Subject: Re: [PATCH v2 11/11] protect architectures where THREAD_SIZE >= PAGE_SIZE against fork bombs
Posted by Michal Hocko on Tue, 21 Aug 2012 09:35:13 GMT
View Forum Message <> Reply to Message

On Thu 09-08-12 17:01:19, Glauber Costa wrote:
> Because those architectures will draw their stacks directly from the
> page allocator, rather than the slab cache, we can directly pass
> __GFP_KMEMCG flag, and issue the corresponding free_pages.
>
> This code path is taken when the architecture doesn't define
> CONFIG_ARCH_THREAD_INFO_ALLOCATOR (only ia64 seems to), and has
> THREAD_SIZE >= PAGE_SIZE. Luckily, most - if not all - of the remaining
> architectures fall in this category.

quick git grep "define *THREAD_SIZE\>" arch says that there is no such
architecture.

> This will guarantee that every stack page is accounted to the memcg the
> process currently lives on, and will have the allocations to fail if
> they go over limit.
>
> For the time being, I am defining a new variant of THREADINFO_GFP, not
> to mess with the other path. Once the slab is also tracked by memcg, we
> can get rid of that flag.
>
> Tested to successfully protect against :(){ :|:& };:

I guess there were no other tasks in the same group (except for the
parent shell), right? I am asking because this should trigger memcg-oom
but that one will usually pick up something else than the fork bomb
which would have a small memory footprint. But that needs to be handled
on the oom level obviously.

> Signed-off-by: Glauber Costa <glommer@parallels.com>
> Acked-by: Frederic Weisbecker <fweisbec@redhat.com>
> CC: Christoph Lameter <cl@linux.com>
> CC: Pekka Enberg <penberg@cs.helsinki.fi>
> CC: Michal Hocko <mhocko@suse.cz>
> CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
> CC: Johannes Weiner <hannes@cmpxchg.org>
> CC: Suleiman Souhlal <suleiman@google.com>

Reviewed-by: Michal Hocko <mhocko@suse.cz>

> ---
> include/linux/thread_info.h | 2 ++
> kernel/fork.c | 4 ++--

> 2 files changed, 4 insertions(+), 2 deletions(-)
>
> diff --git a/include/linux/thread_info.h b/include/linux/thread_info.h
> index ccc1899..e7e0473 100644
> --- a/include/linux/thread_info.h
> +++ b/include/linux/thread_info.h
> @@ -61,6 +61,8 @@ extern long do_no_restart_syscall(struct restart_block *parm);
> # define THREADINFO_GFP (GFP_KERNEL | __GFP_NOTRACK)
> #endif
>
> +#define THREADINFO_GFP_ACCOUNTED (THREADINFO_GFP | __GFP_KMEMCG)
> +
> /*
>   * flag set/clear/test wrappers
>   * - pass TIF_xxxx constants to these functions
> diff --git a/kernel/fork.c b/kernel/fork.c
> index dc3ff16..b0b90c3 100644
> --- a/kernel/fork.c
> +++ b/kernel/fork.c
> @@ -142,7 +142,7 @@ void __weak arch_release_thread_info(struct thread_info *ti) { }
> static struct thread_info *alloc_thread_info_node(struct task_struct *tsk,
>   int node)
> {
> - struct page *page = alloc_pages_node(node, THREADINFO_GFP,
> + struct page *page = alloc_pages_node(node, THREADINFO_GFP_ACCOUNTED,
>   THREAD_SIZE_ORDER);
>
>   return page ? page_address(page) : NULL;
> @@ -151,7 +151,7 @@ static struct thread_info *alloc_thread_info_node(struct task_struct
*tsk,
> static inline void free_thread_info(struct thread_info *ti)
> {
>   arch_release_thread_info(ti);
> - free_pages((unsigned long)ti, THREAD_SIZE_ORDER);
> + free_accounted_pages((unsigned long)ti, THREAD_SIZE_ORDER);
> }
> # else
> static struct kmem_cache *thread_info_cache;
> --
> 1.7.11.2
>
> --
> To unsubscribe from this list: send the line "unsubscribe cgroups" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html

--
Michal Hocko