DMA monitor for Minix Network Driver

Oliver Wilén, Marcus Dypbukt Källman, Marco Godow, Sebastian Veijalainen

Introduction

Direct Memory Access (DMA) allows devices to directly access system memory. When performing memory access without DMA everything will have to go through the CPU meaning it will be occupied during the operation. DMA instead bypasses the CPU by allowing direct access to the memory which means that the CPU is available for other tasks while the transfer is in progress. This improves the performance of the system but also leads to less secure memory access.

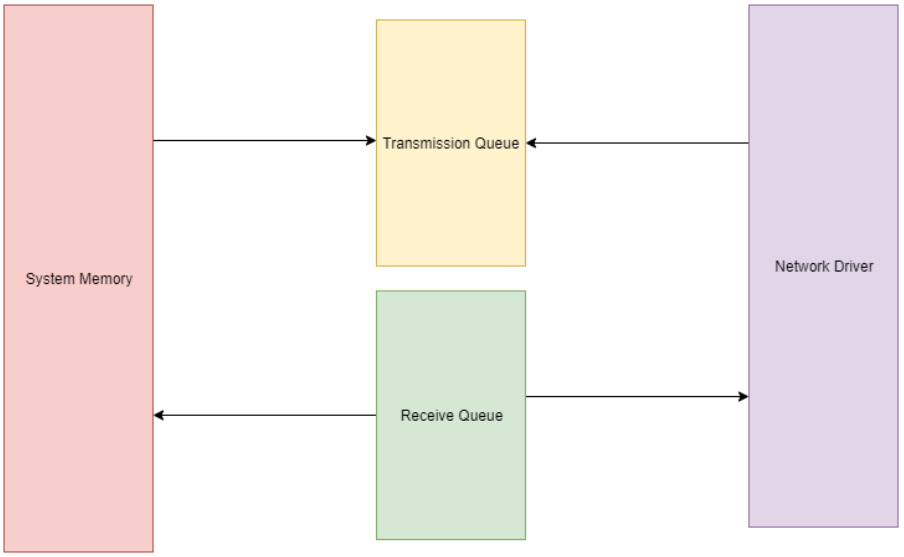
Targeted Vulnerabilities

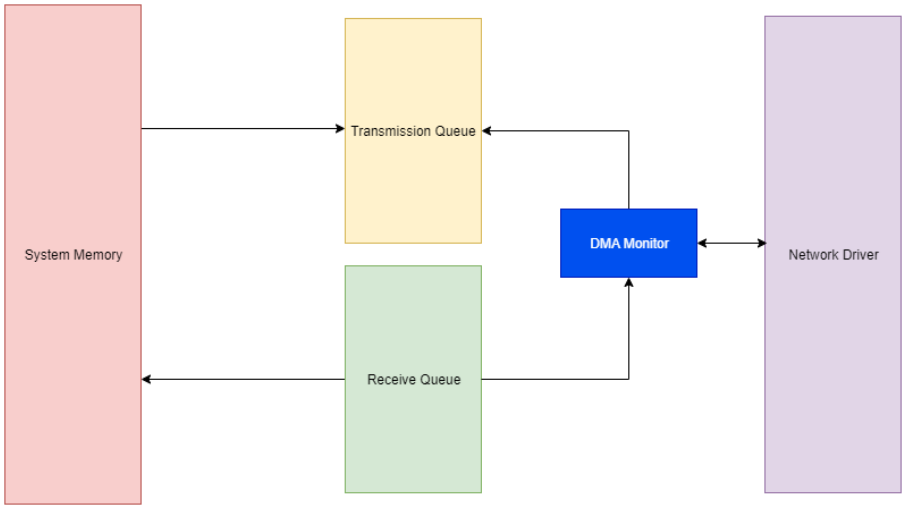
PCI-slot having direct access to system memory means a compromised PCI-device will be able to read and write freely to memory without the supervision of the operating system. This vulnerability is hard to prevent with software since the problem lies with the PCI-slot having DMA, instead, it can be prevented by restricting physical access to exposed DMA-enabled ports.

Another vulnerability is the driver handling the DMA-enabled devices. The driver is what instructs the device where it should read and write. A compromised driver can therefore make a non-compromised device read and write parts of the memory it should and then leak or overwrite sensitive data. If we for example look at a network card, if the driver is corrupted, the address from which the network card will read and send out packets can be altered, and sensitive data be sent through the network card. The same can also be done for where the network card should store packets. This can be hard to detect since the OS is not involved in the memory accesses and the network card can determine if a memory access is faulty.

Project

Our project will try to prevent the second type of attack, where data can be leaked or overwritten, and improve the security of DMA. We will do this by creating a monitor that will sit between the driver and the network card and validate the addresses that the driver tells the card to read and write to. By doing this, a corrupt driver can no longer leak or overwrite parts of memory it shouldn’t since the incorrect addresses will be detected by the monitor. The project will look at the Minix network driver “virtio_net”.





In the diagrams above, we have created an abstraction of the communications between the network driver and the system memory. The Transmission and Receive queues are implemented in VirtIO as a means for the NIC to read or write packages using DMA. The network driver hands out the addresses for the packages that should be read or written. By implementing a monitor between the driver and the queues that verify the addresses before introducing them to the queue, the system gains control over what regions of the memory are accessible.

Goals

Minimal

1. Demonstrate the vulnerabilities of a compromised driver, both data leak and data overwrite.
2. Create a monitor that validates the addresses given to the network card.
3. Be able to detect and prevent memory addresses pointing to kernel memory given by the driver.

4. Demonstrate that the monitor prevents data leaks and overwrites.

Optional

1. Implement our monitor for more drivers or implement it so it works for all drivers.
2. Restrict a corrupt driver to bypass our monitor