The userfaultfd() system call allows one thread to handle page faults for another in user space. It has a number of interesting use cases, including the live migration of virtual machines. There are also some less appealing use cases, though, most of which are appreciated by attackers trying to take control of a machine. Attempts have been made over the years to make userfaultfd() less useful as an exploit tool, but this patch set from Axel Rasmussen takes a different approach by circumventing the system call entirely.

A call to userfaultfd() returns a special file descriptor attached to the current process. Among other things, this descriptor can be used (with ioctl()) to register regions of memory. When any thread in the current process encounters a page fault in a registered area, it will be blocked and an event will be sent to the userfaultfd() file descriptor. The managing thread, on reading that event, has several options for how to resolve the fault; these include copying data into a new page, creating a zero-filled page, or mapping in a page that exists elsewhere in the address space. Once the fault has been dealt with, the faulting thread will continue its execution.

A thread will normally encounter a page fault while running in user space; it may have dereferenced a pointer to a not-present page, for example. But there are times that page faults can happen within the kernel. As a simple example, consider a read() call; if the buffer provided to read() is not resident in RAM, a page fault will result when the kernel tries to access it. At that point, execution will be blocked as usual, but it will be blocked in the kernel rather than in user space.

Zoned storage is a form of storage that offers higher capacities by making tradeoffs in the
kinds of writes that are allowed to the device. It was the topic of a storage and filesystem session led by Luis Chamberlain at the 2022 Linux Storage, Filesystem, Memory-management and BPF Summit (LSFMM). Over the years, zoned storage has been a frequent topic at LSFMM, going back to LSFMM 2013, where support for shingled magnetic recording (SMR) devices, which were the starting point for zoned storage, was discussed.

Chamberlain began with the news that a zoned storage microconference had been accepted for this year’s Linux Plumbers Conference (LPC). He encouraged attendees to submit topics and hoped it was an opportunity to introduce more user-space developers to zoned-storage concepts. LPC will be held September 12-14 in Dublin, Ireland.

Retrieving kernel attributes [4]

At the 2022 Linux Storage, Filesystem, Memory-management and BPF Summit (LSFMM), Amir Goldstein and Miklos Szeredi led a discussion on a new interface for extracting information from kernel objects using the filesystem extended-attributes (xattr) interface. Since Szeredi was not present in Palm Springs, he co-led the session virtually over Zoom audio, which was the only filesystem session with a virtual leader at LSFMM this year. Szeredi’s proposal for an interface of that sort had been posted just the day before the session.

Goldstein started things off by noting that there are several use cases where there is a need for a new API to obtain bits of information from the kernel, so it seems like a good idea to create a common API that can meet those needs. Szeredi proposed the getfattr mechanism, which builds on the xattr interface; Goldstein said that he was happy with the idea, as were Szeredi and Dave Chinner, who suggested the idea a year or so ago. In addition, other than an objection to binary data, Greg Kroah-Hartman was "not unhappy" with the idea.

Szeredi took over to describe the proposal in more detail. The intent is to be able to get attributes from some kernel objects; those could be mounts or inodes, but processes or other objects are possible as well. There are several existing interfaces for getting this kind of information, but each has a different way to access the attributes, so it would be nice to have a unified interface, he said.

The xattr API was repurposed for his proposal. It uses a different namespace for the new attributes, however, in order to ensure that legacy code will not break due to unexpected new attributes. For example, listxattr() would not return attributes from the new namespace. One objection to the interface is that it is not efficient enough if there is a need to retrieve multiple attributes. Szeredi said that would need to be tested to see if it is truly a problem, but if so, the API could be extended with a bulk-retrieval mechanism.

A discussion on readahead [5]

Readahead is an I/O optimization that causes the system to read more data than has been
requested by an application?in the belief that the extra data will be requested soon thereafter.

At the 2022 Linux Storage, Filesystem, Memory-management and BPF Summit (LSFMM), Matthew Wilcox led a session to discuss readahead, especially as it relates to network filesystems, with assistance from Steve French and David Howells. The latency of the underlying storage needs to factor into the calculation of how much data to read in advance, but it is not entirely clear how to do so.

Wilcox began by describing readahead a bit. If user space is reading a file one byte at a time, Linux does not actually read the data that way; instead, it issues reads for a bigger chunk, say 64KB, which gets stored in the page cache. There is a certain amount of latency between the time a page is requested from the storage and when it appears in the page cache; that latency varies greatly over the wide variety of storage types that Linux supports. For network storage, those types can range from locally stored data on multi-gigabit Ethernet to data stored halfway around the world over decidedly slower links. Similarly, for local storage it can range from a 5GB-per-second NVMe SSD to some "crappy USB key picked up from a vendor at a trade show". There is "a lot of stuff to contend with there".

Remote participation at LSFMM [6]

As with many conferences these days, the 2022 Linux Storage, Filesystem, Memory-management and BPF Summit (LSFMM) had a virtual component. The main rooms were equipped with a camera trained on the podium, thus the session leader, so that remote participants could watch; this camera connected into a Zoom conference that allowed participation from afar. In a session near the end of the conference, led by conference organizer Josef Bacik, remote participants were invited to share their experiences?on camera?with those who were there in person. It was an opportunity to discuss what went right?and wrong?with an eye toward improving the experience for future events.

Ric Wheeler was first up; he said that aside from the first few minutes where he could not hear anyone, "it was a good virtual experience". Mel Gorman echoed that and noted that it "was infinitely superior to not being able to participate at all". One thing he noted is that the "raised hands" in the Zoom interface were not monitored in some sessions. There were two sessions that he remembered where the speaker asked for objections or other comments and he was left "screaming at the mic". It is difficult to keep an eye on that when leading a session, he said; it was only a minor detraction from the overall great experience.

Source URL: http://www.tuxmachines.org/node/166371

Links:
[2] https://lwn.net/Articles/897260/
[3] https://lwn.net/Articles/897263/
[4] https://lwn.net/Articles/897420/
[5] https://lwn.net/Articles/897786/
[6] https://lwn.net/Articles/897915/