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

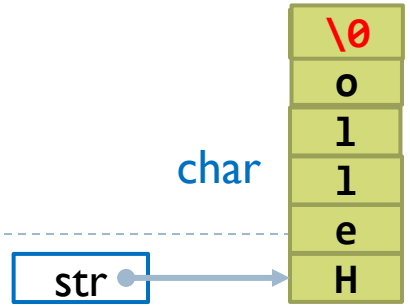
Safe Functions

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

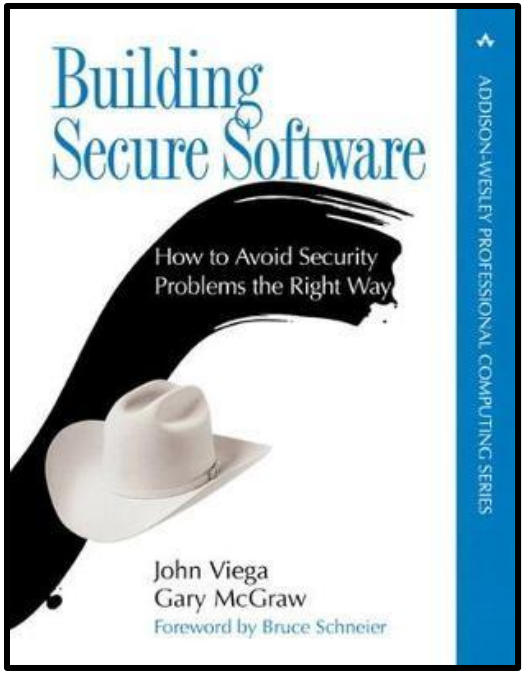
- `strcpy, strcat, sprintf, scanf, sscanf, fscanf, vfscanf, vsscanf, streadd, strecpy, strtrns, realpath, syslog, getenv, getopt, getopt_long, getpass`

Moderate risk

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

Low risk

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



Safe Libraries

libsafe

- Check some common traditional C functions
  - Examines current stack & frame pointers
  - Denies attempts to write data to stack that overwrite return address or any parameters

glib.h

- Provides Gstring type for dynamically growing null-terminated strings in C

Strsafe.h

- A new set of string-handling functions for C and C++.
- Guarantees null-termination and always takes destination size as argument

SafeStr

- Provides a new, high-level data type for strings, tracks accounting info for strings; Performs many other operations.

Glib

- Resizable & bounded

Apache portable runtime (APR)

- Resizable & bounded

Safe Language (Strong Type)

Ada, Perl, Python, Java, C#, and even Visual Basic

- Have automatic bounds checking, and do not have direct memory access

C-derivatives: Rust (Mozilla 2010)

- Designed to be a "safe, concurrent, practical language", supporting functional and imperative-procedural paradigms
- Does not permit null pointers, dangling pointers, or data races
- Memory and other resources are managed through "Resource Acquisition Is Initialization" (RAII).

Go: type-safe, garbage-collected but C-looking language

- Good concurrency model for taking advantage of multicore machines
- Appropriate for implementing server architectures.

Outline

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