INTRODUCTION

COMPUTER & NETWORK SECURITY

CMSC 414

JAN 25 2018
What is security? Why is it so hard to achieve?

Administrative

The security mindset

Analyzing a system’s security
1. Summarize the system
2. Identify the assets
3. Identify the adversaries & threats
4. Identify the vulnerabilities
WHAT IS COMPUTER & NETWORK SECURITY?

• Normally, we are concerned with **correctness**
  • Does the software achieve the desired behavior?

• Security is a form of correctness
  • Does the software prevent “undesired” behavior?

The key difference:

**Security involves an adversary who is active and malicious.**

Attackers seek to **circumvent** protective measures.
WHAT DOES IT MEAN TO BE SECURE?

There is no such thing as security, only degrees of insecurity.

**Goal**: Raise the bar for the attacker

- Too difficult
- Too expensive
- Lower ROI than the next target

Ultimately, we want to mitigate undesired behavior.
WHAT ARE “UNDESIRED” BEHAVIORS?

- Reveals info users wish to hide (confidentiality)
  - Corporate secrets
  - Private data; personally identifying information (PII)

- Modifies information or functionality (integrity)
  - Destroys records
  - Changes data in-flight (think “the telephone game”)
  - Installs unwanted software (spambot, spyware, etc.)

- Denies access to a service (availability)
  - Crashing a website for political reasons
  - Denial of service attack
  - Variant: fairness

This is a subset
ATTACKS ARE COMMON

KEEP CALM AND goto fail;
go to fail;
WHY ARE ATTACKS COMMON?

• Because attacks are derived from design flaws or implementation bugs

• But all software has bugs: so what?

• A normal user never sees most bugs
  • Post-deployment bugs are usually rare corner cases

• Too expensive to fix every bug
  • Normal thought process: “Let’s only fix what’s likely to affect normal users”
WHY ARE ATTACKS COMMON?

Attackers are not normal users

- Normal users avoid bugs/flaws
- Adversaries seek them out and try to exploit them

This extends beyond software: Attacks are possible even with perfect software
• TLS is the de facto protocol for secure online communication

• Heartbleed was a vulnerability in the most popular TLS server
  • A malformed packet allows you to see server memory

• Fix: don’t let the user just tell you how much data to give back

• This was a design flaw
HEARTBLEED

HOW THE HEARTBLEED BUG WORKS:

SERVER, ARE YOU STILL THERE?
IF SO, REPLY "POTATO" (6 LETTERS).

User Meg wants these 6 letters: POTATO.
User Eda wants pages about "irl games". Unlocking secure records with master key 513098573343.
User Meg sends this message: "Hi..."
Server, are you still there? If so, reply "Bird" (4 letters).

Hmm...

User Olivia from London wants pages about "has bees in car why". Note: Files for IP 375.381.183.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 348 connections open. User Brendan uploaded the file self.png (contents: 33b962e70b0ff9b33b0f).

BIRD
User passwords, private keys, personal information...

~40% of “secure” web servers vulnerable
1. Carefully crafted Flash program. When run by the vulnerable Flash player, allows the attacker to execute arbitrary code on the running machine.

2. This program could be **embedded in an Excel spreadsheet**, and run automatically when the spreadsheet was opened.

3. Spreadsheet **attached to an email**, masquerading as a trusted party ("spearphishing")
   - You can forge any "From" address
WHY ARE ATTACKS COMMON?

Because it’s **profitable**

And because a system is **only as secure as its weakest link**

---

**Figure 1:** Infrastructure involved in a single URL’s value chain, including advertisement, click support and realization steps.
WHY ARE ATTACKS COMMON?

• Security is a property of the systems we build

• Many attacks begin by exploiting a vulnerability
  • Vulnerability = defect in hw, sw, protocol, design, ... that can be exploited to yield an undesired behavior
  • Software defect = the code doesn’t “behave correctly”

• Defects arise due to
  • flaws in the design and/or
  • bugs in the implementation
In order to achieve security, we must:

Be able to eliminate bugs and design flaws and/or make them harder to exploit.

Be able to think like attackers.

Develop a foundation for deeply understanding the systems we use and build.
50% of Android apps that use crypto encrypt in this manner
GOALS OF CMSC 414

Be able to eliminate bugs and design flaws and/or make them harder to exploit.

Be able to think like attackers.

Develop a foundation for deeply understanding the systems we use and build.

Software  Hardware  Protocols
Users      Law      Economics
• What is security? Why is it so hard to achieve?

• Administrative

• The security mindset

• Analyzing a system’s security
  1. Summarize the system
  2. Identify the assets
  3. Identify the adversaries & threats
  4. Identify the vulnerabilities
Resources and all this info will be on the class website
  • http://www.cs.umd.edu/class/spring2018/cmsc414-0101

We will be using Piazza
  • You should have been added; let me know if you haven’t
ADMINISTRATIVE: THE TEAM

Michael Bartner
Nirat Saini
Nishant Rodrigues
Omer Akgul
Ronald Cheng
Soumya Indela
Tommy Hegarty
ADMINISTRATIVE: TEXTBOOKS

• None required
  • Mostly in-class and papers posted on website

• Recommended texts, if you are so inclined
  • “Security in Computing”, Pfleeger & Pfleger
  • “Introduction to Computer Security”, Goodrich & Tamassia
  • “Security Engineering”, Ross Anderson
    - Free online: http://www.cl.cam.ac.uk/~rja14/book.html
• The best way to learn is to reinforce

• Lots of security resources (something is always breaking).
  • Krebs on security
  • Bruce Schneier’s blog
  • reddit.com/r/netsec
  • Any other favorites? Let us know on Piazza
What’s in This Course

How do we build software that is secure?

- Memory safety
- Malware
- Web security
- Static analysis
- Design principles
What's in this course

Crypto

What it is, and how to use it responsibly

A black-box approach to crypto
Designing protocols that use crypto
Authentication: proving who you are
Anonymity: hiding who you are
WHAT’S IN THIS COURSE

Attacks on TCP & DNS
Botnets
Underground spam economies

How to build secure networked systems.
WHAT'S IN THIS COURSE

Software Security

How do we build software that is secure?

Crypto

What it is, and how to use it responsibly

Network Security

How to build secure networked systems.

Attacks and defenses across all of these
This is a brief listing of the Top 25 items, using the general ranking.

NOTE: 16 other weaknesses were considered for inclusion in the Top 25, but their general scores were not high enough. They are listed in a separate "On the Cusp" page.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Score</th>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>93.8</td>
<td>CWE-89</td>
<td>Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')</td>
</tr>
<tr>
<td>[2]</td>
<td>83.3</td>
<td>CWE-78</td>
<td>Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')</td>
</tr>
<tr>
<td>[3]</td>
<td>79.0</td>
<td>CWE-120</td>
<td>Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')</td>
</tr>
<tr>
<td>[4]</td>
<td>77.7</td>
<td>CWE-79</td>
<td>Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')</td>
</tr>
<tr>
<td>[6]</td>
<td>76.8</td>
<td>CWE-862</td>
<td>Missing Authorization</td>
</tr>
<tr>
<td>[7]</td>
<td>75.0</td>
<td>CWE-798</td>
<td>Use of Hard-coded Credentials</td>
</tr>
<tr>
<td>[8]</td>
<td>75.0</td>
<td>CWE-311</td>
<td>Missing Encryption of Sensitive Data</td>
</tr>
<tr>
<td>[9]</td>
<td>74.0</td>
<td>CWE-434</td>
<td>Unrestricted Upload of File with Dangerous Type</td>
</tr>
<tr>
<td>[10]</td>
<td>73.8</td>
<td>CWE-807</td>
<td>Reliance on Untrusted Inputs in a Security Decision</td>
</tr>
<tr>
<td>[11]</td>
<td>73.1</td>
<td>CWE-250</td>
<td>Execution with Unnecessary Privileges</td>
</tr>
<tr>
<td>[12]</td>
<td>70.1</td>
<td>CWE-352</td>
<td>Cross-Site Request Forgery (CSRF)</td>
</tr>
</tbody>
</table>
ETHICS AND LEGALITY

• You will be learning about (and implementing and launching) attacks, many of which are in active use today.

• This is not an invitation to use them without the explicit written consent of all parties involved.

• If you want to try something out, then let me know and I will try to help create a safe environment.

• This is not just a question of ethics; to do otherwise would risk violating UMD policies and MD/USA laws.
PREREQUISITE KNOWLEDGE

• You should be reasonably proficient in C and Unix

• You should also be creative and resourceful (those who try to attack your systems will be!)

• Otherwise, this course won’t require any prior knowledge in networking or crypto
WHAT ARE GRADES BASED ON?

• Grade breakdown
  • 50%: Projects (P1-P3: 10%, P4: 20%)
  • Midterms (2 x 12% each)
  • Final (25%)
  • Meet your professor (1%)
• You come by my office at some point *before the last day of classes* and we chat

• Gives me a chance to get to know each of you, learn about your interests, chat plans/research…

• Again: if you are booked during my office hours, just email me to set up a time.
EXAMS

Expected dates

Midterm #1:
Mar 8
12%

Midterm #2:
Apr 19
12%

Final exam:
May 18
25%

Please see the syllabus for information about excused absences
• What is security? Why is it so hard to achieve?

• Administrative

• The security mindset

• Analyzing a system’s security
  1. Summarize the system
  2. Identify the assets
  3. Identify the adversaries & threats
  4. Identify the vulnerabilities
THE SECURITY MINDSET

To anticipate attackers, we must be able to think like attackers.

Uniquely identifiable liquid + Proof of ownership = Proof of ownership

What would an attacker do?

Paint it on someone else’s property and then call the cops.
THE SECURITY MINDSET

To anticipate attackers we must be able to think like attackers

Fill out a card with your address

⇒

They deliver a box of live ants to you

What would an attacker do?

Order them to someone else
THE SECURITY MINDSET

The ability to view a large, complex system and be able to reason about:

- What are the potential security threats?
- What are the hidden assumptions?
- Are the explicit assumptions true?
- How can we mitigate the risks of the system?

Be creative! (Attackers will be)
E-voting analysis

1. Summarize the system as clearly and concisely as possible

1. Pre-election phase
   • Poll worker loads a “ballot definition file” (defines who’s running, colors on the screen, and many more things) on the voting machines with, e.g., USB

2. Voting phase
   (a) Voter obtains a single-use token from poll workers (on smartcard)
   (b) Voter uses the token to interactively vote
   (c) Vote stored encrypted on disk
   (d) Voter token canceled

3. Post-election phase
   • Stored votes decrypted and transported to tabulator
   • Tabulator counts and announces vote
E-voting analysis

2. Identify the assets / goals of the system

- Confidentiality
  - No one knows for whom any given voter voted (except for the voter)

- Integrity
  - Every voter’s vote counted once
  - No voter’s vote changed

- Availability
  - Everyone has the ability to cast their vote

- Usability
  - Easy for the voter to vote (correct language, good UI)
  - Easy for the tabulator to count votes
E-voting analysis

3. Identify the adversaries and threats

- Mickey Mouse
- Donald Duck
- Minnie Mouse

Poll worker could set BDF to print “Mickey Mouse” but record as “Minnie Mouse”

Voter could attempt to generate their own tokens & get ≥2 votes

Because there is no end-to-end verification that a vote was counted, modifying the software could result in complete control

Reading this could reveal who voted for whom. Writing it could change the outcome altogether
E-voting analysis

4. Identify the vulnerabilities

- Ballot definition files are not authenticated
  - How do we know they’re from the election board?
  - Can redefine “Candidate A” as “Candidate B”
  - Viruses

- Smartcards are not authenticated
  - How do we know they’re not user-generated?
  - Possible to make your own and vote multiple times.

- Specific software vulnerabilities
  - Every machine has the same encryption key!
  - Break one, and they all fall

- Votes are shipped unencrypted!

- Votes are stored in the order cast
  - If one can view the data unencrypted, this violates our confidentiality goal
E-voting analysis

Takeaway points

- Analyzing security requires a whole-systems view
  - Hardware
  - Software
  - Data
  - People

- Security is only as strong as the weakest link
  - May have been difficult to break into the building
  - But if the data is sent unencrypted…

- Securing a system can be difficult
  - Interdisciplinary (software, hardware, UI design)
  - Humans are in the loop

- Security through obscurity does not work
  - Especially for high-value assets
  - It’s only a matter of time until someone finds out
We will begin our 1st section: **Software Security**

By investigating **Buffer overflows** and other memory safety vulnerabilities

To prepare: you may want to brush up on your C

Particularly if this seems foreign to you:

```c
char buf[32];
unsigned *ptr = (unsigned*) (buf + 12);
*ptr += 0x1a;
```