Defensive Programming

Secure Programming

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Secure Programming

- The subset of defensive programming concerned with computer security
- the motivation is not as much to reduce the likelihood of failure in normal operation (as if safety were the concern), but to reduce the attack surface
- The programmer must assume that the software might be misused actively to reveal bugs, and that bugs could be exploited maliciously

Example: Buffer Overflow

Risky Programming

```
int risky_func(char *input) {
  char str[1000];
  // ...
  strcpy(str, input);  // Copy input.
  // ...
}
```

Secure Programming

```
int risky_func(char *input) {
  char str[1000];
  // ...
  input[sizeof(str) - 1] = '\0';
  strcpy(str, input);  // Copy input.
  // ...
}
```

Example: Integer Overflow

Risky Programming

```
#define MAX 65535

bool sumIsValid_flawed(unsigned short int x, unsigned short int y) {
  unsigned short int sum = x + y;
  cout << sum << endl;
  return sum <= MAX;
}
```

Secure Programming

```
#define MAX 65535

bool sumIsValid_flawed(unsigned short int x, unsigned short int y) {
   unsigned short int sum = x + y;
   cout << sum << endl;
   return sum >= x && sum >= y && sum <= MAX;
}
```

Defensive Programming

Secure Programming

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur