

[UNIX] BSD Reference Count Overflow in shmat()

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-02/0018.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 02/09/04

To: list@securiteam.com

Date: 9 Feb 2004 11:52:17 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

BSD Reference Count Overflow in shmat()

SUMMARY

While gathering material for a security training Pine Digital Security encountered a reference count overflow condition that could lead to privilege escalation.

DETAILS

Vulnerable systems:

- * FreeBSD version 2.2.0
- * NetBSD version 1.3
- * OpenBSD version 2.6

The `shmat(2)` function maps a shared memory segment, previously created with the `shmget(2)` function, into the address space of the calling process.

This function is implemented in the `sysv_shm.c` file:

--- `sysv_shm.c` lines 317–322 ---

```
vm_object_reference(shm_handle->shm_object);
```

```
rv = vm_map_find(&p->p_vmspace->vm_map,
```

Securiteam: [UNIX] BSD Reference Count Overflow in shmat()

```
shm_handle->shm_object,  
0, &attach_va, size,  
(flags & MAP_FIXED) ? 0 : 1,  
prot, prot, 0);
```

```
if (rv != KERN_SUCCESS) return ENOMEM;
```

-- end of code snippet --

The `shmat(2)` function first increases the reference count on the underlying `vm_object` and then attempts to insert the `vm_object` into the process address space.

The vulnerability occurs because the `shmat(2)` function forgets to decrease the reference count when the `vm_map_find` function returns failure.

Since the caller of `shmat(2)` can specify the address at which the segment should be mapped it is possible to have `vm_map_find` return failure and thus end up with stale references.

Exploitability:

This vulnerability can be exploited (reliably) by local users:

One would first create a shared memory segment using the `shmget(2)` function and create two separate mappings at different locations in the process address space using the `shmat(2)` function.

After making around $2^{32}-2$ (invalid) calls to the `shmat(2)` function the reference count of the underlying `vm_object` will wrap around to 1.

After deleting one of our mappings using the `shmdt(2)` function the underlying `vm_object` will be freed and we will still have one (extraneous) mapping hanging around.

One would then invoke some magic trickery and execute a suid binary that will reuse the freed `vm_object` for its stack segment.

At this point one could write directly into the stack segment of the suid binary (using the extraneous mapping) and thus escalate one's privileges easily.

Patches:

The various CVS repositories should be updated.

Impact:

Local users can elevate their privileges.

ADDITIONAL INFORMATION

The original advisory can be found at:

<<http://www.pine.nl/press/pine-cert-20040201.txt>>
<http://www.pine.nl/press/pine-cert-20040201.txt>.

Securiteam: [UNIX] BSD Reference Count Overflow in shmat()

The information has been provided by <mailto:joost@pine.nl> Joost Pol.

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.