CS3235 Part2 Lec3

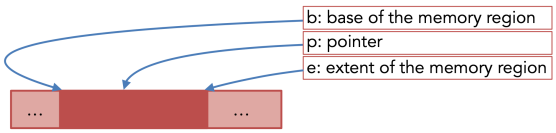
$_/|/|U_Ch@NgRu!$

December 16, 2021

1 Spatial safety

Memory access should be limited to regions that belong to an object
Access is allowed if and only if

$b \leq p \leq e$- sizeof(typeof(p))



1.1 Partial safety check

We check if it's safe to access address p using check+safety()

- check_safety(base,pointer, extent,size)

```
void check_safety(b,p, e,s){
    if ((p<b)||(p+s>e)){
        abort();
    }
}
```

  - here p+s may result in integer overflow

  Therefore, a correct form can be

```
void check_safety(b,p,e,s){
    if ((p<b) ||(p+s>e)||(p+s)<p){
        abort();
    }
```

```
}
```

2 Temporal safety

Undefined memory:

- unallocated
- uninitialized
- deallocated

  e.g. dereferencing a freed pointer(dangling pointer)

```
int * p = malloc(sizeof(int)*n);
free(p);
printf(*p);
```

  e.g. Uninitialized pointer

```
int *p;
*p = 1;
```

3 Control flow integrity

Control flow integrity states that a program's behavior should be normal(as expected)

3.1 Control flow graph

In computer science, a control-flow graph (CFG) is a representation, using graph notation, of all paths that might be traversed through a program during its execution;
Control flow integrity requires to model the expected behavior of the model as its CFG

3.2 Inline reference monitor

Instructions inserted into the program to check whether or not the program behaves as expected

1. Build a list of possible call/return targets(from the binary or during compilation)
2. Monitor the control flow of the program. Before call, check its validity to ensure that it follows CFG

- Direct call(with constant target): need not be monitored, as the target address cannot be changed(why? RTBD)
- Indirect call(should focus on)(e.g. call, ret, jmp)

3. Insert label just before the target address of an indirect call
4. Insert code to check the label of the target, and abort if the label(does not match the CFG)

```
sort2(int a[], int b[], int len)
{
    sort(a, len, lt);
    sort(b, len, gt);
}
```

```
bool lt (int x, int y) {
    return x < y;
}
bool gt (int x, int y) {
    return x > y;
}
```

```
sort2():
...
call sort
...
call sort
...
ret ...
```

```
sort():
...
call ... R
    ...
ret ...
```

```
lt():
...
ret ...
```

```
gt():
...
ret ...
```

RTBD

4 Multiple stages of attack

1. Vulnerability in a system is discovered
2. vulnerability is exploited to gain access to the system
3. Attacker gets foothold on the system by escalating privileges, installing backdoor etc.
    - Rootkit help an attacker to gain a stronger foothold on the system
4. Attack use system access to steal information or conduct other attack
5. Compromise is detected and incident response is executed

- Rootkit prevents or delay the detection of attack by hidening attcker's resource on the system

  The attacker installs a rootkit which is a set of programs and code that allows a permanent or consistent, undetectable presence on a computer.

4.1 Rootkit

A rootkit is a clandestine computer program designed to provide continued privileged access to a computer while actively hiding its presence. The purpose of a rootkit is NOT to gain access to a system, but to preserve existing access. An exploit must be used to gain access to the host before a rootkit can be deployed.
Goals: Hide malicious resources (e.g., processes, files, registry keys, open ports, etc.), and provide hidden backdoor access
Different types:

- Applications
- libraries
- kernel
- hardware

5 Designing Secure Systems

- Principle of least privilege
- Privilege separation and isolation(compartment)
- Defense in depth
  - Use more than one security mechanism
  - Secure the weakest link
  - Fail securely
- Simple systems
- Access control

5.1 Principle of least privilege

5.1.1 Definition of privilege

Ability to access or modify a resource

5.1.2 Definition of principle of least privilege

A system module should only have the minimal privilege needed for its intended purposes

5.2 Privilege separation and isolation (compartmentalization)

- Separate the system into isolated compartments
- Limit interaction between compartments

5.3 Defense in depth

5.4 Simple systems

5.5 Access Control

6 Example of designing secure system: Email Agent

RTBD

7 Access Control

- Authentication, e.g. via username and password
- Access reaquests pass through reference monitor(gatekeeper)
- System must not allow monitor to be bypassed

7.1 Access control matrix

RTBD

7.2 Implementation Concepts

7.2.1 Access control list (ACL)

Store column of matrix with the resource

7.2.2 Capability

- User holds a "ticket" for each resource
- Two variations
  - store row of matrix with user, under OS control
  - unforgeable ticket in user space