Elias Lundgren
Erik Persson
Linus Below Blomkvist
Fredrik Gölman

Project Specification: VirtIO Queue Monitor

Introduction

VirtIO is a virtualization standard that enables users to perform IO operations with minimal overhead. Through an abstraction layer it provides a protocol for network and disk device drivers.

Driver access to memory is done through a queue built up of frames, containing memory address and size for the read/write. VirtIO is however not built to adhere to security protocols set by a capability based kernel, this creates a vulnerability when virtIO is integrated into s3k. Simply put, the virtIO disk driver has the ability to access any memory without adhering to the rules defined by the s3k capabilities.

To combat this, we want to insert a monitor on the queue to always check that the process has enough capabilities to execute the next action in the queue. There are two actions which we wish to monitor: When a new frame is inserted into the queue, and when a capability is withdrawn from the driver. When these are executed we need to assess if the memory in each frame in the queue is within the drivers defined capabilities.

This is important to stop an attacker from reading, writing or executing things in memory that they don’t have access to. A monitor on the queue would entail some hits to performance, but like other secure microkernels, the focus is not generally on performance but reliability and security.

Attacker Perspective

Currently an attacker could use the virtIO Queue to access portions of memory without having the required capabilities. This opens up s3k to code injection and execution and in general full access to the memory.

The VirtIO driver doesn’t currently allow reading and writing to selected memory portions. A hacker could however modify their driver to do this with just a few changes to the current code. In this project we will demonstrate this by editing the function *virtio_disk_rw* in driver file *virtio_disk.c*. This will allow the attacker to use the driver to directly access memory by calling the read/write function, giving them full control over the system.

Defender Perspective

We want to keep track of which memory ranges each process can read from and write to during runtime by implementing a monitor that's used as a decision maker when the driver wishes to add something to its own queue, as well as when another process withdraws a capability from the driver.

For the first issue we want to check with the monitor each time an item is added to the queue in the VirtIO Driver. This can be done using IPC, sending a request to the Monitor with each new frame it wants to add to the queue, and then awaiting a response whether it may add it or not.

For the second issue we want to stop the driver from reading memory segments that it previously had access to but have been withdrawn. This may be achieved by modifying the kernel to confirm with the monitor each time a capability is revoked. If the memory belongs to the driver, we don't revoke the capability and throw an error in the kernel and begin shutting down the driver. The app that is trying to revoke the memory can then loop on this revocation until the driver is shut down, and then properly revoke the capability without any intervention.

This is the first step, but shutting down the driver on each change of capability might not be optimal. A further requirement for this project could be to reevaluate the entire queue when a capability is removed, and remove frames that access illegal memory addresses.

Requirements

- Attack: Modified VirtIO driver that demonstrates the vulnerability
- Defense1: Check with monitor upon adding something to the queue
- Defense2: Check with monitor upon revoking a capability belonging to driver
- Sufficient performance