Introduction

CMSC 414: Computer and Network Security
Spring 2016
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?
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 are “undesired” behaviors?
What are “undesired” behaviors?

• Reveals info users wish to hide (confidentiality)
  • Corporate secrets
  • Private data; personally identifying information (PII)
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.)
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
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
From just the past 9 months or so:
Why are attacks common?
Why are attacks common?

• Security is a property of the systems we build

• Many attacks begin by exploiting a vulnerability
  • Vulnerability = software defect that can be exploited to yield an undesired behavior
  • Software defect = the code doesn’t “behave correctly”

• Software defects arise due to
  • flaws in the design and/or
  • bugs in the implementation
Why are attacks common?

- Because attacks derive from **design flaws** and/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
  - 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
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
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.
In order to achieve security, we must:

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

Widespread misuse of crypto
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.

Widespread misuse of crypto

This is an encrypted image
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.

Widespread misuse of crypto

50% of Android apps that use crypto encrypt in this manner
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.
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.

**Software**  **Hardware**  **Protocols**

**Users**  **Law**  **Economics**
The 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
This time

• What is security?

• Administrative

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

• Trusting trust
Administrative

Communicating

• Resources and all this info will be on the class website
  • http://www.cs.umd.edu/class/spring2016/cmsc414

• Who
  • Me: Dave Levin (dml@cs.umd.edu)
  • TAs: Frank Cangialosi
        Jacob Hammontree
        Lee Williams
        Chengxi Ye
  • Office hours are on the website
    • If my office hours don’t work for you, email me and set up a time

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

Outside reading

• 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?
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?

**Software Security**

**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?

- Software Security
- Crypto
- Network Security

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>
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’re grades based on?

• Grade breakdown
  • 50%: Projects (P1-P3: 10%, P4: 20%)
  • Midterms (2 x 12% each)
  • Final (25%)
  • Meet your instructor (1%)
Meet your instructor (that’s me!)

• 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.
Midterms & Exams

Expected dates

Midterm #1: Mar. 10
12%

Midterm #2: Apr. 21
12%

Final exam: May 18
25%

Please see the syllabus for information about excused absences
Trusting Trust
Is anything really “secure”? 
Is anything really “secure”?

• Security requires context
  • What is the **threat model**? What can the attacker do?
  • What are the **assets** you seek to protect?
  • Whom and what do you **trust**?
Is anything really “secure”?

• Security requires context
  • What is the \textit{threat model}? What can the attacker do?
  • What are the \textit{assets} you seek to protect?
  • Whom and what do you \textit{trust}?

• “Trust no one!”
  • That’s the spirit!
  • But how did you compile your code again?
  • Who built your OS? Your hardware?…
Is anything really “secure”?  

- Security requires context
  - What is the **threat model**? What can the attacker do?
  - What are the **assets** you seek to protect?
  - Whom and what do you **trust**?

- “Trust no one!”
  - That’s the spirit!
  - But how did you compile your code again?
  - Who built your OS? Your hardware?…

**Required reading**

“Reflections on Trusting Trust”
Ken Thompson
What does a security vulnerability look like?

*Brief look at the Heartbleed vulnerability*
Next time

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;
```