Welcome to Hacknight

OSIRIS

Offensive Security, Incident Response, and Internet Security Laboratory

Room 219 of Rogers Hall,
5 MetroTech Center, Brooklyn

osiris@osiris.cyber.nyu.edu

Come hang out with us!

Hiring partners:

Google, Facebook, IBM, Bloomberg, Capsule8, BAE Systems, Raytheon, NSA, Palo Alto Networks, Riot Games, Square, Salseforce, JPMorgan Chase & Co, Etsy, MWR Labs, Include Security, NCC Group, Uber, Cisco, Etsy, CTFd, Synack, FBI, OPTIV, TD Bank, Red Balloon Security, Vector35 LLC, Trail of Bits, New York City Cyber Command....

OSIRIS

Kent Ma

About me

- CSAW CTF Lead
- Email: kent@osiris.cyber.nyu.edu
- I like to cheat in video games

OSIRIS

Hacknight

- Lectures and workshops
  - RH217 every Thursday at 6 PM
  - Workshops will be hosted on wargames.osiris.cyber.nyu.edu
- Come hang out at the OSIRIS Lab!
  - RH219 (Right next door)
  - recruit.osiris.cyber.nyu.edu <- Hacknight teaches everything for this

Warning

This course will be teaching potentially dangerous techniques. By staying here, you agree that all of the knowledge gained will be used in an ethical and safe manner.

The material taught in this course can cause harm to people and computer systems. If you cause harm either accidentally or intentionally, **it will get you fined or in jail.**

There are many other fields of security, but our focus will be on offensive application security

Why?

There are many other fields of security, but our focus will be on offensive application security

Why?

- We can make developers cry

There are many other fields of security, but our focus will be on offensive application security

Why?

- We can make developers cry
- How can you defend against something you don't understand?

OSIRIS

Overall Outline

Week 1	Introduction
Week 2, 3	Server-Side Web Security
Week 4	Client-Side Web Security
Week 5, 6, 7	Reverse engineering
Week 8, 9, 10	Memory Corruption
Week 11, 12	Heap Exploitation
Week 13	Kernel Exploitation

Week 1 - Introduction

Kent Ma

OSIRIS

Terminology

Vulnerability - A bug that has potential to be used in an exploit

Exploit (noun) - Crafted data of that uses a bug to make software do something Unintended

0day - Unknown or unpatched vulnerability that can be used by an exploit

OSIRIS

Outline

- What are we actually looking for?
- Where do we look?
- How do we look?

What are we looking for?

Types of Flaws

The rest of Hack Night will be focused on these bugs in depth with more advanced techniques

- Design flaws
- Implementation bugs
- Operational flaws

OSIRIS

Design Flaws

- Fundamental flaws in application architecture
- Application is working exactly as designed - for better or worse
- These are often subtle, devastating, and a massive pain to fix

Types of Design Flaws

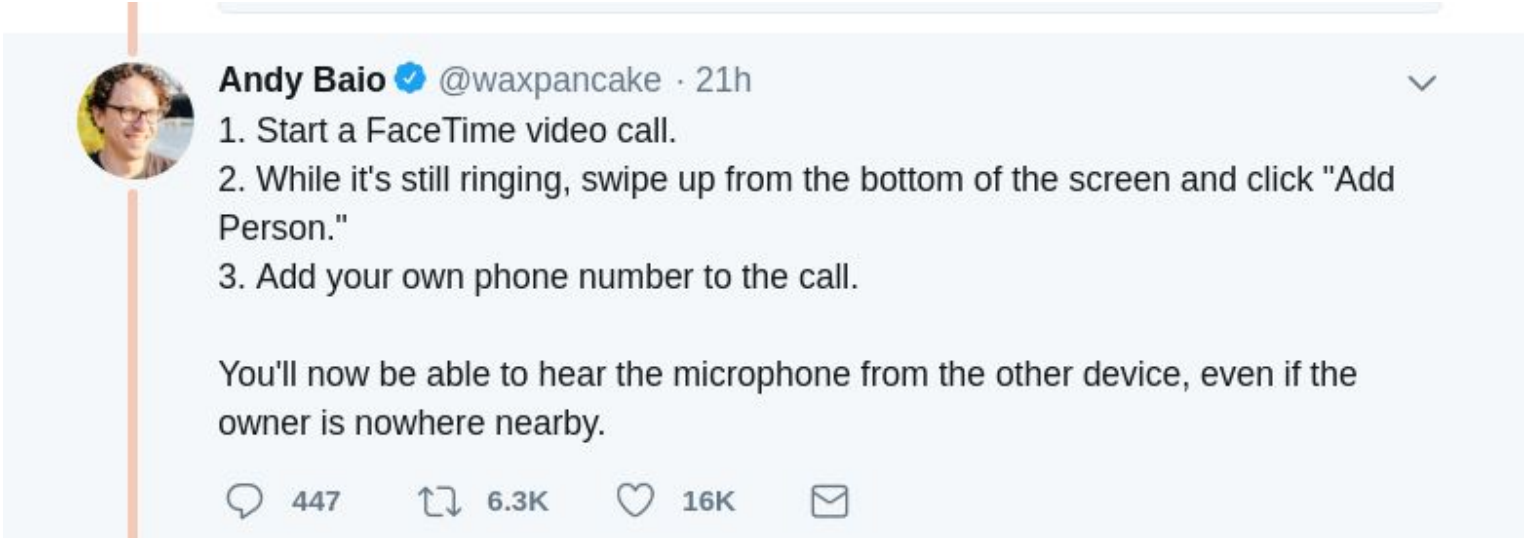
- Authentication
- Authorization
- Business logic
- Cryptography

Design Flaws - Authentication

- Lack of authentication
- Bypasses
  - Backdoors/Default creds
  - Logic bugs

OSIRIS

Authentication Bypass - Facetime



Design Flaws - Authorization

- No/poor authorization checks
- Granting privilege to lower privileged users
- Confusion between authorization contexts

OSIRIS

Authorization - Dominos App

- Authorization for payment of order is done client-side

  Change request to the server payment:

  ```
  <reason>DECLINED</reason>
  ```

  Changed to

  ```
  <reason>ACCEPTED</reason>
  ```

Business Logic Bugs

- Multi-step processes happening out of order
- Race conditions
- Time of check vs time of use (TOCTOU)

Business Logic - libssh Auth Bypass

“**By presenting the server an SSH2_MSG_USERAUTH_SUCCESS** message in place of the SSH2_MSG_USERAUTH_REQUEST message which the server would expect to initiate authentication, the attacker could successfully authentciate [sic] without any credentials.” (CVE-2018-10933)

Business Logic - libssh Auth Bypass

“**By presenting the server an SSH2_MSG_USERAUTH_SUCCESS** message in place of the SSH2_MSG_USERAUTH_REQUEST message which the server would expect to initiate authentication, the attacker could successfully authentciate [sic] without any credentials.” (CVE-2018-10933)

- Sending “SUCCESS” out of order - server mistakenly assumes that the rest of the authentication logic was done already

OSIRIS

Business Logic Bugs - Starbucks

- Starbucks.com has personal accounts for gift cards
- API endpoints to hit to transfer money:

```
starbucks.com/step1?amount=<amount>&from=wallet1&to=wallet2
```

OSIRIS

Business Logic Bugs - Starbucks

`starbucks.com/step1?amount=<amount>&from=wallet1&to=wallet2`

The code might look something like this:

```
if from.balance - amount > 0:
    from.balance -= amount
    to.balance += amount
```

OSIRIS

Time of Check vs Time of Use (TOCTOU)

What happens if multiple transactions happen at once?

wallet1 has 10 dollars on it

```
starbucks.com/step1?amount=9&from=wallet1&to=wallet2
starbucks.com/step1?amount=9&from=wallet1&to=wallet2
```

OSIRIS

Business Logic Bugs - TOCTOU

```
from has 10
to has 0

if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount
```

```
if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount
```

OSIRIS

Business Logic Bugs - TOCTOU

```
from has 10
to has 0
```

```
if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount
```

```
if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount
```

OSIRIS

Business Logic Bugs - TOCTOU

```
from has 10
to has 0

if from.balance - amount > 0: # 10 > 0
     from.balance -= amount
     to.balance += amount
```

```
if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount
```

OSIRIS

Business Logic Bugs - TOCTOU

```
from has 1
to has 0

if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount
```

```
if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount
```

OSIRIS

Business Logic Bugs - TOCTOU

```
from has 0
to has 9

if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount

Done!
```

```
if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount
```

OSIRIS

Business Logic Bugs - TOCTOU

```
from has -8
to has 9

if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount

Done!
```

```
if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount
```

OSIRIS

Business Logic Bugs - TOCTOU

```
from has -8
to has 18

if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount

Done!
```

```
if from.balance - amount > 0:
     from.balance -= amount
     to.balance += amount

Done!
```

Implementation Bugs

- Failures in implementation that cause the application to perform incorrectly
  - Invalid input causing **undefined behavior**
  - Misuse of API
  - Miscalculation
  - No verification

Injection Bugs

Several bugs classes follow this same idea

- Special characters which have no meaning in one context can have meaning in others
- Look for communication/context switches between different execution

OSIRIS

Injection Bugs - Command Injection

```
os.system(“ping ” + ip)
```

OSIRIS

Injection Bugs - Command Injection

```
os.system(“ping ” + ip)
ip = “127.0.0.1; cat /etc/passwd”
```

OSIRIS

Injection Bugs - Command Injection

```
os.system(“ping ” + ip)
ip = “127.0.0.1; cat /etc/passwd”
```

- Occurs because the IP string is moved to a different context (shell with system()) where it has different behavior

OSIRIS

Integer Bugs - Under/Overflow

```
unsigned int level = 0;
level--;
print(level);
```

4294967295

```
int level = 2147483647;
level++;
print(level);
```

-2147483648

OSIRIS

Integer Underflow?

```
unsigned short level = 1;
if ((level - 5) < 0) {
    login();
}
```

- Does `login()` get called?
- `level` should underflow to **`MAX_INT`**, shouldn't it?

OSIRIS

Integer Underflow?

```
unsigned short level = 1;
if ((level - 5) < 0) {
    login();
}
```

- Trick question! Login *is* called
- What type is everything?

OSIRIS

Integer Underflow?

```
unsigned short level = 1;
if ((level - 5) < 0) {
    login();
}
```

- level is unsigned short

OSIRIS

Integer Underflow?

```
unsigned short level = 1;
if ((level - 5) < 0) {
    login();
}
```

- level is unsigned short
- What type is 5? It's an int

OSIRIS

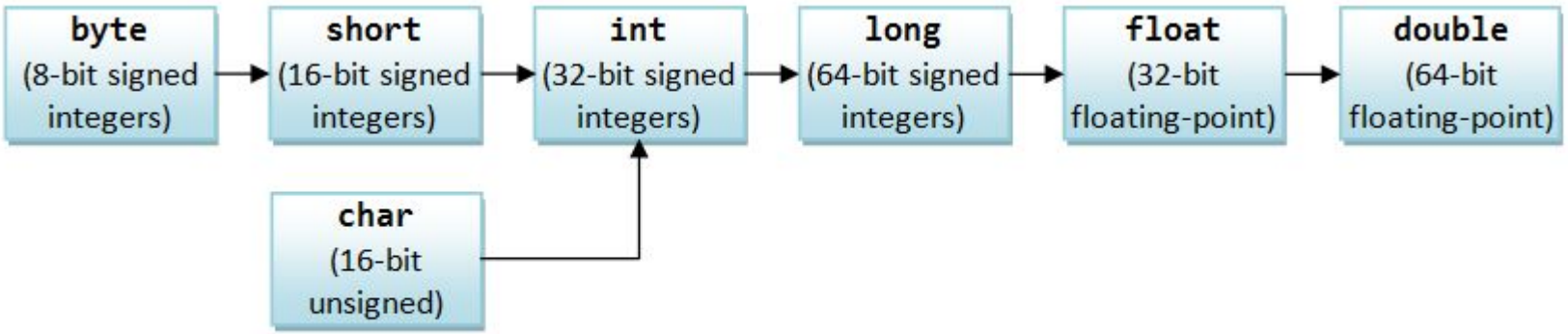
Integer Underflow?

```
unsigned short level = 1;
if ((level - 5) < 0) {
    login();
}
```

- level is unsigned short
- What type is 5? It's an int
- What happens when you do something with two types of different sizes?

Integer Promotions

- Smaller type is converted to “Larger” type **even if sign-ness is different**



Orders of Implicit Type-Casting for Primitives

Memory Corruption

- The most classic implementation bug
- We'll have several Hack Night weeks dedicated to this topic
- For now, just keep in mind that segfaults in your code aren't "just errors"

Operational Flaws

Dumb mistakes that get you bug bounties

- Misconfigurations
- Insecure defaults
- Exposing sensitive external components
- Missing mitigations

Operational Flaws

- We won’t focus too much on these
- If you notice potential for one when using/auditing a web application, scan the internet for it for $$$
- https://www.shodan.io/

OSIRIS

Operational Flaws - Exposed .git

- .git is the hidden folder created with repository metadata + code stored as git objects
- Common mistake: `$ git clone site.git /var/www/html`
- We can recover the site by going to http://site.com/.git and using

```
$ git checkout --
```

Where do we look?

There’s never enough time

- There’s a lot of code
- You start with little knowledge of the code
- People write ugly, hard-to-read code
- Timed engagements

Smelly Traditional Targets

- Input sources
- Authentication/filtering mechanisms
- Complex things (protocols, parsing, etc)
- Deeply coupled objects/classes/boundaries

Smelly Meta Targets

- Old/unmaintained code (check those copyright dates)
- Timing of code checked in
  - Near release dates
  - Around the same time as other known buggy code
- Developers that wrote bad code write more bad code elsewhere
- Previously buggy code
  - Don’t trust fixes!
- Things that directly tell you that the code sucks
  - FIXME, TODO, XXX, swearing, typos, github issues, etc

Coupled Components

- What are the components of the application?
- How are different components and boundaries connected?
- Are any components tightly coupled with each other? Where else are those used?
- Coupled components may have unintended side effects with their coupled pair when one of them is used elsewhere

Roles and Goals

- How are privileges defined?
- Which components need which privileges?
- How are users and groups used? (Authentication, identification)
- Are there ambiguities in roles for users?

How do we look?

OSIRIS

Goals

- Understand the application
- Find bugs, of course :)

OSIRIS

How do we look?

- Different tactics take different amounts of mental capacity
  - You can’t be at 100% capacity 100% of the time

Let’s go over some of the tactics we can rotate through

Source Code Auditing

Reading source code to find **vulnerabilities**

- Tedious and kind of a pain
- Requires thorough understanding of the language

Tactics - grep

- Search through the code base for bad functions
- Easy to do, takes little effort
- Tracing back to exploitable inputs is a pain
- Sometimes works
- Won’t give you more understanding of the application

Tactics - Input Tracing

- What type of input can we find in the application?
- Where does it flow?
- What’s the point of the input?
- What checks are done on the data? Where aren’t they done?

OSIRIS

Tactics - Modeling Boundaries

- Look at separate components of a target and diagram where they communicate
- Note authorization levels for communication between each component

Fuzz testing (Fuzzing)

- Basic idea: throw random inputs at the program and watch things break
- Works surprisingly well - lots of research into optimizing/improving fuzzing
- Think of it as searching over all possible inputs for any cases that are bugs

We won't go too deeply into fuzzing today, but you're free to chat with me afterwards about it

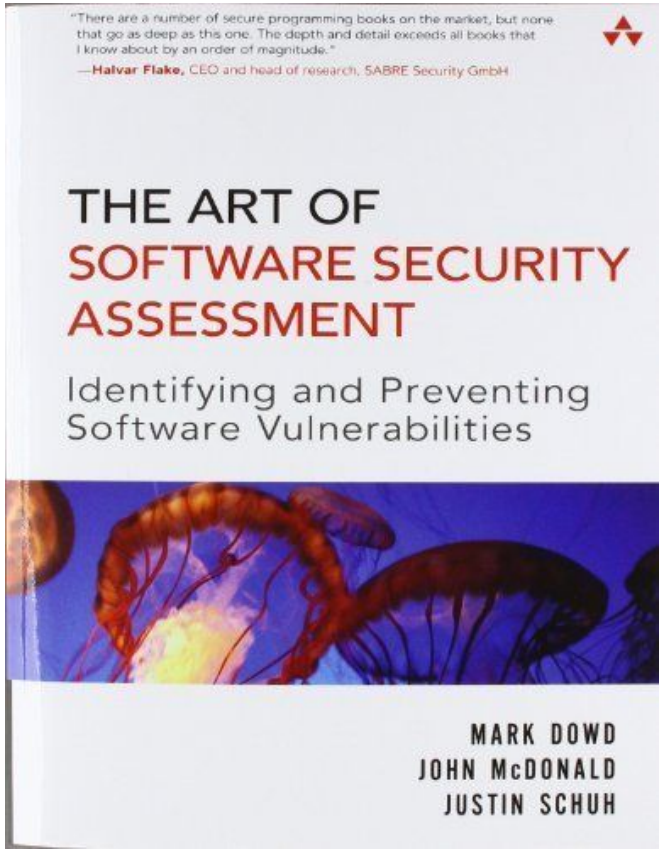
See https://labs.mwrinfosecurity.com/blog/what-the-fuzz/

OSIRIS

Resources

The Art of Software Security Assessment, Mark Dowd

- No seriously, this is THE book



OSIRIS

Workshop 1

wargames.osiris.cyber.nyu.edu