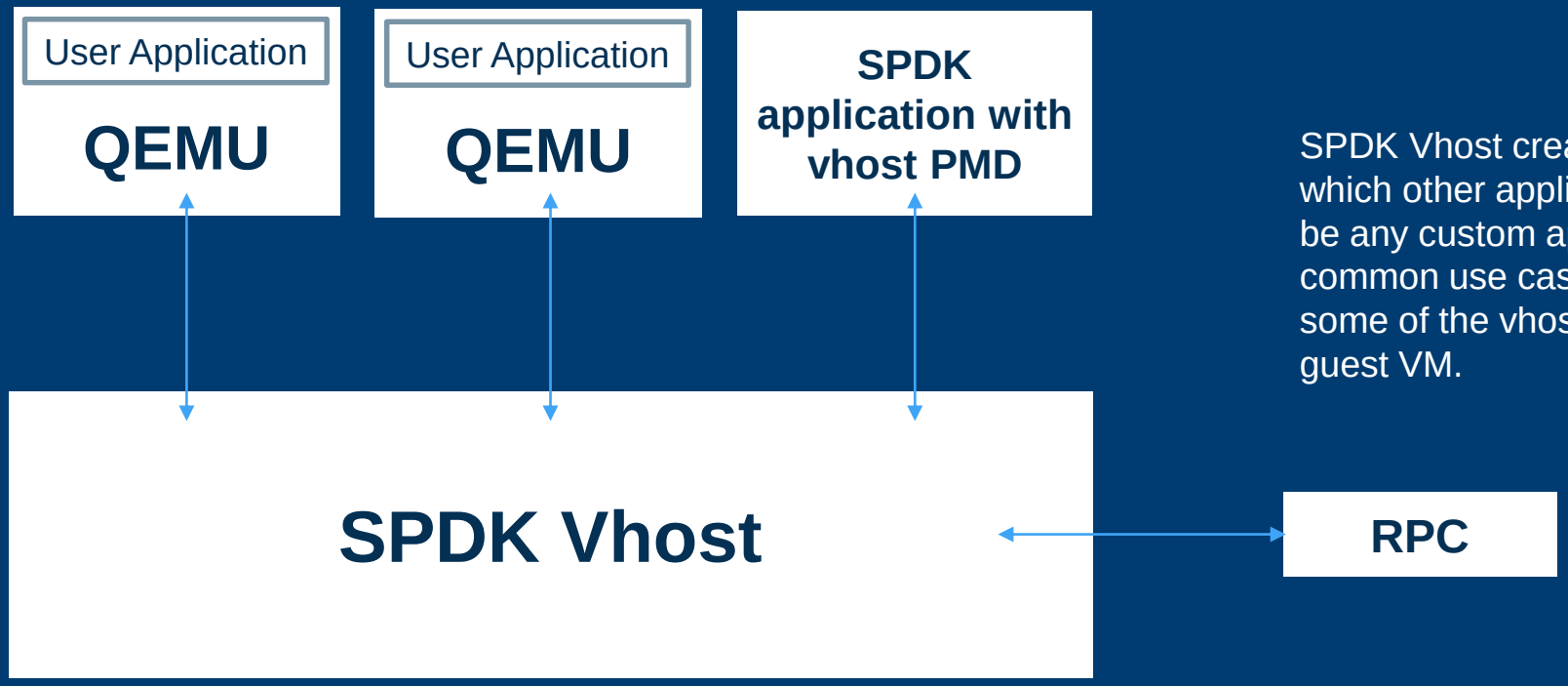
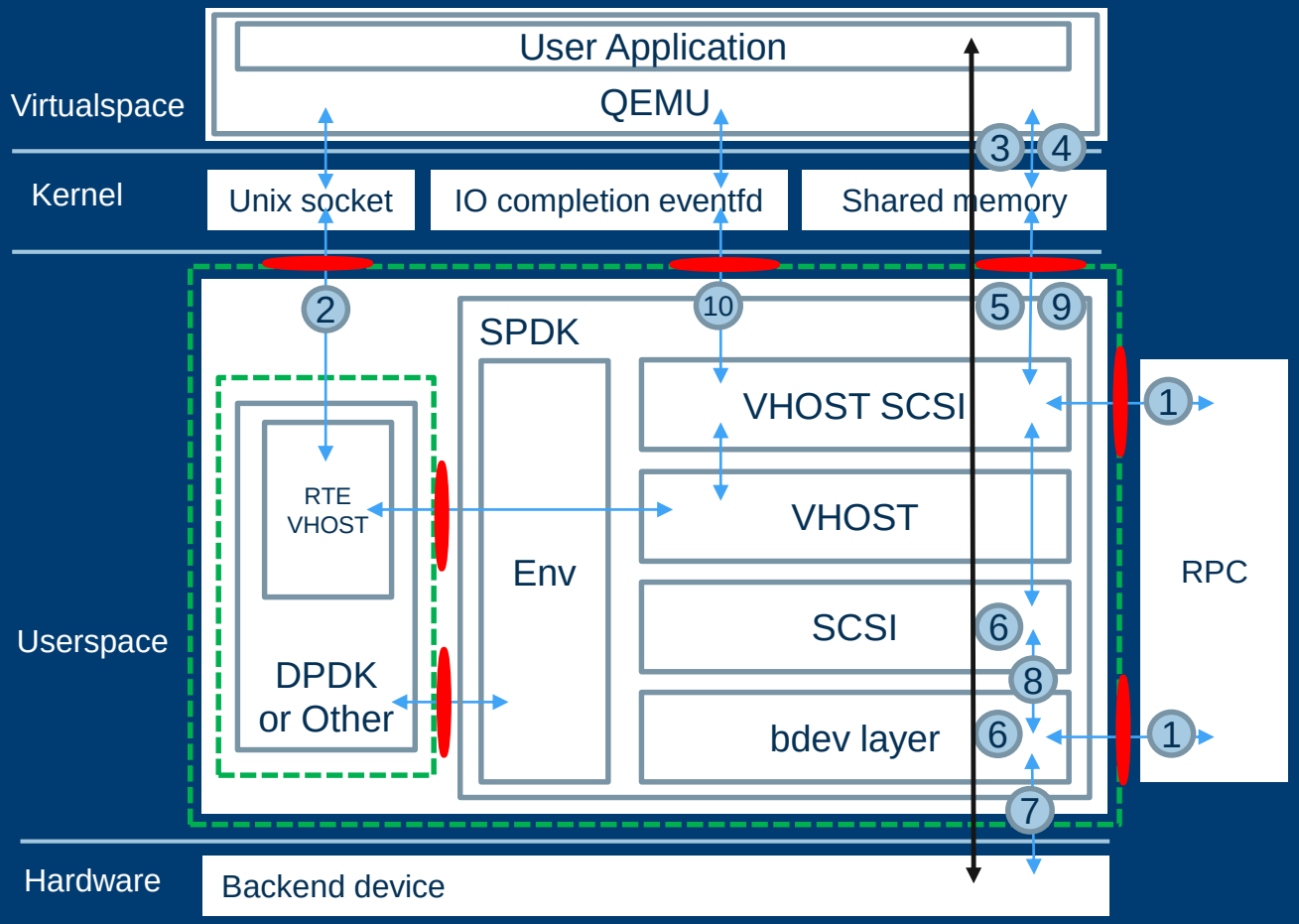
# USE CASE: VHOST INTEGRATION

- User has decided to integrate SPDK vhost stack into his existing virtualization solution
- Related hardware and software components (including bdev interface) were configured correctly so they are assumed to be safe in this use case
- Users can create custom SPDK vhost applications,
  but this document focuses on the default one




SPDK Vhost creates Unix domain sockets which other applications connect to. It could be any custom app connecting, but the most common use case is QEMU which offers some of the vhost communication to the guest VM.

# SYSTEM DIAGRAM




High Level Flow:

1) System admin starts an SPDK app and configures it through RPC
2) SPDK creates a vhost Unix domain socket(s)
3) System admin starts a QEMU instance and instructs it to connect to the SPDK vhost socket
4) QEMU shares the entire VM's memory with SPDK
5) QEMU creates a virtio-pci device to be used inside the VM
6) Virtio-PCI driver inside the VM initializes the device, sets up I/O virtqueues
7) QEMU, in response to device initialization, sends the relevant data over the vhost socket
8) User application inside the VM sends an I/O request
9) Virtio-PCI driver adds a Virtio I/O entry to a virtqueue
10) SPDK detects request by continously polling shared I/O queues
11) SPDK processes the request, first in lib/scsi then in bdev layer
12) Request is sent to backend device
13) Callback is called from bdev layer
14) Vhost updates the request status by modifying shared queues
15) Vhost notifies the application about completion by writing to completion eventfd


Assets:

A. Data
B. SPDK Application
C. Shared Memory
D. Sockets
E. Env
F. QEMU
G. RTE_VHOST

* vhost-blk is an alternative option for vhost-scsi. If vhost-blk is used, the SCSI block can be ommited on the diagram.

# ATTACK SURFACES

| System Element | Compromise Type(s) | Assets exposed | Attack Method |
|---|---|---|---|
| Vhost socket | Denial of service, System integrity | RTE_VHOST, Shared memory, QEMU, SPDK app | Malformed vhost-user commands |
| Completion eventfd | Denial of service | Sockets, RTE_VHOST | Unexpected eventfd writes |
| Shared memory | Data disclosure, Denial of service, System integrity | Data, RTE_VHOST, SPDK app | Malformed IO requests |
| ENV/RTE VHOST interface | Data disclosure, System integrity | Env, RTE_VHOST, Data, QEMU, Sockets, Shared Memory, SPDK app | DLL Injection |
| RPC interface | Data disclosure, System integrity | Sockets, RTE_VHOST, SPDK app | Malformed JSON-RPC requests, Changing SPDK configuration at runtime |

# THREAT MATRIX

| Assets / Surface | Data | SPDK application | Shared memory | Sockets | Env | QEMU | RTE_VHOST |
|---|---|---|---|---|---|---|---|
| QEMU socket interface | | Y | | Y | | Y | Y |
| Completion eventfd | | | | | | | Y |
| Shared memory | Y | Y | Y | | | | Y |
| ENV/RTE VHOST interface | Y | Y | Y | Y | Y | Y | Y |
| RPC interface | | Y | | Y | | | Y |

# ADVERSARIES IN SCOPE

| Persona | Motivation | Attacker Type | Starting Privilege Level | Skill and Potential Effort level |
|---|---|---|---|---|
| Malicious VM User | Wants to snoop data/disrupt users on system | Software Adversary in a VM | None | Unskilled, gives up easily |
| Malicious vhost client | Wants to snoop data/disrupt users on system | Vhost driver Software Adversary | None | Proficient level of skill, does not give up easily |
| Malicious RPC Admin | Denial Of Service | Network Adversary | None | Proficient level of skill, does not give up easily |

* host system software adversary is out of scope because such adversaries have permissions to defeat all mitigations. User needs to ensure appropriate deployment  policies are in place to prevent system level software adversaries

# THREAT/ATTACK SURFACE MATRIX

| Asset\Attack Surface | QEMU socket interface | Completion eventfd | Shared memory | ENV/ RTE_VHOST interface | RPC interface |
|---|---|---|---|---|---|
| Data availability | Y | Y | Y | Y | Y |
| Data confidentiality | | | | Y | |
| Data integrity | Y | Y | Y | Y | Y |
| Shared memory resources | | | | Y | Y |
| Unix sockets | Y | Y | | Y | Y |
| App configuration file | | | | | |

# THREATS

| ID | Threat | Assets | Protections Req'd | Adversary | Attack Point | Technique | Mitigation |
|---|---|---|---|---|---|---|---|
| 1 | Malformed vhost-user commands | data availability, sockets, shared memory | B G | Software adversary in a VM | vhost-user communication | Connect to a vhost socket and send malformed messages to stall or crash the SPDK app | SW to validate input before use |
| 2 | Malformed I/O requests | data availability, hared memory | B E | Software adversary in a VM | vhost-user communication | Connect as a client and try to setup invalid memory region to cause an error on host application | SW to validate input before use |
| 3 | Deinitialization of nonexisting virtqueues | data availability, sockets, shared memory | B G | Software adversary in a VM | vhost-user communication | Connect as a client and try to deinitialize nonexisting virtqueus to cause an error on host application | SW to validate input before use |
| 4 | Repeated reconnect | data availability | B G | Software adversary in a VM, Vhost driver software adversary | vhost-user communication | Repeatedly connect and disconnect causing SPDK to map/unmap memory regions and stall other, legitimate connections | SW to implement smart QoS and thread balancing policy and leverage system privileges to restrict access to socket |

* Protections Req'd lists assets from system diagram

# THREATS

| ID | Threat | Assets | Protections Req'd | Adversary | Attack Point | Technique | Mitigation |
|---|---|---|---|---|---|---|---|
| 5 | Overlapping queue addresses | data availability, sockets, shared memory | B | Software adversary in a VM | virtio data | Connect as a client and try to setup a queues with overlapping addresses to cause infinite loop or other error on host application | SW to validate input before use |
| 6 | Invalid unix socket | data availability | B G | Software adversary in a VM | vhost-user communication | Connect as a client and provide socked used for connection as ex. completion evenfd to cause infinite loop or other error on host | SW to validate input before use |
| 7 | Mutable virtio requests | data availability, data integrity, data confidentiality, sockets, shared memory | B | Vhost driver software adversary | shared memory | Modify virtio request during it being processing by host SPDK app to try to bypass error checking | SW to guarantee immutability of potentially dangerous request data such as addresses, ranges, pointers and leverage system privileges to restrict access to socket |
| 8 | Malicious RPC commands | data availability, data integrity | B G | Network adversary | RPC commands | Try to cause a race condition or other error by sending possiblly conflicting RPC commands, ex hotremoving vhot controller and backend device at the same time | SW to implement privileged commands and/or monitoring system and leverage system privileges to restrict access to socket |

# THREATS

| ID | Threat | Assets | Protections Req'd | Adversary | Attack Point | Technique | Mitigation |
|---|---|---|---|---|---|---|---|
| 9 | Mutable backend device pointer | data confidentiality, data availability | B | Network adversary | RPC commands | Reconfigure vhost controller to use different (available to the RPC user) backend device while it's in use, in order to get access to user data | SW to prevent mutating the controller while it is in use and/or implement privileged access to that functionality |
| 10 | Malicious vhost events | data availability, sockets, shared memory | B G | Software adversary in a VM | vhost-user communication | Connect as a client and provide socked used for connection as ex. completion evenfd to cause infinite loop or other error on host application | SW to validate input before use |
| 11 | WRITEs to I/O queue memory | data availability, data integrity, sockets, shared memory | B G | Vhost driver software adversary | virtio data | As virtio client, issue WRITEs to I/O queue memory to make SPDK send itself new I/Os to process and waste CPU cycles without any VM interaction | SW to implement smart QoS policy and leverage system privileges to restrict access to socket |