Hash Suite - Windows password security audit tool. GUI, reports in PDF.

[<prev] [next>] [day] [month] [year] [list]

Date: Thu, 25 Nov 2021 19:15:19 +0000
From: Nadav Amit <namit@...are.com>
To: "oss-security@...ts.openwall.com" <oss-security@...ts.openwall.com>
CC: Mike Kravetz <mike.kravetz@...cle.com>, Greg Kroah-Hartman
        <gregkh@...uxfoundation.org>, Security Officers <security@...nel.org>, Andrew
 Morton <akpm@...ux-foundation.org>
Subject: CVE-2021-4002: Linux kernel: Missing TLB flush on hugetlbfs

On Linux kernel 3.6 and later it is possible for an attacker to leak or change data that resides on hugetlbfs. Such data can reside on hugetlbfs, for instance, if the victim runs mmap() using the MAP_HUGETLB or shmget() with SHM_HUGETLB. If a victim maps executable code onto hugetlbfs, the executable can be modified as well.

The bug is caused due to a missing TLB flush when unmapping of a page of PMDs is performed by clearing a PUD. While the comment in the code claims that it is safe, it is not, since no flush would take place under these circumstances (unless, of course, it was needed for some other reason).

Apparently the bug existed since commit 24669e58477e ("hugetlb: use mmu_gather instead of a temporary linked list for accumulating pages)" which means that it existed since kernel 3.6. There might be some mitigating factors in certain older kernels on certain architectures. For instance, x86 performed TLB flushes on huge-pages more eagerly in the past.

Fix:

The fix is upstreamed as commit a4a118f2eead ("hugetlbfs: flush TLBs correctly after huge_pmd_unshare”). Backporting of the fix to older kernels is in progress.

To fix the bug a call to tlb_flush_pmd_range() is needed from __unmap_hugepage_range() when huge_pmd_unshare() succeeds, and forcing a flush before returning from __unmap_hugepage_range().

Details:

An attacker can using shmget() 512 pages of 2MB map twice which are aligned to PUD alignment and fault in some of the pages. As the pages are properly aligned, the kernel would share a PUD between the mappings. Later the attacker would remove the mappings and the shared memory segments.

The first mapping that is removed does not trigger a TLB flush due to a bug in __unmap_hugepage_range(). Later, if the kernel reallocates the huge-pages to another process shortly after, an attacker would be able to read and write these huge-pages for some time (until TLB flush happens for some other reason later on).

A proof of concept is attached. The PoC creates a child process that allocates a huge-page and this data is leaked back to the parent. There is no need for the attacker to be the parent of the child and this is only implemented in such manner for simplicity. The PoC fails on the first iteration on my system repeatedly (although it is written to make multiple attack attempts).

To make it work the PoC work, configure the number of pages to 512 ("echo 512 > /proc/sys/vm/nr_hugepages"), so huge-pages will be available for the PoC and will be reused by the victim.

Download attachment "poc.c**" of type "**application/octet-stream**" (3271 bytes)**

Powered by blists - more mailing lists

Please check out the Open Source Software Security Wiki, which is counterpart to this mailing list.

Confused about mailing lists and their use? Read about mailing lists on Wikipedia and check out these guidelines on proper formatting of your messages.

OPENWALL OPENVZ