University of BRISTOL

Computer System B -Security

Introduction to Software Vulnerabilities Part 1

Sanjay Rawat

Outline

- Memory corruption bugs
  - Memory Safety
- Examples
  - Buffer Overflows
  - Integer Overflow
  - Format String bug

Memory Corruption bugs

- An example of programming errors
- *Property* of specific type of languages, called memory unsafe languages.
- One of the most used bugs for exploitation

Programming bugs vs Design flaws

- Intended behavior
  - Functional
  - Security policy/objectives

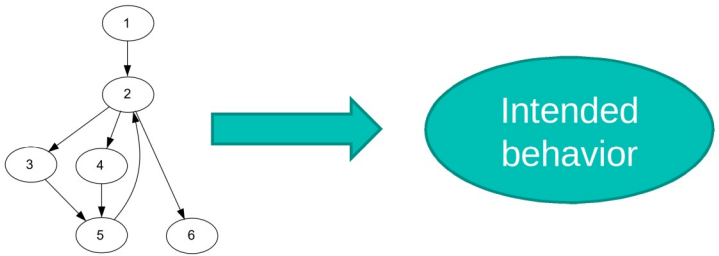
Programming bugs vs Design flaws

- Intended behavior
  - Functional
  - Security policy/objectives

Programming bugs vs Design flaws

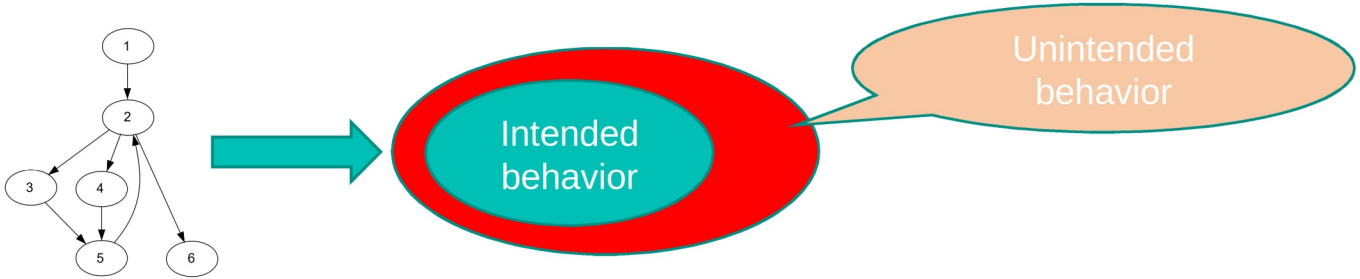
- Intended behavior
  - Functional
  - Security policy/objectives

Security feature (crypto)



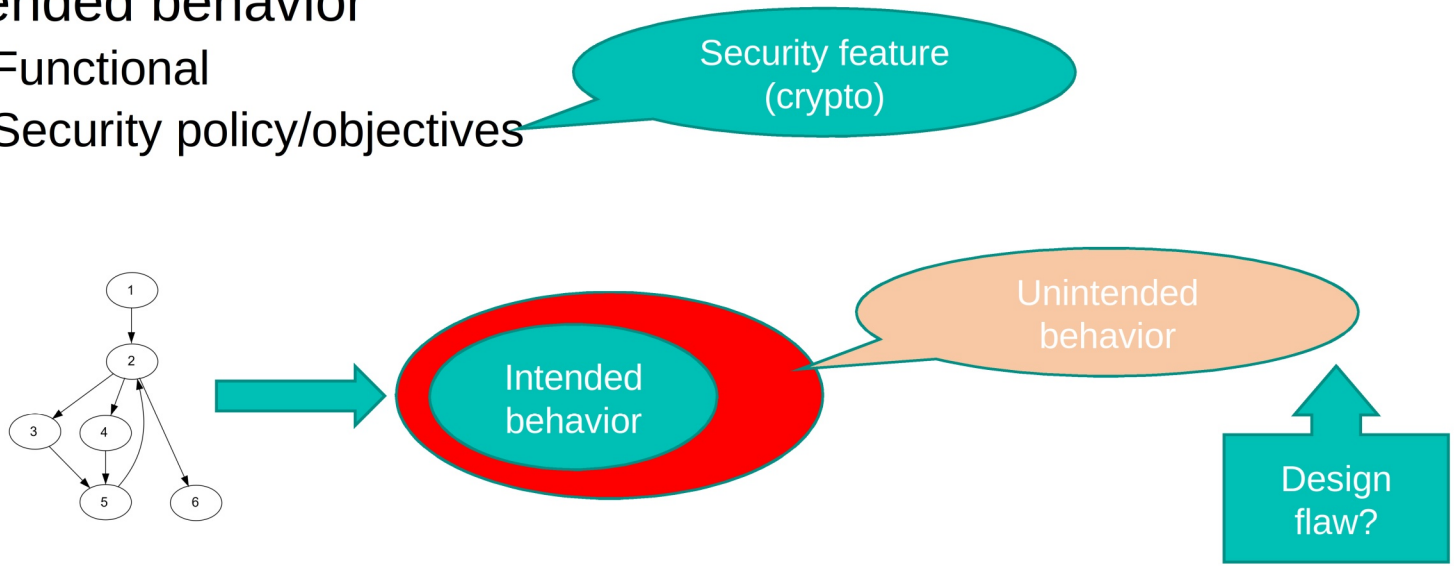
Programming bugs vs Design flaws

- Intended behavior
  - Functional
  - Security policy/objectives



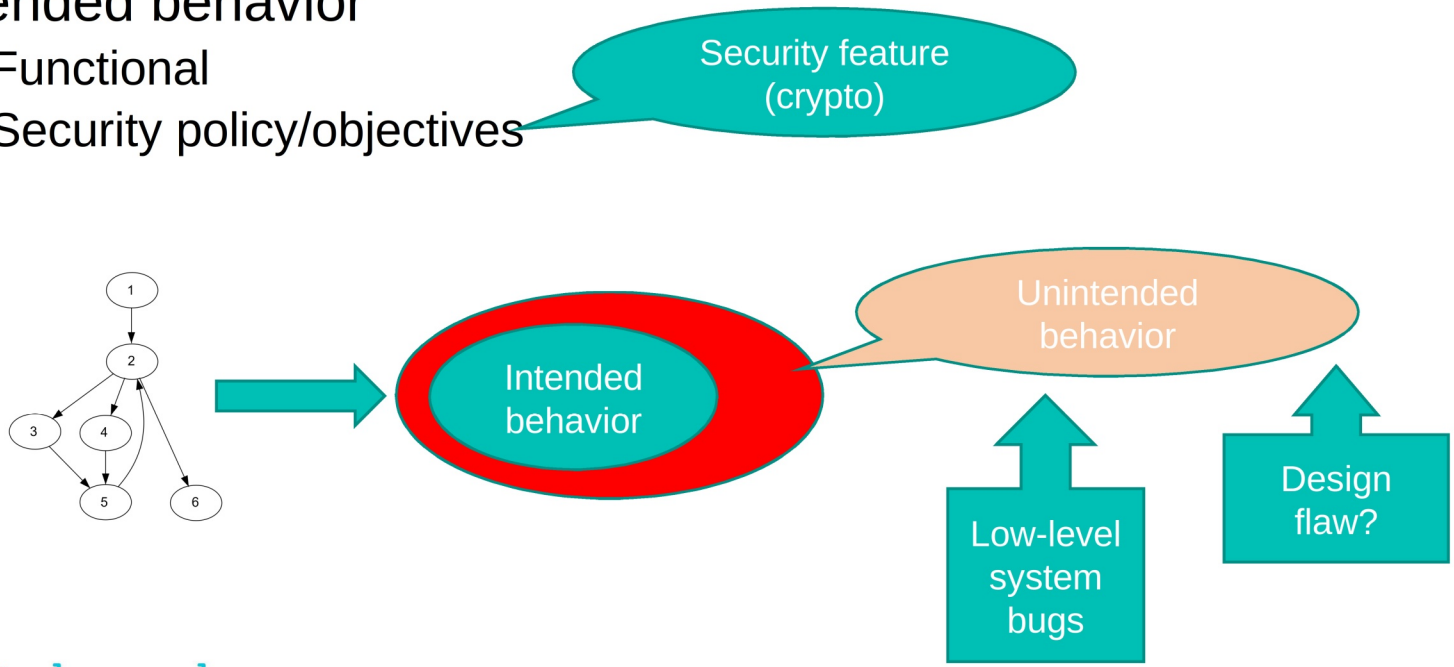
Programming bugs vs Design flaws

- Intended behavior
  - Functional
  - Security policy/objectives



Programming bugs vs Design flaws

- Intended behavior
  - Functional
  - Security policy/objectives



Design Flaws

- A design flaw is a weakness in a software program’s architecture/design.
- Is fundamentally *language-independent.*
- Ex. insecure use of cryptography, absence of checking for authentication etc.

Programming errors

- Error introduced based on the semantics of a particular language.
- For our purpose, we focus on memory safety properties of a language.
  - Spatial and temporal safety.

Memory Safety

- When object boundary access is violated via pointers.
- Spatial Safety[1]:
  - A spatial safety violation is an error in which a pointer is used to access the data at a location in memory that is outside the bounds of an allocated object.
  - The error is ‘spatial’ in the sense that the dereferenced pointer refers to an incorrect location in memory.

[1]: MemSafe: ensuring the spatial and temporal memory safety of C at runtime

example

```
int x;
int buf[4];
buf[5] = 3; /* overwrites x */
```

example

```
int x;
int buf[4];
buf[5] = 3; /* overwrites x */
```

Your C specification does not

example

```
int x;
int buf[4];
buf[5] = 3; /* overwrites x */
```

Your C specification does not complain about this behavior!

Memory Safety conti..

- Temporal Safety[1]
  - A temporal safety violation is an error in which a pointer is used in an attempt to access or deallocate an object that has already been deallocated.
  - The violation is ‘temporal’ in the sense that the pointer use occurs at an invalid instance during the execution of the program (i.e., after the object to which it refers has been deallocated).

[1]: MemSafe: ensuring the spatial and temporal memory safety of C at runtime

Example

```
int *p, *q;
p = (int )malloc(100*sizeof(int));
q = p;
….
free(p);
…
.. *q ..//dangling pointer
```

C and C++

- C and C++:
  - Popular programming languages.
  - Many vulnerabilities that have been reported to the CERT/CC have occurred in programs written in one of these two languages.
  - By design, these are *memory unsafe language.*

What is the Problem with C?

- C does not protect programmers.
  - Problems arise from an imprecise understanding of the semantics of logical abstractions and how they translate into machine-level instructions.

What is the Problem with C?

- Programmer errors
  - Failing to prevent writing beyond the boundaries of an array,
  - Failing to catch integer overflows and truncations,
- Lack of *type safety.*
  - Operations can legally act on signed and unsigned integers of differing lengths using implicit conversions and producing unrepresentable results.

What is the Problem with C?

- Short term solutions:
  - Educating developers in how to program securely by recognizing common security flaws and applying appropriate mitigations.
- Long term solutions:
  - Language standard, compilers, and tools evolve.

Legacy Code

- A significant amount of legacy C code was created (and passed on) before the standardization of the language.
- Legacy C code is at higher risk for security flaws because of the looser compiler standards and is harder to secure because of the resulting coding style.