Universität des Saarlandes
Lehrstuhl für Softwaretechnik
Prof. Andreas Zeller, Stephan Neuhaus
Design of Secure Software Systems
Exercise 9

Total number of points obtainable: 9, plus 6 bonus points.

1 In-Band Error Signalling

The C function *atoi*(3) converts a string to an integer. The prototype for *atoi*(3) is

```
#include <stdlib.h>

extern int atoi(const char *nptr);
extern long int strtol(const char *nptr, char **endptr, int base);
```

The manpages for *atoi*(3) and *strtol*(3) have this to say:

> "The *atoi*() function converts the initial portion of the string pointed to by *nptr* to **int**. The behaviour is the same as *strtol*(nptr, (**char**$**$)*NULL*, 10); [...] The *strtol*() function returns the result of the conversion, unless the value would underflow or overflow. If an underflow occurs, *strtol*() returns *LONG_MIN*. If an overflow occurs, *strtol*() returns *LONG_MAX*. In both cases, *errno* is set to *ERANGE*."

- **(3 points)** Assuming that your C library is conforming to this manpage, how can you reliably distinguish between an underflow and an input that just happens to have the same value as *LONG_MIN*? Please note that no standard-conforming library function will set *errno* if it succeeds.
- **(3 points)** What is returned for the string "`abcd`" on your system?
- **(3 points)** How can you distinguish between a string that contains no numeric characters and the string "`0`"?

2 Validation and Canonicalization

Warning: difficult and time-consuming exercise ahead! (6 bonus points) Take a look at RFC 1738 at `http://www.faqs.org/rfcs/rfc1738.html` and write a C function to canonicalize URLs for the "http" scheme. Reject invalid URLs.