



# SA30228 / CVE-2008-1105

**Generated by Secunia**

**29 May, 2008**

*5 pages*

## Table of Contents

<b>Introduction</b>	<b>2</b>
<b>Technical Details</b>	<b>2</b>
<b>Exploitation</b>	<b>4</b>
<b>Characteristics</b>	<b>4</b>
<b>Tested Versions</b>	<b>5</b>
<b>Fixed Versions</b>	<b>5</b>
<b>References</b>	<b>5</b>

## Introduction:

=====

A boundary error in Samba can be exploited to cause a heap-based buffer overflow and compromise a vulnerable system.

## Technical Details:

=====

Samba is a popular open source implementation of the SMB (Service Message Block) protocol.

An SMB packet starts with one byte containing the SMB command followed by three bytes containing the length of the SMB data.

When Samba receives an SMB packet with an overly long length in a client context, a heap-based buffer overflow takes place. This can be exploited to execute arbitrary code on an affected system.

The client-side implementation of Samba is included in the "libsmbclient" library. When communication with a server is started, an input buffer is allocated with a size of "CLI\_BUFFER\_SIZE + 4 + SAFETY\_MARGIN" (66563) bytes by the "cli\_initialise()" function.

```
[libsmb/clientgen.c:282]
struct cli_state *cli_initialise(void)
{
...
    cli->bufsize = CLI_BUFFER_SIZE+4;    <-- 65535 + 4
...
    cli->inbuf = (char *)SMB_MALLOC(cli->bufsize+SAFETY_MARGIN);    <-- ... + 1024

#include/client.h:29]
#define CLI_BUFFER_SIZE (0xFFFF)

#include/smb.h:40]
#define SAFETY_MARGIN 1024
```

The client then opens a TCP connection to the SMB server (usually on TCP port 139 or 445). A protocol negotiation follows via a call to "cli\_negprot()". The purpose of "cli\_negprot()" is to decide which version of the SMB protocol is to be used with the selected server. The function sends a "negprot" command and calls "cli\_receive\_smb()" in order to receive the response.

NOTE: "cli\_receive\_smb()" is called throughout the SMB session whenever an SMB packet is expected. The protocol negotiation phase is only given as an example.

```
[libsmb/cliconnect.c:1222]
BOOL cli_negprot(struct cli_state *cli)
...
    SCVAL(cli->outbuf, smb_com, SMBnegprot);
    cli_setup_packet(cli);
...
    cli_send_smb(cli);    <-- send the negprot command
    if (!cli_receive_smb(cli))    <-- receive SMB packet
```

"cli\_receive\_smb()" calls "client\_receive\_smb()" with the client input buffer ("cli->inbuf") passed via the second parameter.

```
[libsmb/clientgen.c:82]
BOOL cli_receive_smb(struct cli_state *cli)
...
    ret = client_receive_smb(cli->fd,cli->inbuf,cli->timeout);
```

"client\_receive\_smb()" is a wrapper for "receive\_smb\_raw()", calling the function with the received parameters unmodified (source code not included). "receive\_smb\_raw()" calls "read\_smb\_length\_return\_keepalive()" in order to extract the length of the SMB packet.

```
[lib/util_sock.c:664]
BOOL receive_smb_raw(int fd, char *buffer, unsigned int timeout)
{
    ssize_t len,ret;
...
    len = read_smb_length_return_keepalive(fd,buffer,timeout);
```

"read\_smb\_length\_return\_keepalive()" reads 4 bytes from the socket and calls the "smb\_len" macro with the buffer as a parameter.

```
[lib/util_sock.c:601]
static ssize_t read_smb_length_return_keepalive(int fd, char *inbuf, unsigned int timeout)
...
    ok = (read_data(fd,inbuf,4) == 4);    <-- read four bytes
...
    len = smb_len(inbuf);                <-- extract the length
```

"smb\_len" extracts the length from the two last bytes and from the last bit of the third byte. This allows a maximum length equal to 0x1FFFF (131071 decimal).

```
[include/smb_macros.h:191]
#define smb_len(buf) ((PVAL(buf,3)|(PVAL(buf,2)<<8)|((PVAL(buf,1)&1)<<16))
```

After extracting the length, "receive\_smb\_raw()" continues its execution by testing if the length goes above "BUFFER\_SIZE + (SAFETY\_MARGIN/2)". If the test yields a positive result, the function returns. Otherwise, the whole SMB packet is read into the input buffer at offset 4 (whole SMB packet == length bytes).

```
[lib/util_sock.c:692]
    if (len > BUFFER_SIZE + (SAFETY_MARGIN/2)) {
...
        <-- error
    }
...
    ret = read_data(fd,buffer+4,len);
```

"SAFETY\_MARGIN" has a constant value of 1024 (as previously noted), resulting in a maximum SMB packet length value of "BUFFER\_SIZE" + 512. "BUFFER\_SIZE" is equal to 128\*1024 if the system on which Samba was compiled supports 64-bit file offsets (i.e. files over 2 GB are supported), and equal to 0xFFFF on other systems. Support for 64-bit offsets is included in all modern operating systems (e.g. current Linux distributions) and it is given by the "LARGE\_SMB\_OFF\_T" define (which internally is set to 1 due to sizeof(off\_t) being 8).

```
[include/smb.h:34]
#if defined(LARGE_SMB_OFF_T)
#define BUFFER_SIZE (128*1024)
#else /* no large readwrite possible */
#define BUFFER_SIZE (0xFFFF)
#endif

[include/includes.h:495]
# if (defined(HAVE_EXPLICIT_LARGEFILE_SUPPORT) && defined(HAVE_OFF64_T)) || (defined(SIZEOF_OFF_T) && (SIZEOF_OFF_T == 8))
#   define LARGE_SMB_OFF_T 1
# endif
```

The imposed limit of "BUFFER\_SIZE" + 512 (equal to 0x20200) is greater than the maximum length of an SMB packet (0x1FFFF), allowing a maximum number of 0x1FFFF (131071 decimal) bytes. This will result in a buffer overflow when the network data is read into the client buffer (having 66559 bytes left after the first four bytes). The overflow will result in a maximum of 64512 bytes on the heap being overwritten with user-controlled data.

#### Exploitation:

=====

Exploiting the vulnerability for code execution is possible in GLIBC versions 2.6 and 2.7 by overwriting vital fields of the next heap chunk. Code execution on other GLIBC versions has not been proven, but cannot be ruled out.

The buffer overflow can be triggered by tricking a user into connecting to a malicious SMB server (e.g. by clicking an "smb://" link).

The vulnerability can also be exploited without user interaction if a Samba server is configured to run as a domain master browser (the "domain master" or "domain logons" and the "wins support" options are enabled in smb.conf). Specifically, an attacker can register itself as a local master browser and trick the Samba server into establishing an SMB session (as a client) to the attacker's machine. Another scenario in which a WINS server used by Samba can be tricked into registering a malicious domain master browser is also viable.

Secunia Research has developed a PoC for the vulnerability. This is available to customers on Secunia Proof of Concept and Exploit Code Services.

#### Characteristics:

=====

#### Detection:

Look for SMB connections established to remote hosts (usually on TCP ports 139 or 445). Received SMB packets with a length over 66559 should be considered malicious.

#### Verification:

Create a server listening on port 445, which sends SMB packets with a length of 0x1FFFF bytes. Connect to the server using smbclient (e.g. "smbclient -L <host>"). A "smbclient" binary coming from a vulnerable Samba package should crash.

## Identification:

Samba versions 3.0.28a and 3.0.29 are confirmed to be vulnerable. Prior versions may also be affected. In order to determine the version of the installed Samba package execute one of the installed Samba binaries with the "--version" option attached (e.g. "smbd --version"). If no binaries are present, construct a program linked against libsmbclient, calling the "samba\_version\_string()" function.

E.g.:

```
void main() { printf("%s", samba_version_string()); }
```

## Tested Versions:

=====

The vulnerability was analysed on Ubuntu 8.04 using Samba 3.0.28a.

## Fixed Versions:

=====

The vendor has released version 3.0.30, which fixes the vulnerability by properly checking the size of incoming SMB packets. A patch for version 3.0.29 is also available.

## References:

=====

SA30228:

<http://secunia.com/advisories/30228/>

CVE-2008-1105:

[http://secunia.com/cve\\_reference/CVE-2008-1105/](http://secunia.com/cve_reference/CVE-2008-1105/)

Secunia Research:

[http://secunia.com/secunia\\_research/2008-20/](http://secunia.com/secunia_research/2008-20/)

Samba:

<http://www.samba.org/samba/security/CVE-2008-1105.html>