

0. heap overflow attack

Stack overflow and heap buffer overflow

1. suppose the software has these bugs. How to prevent exploit?

(1) non-executable page NX bit

+ NX mechanism helps prevent executing code from stack.

- It does not prevent executing existing code (e.g., glibc functions) with user/malicious input

(2) address space layout randomization

Makes it difficult to guess the location of library functions

(3) ways to catch stack overflow

Using 'canary bit'. Change compiler to insert padding between return address location and local variable. The padding will be checked at return.

2. how to detect these bugs from software?

2.1 static analysis

2.2 dynamic analysis: mark certain memory location as non-writable and monitor accesses to these locations

(1) stack overflow: the location that holds return address

(2) heap overflow: Q: which part is non-writable?

Q: do you have other ways to detect heap overflow?

Q: how to conduct the monitoring? Any fast way of monitoring?

Bugs as deviant behavior

1. checker: metal compiler

a tool to enable you write compiler extension

write state transition graph, error reporting condition, etc.

certain type of statement triggers certain state transition; certain state transition triggers error reporting

e.g. transition: lock(x) → x is locked; unlock(x) → x is unlocked

rule: x cannot be unlocked from normal state; x cannot be locked from locked state

the question is how do we know the `rule`?

2. how to get the rule?

Infer from program; backward AND forward propagation

Certain action → certain inference

(1) MUST belief: (find one evidence)

About pointer: if you check ptr, ptr is unknown before and ...; if you assign ...

About user pointer

About IS_ERR

(2) May belief: use statistical (find common pattern)

Template: NO <A> after (no use argument after function call) [expert knowledge]

Template: must follow <a>

Does f's return value need to be checked? (why this is different from IS_ERR)

3. latent specification and other manual labor

BUG, panic

Lock, ioctl, alloc, free, knowledge

List all files that do `paranoid` (copyout, copyin)

4. about static analysis: summary

Follow up:

1. what are other places where you can get information?

Document

Comments

2. MAY belief template still quite limited

PR-Miner

Security check (SELinux)

3. false positive (this IS an issue)

The authors moved on to testing; KLEE