SC3010
Computer Security

Lecture 4: Software Security (III)

Outline

- **Safe Programing**
- **Software Testing**
- **Compiler and System Support**

Outline

- **Safe Programing**
- **Software Testing**
- **Compiler and System Support**

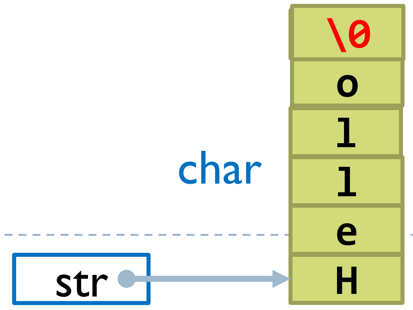
Safe Functions

Root cause: unsafe C lib functions have no range checking

- `strcpy (char *dest, char *src)`
- `strcat (char *dest, char *src)`
- `gets (char *s)`

- Use "safe" versions of libraries:
  - `strncpy (char *dest, char *src, int n)`
    - Copy *n* characters from string src to dest
    - Do not automatically add the NULL value to dest if *n* is less than the length of string src. So it is safer to always add NULL after `strncpy`.
  - `strncat (char *dest, char *src, int n)`
  - `fgets(char *BUF, int N, FILE *FP);`
  - Still need to get the byte count right.

```
char str[6];
strncpy(str, "Hello, World", 5);
str[5] = '\0';
```



Assessment of C Library Functions

Extreme risk

- `gets`

High risk

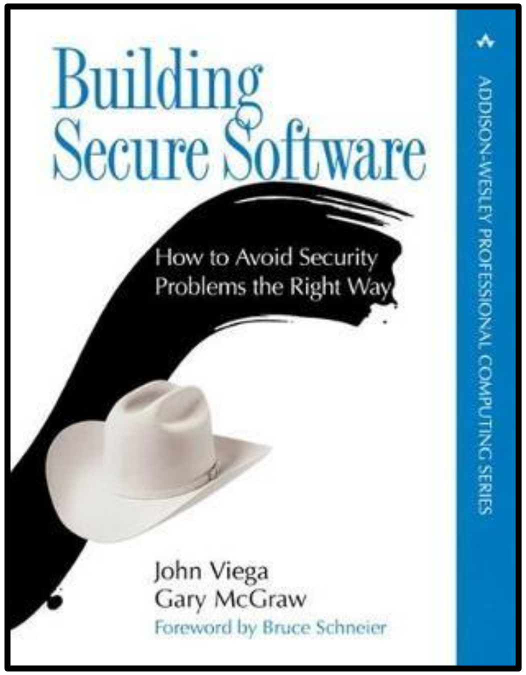
- `strcpy, strcat, sprintf, scanf, sscanf, fscanf, vfscanf, vsscanf, streadd, strecpy, strtrns, realpath, syslog, getenv, getopt, getopt_long, getpass`

Moderate risk

- `getchar, fgetc, getc, read, bcopy`

Low risk

- `fgets, memcpy, snprintf, strccpy, strcadd, strncpy, strncat, vsnprintf`


Building Secure Software
How to Avoid Security Problems the Right Way
John Viega
Gary McGraw
Foreword by Bruce Schneier
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES