IOActive Security Advisory

Title	Heap corruption in Tor when escaping input for logging
Severity	Critical (for the affected platforms)
Date Discovered	January 2009
Date Reported	January 20, 2009
Date Disclosed	June 8, 2009
Author	Ilja van Sprundel

Affected Products

Tor 0.2.0.32 and 0.2.1.10-alpha were verified; however, earlier versions are also likely to be vulnerable.

Description

There is a potential heap corruption bug in Tor when escaping data for logging purposes. Due to the nature of this bug, only certain deployments of Tor are vulnerable and the bug can only be triggered from certain locales.

Technical Details

```
Src/common/util.c
/** Allocate and return a new string representing the contents of
 *  <b>s</b>,
 * surrounded by quotes and using standard C escapes.
 *
 * Generally, we use this for logging values that come in over
 * the network to
 * keep them from tricking users, and for sending certain values
 * to the
 * controller.
 *
 * We trust values from the resolver, OS, configuration file, and
 * command line
 * to not be maliciously ill-formed.  We validate incoming
 * routerdescs and
 * SOCKS requests and addresses from BEGIN cells as they're
 * parsed;
 * afterwards, we trust them as non-malicious.
 */
char *
esc_for_log(const char *s)
{
  const char *cp;
  char *result, *outp;
  size_t len = 3;
```

```
if (!s) {
  return tor_strdup("");
}

for (cp = s; *cp; ++cp) {
  switch (*cp) {
    case '\\':
    case '\"':
    case '\'':
      len += 2;
      break;
    default:
      if (TOR_ISPRINT(*cp) && ((uint8_t)*cp)<127)
        ++len;
      else
        len += 4;
      break;
  }
}

result = outp = tor_malloc(len);
*outp++ = '\"';
for (cp = s; *cp; ++cp) {
  switch (*cp) {
    case '\\':
    case '\"':
    case '\'':
      *outp++ = '\\';
      *outp++ = *cp;
      break;
    case '\n':
      *outp++ = '\\';
      *outp++ = 'n';
      break;
    case '\t':
      *outp++ = '\\';
      *outp++ = 't';
      break;
    case '\r':
      *outp++ = '\\';
      *outp++ = 'r';
      break;
    default:
      if (TOR_ISPRINT(*cp) && ((uint8_t)*cp)<127) {
        *outp++ = *cp;
      } else {
        tor_snprintf(outp, 5, "\\%03o", (int)(uint8_t) *cp);
        outp += 4;
      }
      break;
  }
}

*outp++ = '\"';
*outp++ = 0;
```

```
  return result;
}
...
Src/common/compat.h
#define TOR_ISPRINT(c)   isprint((int)(unsigned char)(c))
```

The function `esc_for_log()` is used to escape logging data that may potentially be tainted, and it contains two loops, the first of which scans input `string(s)` and calculates how many bytes are needed to escape a specific character. After the first loop, the calculated number of bytes is allocated and the second loop reviews the `string(s)`, escaping them if necessary, and then copying them into the allocated buffer.

There are three cases in which a heap overflow can occur in the second loop: `\r`, `\n`, and `\t`. This occurs because these cases are not explicitly handled in the first loop, so the code falls to the default case, which looks like:

```
      default:
        if (TOR_ISPRINT(*cp) && ((uint8_t)*cp)<127)
          ++len;
        else
          len += 4;
        break;
```

The second loop assumes that the else case will always be true for `\r`, `\n`, and `\t`, which is not, in fact, true. If `TOR_ISPRINT`—which is equivalent to `isprint()`—declares something printable, heap corruption can occur. Contrary to popular belief, `isprint()` does not contain a static table of printable characters and, instead, refers to the current locale, which may believe that `\r`, `\n`, or `\t` are printable. Whether this occurs depends heavily on the operating system and the specific locale. For example, most Windows locales (though not C or POSIX) would identify `\t` as printable.

Remediation

Versions 0.2.0.33 and 0.2.1.11-alpha include a solution that involves not using the `is*()` functions provided by the operating system; they implement a customized `TOR_IS*()` function instead.

The full patch can be obtained from

<https://svn.torproject.org/cgi-bin/viewcvs.cgi?rev=18189&view=rev>