

# [TOOL] TESO Burneye Unwrapper

*Source:* <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-07/0065.html>

---

*From:* [support@securiteam.com](mailto:support@securiteam.com)

*Date:* 07/13/02

From: [support@securiteam.com](mailto:support@securiteam.com)

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: Sat, 13 Jul 2002 00:48:17 +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.

-----

TESO Burneye Unwrapper

---

## DETAILS

Burndump is a LKM that strips off the TESO Burneye protection from encrypted executables. You must be able to run the executable. When the program is unwrapped, you do not need the host-fingerprint or the password anymore and the ELF file can be reverse engineered without the Burneye anti-debugger tricks.

Tool code:

/\* burndump.c -- burneye unwrapper by [ByteRage] ([byterage@yahoo.com](mailto:byterage@yahoo.com))

\* <http://www.byterage.cjb.net>

\*

\* this LKM can only unwrap binaries that can be run!

\* -> you need the password if the binary is password protected

\* -> it can not remove the protection when the binary is

\* host-fingerprint protected, and you are not allowed to run

\* the binary

\*

\* -> compile with `gcc -c burndump.c`

\* (`gcc -c -I/usr/src/linux/include burndump.c` on some systems)

\* -> load kernel module with `insmod burndump`

\* -> run 7350 binary

\* -> unload kernel module with `rmmod burndump`

\* -> unwrapped binary will be in `./burnout`

```

*
* tested under a linux 2.4.x kernel
*/

#define MODULE
#define __KERNEL__
#include <linux/kernel.h>
#include <linux/module.h>
#include <asm/unistd.h>
#include <asm/uaccess.h>
#include <sys/syscall.h>
#include <linux/slab.h>

extern void* sys_call_table[];

int (*open)(char *, int, int);
int (*write)(int, char *, int);
int (*close)(int);

int (*old_brk) (void *addr);

asmlinkage int wrapped_brk(void *addr) {
    unsigned int fd;
    unsigned long codeptr, ELFsign, m, n, fnd, cntelf;
    unsigned long phoff, shoff, offset, filesz, totalsize;
    unsigned short phentsize, phnum, shentsize, shnum;
    mm_segment_t old_fs;
    /* only unwrap burneye protected executables with uid root */
    if (current->uid == 0) {
        old_fs = get_fs();
        printk("<1>brk(%08X) (PID:%d) (start code:%08X) (end code:%08X)\n",
addr, current->pid, current->mm->start_code, current->mm->end_code);
        set_fs(get_ds());
        /* start_code + 1 because we don't want to dump the whole
        burneye ELF itself */
        codeptr = current->mm->start_code + 1;
        /* caller == burneye ??? */
        if ((codeptr >> 16) == 0x0537) {
            printk("<1> 7350 signature 0x0537 found!\n");
            cntelf = 1;
        }
    }
    findelf:
    n = -1; m = 0;
    while(m < cntelf) {
        fnd = 0;
        while(fnd == 0) {
            n++;
            if ((codeptr+n+2) < current->mm->end_code) {
                if (*(unsigned long*)(codeptr+n) == 0x464C457F)
                    fnd = 1;
            } else {
                printk("<1> No more ELF signatures found! Returning...\n");
            }
        }
    }
}

```

```

    return old_brk(addr);
}
}
m++;
}
ELFsign = codeptr+n;
printk("<1> ELF signature found at %08X...\n", ELFsign);
printk("<1> Calculating size ...\n");
totalsize = 0;
phoff = *(unsigned long *)(ELFsign+28);
phentsize = *(unsigned short *)(ELFsign+42);
phnum = *(unsigned short *)(ELFsign+44);
totalsize = phoff+(phnum*phentsize);
for(n = 0; n < phnum; n++) {
    offset = *(unsigned long *)(ELFsign+phoff+(n*phentsize)+4);
    filesz = *(unsigned long *)(ELFsign+phoff+(n*phentsize)+16);
    if (offset+filesz > totalsize)
        totalsize = offset+filesz;
}
shoff = *(unsigned long *)(ELFsign+32);
shentsize = *(unsigned short *)(ELFsign+46);
shnum = *(unsigned short *)(ELFsign+48);
if (shoff+(shnum*shentsize) > totalsize)
    totalsize = shoff+(shnum*shentsize);
for(n = 0; n < shnum; n++) {
    offset = *(unsigned long *)(ELFsign+shoff+(n*shentsize)+16);
    filesz = *(unsigned long *)(ELFsign+shoff+(n*shentsize)+20);
    if (offset+filesz > totalsize)
        totalsize = offset+filesz;
}
printk("<1> Total size : %d bytes\n", totalsize);
if (totalsize == 271) {
    printk("<1> ELF is part of burneye engine... Skipping...\n");
    cntelf++;
    goto findelf;
}
printk("<1> Dumping binary ...\n");
fd = open("burnout", O_CREAT|O_RDWR|O_EXCL, 0700);
write(fd, (void *)ELFsign, totalsize);
close(fd);
printk("<1> Done!\n");
}
set_fs(old_fs);
}
return old_brk(addr);
}

int init_module(void) {
    old_brk = sys_call_table[__NR_brk];
    sys_call_table[__NR_brk] = wrapped_brk;
    open = sys_call_table[__NR_open];

```

Securiteam: [TOOL] TESO Burneye Unwrapper

```
write = sys_call_table[__NR_write];
close = sys_call_table[__NR_close];
printk("<1>burndump loaded...\n");
return 0;
}

void cleanup_module(void) {
    sys_call_table[__NR_brk] = old_brk;
    printk("<1>burndump unloaded...\n");
}
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:[byterage@yahoo.com](mailto:byterage@yahoo.com)>  
ByteRage.

=====

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](mailto:list-unsubscribe@securiteam.com)  
In order to subscribe to the mailing list, simply forward this email to: [list-subscribe@securiteam.com](mailto: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.

- 
- **Previous message:** [support@securiteam.com](mailto:support@securiteam.com): "[TOOL] Bigeye, Service Emulation Tool"
  - **Messages sorted by:** [\[ date \]](#) [\[ thread \]](#) [\[ subject \]](#) [\[ author \]](#) [\[ attachment \]](#)