Talos Vulnerability Report

TALOS-2020-1187

Genivia gSOAP WS-Addressing plugin code execution vulnerability

JANUARY 5, 2021

CVE NUMBER

CVE-2020-13576

Summary

A code execution vulnerability exists in the WS-Addressing plugin functionality of Genivia gSOAP 2.8.107. A specially crafted SOAP request can lead to remote code execution. An attacker can send an HTTP request to trigger this vulnerability.

Tested Versions

Genivia gSOAP 2.8.107

Product URLs

https://www.genivia.com/products.html#gsoap

CVSSv3 Score

9.8 - CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CWE

CWE-680 - Integer Overflow to Buffer Overflow

Details

The gSOAP toolkit is a C/C++ library for developing XML-based web services. It includes several plugins to support the implementation of SOAP and web service standards. The framework also provides multiple deployment options including modules for both IIS and Apache, standalone CGI scripts and its own standalone HTTP service.

One of the many plugins provided by gSOAP includes the wsa plugin for supporting the WS-Addressing specification which provides an asynchronous mechanism for routing SOAP requests and responses. The specification includes a element for providing URI parameters to a number of different parts of both requests and responses. The URIs may include a username and password for the resource in a standard format. http://user:password@somehost.com A buffer overflow vulnerability existing in the parsing of these extra parameters.

It starts by looking for the `:` in the uri and assumes that if it's not part of `://` that is is the seperator between the username and password.

```
21234   s = strchr(endpoint, ':');
21235   if (s && s[1] == '/' && s[2] == '/') // Skip the :// part of the parameter
21236     s += 3;
21237   else // Assume it's the seperator and start from the beginning of the element
21238     s = endpoint;
```

Processing continues by trying to find the `@` to seperate the user:pass from the hostname. At this point, parsing of the username and password occurs assuming we have found both seperators (`@` and `:`)

```
21240   t = strchr(s, '@'); // attempts to find the seperator
21241
21242   if (t && *s != ':' && *s != '@')
21243   {
21244     size_t l = t - s + 1;  // Calculate the size of the user:pass part of the string
21245     char *r = (char*)soap_malloc(soap, l); // Allocate enough storage to hold both the user and pass
21246     n = s - endpoint;
21247     if (r)
21248     {
21249       s = soap_decode(r, l, s, ":@");  // s is still pointing to the beginning of the data in this element
21250       soap->userid = r; // r is now a copy of the user part of the string up to the :
21251       soap->passwd = SOAP_STR_EOS;
21252       if (*s == ':') // s now points to the : seperator
21253       {
21254         s++; // Step past the seperator and now we should be pointing to the password section
21255         if (*s != '@') // Make sure the password isn't empty
21256         {
21257           l = t - s + 1; // t points to the @ seperator so here we calculate the length of the password by subtracting between the two
seperators
21258           r = r + strlen(r) + 1; // r currently points to the copied username so we skip past to copy the password into the same
buffer.
21259           s = soap_decode(r, l, s, "@"); // l is now a very large number allowing us to write us much data to the head as we like and
cleanly terminate the copy with an @
21260           soap->passwd = r;
21261         }
21262       }
21263     }
21264     s++;
21265     soap_strcpy(soap->endpoint + n, sizeof(soap->endpoint) - n, s);
21266   }
```

Here the code makes an assumption about the order of the `:` and `@` seperators. It assumes that the `@` (`t`) is after `:`(`s`) and calculates the size for the second soap_decode based on this assumption. If the `:` comes after the `@`, this calculation causes the calculated size to be negative and wrap around and become a very large length value to `soap_decode` to parse the password.

Within soap_decode, an attempt to copy the data into a the new heap buffer occurs. As the length is a very large number and the counter is counting backwards, we are able to write an arbitrary amount of data past our destination buffer.

```
7919 soap_decode(char *buf, size_t len, const char *val, const char *sep)
7920 {
7921   const char *s;
7922   char *t = buf;
7923   size_t i = len;
7924   for (s = val; *s; s++)
7925     if (*s != ' ' && *s != '\t' && !strchr(sep, *s))
7926       break;
7927   if (len > 0)
7928   {
7929     if (*s == '"')
7930     {
7931       s++;
7932
7933       while (*s && *s != '"' && --i)
7934         *t++ = *s++;
7935     }
```

Crash Information

```
Starting program: /gsoap-2.8/gsoap/samples/wsa/wsademo 8080
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Server is running
malloc(): memory corruption

Program received signal SIGABRT, Aborted.
__GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:51
51      ../sysdeps/unix/sysv/linux/raise.c: No such file or directory.
(gdb) bt
#0  __GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:51
#1  0x00007ffff70af8b1 in __GI_abort () at abort.c:79
#2  0x00007ffff70f8907 in __libc_message (action=action@entry=do_abort, fmt=fmt@entry=0x7ffff7225dfa "%s\n") at
../sysdeps/posix/libc_fatal.c:181
#3  0x00007ffff70ff97a in malloc_printerr (str=str@entry=0x7ffff722406e "malloc(): memory corruption") at malloc.c:5350
#4  0x00007ffff7103a04 in _int_malloc (av=av@entry=0x7ffff745ac40 <main_arena>, bytes=bytes@entry=72) at malloc.c:3738
#5  0x00007ffff710616c in __GI___libc_malloc (bytes=72) at malloc.c:3057
#6  0x000055555556ddae in soap_malloc (soap=soap@entry=0x7ffff7fba010, n=56, n@entry=48) at stdsoap2_ssl.c:10428
#7  0x000055555556de7f in soap_strdup (soap=soap@entry=0x7ffff7fba010, s=s@entry=0x555555589430
"http://www.w3.org/2005/08/addressing/soap/fault") at stdsoap2_ssl.c:2806
#8  0x000055555557eedd in soap_try_connect_command (soap=soap@entry=0x7ffff7fba010, http_command=http_command@entry=2000,
    endpoint=endpoint@entry=0x55555579db40 "http://%@AAAA:", 'B' <repeats 186 times>..., action=action@entry=0x555555589430
"http://www.w3.org/2005/08/addressing/soap/fault")
    at stdsoap2_ssl.c:21486
#9  0x0000555555555582cdb in soap_connect_command (soap=soap@entry=0x7ffff7fba010, http_command=http_command@entry=2000,
    endpoints=0x55555579db40 "http://%@AAAA:", 'B' <repeats 186 times>..., action=0x555555589430
"http://www.w3.org/2005/08/addressing/soap/fault") at stdsoap2_ssl.c:21455
#10 0x0000555555555582de0 in soap_connect (soap=soap@entry=0x7ffff7fba010, endpoint=<optimized out>, action=<optimized out>) at
stdsoap2_ssl.c:21408
#11 0x0000555555555562ebb in soap_wsa_fault_subcode_action (soap=soap@entry=0x7ffff7fba010, flag=flag@entry=1,
faultsubcode=faultsubcode@entry=0x555555589856 "wsa5:ActionNotSupported",
    faultstring=faultstring@entry=0x555555589d50 "The [action] cannot be processed at the receiver.", faultdetail=faultdetail@entry=0x0,
action=action@entry=0x0)
    at ../../plugin/wsaapi.c:1088
#12 0x0000555555555563145 in soap_wsa_fault_subcode (faultdetail=0x0, faultstring=0x555555589d50 "The [action] cannot be processed at the
receiver.",
    faultsubcode=0x555555589856 "wsa5:ActionNotSupported", flag=1, soap=0x7ffff7fba010) at ../../plugin/wsaapi.c:1022
#13 soap_wsa_sender_fault_subcode (faultdetail=0x0, faultstring=0x555555589d50 "The [action] cannot be processed at the receiver.",
faultsubcode=0x555555589856 "wsa5:ActionNotSupported",
    soap=0x7ffff7fba010) at ../../plugin/wsaapi.c:1126
#14 soap_wsa_error (soap=soap@entry=0x7ffff7fba010, fault=fault@entry=wsa5__ActionNotSupported, info=0x0) at ../../plugin/wsaapi.c:1428
#15 0x0000555555555563a7d in soap_wsa_set_error (soap=0x7ffff7fba010, c=0x55555579bbf0, s=0x55555579bc20) at ../../plugin/wsaapi.c:1623
#16 0x000055555555558535f in soap_set_fault (soap=soap@entry=0x7ffff7fba010) at stdsoap2_ssl.c:22051
#17 0x000055555555561f3 in soap_send_fault (soap=0x7ffff7fba010) at stdsoap2_ssl.c:22314
#18 0x000055555555588f3 in main (argc=2, argv=<optimized out>) at wsademo.c:85
```

Timeline

2020-11-05 - Vendor Disclosure

2020-12-16 - Vendor advised patch released on 2020-11-20

2021-01-05 - Public Release

CREDIT

Discovered by a member of Cisco Talos.
