Syscall-based Isolation for Monolithic OS Kernels

Yousuke Tanimoto
TUAT
Tokyo, Japan
tanimoto@asg.cs.tuat.ac.jp

Hiroshi Yamada
TUAT
Tokyo, Japan
hiroshiy@cc.tuat.ac.jp

Memory isolation is key to mitigating undefined behaviors of commodity operating systems (OSes). Since these kernels are typically monolithic, their functionalities run in the same address space. Errors propagating from the buggy components or hijacked kernel contexts can damage all kernel memory objects. The isolation mechanism builds multiple protection domains, traps any access from one to the others' objects, and forces the faulted OS kernel to fail-stop. Appropriate isolation in the kernels prevents failed components from accessing healthy objects and minimizes their effect as much as possible. Modern CPUs offer hardware support to enforce memory isolation, such as Intel Memory Protection Keys for Supervisor Pages (PKS) and ARM Memory Domains.

Although the existing isolation techniques, which are typically *component-based*, effectively force OS kernels to perform a fail-stop on illegal accesses to protected objects, *intra-component errors*, caused by some complicated bugs, can evade a component-based isolation and trigger undefined behaviors in the affected OS kernels as they propagate. An intra-component error is an error that illegally updates memory objects inside an OS component so that the component-based isolation cannot prevent it from propagating. For example, the CVE-2022-0817 [1] vulnerability, known as Dirty Pipe, allows attackers to write to a read-only pipe buffer into the file via splice() by exploiting a merge check miss, which means that the error propagation is confined inside only the file system and the attack is done completely. Also, existing component-specific approaches cannot be applied to the entire body of OS kernels; errors can propagate over the kernel entities outside the protected component.

This paper presents *Iv6*, an OS kernel whose internals are isolated to mitigate intra-component error propagation. To enhance the memory isolations, Iv6 offers a system call (syscall)-based isolation where the kernel process context can access only kernel objects necessary for the issued syscall execution; it assigns different access permissions to different kernel contexts. The kernel context in a syscall accesses only the kernel objects necessary for its execution, and Iv6 forbids it from reading and writing the other objects, even those in the same component. When a process reads or writes objects unrelated to the current syscall context, Iv6 detects the illegal accesses and forces the running OS kernel to fail-stop. For example, open() can update proc, which is a process control block in xv6, to connect it to a new file descriptor but write() cannot modify it. In another example, write() can access pipe but open cannot do so. Iv6 makes use of CPU-supported protection keys; our prototype uses Intel PKS to assign different access permissions to kernel objects to syscalls in a lightweight manner.

Iv6 introduces *memory domains*, each of which has access permission for memory objects based on Intel PSK and is associated with one syscall. A kernel context is executed with one memory domain, and the accesses to kernel objects in the context follow its permission. When a process issue open(), Iv6 sets open()'s memory domain and executes its kernel context. It releases the access control before returning to the user space. Iv6 conceptually generates the same number of memory domains as the syscalls. In addition, it prepares several memory domains for interrupt handlers to mitigate their failures. The management is performed by two subsystems: the *domain allocator* and *domain switcher*. The domain allocator allocates memory domains to all the syscalls in an on-demand manner, while the domain switcher safely sets a memory domain based on the issued syscall. When a process issues a syscall and the domain switcher does not find the corresponding memory domain, it requests the domain allocator to create a new memory domain and the domain switcher associates the domain to the issued syscall. The domain allocator also provides more memory domains than the number of physical PKEYs by extending VDom [2]. The domain switcher switches memory domains so safely that damaged or hijacked kernel contexts cannot change their memory domains arbitrarily and update their access rights.

We prototyped Iv6 into an x86-64-based xv6 integrated with a lwIP-based network stack. Our prototype, yet to be well-optimized, controls 37 syscalls (21 original xv6 syscalls and 16 socket syscalls). We conducted experiments using it, and the experimental results show that Iv6 successfully isolates kernel contexts on a syscall basis with up to 1.22× runtime overhead. We are now optimizing the prototype, measure its memory space ooverhead, and conducting experiments with more complex workloads like web servers.

References

[1] CVE Program. CVE-2022-0817, Accessed: 2023-12-20. https://www.cve.org/CVERecord?id=CVE-2022-0817.

[2] Ziqi Yuan, Siyu Hong, Rui Chang, Yajin Zhou, Wenbo Shen, and Kui Ren. Vdom: Fast and unlimited virtual domains on multiple architectures. In *Proc. of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '23)*, pages 905–919, 2023.