CMSC388N:

Build It, Break It, Fix It: Competing to Secure Software

Lecture 8 - Wrap Up

Prof. Daniel Votipka

Winter 2020

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

The Plan

- Breaks found
- Secure development reflection
- Security best practices
- In-class survey time!

Break Review

- Lots of crashes!
- Incorrect implementation of spec
- Security properties not specifically stated
- Non-logic problems

Incorrect Implementation

- Admin password not checked
- Exit and set default delegator by non-admin
- Default delegator rights provided after creation
- Too many rights for the hub (e.g., read)
- Incorrect delegation checks
- No rollback on failure
- Rule permission checking

Incorrect Implementation

- Admin password not checked
- Exit and set default delegator by non-admin
- Default delegator rights provided after creation
- Too many rights for the hub (e.g., read)
- Incorrect delegation checks
- No rollback on failure
- Rule permission checking

Who do you think this is?

Un(der)specified

- Anyone login
- Leak of variable type information
- Circular delegation
- Multiple delegation paths

Circular Delegation

admin

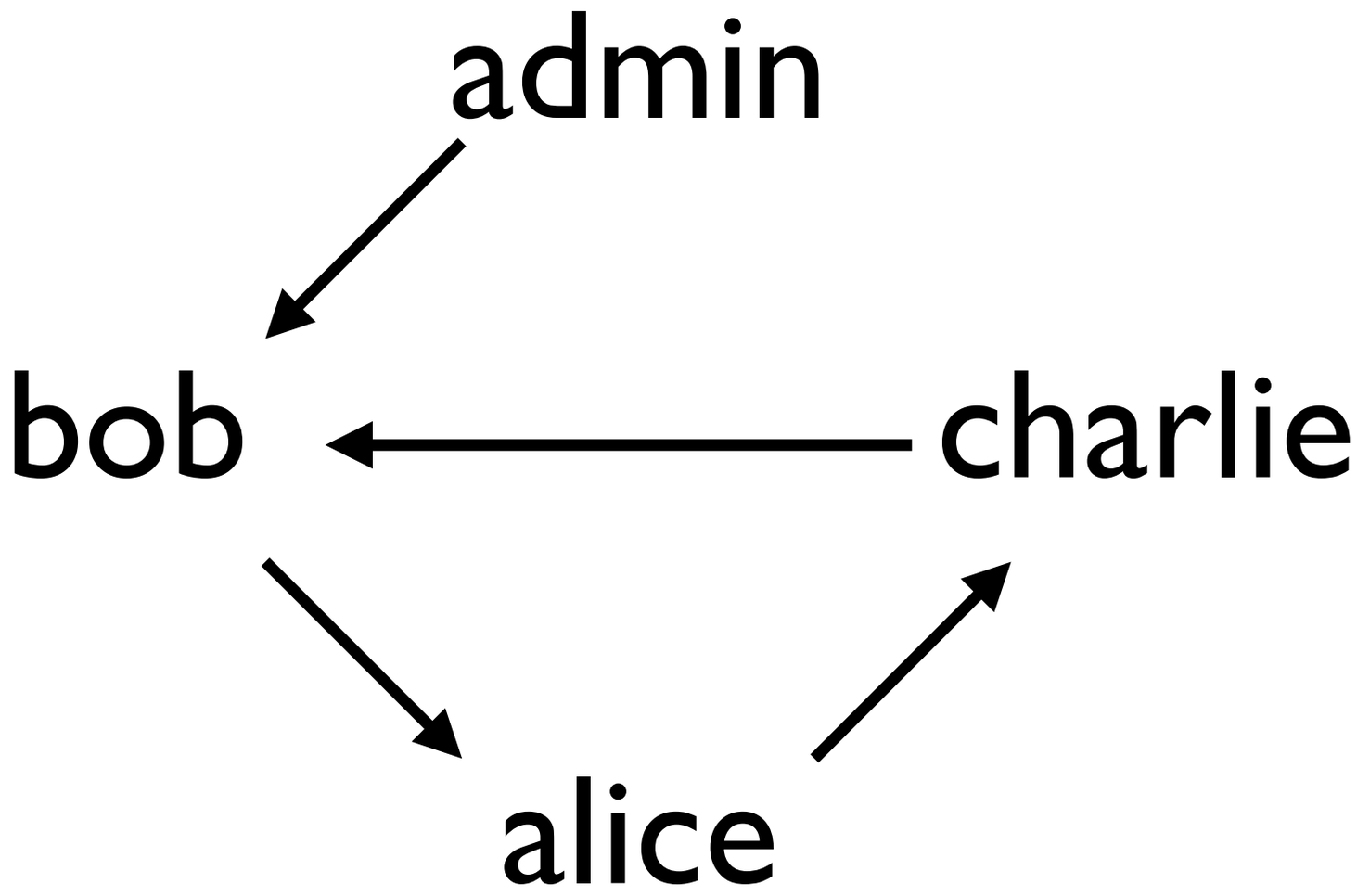
Circular Delegation

admin

bob charlie

alice

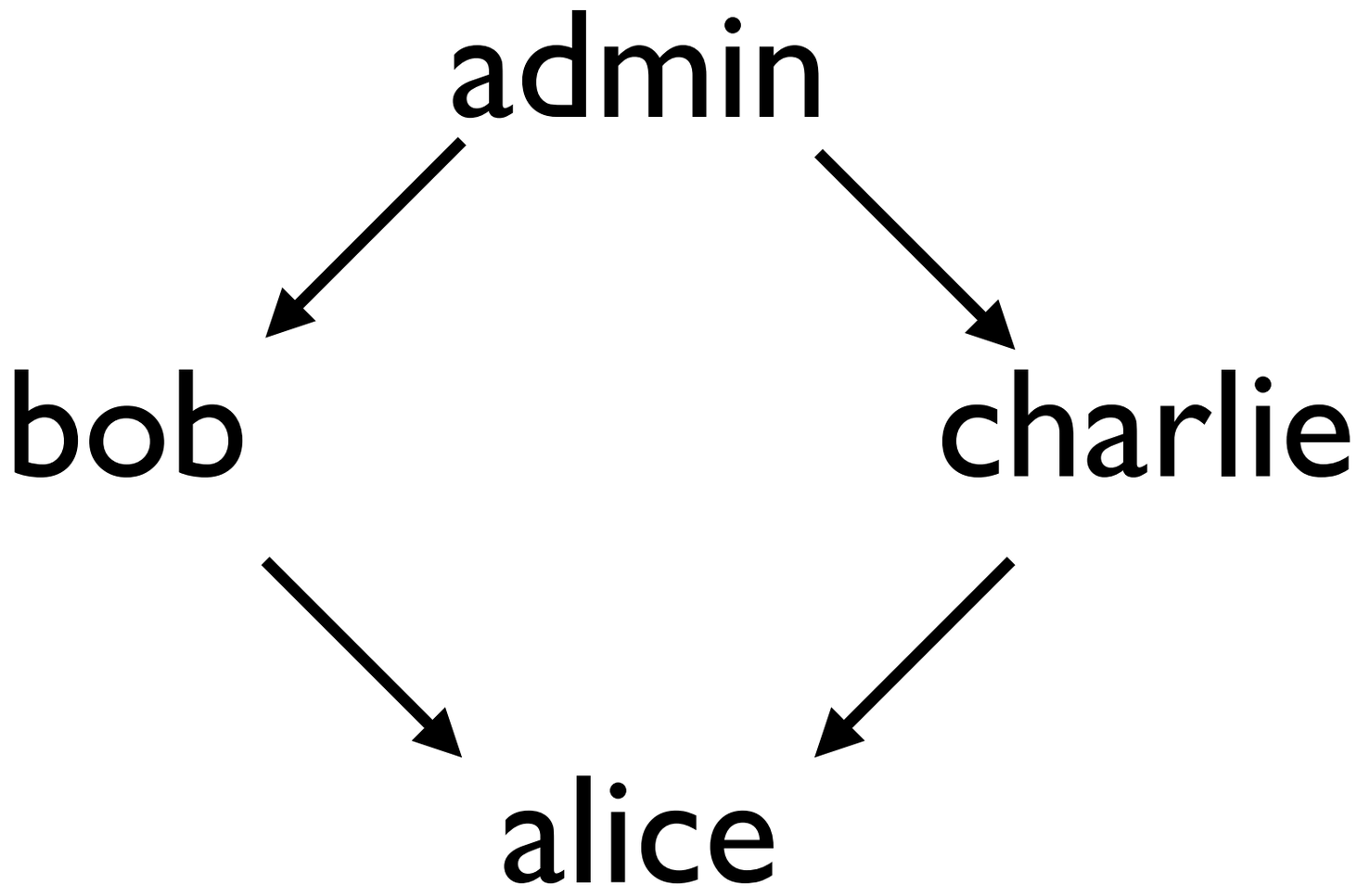
Circular Delegation



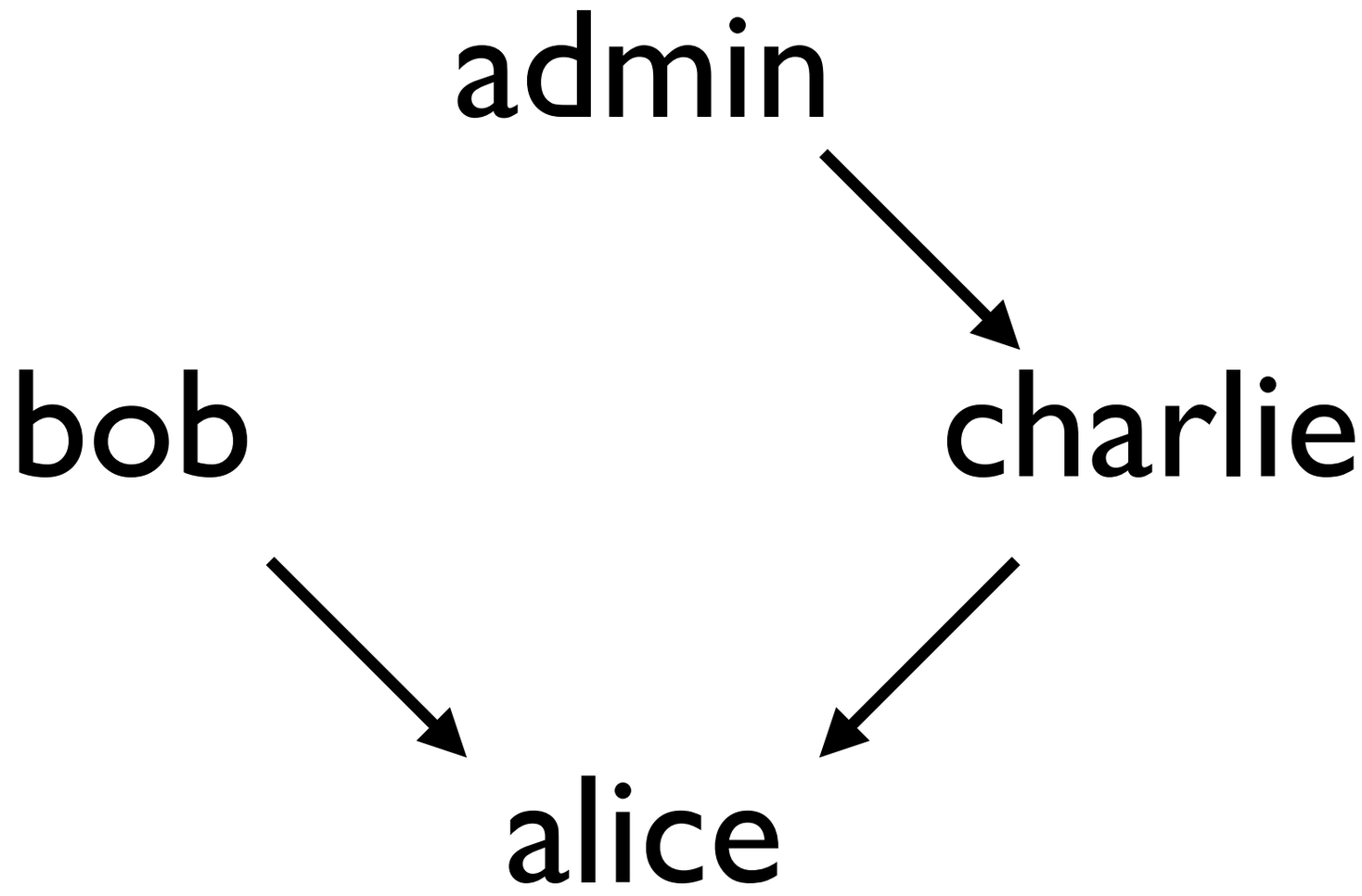
Circular Delegation

```
hasPermission(var, principal, right):
    if principal is "admin":
        return true
    else:
        delegator = getDelegator(var, principal, right)
        if there is a delegator:
            return hasPermission(var, delegator, right)
        else:
            return false
```

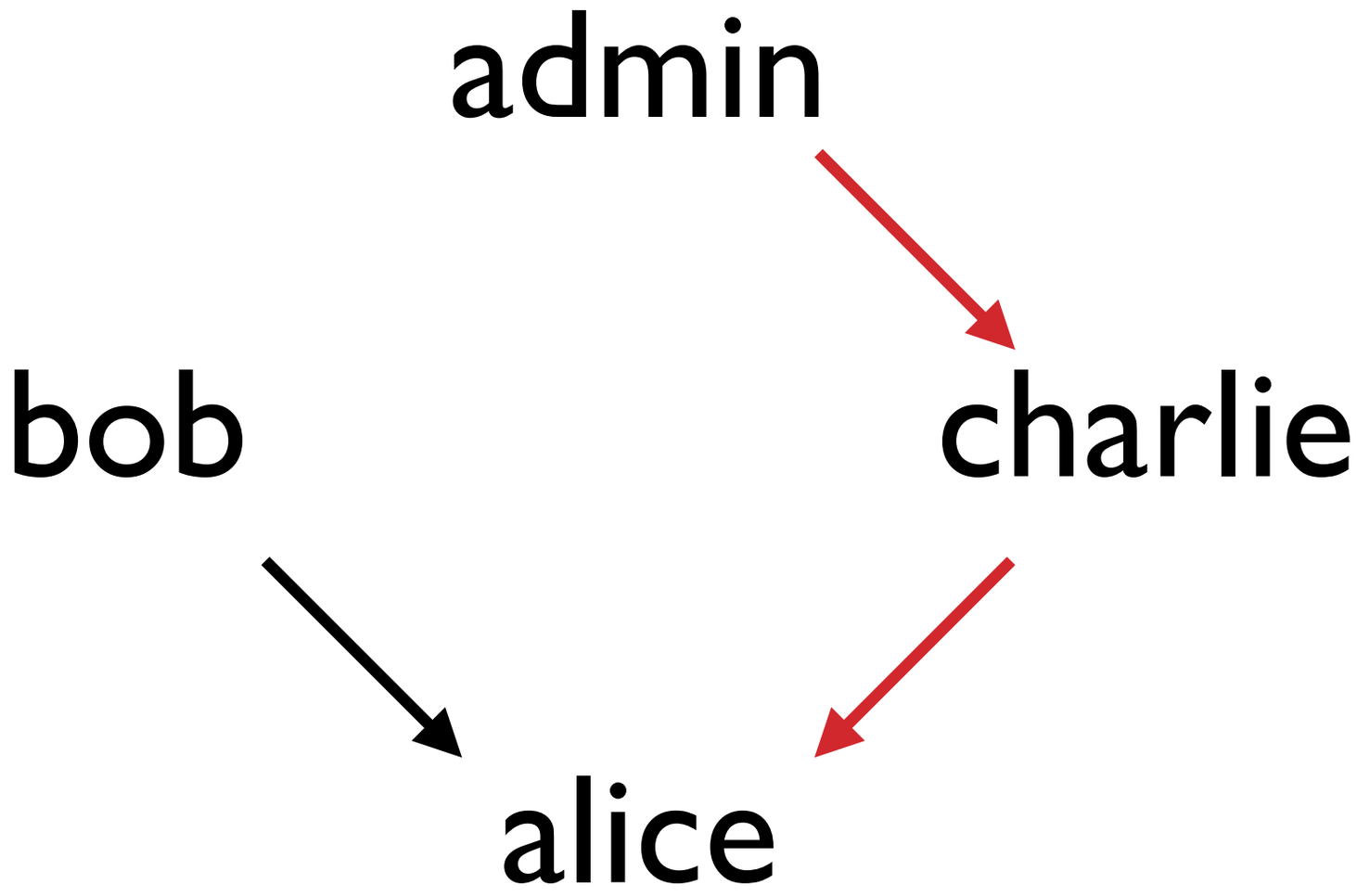
Multiple Paths



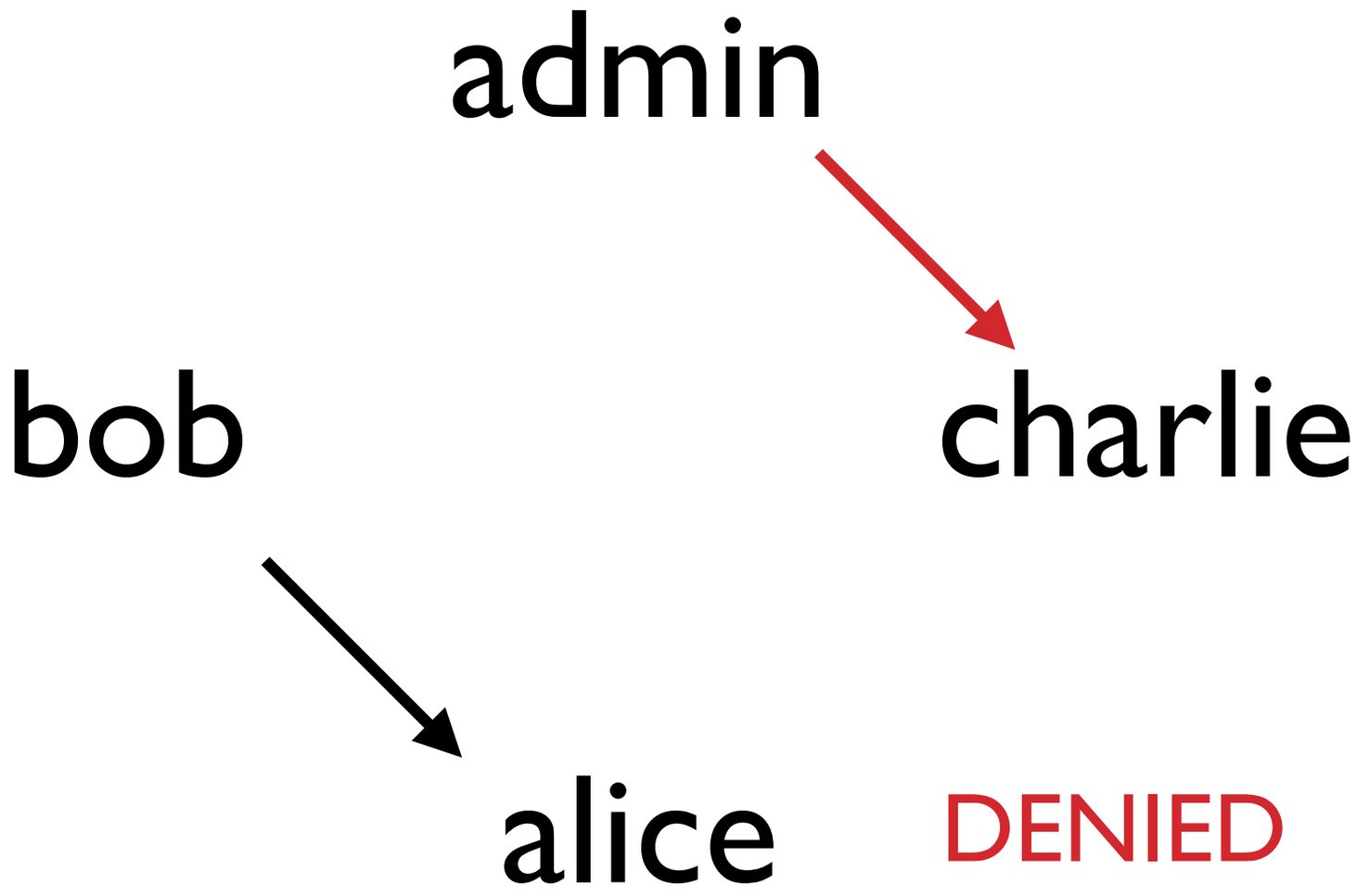
Multiple Paths



Multiple Paths



Multiple Paths



Non-logic problems

- Bugs in libraries used
- Integer overflow
- Code injection

Integer Overflow

```
as principal admin password "admin" do
   create principal bob "bob"
   set count = 1
   set delegation count admin write -> bob
   return count
***
```

Integer Overflow

```
as principal admin password "admin" do
   create principal bob "bob"
   set count = 1
   set delegation count admin write -> bob
   return count
***
as principal bob password “bob" do
   set count = 2147483647
   return x
***
```

Integer Overflow

```
as principal admin password "admin" do
   create principal bob "bob"
   set count = 1
   set delegation count admin write -> bob
   return count
***
as principal bob password “bob" do
   set count = 0b11111111111111111111111111111111
   return x
***
```

Integer Overflow

```
as principal admin password "admin" do
   create principal bob "bob"
   set count = 1
   set delegation count admin write -> bob
   return count
***
as principal bob password “bob" do
   set count = 0b11111111111111111111111111111111
   return x
***
as principal admin password "admin" do
   set count = count + 1
   if count < 0 then set door = 0
   return count
***
```

Integer Overflow

```
as principal admin password "admin" do
   create principal bob "bob"
   set count = 1
   set delegation count admin write -> bob
   return count
***
as principal bob password “bob" do
   set count = 0b11111111111111111111111111111111
   return x
***
as principal admin password "admin" do
   set count = 0b10000000000000000000000000000000 (-1)
   if count < 0 then set door = 0
   return count
***
```

Code Injection

```
os.system(“<call compiled executable> " + port  + " " +
config + " " + admin + " " + hub)
```

```
output = eval(<expr>)
```

Code Injection

```
os.system(“<call compiled executable> " + port  + " " +
config + " " + admin + " " + hub)
```

./server 1024 config.json "; ls " "; rm -rf /"

```
output = eval(<expr>)
```

Code Injection

```
os.system(“<call compiled executable> " + port  + " " +
config + " " + admin + " " + hub)
```

./server 1024 config.json "; ls " "; rm -rf /"

```
output = eval(<expr>)
```

set x = __import__(‘os’).system(‘rm -rf /’) + 1

Secure Development Reflection

- What worked?
- What didn't work?
- What would you do differently?

Secure Development Reflection

- What worked?
- What didn't work?
- What would you do differently?

Secure Development Reflection

- What worked?
- What didn't work?
- What would you do differently?

Secure Development Reflection

- What worked?
- What didn't work?
- What would you do differently?

Best Practices

- Limit security-specific code
- Graceful error handling (and logging)
- In-depth design/threat modeling
- Test cases are important!
- Code review!

In-class Survey Time!

- End-of-course survey
  - link emailed to you
- Work on final design document