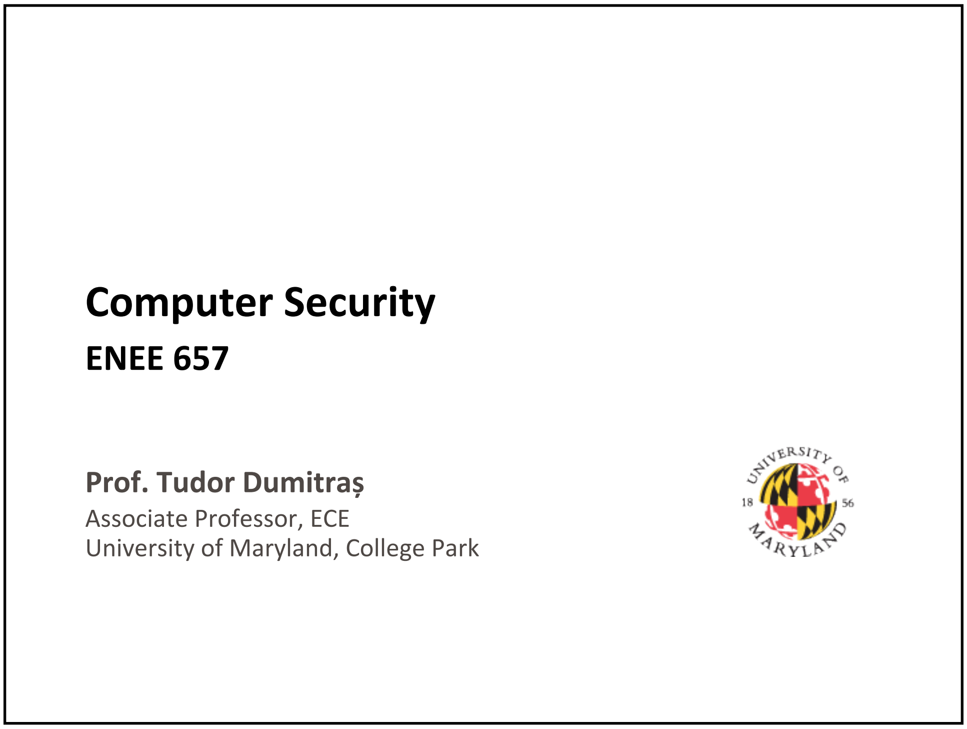
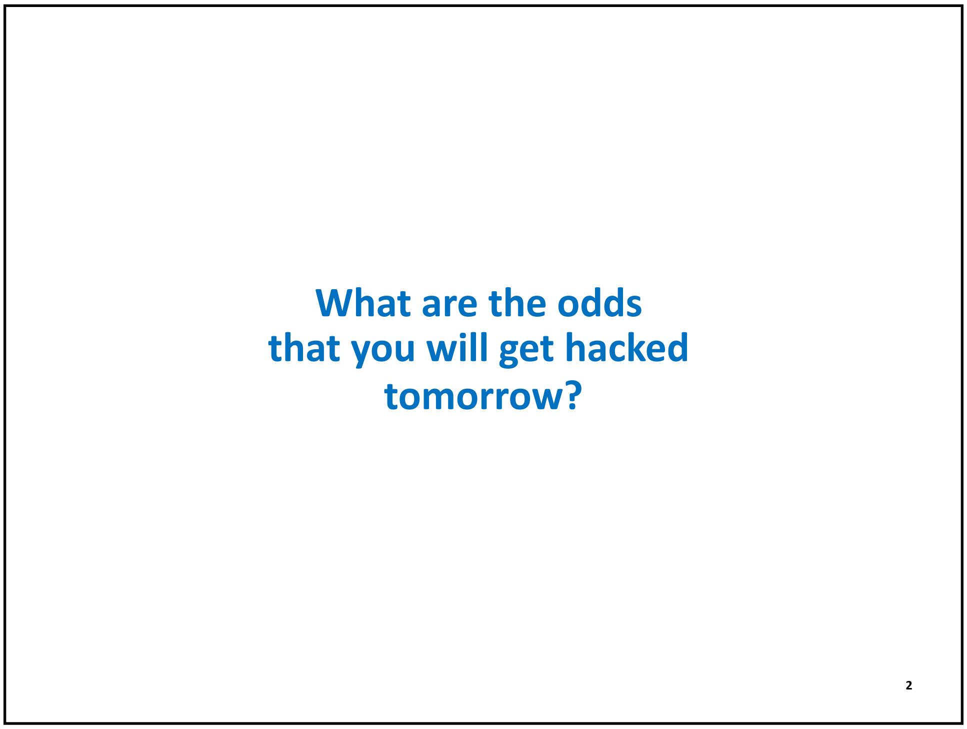


Computer Security

ENEE 657

Prof. Tudor Dumitraș
Associate Professor, ECE
University of Maryland, College Park



What are the odds
that you will get hacked
tomorrow?

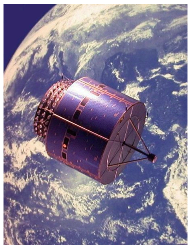
How Vulnerable Are You To Malware?

- We systematically measured amount of malware on **4 million hosts** in 44 countries
  [*The Global Cyber-Vulnerability Report*, Springer, 2015]
- Top 5:
  - South Korea, India, Saudi Arabia, China, Malaysia, Russia
- United States: 10th safest

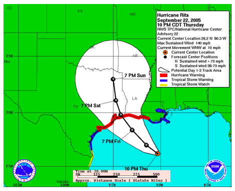
Range of adversary capabilities

Perceived vs. objective security

Understanding Computer Security



+


Hurricane Rita
September 22, 2005
10 PM CDT Thursday
NWS TPC/National Hurricane Center
Advisory 22
Current Center Location 26.2 N 90.3 W
Max Sustained Wind 140 mph
Current Movement WNW at 10 mph
Current Center Location
Forecast Center Positions
H Sustained wind > 73 mph
S Sustained wind 39-73 mph
Potential Day 1-3 Track Area
Hurricane Warning
Tropical Storm Warning
Tropical Storm Watch
7 PM Sun
7 PM Sat
7 PM Fri
10 PM Thu
CO
KS
MO
OK
AR
NM
TX
LA
Mexico
35N
30N
25N
105W
100W
95W
90W
85W
True at 30.00N
SM 125 250 375 500
Approx. Distance Scale (Statute Miles)

+

Security Measurements

Inference and Prediction

(Adversarial) Machine Learning

About Your Instructor

Tudor Dumitraș

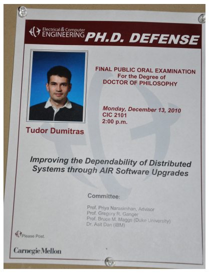
Office: IRB 5228

Email: `tdumitra@umiacs.umd.edu`

Course Website: http://ter.ps/enee657

My Story

- 2000s: Carnegie Mellon University
  - Ph.D. in distributed systems



Electrical & Computer ENGINEERING **PH.D. DEFENSE**

FINAL PUBLIC ORAL EXAMINATION
For the Degree of
DOCTOR OF PHILOSOPHY

Monday, December 13, 2010
CIC 2101
2:00 p.m.

Tudor Dumitras

Improving the Dependability of Distributed Systems through AIR Software Upgrades

Committee:

Prof. Priya Narasimhan, Advisor
Prof. Gregory R. Ganger
Prof. Bruce M. Maggs (Duke University)
Dr. Asit Dan (IBM)

Please Post.

Carnegie Mellon

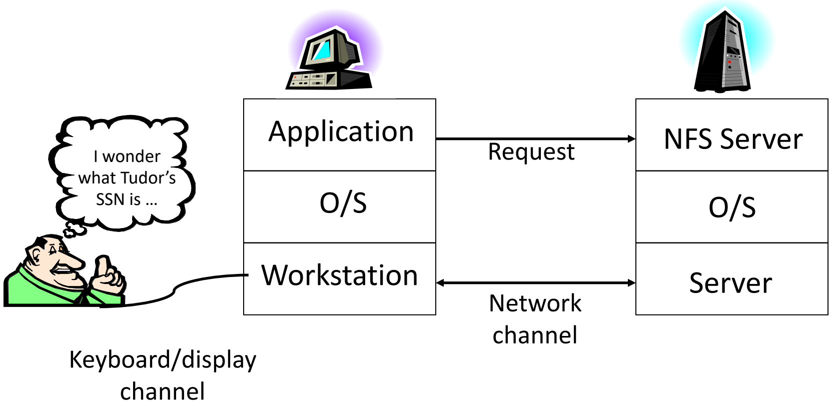
- 2010: Symantec Research Labs

- Since 2013: UMD
  - Maryland Cybersecurity Center (MC2)

ENEE 657 in a Nutshell

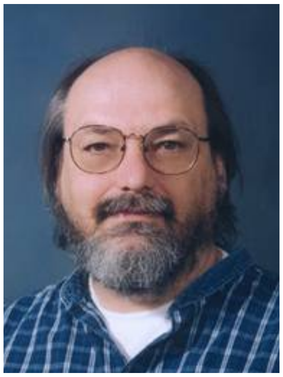
- ENEE 657 is a **graduate**-level security course
  - Learn by **reading**, **explaining** and **doing**
  - **Project oriented**: develop to a degree that would merit **publication** in one of the **workshops** associated with the USENIX Security Symposium 2020
- Aims to prepare you for **research in security**
  - **Not** a tutorial or comprehensive course on these topics
  - Instead, exploring a range of topics to illustrate some of the current research challenges
  - Targeted at students who want to conduct research in the area or who are more generally interested in security as it applies to their fields

Who Can You Trust?



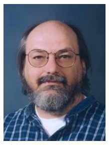
- Where is the request "from"?
  - The user? The workstation? The application? The network channel? All of the above?
  - Which of these actors do you trust?

Ken Thompson



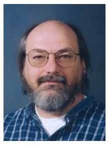
ACM Turing Award, 1983

“Reflections on Trusting Trust”



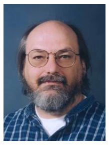
- What software can we trust?
- Example: any operating system includes a program checking whether users are allowed to log in
  - "login" or "su" in Unix
  - Is the login binary from Windows/Mac OS/Ubuntu/etc. trustworthy?
  - Does it send your password to someone?
  - Does it have backdoor for a “special” remote user?
- Can't trust the binary, so check source code or write your own, recompile
- Does this solve problem?

“Reflections on Trusting Trust” – cont’d



- Who wrote the compiler?
- Compiler looks for source code that looks like the login process, inserts backdoor into it
- Ok, inspect the source code of the compiler… Looks good? Recompile the compiler!
- Does this solve the problem?

“Reflections on Trusting Trust” – cont’d



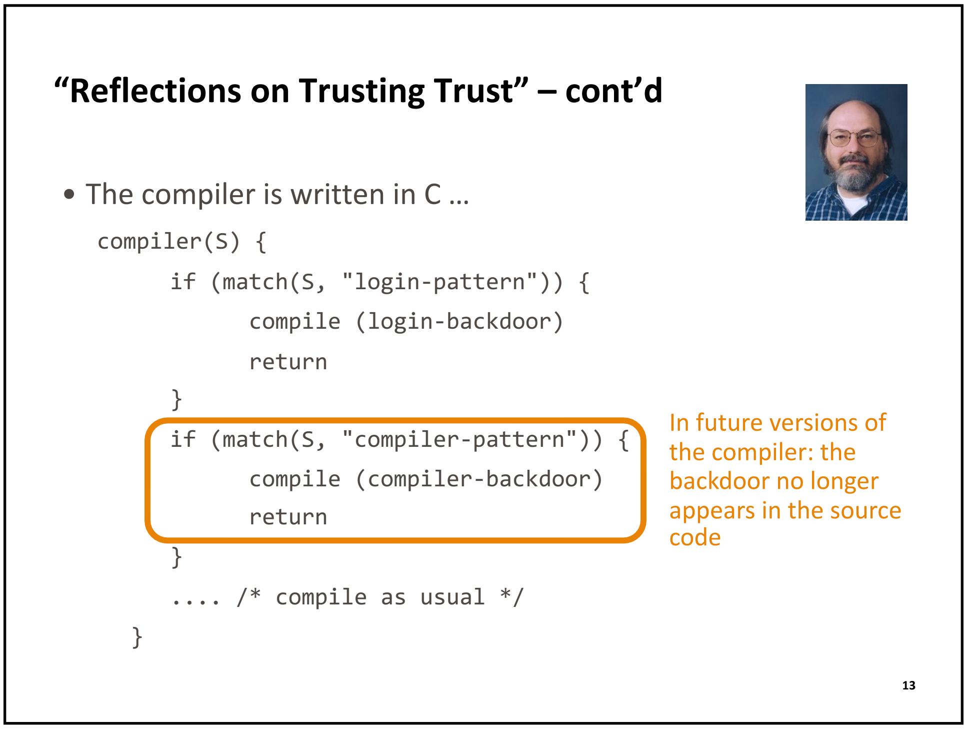
- The UNIX login program is compiled by a C compiler
  - The C compiler was also compiled by an (older) C compiler
- Aside: how does the compiler handle special characters?

```
…
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c== 'n')
        return('\n');
if(c == 'v')
        return(11);
…
```

When adding a new special character to the C language, must specify the character code

```
…
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c== 'n')
        return('\n');
if(c == 'v')
        return('\v');
…
```

In future versions of the compiler: use the special character

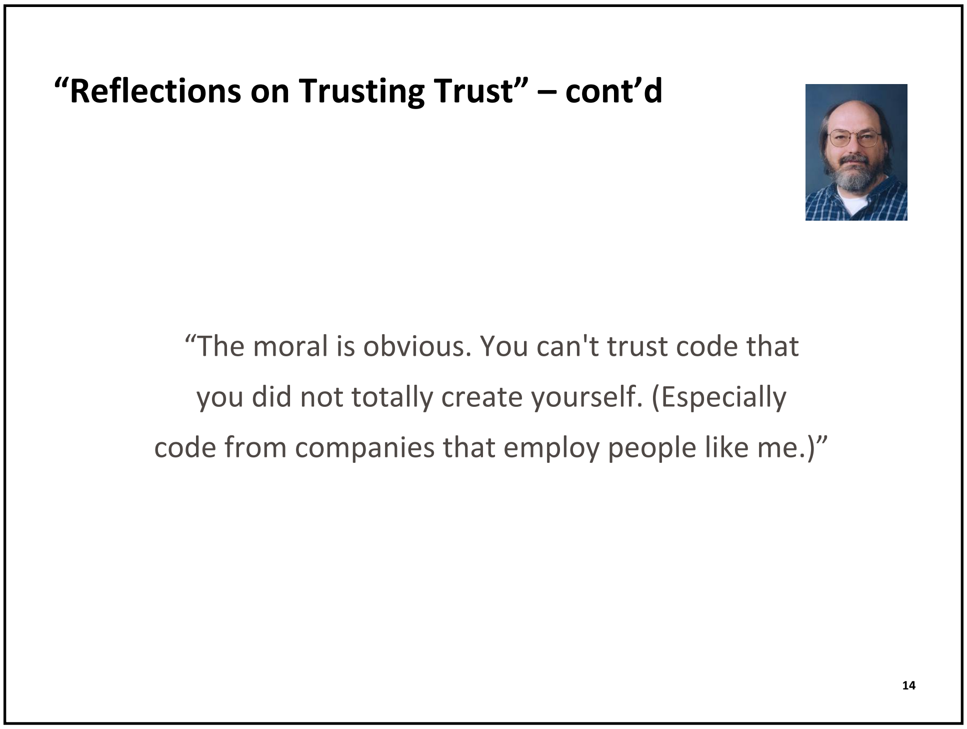


“Reflections on Trusting Trust” – cont’d

- The compiler is written in C …

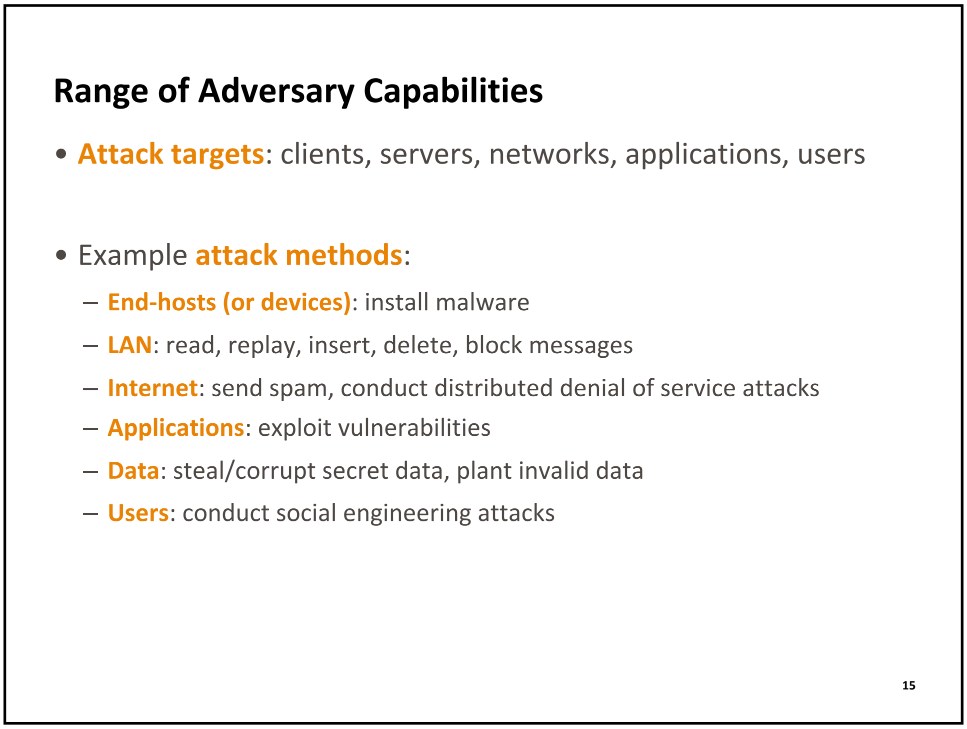
```
compiler(S) {
      if (match(S, "login-pattern")) {
            compile (login-backdoor)
            return
      }
      if (match(S, "compiler-pattern")) {
            compile (compiler-backdoor)
            return
      }
      .... /* compile as usual */
   }
```

In future versions of the compiler: the backdoor no longer appears in the source code



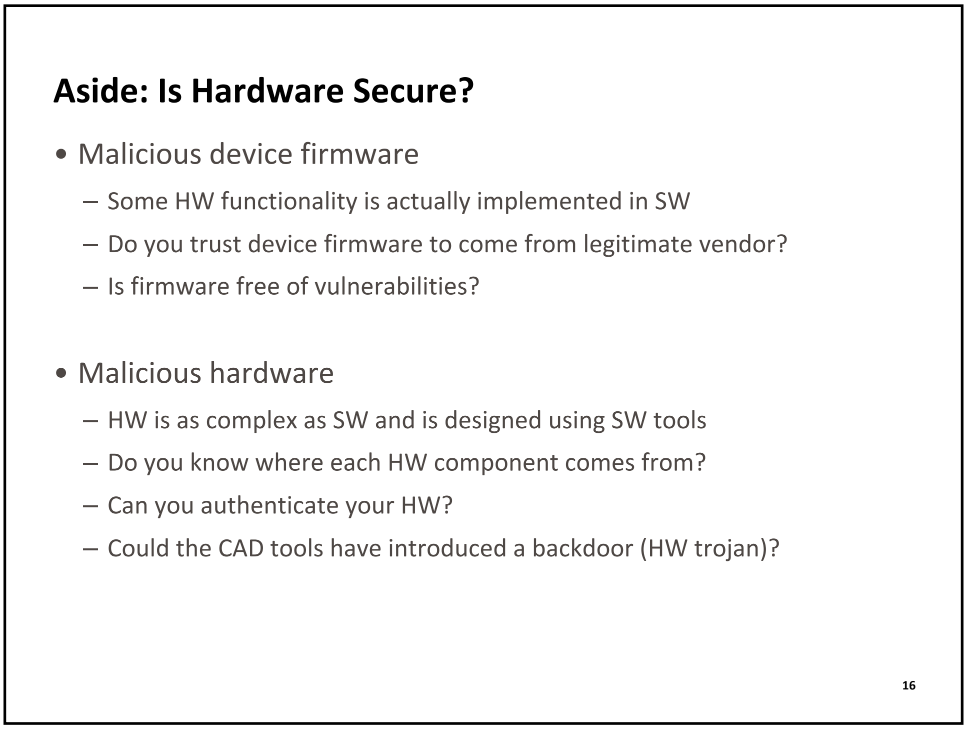
“Reflections on Trusting Trust” – cont’d

“The moral is obvious. You can't trust code that you did not totally create yourself. (Especially code from companies that employ people like me.)”



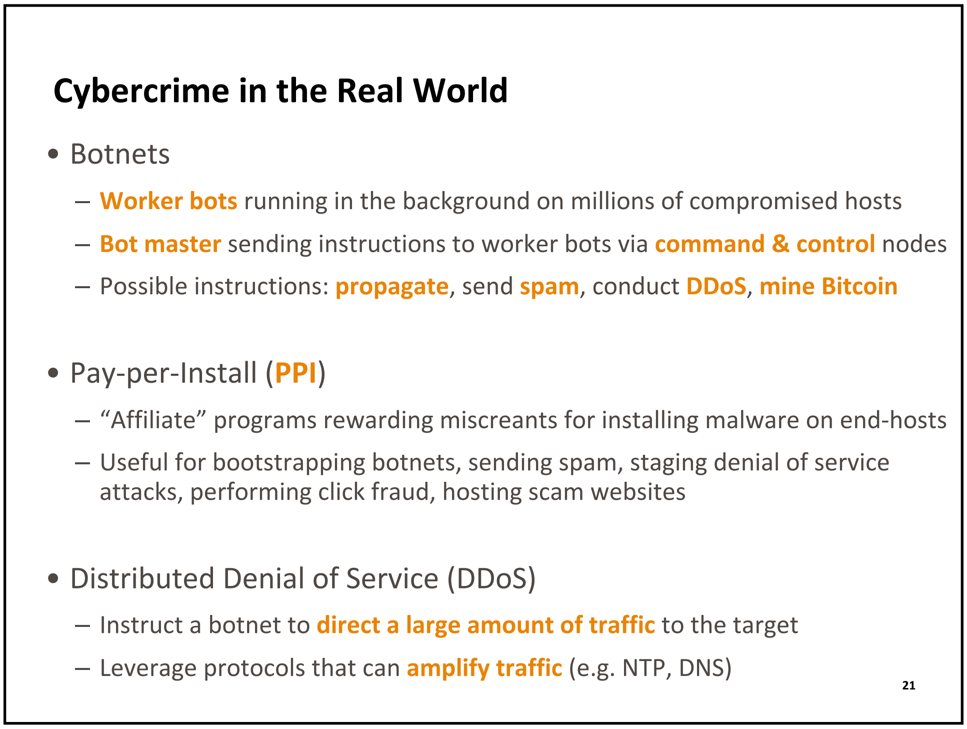
Range of Adversary Capabilities

- **Attack targets**: clients, servers, networks, applications, users
- Example **attack methods**:
  - **End-hosts (or devices)**: install malware
  - **LAN**: read, replay, insert, delete, block messages
  - **Internet**: send spam, conduct distributed denial of service attacks
  - **Applications**: exploit vulnerabilities
  - **Data**: steal/corrupt secret data, plant invalid data
  - **Users**: conduct social engineering attacks



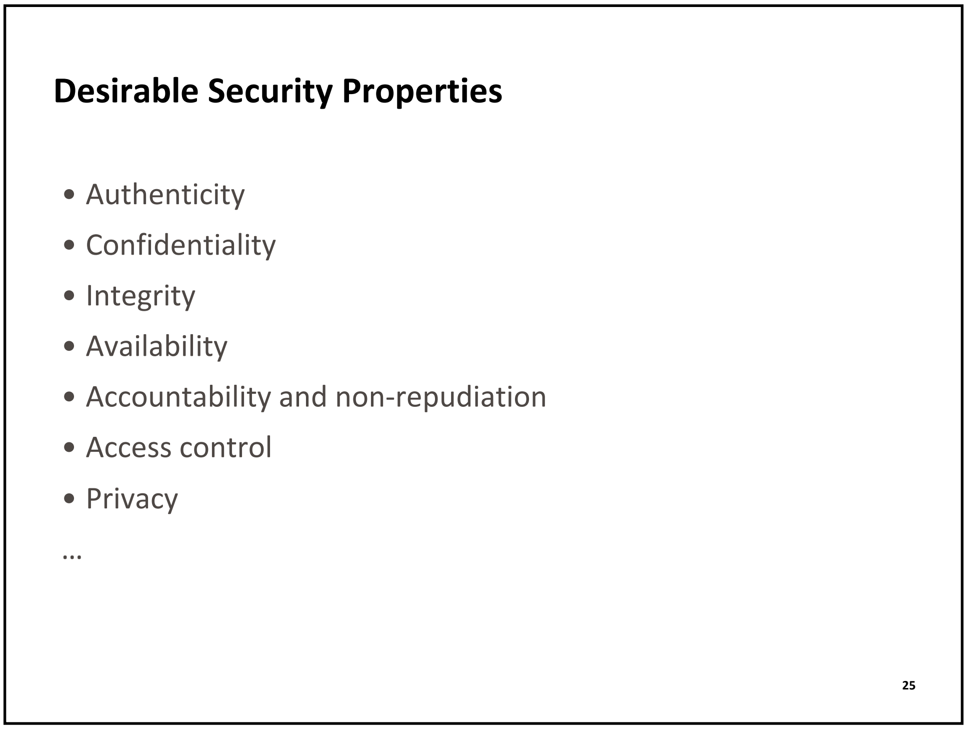
Aside: Is Hardware Secure?

- Malicious device firmware
  - Some HW functionality is actually implemented in SW
  - Do you trust device firmware to come from legitimate vendor?
  - Is firmware free of vulnerabilities?
- Malicious hardware
  - HW is as complex as SW and is designed using SW tools
  - Do you know where each HW component comes from?
  - Can you authenticate your HW?
  - Could the CAD tools have introduced a backdoor (HW trojan)?



Cybercrime in the Real World

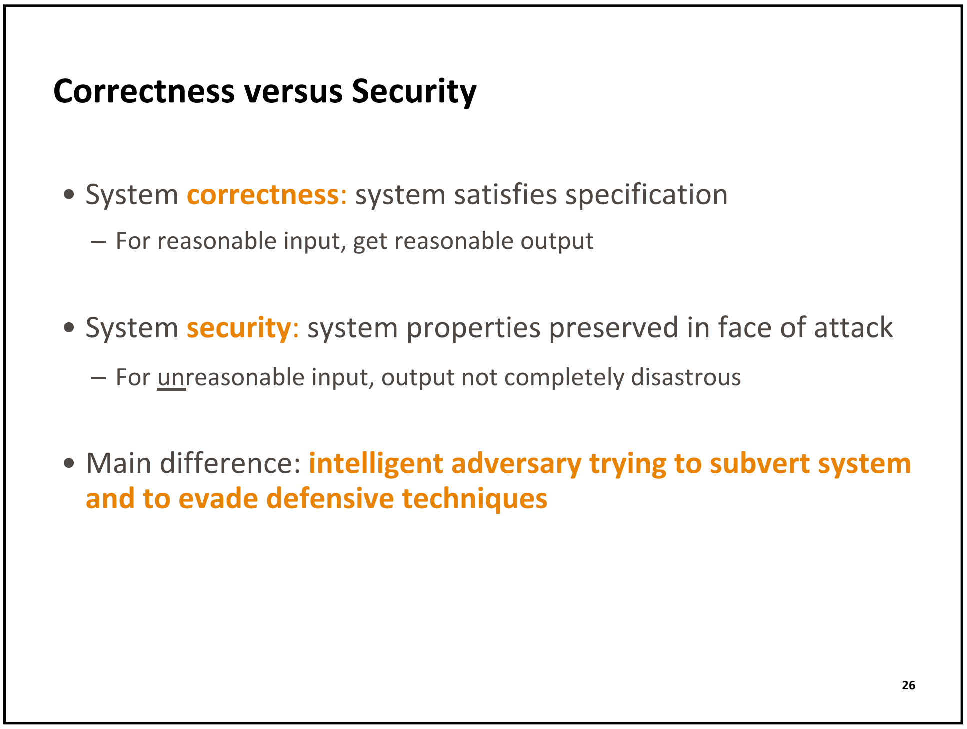
- Botnets
  - **Worker bots** running in the background on millions of compromised hosts
  - **Bot master** sending instructions to worker bots via **command & control** nodes
  - Possible instructions: **propagate**, send **spam**, conduct **DDoS**, **mine Bitcoin**
- Pay-per-Install (**PPI**)
  - "Affiliate" programs rewarding miscreants for installing malware on end-hosts
  - Useful for bootstrapping botnets, sending spam, staging denial of service attacks, performing click fraud, hosting scam websites
- Distributed Denial of Service (DDoS)
  - Instruct a botnet to **direct a large amount of traffic** to the target
  - Leverage protocols that can **amplify traffic** (e.g. NTP, DNS)



Desirable Security Properties

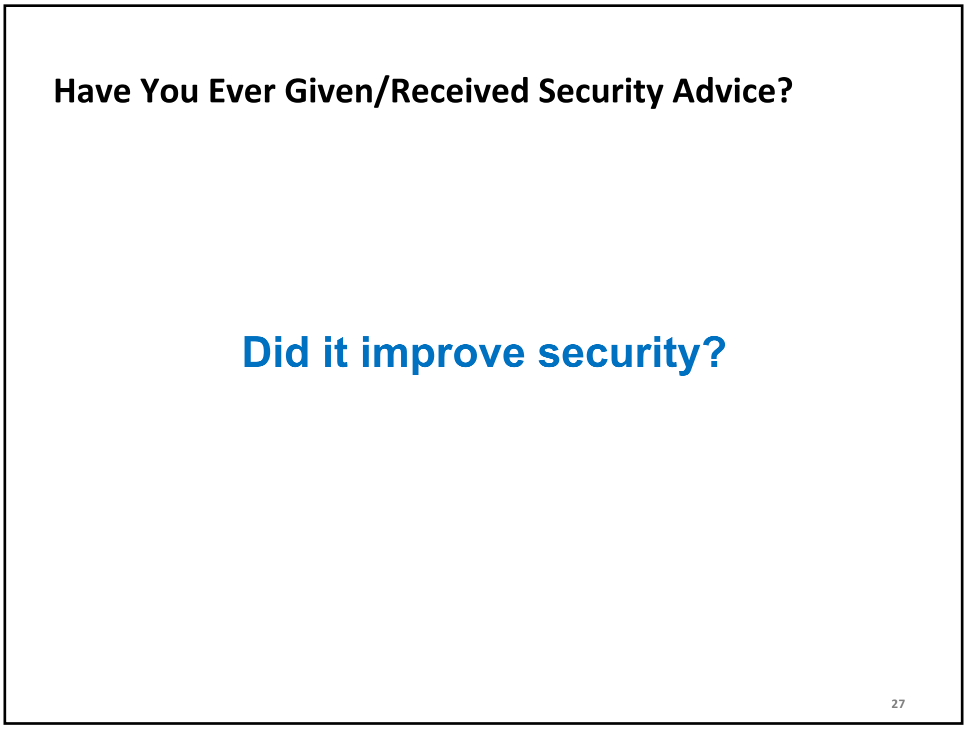
- Authenticity
- Confidentiality
- Integrity
- Availability
- Accountability and non-repudiation
- Access control
- Privacy

…



Correctness versus Security

- System **correctness**: system satisfies specification
  - For reasonable input, get reasonable output
- System **security**: system properties preserved in face of attack
  - For unreasonable input, output not completely disastrous
- Main difference: **intelligent adversary trying to subvert system and to evade defensive techniques**

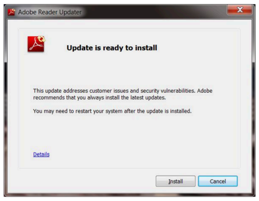


Have You Ever Given/Received Security Advice?

Did it improve security?

To Patch Or Not To Patch?

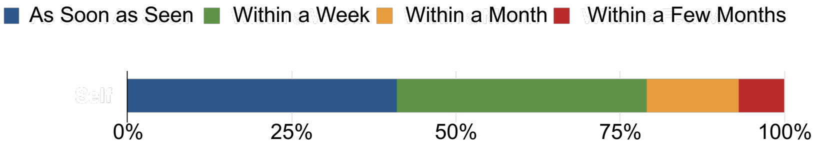
- Common advice:
  - Educate users about the importance of patching software vulnerabilities



Update is ready to install

This update addresses customer issues and security vulnerabilities. Adobe recommends that you always install the latest updates.

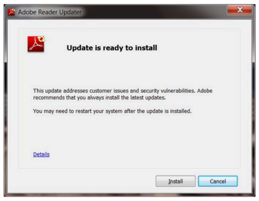
You may need to restart your system after the update is installed.



E. Redmiles, Z. Zhu, D. Kuchhal, T. Dumitraş, and M. Mazurek, 'Asking for a Friend: Evaluating Response Biases in Security User Studies.' CCS 2018

To Patch Or Not To Patch?

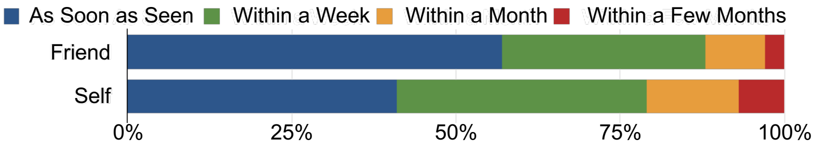
- Common advice:
  - Educate users about the importance of patching software vulnerabilities



Update is ready to install

This update addresses customer issues and security vulnerabilities. Adobe recommends that you always install the latest updates.

You may need to restart your system after the update is installed.



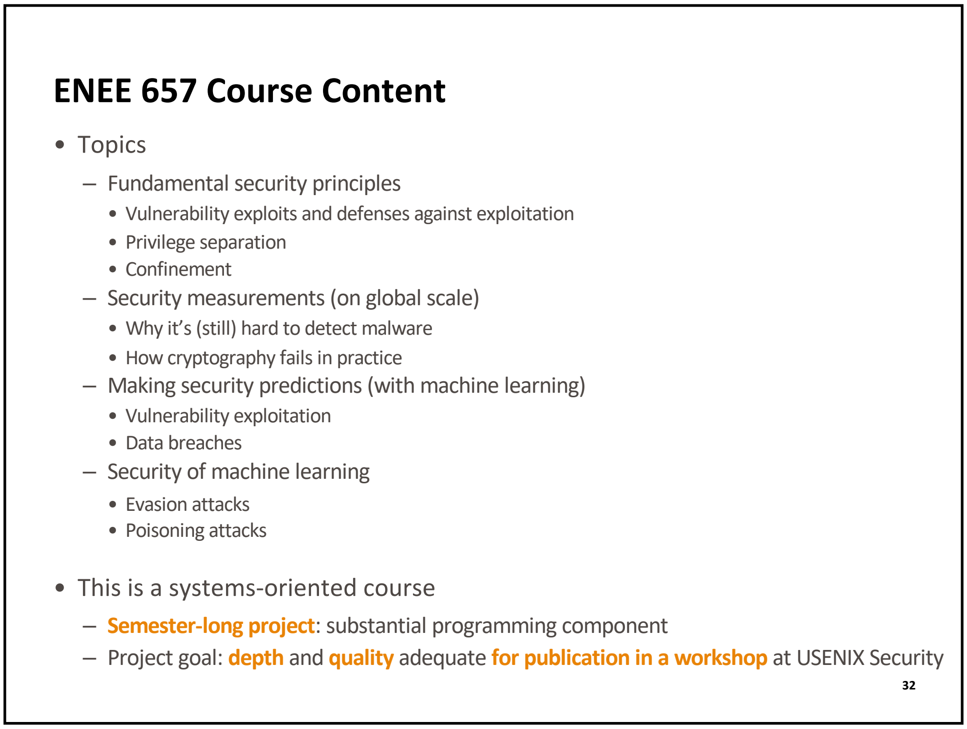
Security Advice: Perceived effectiveness != Actual effectiveness

E. Redmiles, Z. Zhu, D. Kuchhal, T. Dumitraş, and M. Mazurek, 'Asking for a Friend: Evaluating Response Biases in Security User Studies.' CCS 2018

ENEE 657 Logistics

ENEE 657 In A Nutshell

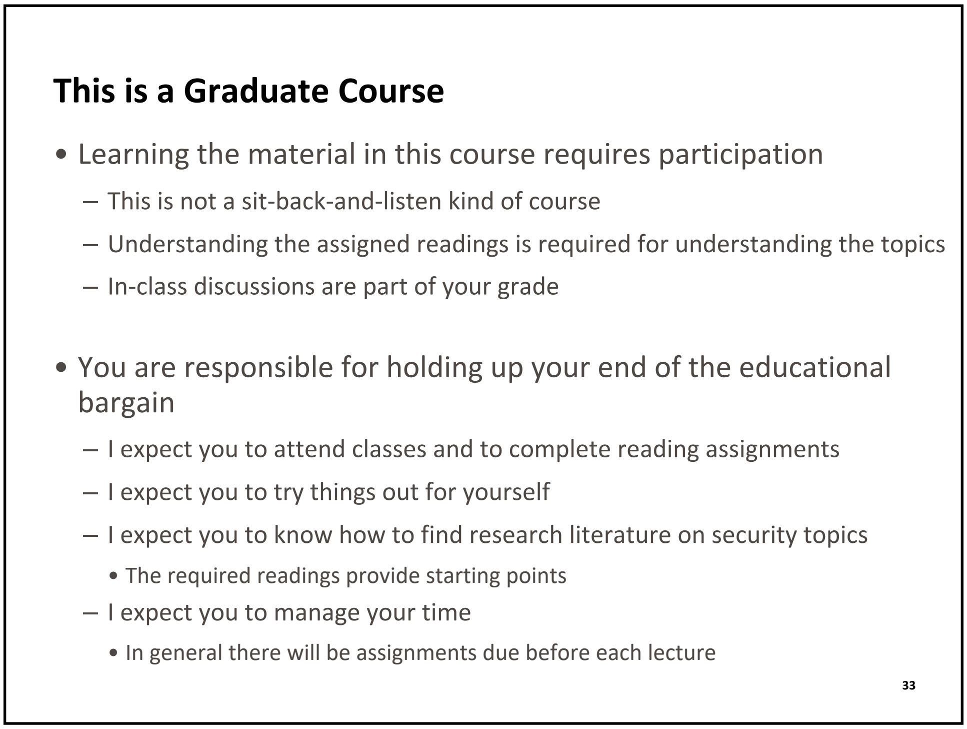
- Course objectives
  - Gain thorough **grounding** in computer security
    - Understand **attacks** and **defenses**
    - Learn to reason about their **effectiveness in the real world**
  - Prepare you to collaborate with **security researchers**
    - **Think critically** about recent advances in security
    - Learn how to **discuss** security topics intelligently
- What ENEE 657 is not
  - A course on cryptography
  - A course on theoretical security



ENEE 657 Course Content

- Topics
  - Fundamental security principles
    - Vulnerability exploits and defenses against exploitation
    - Privilege separation
    - Confinement
  - Security measurements (on global scale)
    - Why it’s (still) hard to detect malware
    - How cryptography fails in practice
  - Making security predictions (with machine learning)
    - Vulnerability exploitation
    - Data breaches
  - Security of machine learning
    - Evasion attacks
    - Poisoning attacks
- This is a systems-oriented course
  - **Semester-long project**: substantial programming component
  - Project goal: **depth** and **quality** adequate **for publication in a workshop** at USENIX Security

32



This is a Graduate Course

- Learning the material in this course requires participation
  - This is not a sit-back-and-listen kind of course
  - Understanding the assigned readings is required for understanding the topics
  - In-class discussions are part of your grade
- You are responsible for holding up your end of the educational bargain
  - I expect you to attend classes and to complete reading assignments
  - I expect you to try things out for yourself
  - I expect you to know how to find research literature on security topics
    - The required readings provide starting points
  - I expect you to manage your time
    - In general there will be assignments due before each lecture

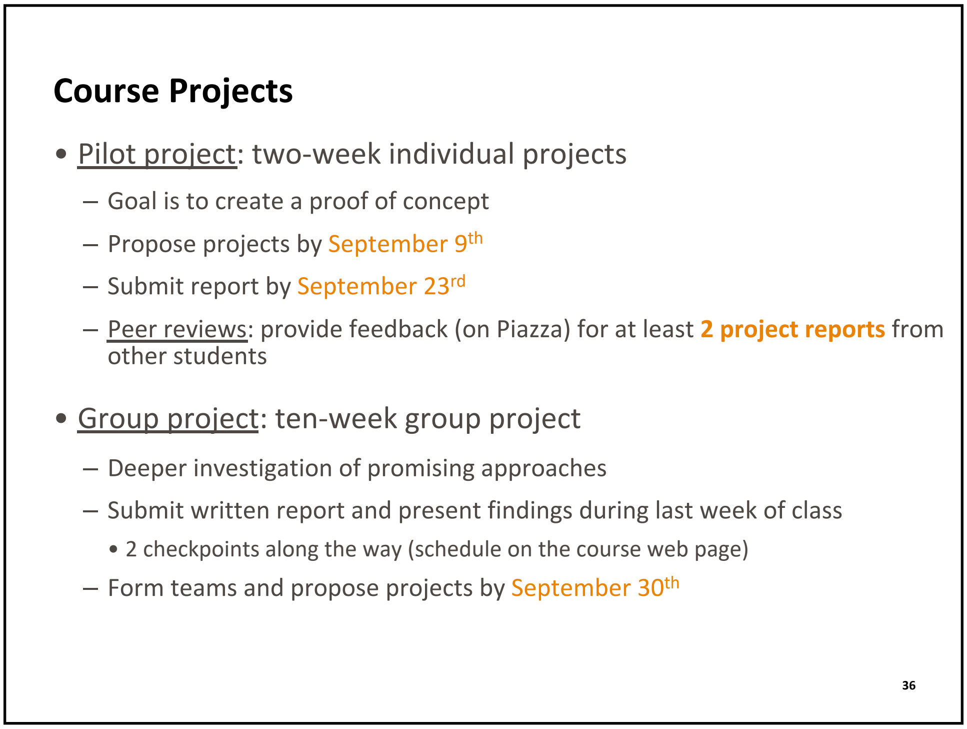
33

Homeworks

- Goal: refresh background material
  - Buffer overflow
  - Data analytics
- First homework
  - Will introduce the material on Wednesday
  - Homework will be due on September 6th

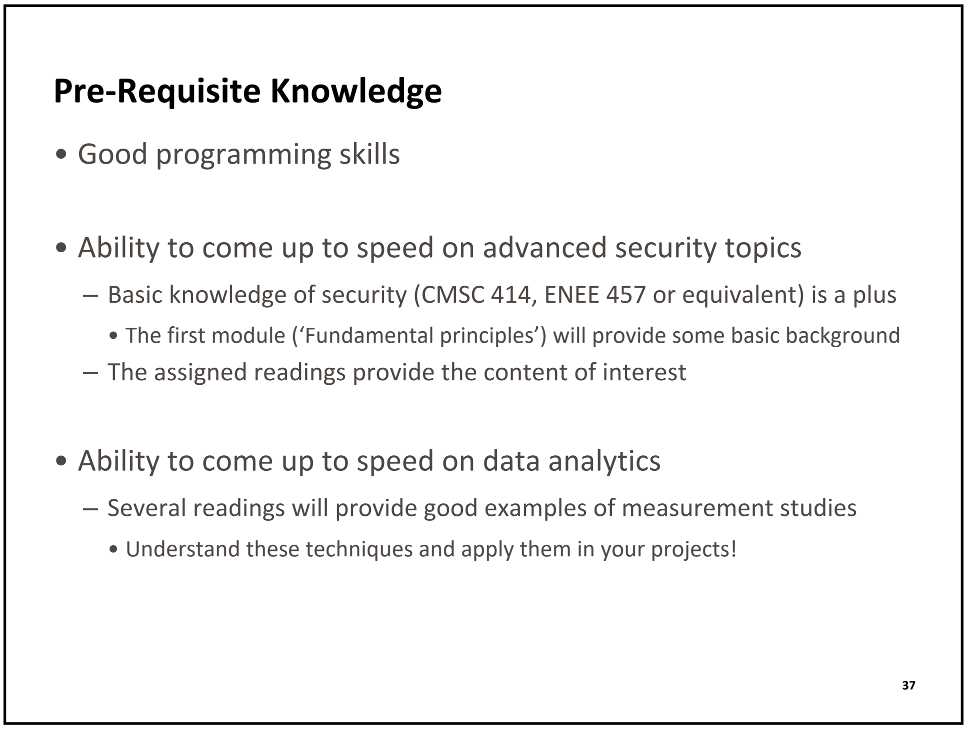
Reading Assignments

- Readings: 1-2 papers before each lecture
  - Not light reading – some papers require several readings to understand
  - Check course web page (still in flux) for next readings and links to papers
- Paper critiques: post a critique of each paper on Piazza
  - Provide feedback on at least 2 critiques from other students, to start the debate
  - More on this later
- In-class paper discussions: debate contributions and weaknesses
  - Structured discussion, inspired by competitive debating
  - Open discussion with whole class afterward
  - More on this later
- Discussion summaries: scribe posts summary to Piazza
  - More on this later



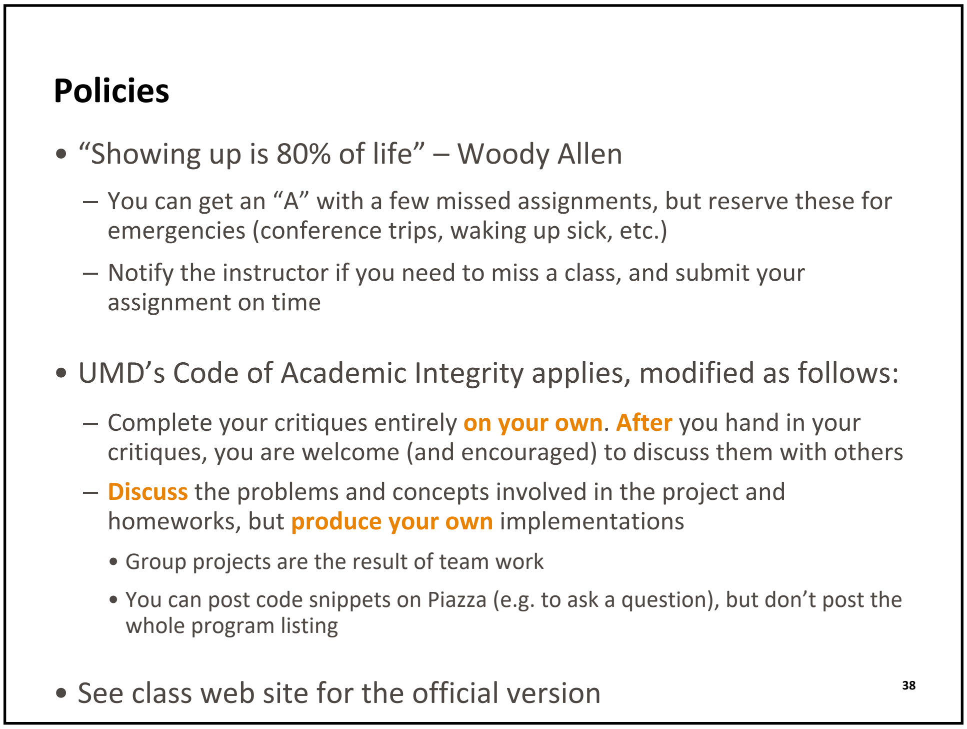
Course Projects

- Pilot project: two-week individual projects
  - Goal is to create a proof of concept
  - Propose projects by September 9th
  - Submit report by September 23rd
  - Peer reviews: provide feedback (on Piazza) for at least **2 project reports** from other students
- Group project: ten-week group project
  - Deeper investigation of promising approaches
  - Submit written report and present findings during last week of class
    - 2 checkpoints along the way (schedule on the course web page)
  - Form teams and propose projects by September 30th



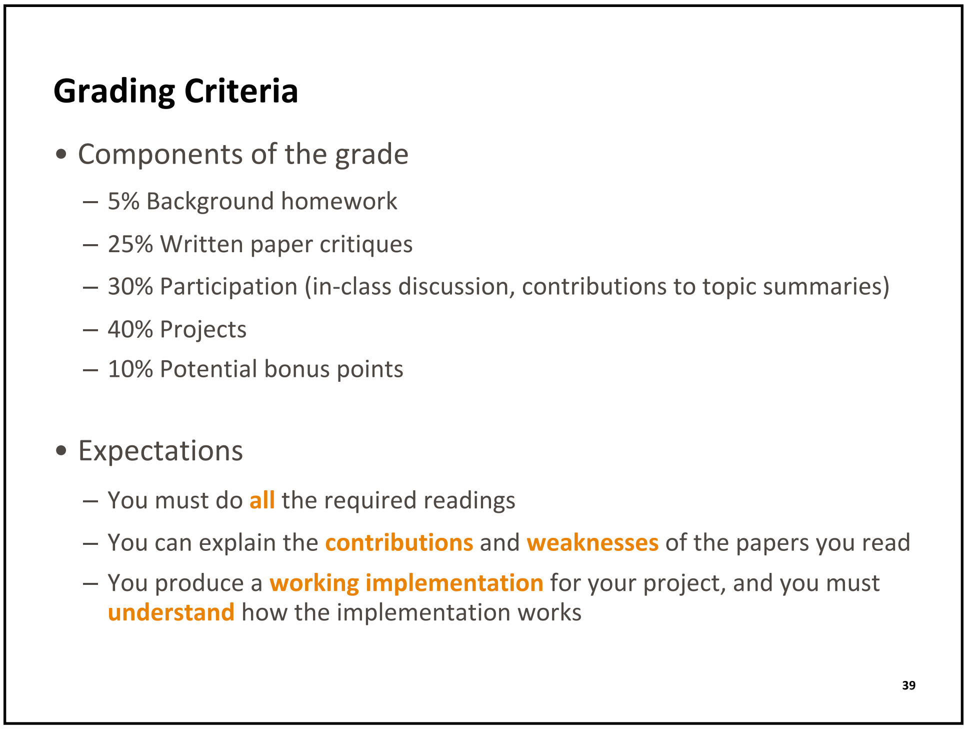
Pre-Requisite Knowledge

- Good programming skills
- Ability to come up to speed on advanced security topics
  - Basic knowledge of security (CMSC 414, ENEE 457 or equivalent) is a plus
    - The first module (‘Fundamental principles’) will provide some basic background
  - The assigned readings provide the content of interest
- Ability to come up to speed on data analytics
  - Several readings will provide good examples of measurement studies
    - Understand these techniques and apply them in your projects!



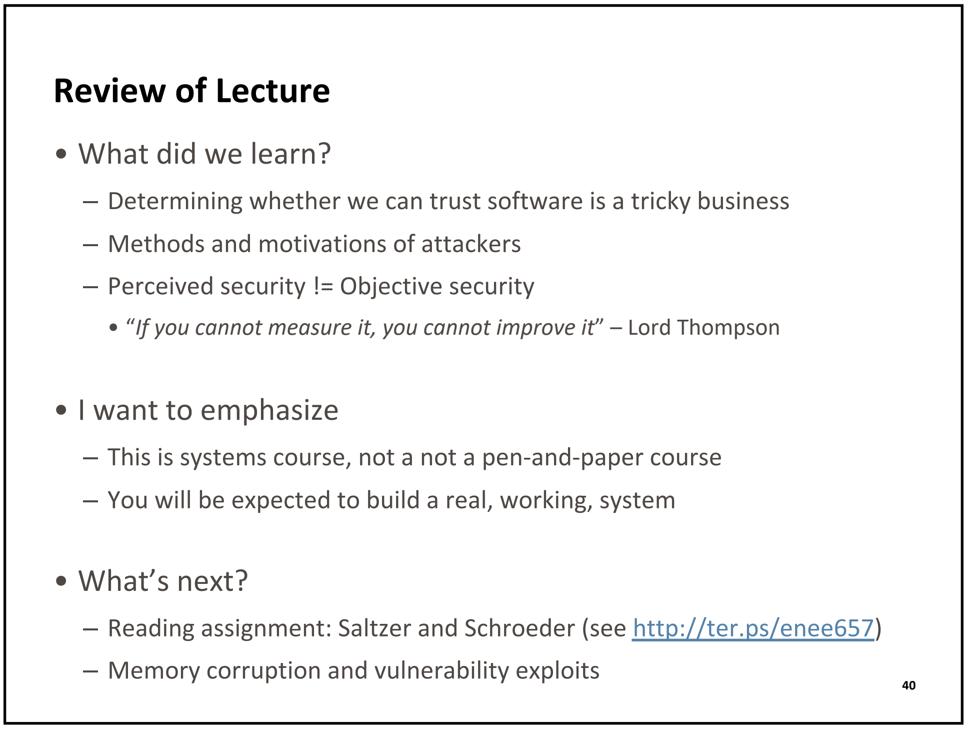
Policies

- “Showing up is 80% of life” – Woody Allen
  - You can get an “A” with a few missed assignments, but reserve these for emergencies (conference trips, waking up sick, etc.)
  - Notify the instructor if you need to miss a class, and submit your assignment on time
- UMD’s Code of Academic Integrity applies, modified as follows:
  - Complete your critiques entirely **on your own**. **After** you hand in your critiques, you are welcome (and encouraged) to discuss them with others
  - **Discuss** the problems and concepts involved in the project and homeworks, but **produce your own** implementations
    - Group projects are the result of team work
    - You can post code snippets on Piazza (e.g. to ask a question), but don’t post the whole program listing
- See class web site for the official version



Grading Criteria

- Components of the grade
  - 5% Background homework
  - 25% Written paper critiques
  - 30% Participation (in-class discussion, contributions to topic summaries)
  - 40% Projects
  - 10% Potential bonus points
- Expectations
  - You must do **all** the required readings
  - You can explain the **contributions** and **weaknesses** of the papers you read
  - You produce a **working implementation** for your project, and you must **understand** how the implementation works



Review of Lecture

- What did we learn?
  - Determining whether we can trust software is a tricky business
  - Methods and motivations of attackers
  - Perceived security != Objective security
    - “*If you cannot measure it, you cannot improve it*” – Lord Thompson

- I want to emphasize
  - This is systems course, not a not a pen-and-paper course
  - You will be expected to build a real, working, system

- What’s next?
  - Reading assignment: Saltzer and Schroeder (see http://ter.ps/enee657)
  - Memory corruption and vulnerability exploits