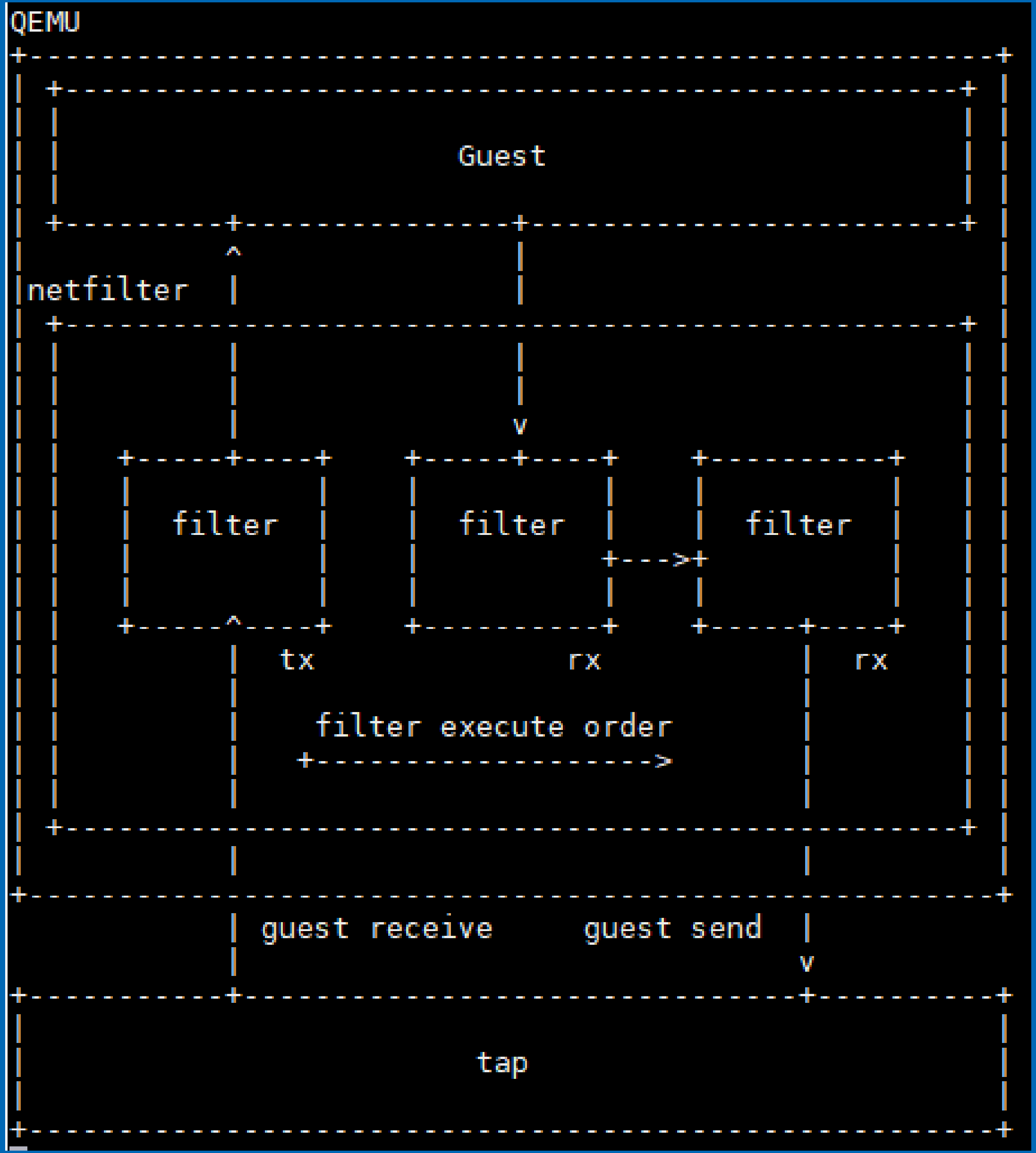


Picture from Thomas Graf's presentation

QEMU Userspace eBPF Development

- Introduce uBPF project to QEMU.
- Add eBPF infrastructure and helpers.
- Start from introduce first uBPF based module/hook ubpf-filter for network packet filtering/tracing (aka tcpdump), classification .

uBPF Network Filter Arch



```
QEMU
+------------------------------------------------------+
| +--------------------------------------------------+ |
| |                                                  | |
| |                      Guest                       | |
| |                                                  | |
| +---------+---------------+------------------------+ |
|           ^               |                          |
|netfilter  |               |                          |
| +--------------------------------------------------+ |
| |         |               |                        | |
| |         |               |                        | |
| |         |               v                        | |
| |   +-----+----+    +-----+----+    +----------+   | |
| |   |          |    |          |    |          |   | |
| |   |  filter  |    |  filter  |    |  filter  |   | |
| |   |          |    |          +--->+          |   | |
| |   |          |    |          |    |          |   | |
| |   +-----^----+    +----------+    +-----+----+   | |
| |         |  tx              rx           |  rx    | |
| |         |                               |        | |
| |         |    filter execute order       |        | |
| |         |   +------------------->       |        | |
| |         |                               |        | |
| +--------------------------------------------------+ |
|           |                               |          |
+------------------------------------------------------+
            | guest receive     guest send  |
            |                               v
+-----------+-------------------------------+----------+
|                                                      |
|                         tap                          |
|                                                      |
+------------------------------------------------------+
```

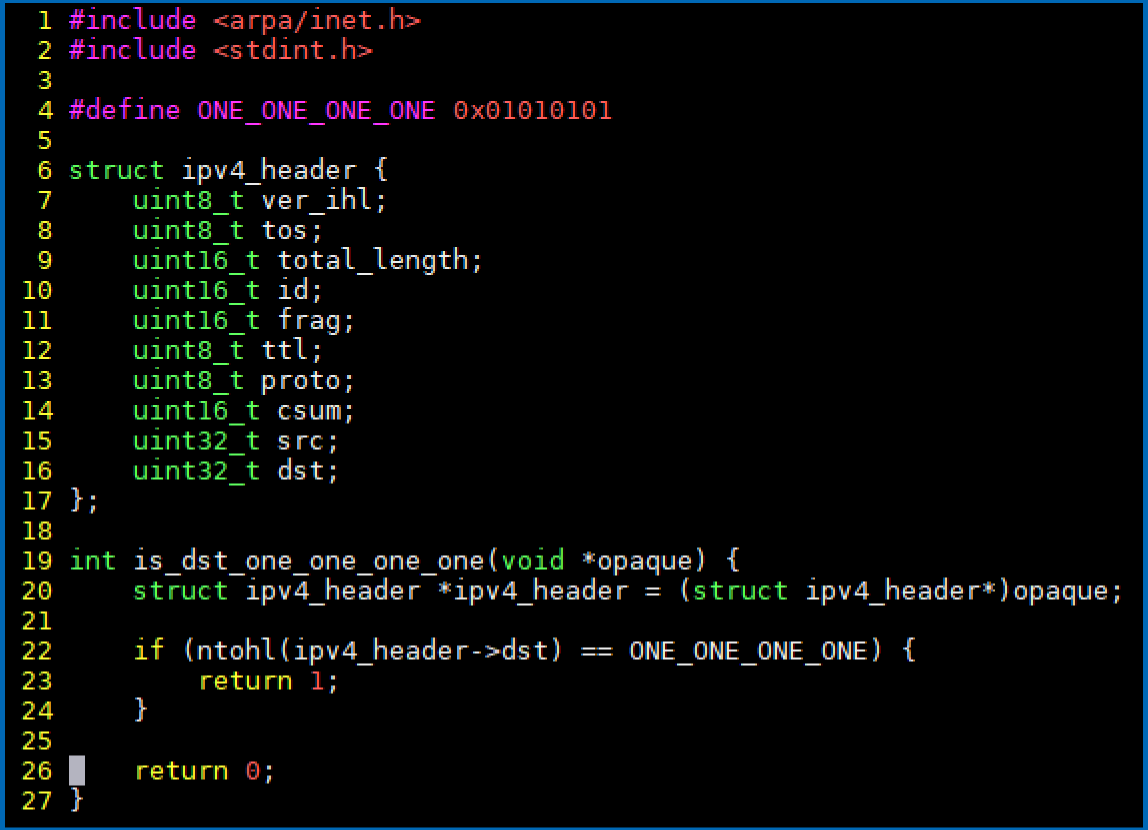
QEMU uBPF Filter Benefits

Make admin have ability to create any local policy for each VM to filter corresponding packages according to business workload. For example:

- Do Firewalls for DDoS mitigation…etc…
- Packet feature recognition filtering.
- Etc…

QEMU uBPF Filter Usage

Prepare your eBPF program



```c
#include <arpa/inet.h>
#include <stdint.h>

#define ONE_ONE_ONE_ONE 0x01010101

struct ipv4_header {
    uint8_t ver_ihl;
    uint8_t tos;
    uint16_t total_length;
    uint16_t id;
    uint16_t frag;
    uint8_t ttl;
    uint8_t proto;
    uint16_t csum;
    uint32_t src;
    uint32_t dst;
};

int is_dst_one_one_one_one(void *opaque) {
    struct ipv4_header *ipv4_header = (struct ipv4_header*)opaque;

    if (ntohl(ipv4_header->dst) == ONE_ONE_ONE_ONE) {
        return 1;
    }

    return 0;
}
```

intel

Compile and load eBPF program to uBPF filter

- Compile the eBPF C program with LLVM
  - clang -O2 -target bpf -c ubpf-test.c -o demo_ubpf.o
- Load the eBPF byte code to QEMU ubpf-filter module.
  - -object filter-ubpf,netdev=hn0,id=ubpf1,queue=tx,ip-mode=on,ubpf-handler=demo_ubpf.o
- No need to reboot the VM, Enjoy!

Current status

- Submitted the RFC version to community. Including load uBPF as shared library and the first effective module network uBPF filter.
- URL:https://www.mail-archive.com/qemu-devel@nongnu.org/msg894623.html
- The initial version focus on programmability, No PREVAIL eBPF Verifier support.

Futrue of QEMU Userspace eBPF

- Collect community comments for this series.
- Add more helpers.
- Add more hook points.
- eBPF map support.
- .....

Disclaimers

Q&A

intel®