2016 年 6 月 8 日，星期三

漏洞聚焦：ESnet iPerf3 JSON parse_string UTF 代码执行漏洞

此漏洞的发现者为高级研究工程师 Dave McDaniel。

概要

iPerf 是一款网络测试应用，通常部署在客户端/服务器配置中，通过创建 TCP 和/或 UDP 连接，来测量系统间的可用网络带宽。对于每个连接，iPerf 可以报告最大带宽、损耗和其他性能相关指标。它通常用于评估和量化网络优化的效果，以及获取与网络性能有关的基准指标。

iPerf3 由 ESnet 和劳伦斯伯克利国家实验室共同开发。开发人员利用派生的 cJSON 库，在原始 iPerf 应用的基础上进行了全面重新设计。思科 Talos 最近发现派生版本的 cJSON 库存在一个漏洞，可能导致在运行 iPerf3 服务器守护程序的系统上实施远程代码执行 (RCE)。此漏洞与派生的 cJSON 库解析 UTF-8/16 字符串的方式有关。当前有多种公共 iPerf3 服务器可以通过互联网进行访问，都有可能会受到此漏洞远程攻击的影响。尽管底层 cJSON 库的创建者之前已发布解决此漏洞的修补程序，但 iPerf3 3.1-1 附带的 cJSON 版本仍存在漏洞。iPerf3 应用的更新版本可在此处获取。

详细信息

在处理 Unicode 字符串时，JSON 规范要求使用“\u”令牌，后接四个十六进制数字。iPerf3 中的漏洞位于内置的 cJSON 库中，漏洞原因与处理 UTF-8/16 字符串时 parse_string() 函数处理内存分配的方式有关。接收到 UTF-8/16 字符串后，此函数会尝试评估字符串，根据接收的输入大小，确定适当的堆分配。

当遇到包含 ‘ “ ‘ 字符的 UTF-8/16 字符串时，sscanf() 函数会尝试读取该字符串。由于所需格式的原因，当遇到第一个 ‘ “ ‘ 时，sscanf() 会立即停止读取字符串，但是缓冲区指针会以四个字节的增量自动增加，无论提供的字符串的实际长度是多少。这会产生逻辑错误，导致持续向保留的块写入数据，直到达到循环结束为止。攻击者可以通过提供一个足够长的字符串来制造堆溢出条件，导致堆数据块报头覆盖。这也有可能会导致远程代码执行。

```
236 static const char *parse_string( cJSON *item, const char *str )
237 {
[...snip...]
250     /* Skip escaped quotes. */
251     while ( *ptr != '\"' && *ptr && ++len ) // extract key from 'ptr'
252         if ( *ptr++ == '\\' )
253             ptr++;
254
255     if ( ! ( out = (char*) cJSON_malloc( len + 1 ) ) ) // Malloc Block Size of 0x3 (
actually              reserves 0x20 bytes)
256         return 0;
257
258     ptr = str + 1;  // move ptr past the first quote of the key ("\uC" in this case)
259     ptr2 = out;     // ptr2 will store the resulting cstring
260     while ( *ptr != '\"' && *ptr ) { // until the second quote or backslash is found
261         if ( *ptr != '\\' )         // the slash in \uC is found
262             *ptr2++ = *ptr++;       // ptr2 will store the unescaped contents of ptr overflowing
the      block allocated for only 3 bytes
263         else {
264             ptr++;                  // skip the slash found and check the escaped char
265             switch ( *ptr ) {
266                 case 'b': *ptr2++ ='\b'; break;
267                 case 'f': *ptr2++ ='\f'; break;
268                 case 'n': *ptr2++ ='\n'; break;
269                 case 'r': *ptr2++ ='\r'; break;
270                 case 't': *ptr2++ ='\t'; break;
271                 case 'u':                       // UTF8/16 found
272                     /* Transcode utf16 to utf8. */
273                     /* Get the unicode char. */
274                     sscanf( ptr + 1,"%4x", &uc ); // sscanf stops at non-hex char '"'
275                     ptr += 4;                     // ptr inc'd by 4 regardless of sscanf result,
skipping the first quote and continuing to read in the data beyond the bounds of the heap structure
allocated to store it.
```

然后，程序会对照密钥列表进行对象比对，如果未找到匹配项，则删除对象，以释放此字符串。在尝试删除对象时，程序会读取内存中损坏的数据块，然后释放其空间。

```
3833    static void
3834    _int_free (mstate av, mchunkptr p, int have_lock)
3835    {
3836      INTERNAL_SIZE_T size;        /* its size */
3837      mfastbinptr *fb;             /* associated fastbin */
3838   mchunkptr nextchunk;          /* next contiguous chunk */
3839   INTERNAL_SIZE_T nextsize;     /* its size */
3840   int nextinuse;                /* true if nextchunk is used */
3841   INTERNAL_SIZE_T prevsize;     /* size of previous contiguous chunk */
3842   mchunkptr bck;                /* misc temp for linking */
3843   mchunkptr fwd;                /* misc temp for linking */
3844
3845   const char *errstr = NULL;
3846   int locked = 0;
3847
3848   size = chunksize (p);
[...]
3960     nextchunk = chunk_at_offset(p, size);

:: in __GI___libc_free (malloc.c:3848)
3848   size = chunksize (p);

>>> p/x size
$70 = 0x20

>>> p/x *p
$71 = {
prev_size = 0x0,
size = 0x21,
fd = 0x454545454545440c,
bk = 0x4646464646464545,
fd_nextsize = 0x4747474747474646,
bk_nextsize = 0x7d4747
}
```

随后，程序会尝试向后合并，如果遇到已损坏的堆数据块报头，则会导致程序崩溃。

```
686       if (!prev_inuse (p)) /* consolidate backward */
687         {
688           p = prev_chunk (p);
689           unlink (ar_ptr, p, bck, fwd);
690         }
```

测试版本

Talos 已对以下版本进行测试：

iperf 3.1.1 2015-10-16
iperf3 3.1-1 2015-11-02

结论

cJSON 库已经过官方更新，解决了此漏洞。cJSON 库的用户可以从官方 cJSON Github 存储库获取最新版本，最新版本包含若干与 cJSON 已发现的安全漏洞有关的修补程序。使用 iPerf3 应用的用户应尽快更新到 iPerf3 的最新版本，以解决此漏洞。

TALOS-2016-0164 是通过 SID 39165 检测到的。

有关此漏洞的完整详细信息，请参阅此处的公告。
发布者：Edmund Brumaghin；发布时间：下午 4:07
标签：cve-2016-4303、esnet、补丁、漏洞、漏洞聚焦