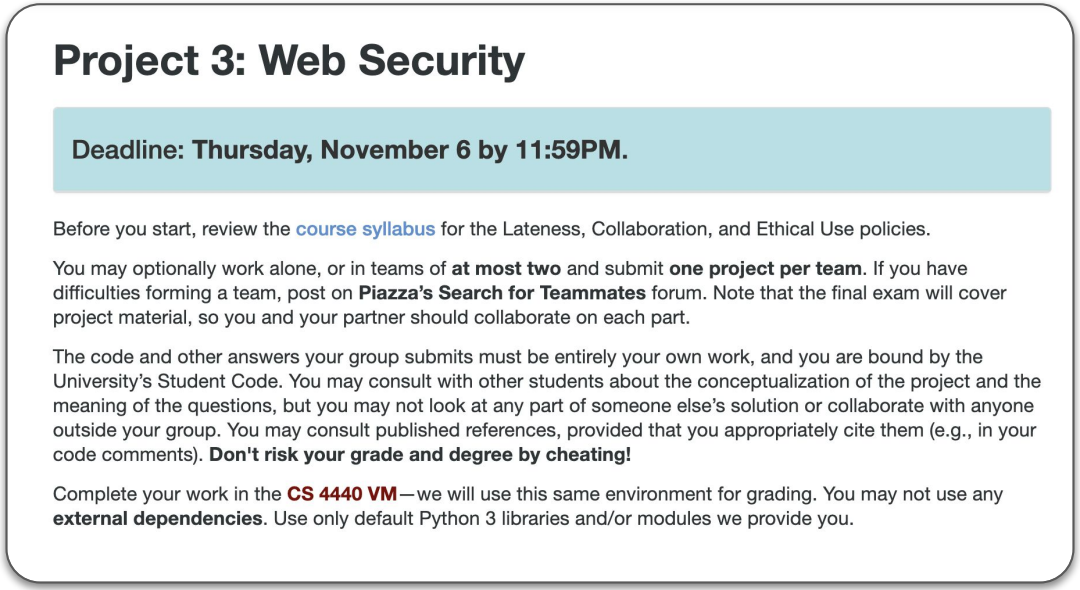
Week 11: Lecture A

Secure Authentication

Tuesday, November 4, 2025

Announcements

- **Project 3: WebSec** released
  - **Deadline: this Thursday** by 11:59PM



Project 3: Web Security

Deadline: Thursday, November 6 by 11:59PM.

Before you start, review the course syllabus for the Lateness, Collaboration, and Ethical Use policies.

You may optionally work alone, or in teams of **at most two** and submit **one project per team**. If you have difficulties forming a team, post on **Piazza's Search for Teammates** forum. Note that the final exam will cover project material, so you and your partner should collaborate on each part.

The code and other answers your group submits must be entirely your own work, and you are bound by the University's Student Code. You may consult with other students about the conceptualization of the project and the meaning of the questions, but you may not look at any part of someone else's solution or collaborate with anyone outside your group. You may consult published references, provided that you appropriately cite them (e.g., in your code comments). **Don't risk your grade and degree by cheating!**

Complete your work in the **CS 4440 VM**—we will use this same environment for grading. You may not use any **external dependencies**. Use only default Python 3 libraries and/or modules we provide you.

Project 3 progress

Working on Part 1

0%

Finished Part 1, working on Part 2

0%

Finished Part 2, working on Part 3

0%

Finished with everything!

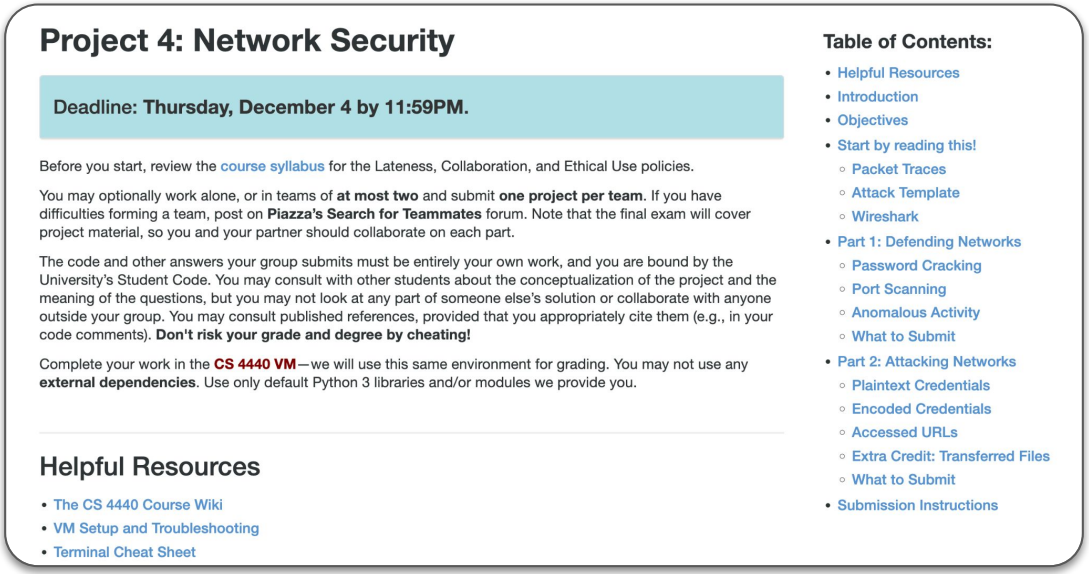
0%

Haven't started yet :(

0%

Announcements

- **Project 4: NetSec** released
  - **Deadline:** Thursday, December 4th by 11:59PM



Project 4: Network Security

Deadline: Thursday, December 4 by 11:59PM.

Before you start, review the course syllabus for the Lateness, Collaboration, and Ethical Use policies.

You may optionally work alone, or in teams of **at most two** and submit **one project per team**. If you have difficulties forming a team, post on **Piazza's Search for Teammates** forum. Note that the final exam will cover project material, so you and your partner should collaborate on each part.

The code and other answers your group submits must be entirely your own work, and you are bound by the University's Student Code. You may consult with other students about the conceptualization of the project and the meaning of the questions, but you may not look at any part of someone else's solution or collaborate with anyone outside your group. You may consult published references, provided that you appropriately cite them (e.g., in your code comments). **Don't risk your grade and degree by cheating!**

Complete your work in the **CS 4440 VM**—we will use this same environment for grading. You may not use any **external dependencies**. Use only default Python 3 libraries and/or modules we provide you.

Helpful Resources

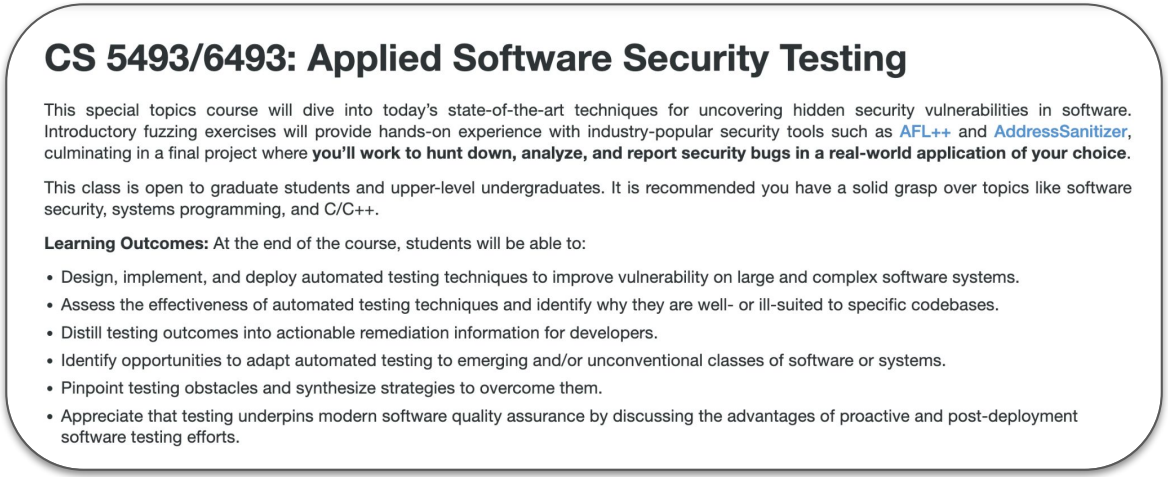
- The CS 4440 Course Wiki
- VM Setup and Troubleshooting
- Terminal Cheat Sheet

Table of Contents:

- Helpful Resources
- Introduction
- Objectives
- Start by reading this!
  - Packet Traces
  - Attack Template
  - Wireshark
- Part 1: Defending Networks
  - Password Cracking
  - Port Scanning
  - Anomalous Activity
  - What to Submit
- Part 2: Attacking Networks
  - Plaintext Credentials
  - Encoded Credentials
  - Accessed URLs
  - Extra Credit: Transferred Files
  - What to Submit
- Submission Instructions

Interested in fuzzing?

- **Spring 2026: CS 5493/6493: Applied Software Security Testing**
  - **Everything you'd ever want to know about fuzzing for finding security bugs!**
  - Course project: team up to fuzz **a real program** (of your choice), and find and report its bugs!
  - http://cs.utah.edu/~snagy/courses/cs5493/



CS 5493/6493: Applied Software Security Testing

This special topics course will dive into today's state-of-the-art techniques for uncovering hidden security vulnerabilities in software. Introductory fuzzing exercises will provide hands-on experience with industry-popular security tools such as AFL++ and AddressSanitizer, culminating in a final project where **you'll work to hunt down, analyze, and report security bugs in a real-world application of your choice**.

This class is open to graduate students and upper-level undergraduates. It is recommended you have a solid grasp over topics like software security, systems programming, and C/C++.

Learning Outcomes: At the end of the course, students will be able to:

- Design, implement, and deploy automated testing techniques to improve vulnerability on large and complex software systems.
- Assess the effectiveness of automated testing techniques and identify why they are well- or ill-suited to specific codebases.
- Distill testing outcomes into actionable remediation information for developers.
- Identify opportunities to adapt automated testing to emerging and/or unconventional classes of software or systems.
- Pinpoint testing obstacles and synthesize strategies to overcome them.
- Appreciate that testing underpins modern software quality assurance by discussing the advantages of proactive and post-deployment software testing efforts.

Announcements

utahsec

See Discord for meeting info!

utahsec.cs.utah.edu

Kahlert School of Computing

Graduate Program Open House

Why PhD?

Information session for prospective graduate students

- What to expect from graduate school
- Reasons to pursue graduate career
- Perspective of alumni and current students
- How to prepare your application (and a statement of purpose)

RSVP / Zoom links:

November 14, 3:00pm – 5:00pm

MEB 3147 (LCR) and Zoom (free pizza—please RSVP)

Questions?

Last time on CS 4440...

Attacks on Security Properties
Denial of Service Attacks

Basic Security Properties

- **Confidentiality: ???**
- **Authenticity: ???**
- **Integrity: ???**
- **Access Control: ???**
- **Availability: ???**

Basic Security Properties

- **Confidentiality:** Concealment of information or resources
  - **Attacks: intercept credentials, info**
- **Authenticity:** Identification and assurance of info origin
  - **Attacks: SMTP header spoofing**
- **Integrity:** Preventing improper and unauthorized changes
  - **Attacks: tampering HTML over HTTP**
- **Access Control:** Enforce who is allowed access to what
  - **Attacks: web app code injection**
- **Availability:** Ability to use desired information or resource
  - **Attacks: denial of service**

DoS: Denial of Service

- **Goal: ???**

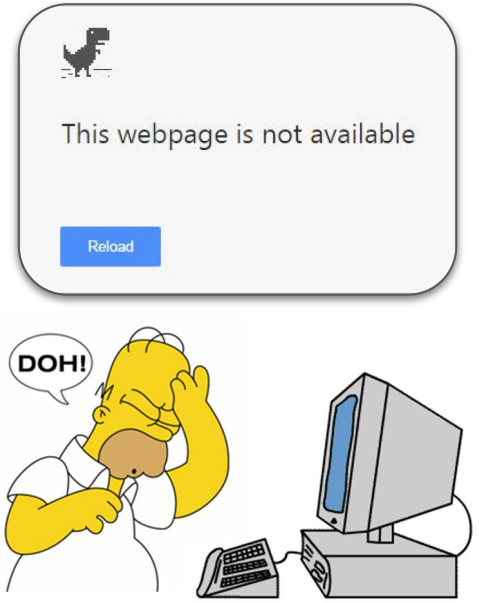
DoS: Denial of Service

- **Goal:** make a service unusable, usually by overloading the server or network
- **How?**

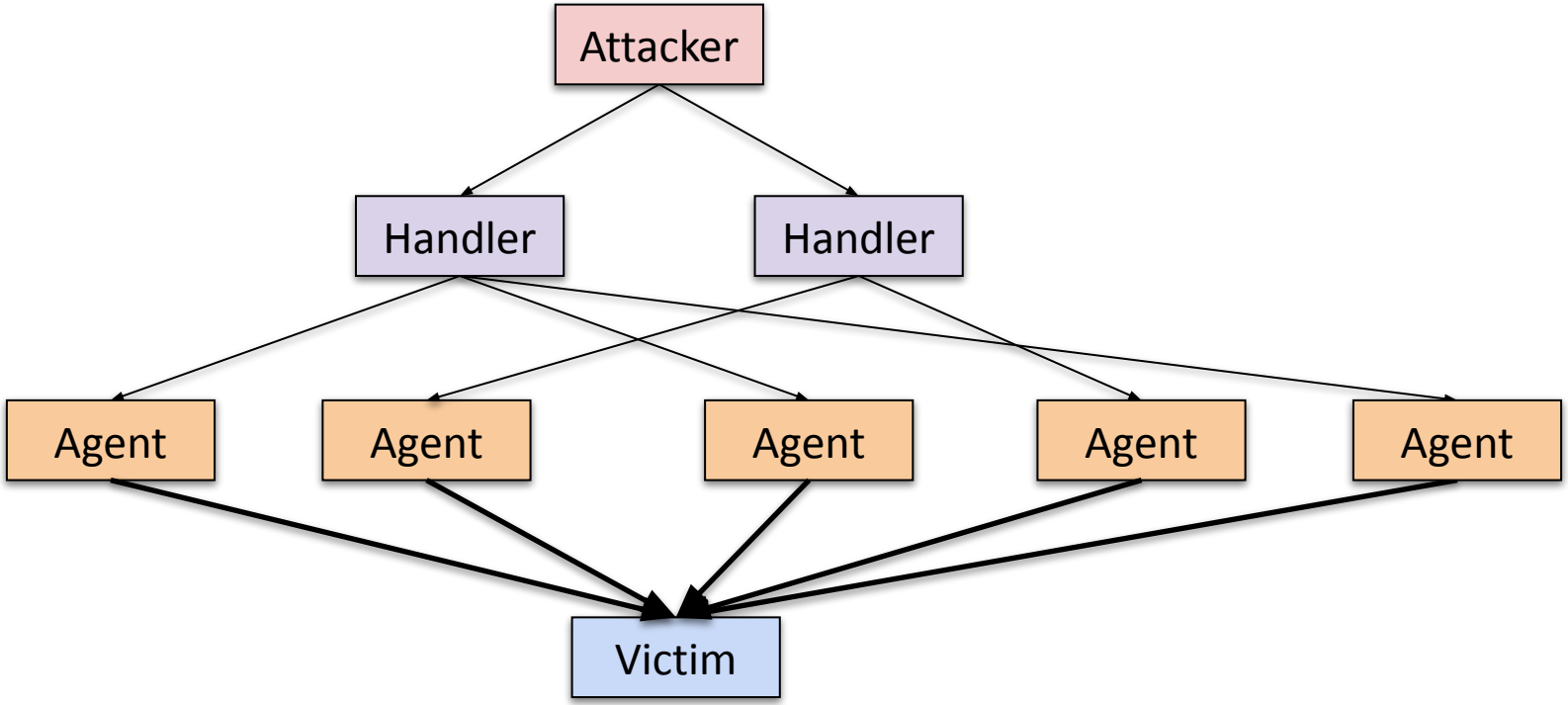
DoS: Denial of Service

- **Goal:** make a service unusable, usually by overloading the server or network

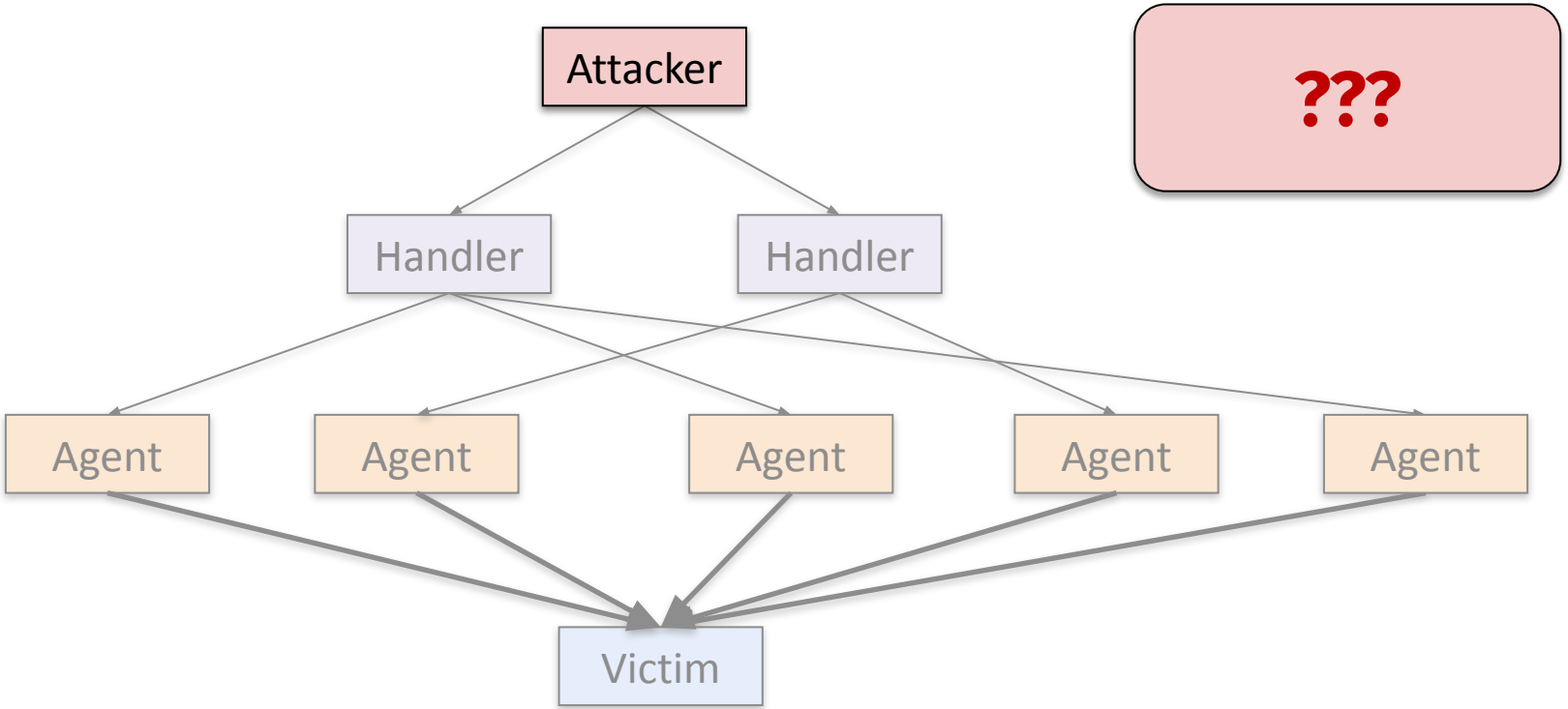
- **How?**
  - Trigger the host to **crash**
    - Application-based DoS
    - Memory corruption
  - Consume host's **resources**
    - TCP SYN floods
    - ICMP ECHO (ping) floods
  - Consume host's **bandwidth**
    - UDP floods
    - ICMP floods



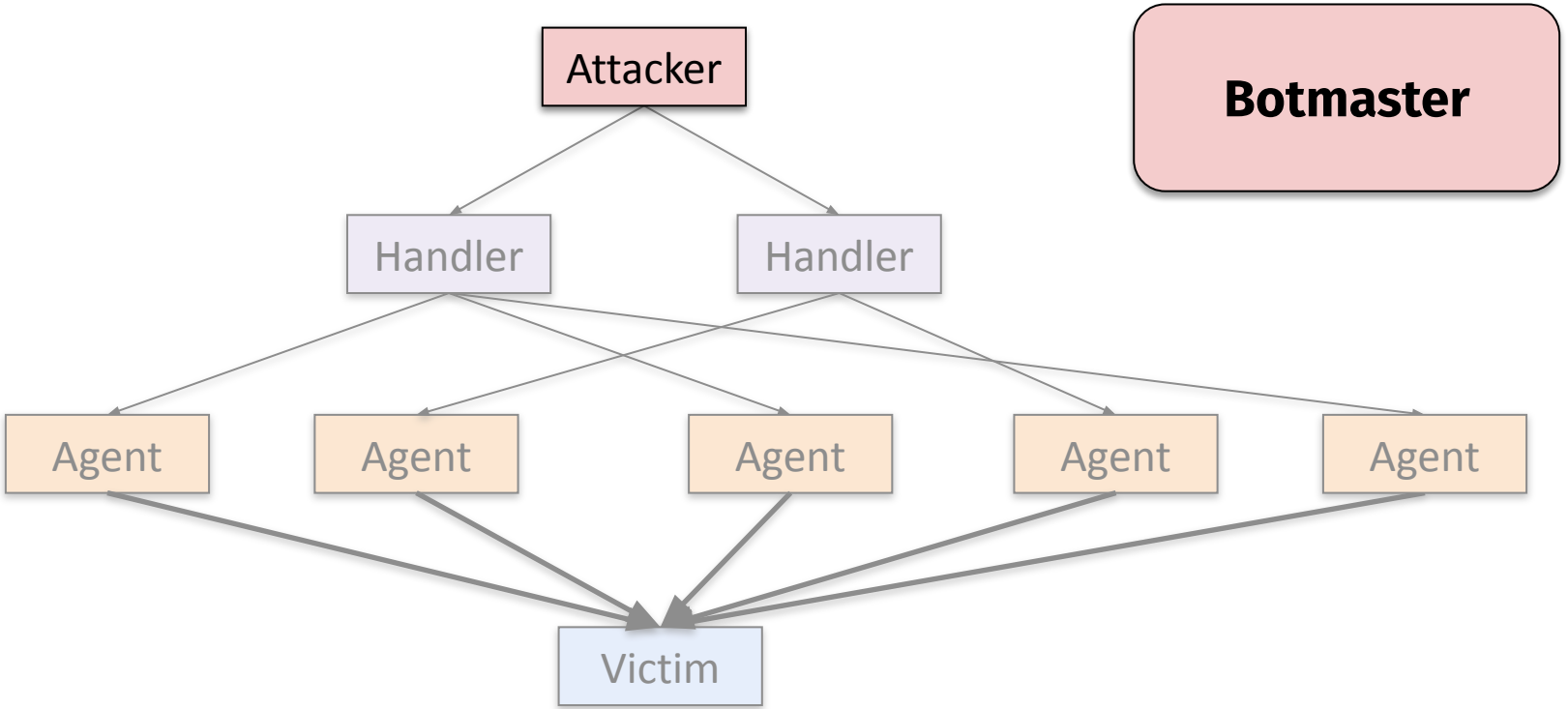
Distributed DoS Attacks (DDoS)



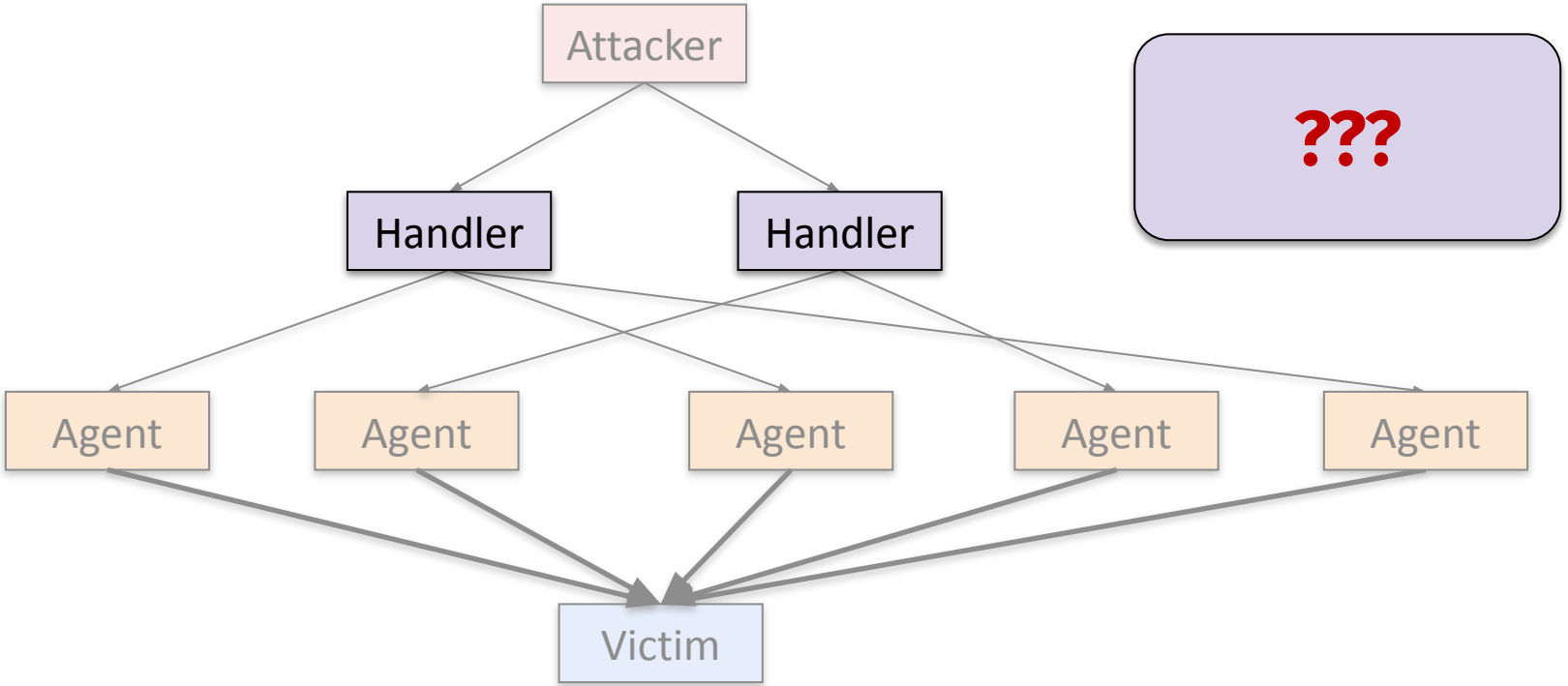
Distributed DoS Attacks (DDoS)



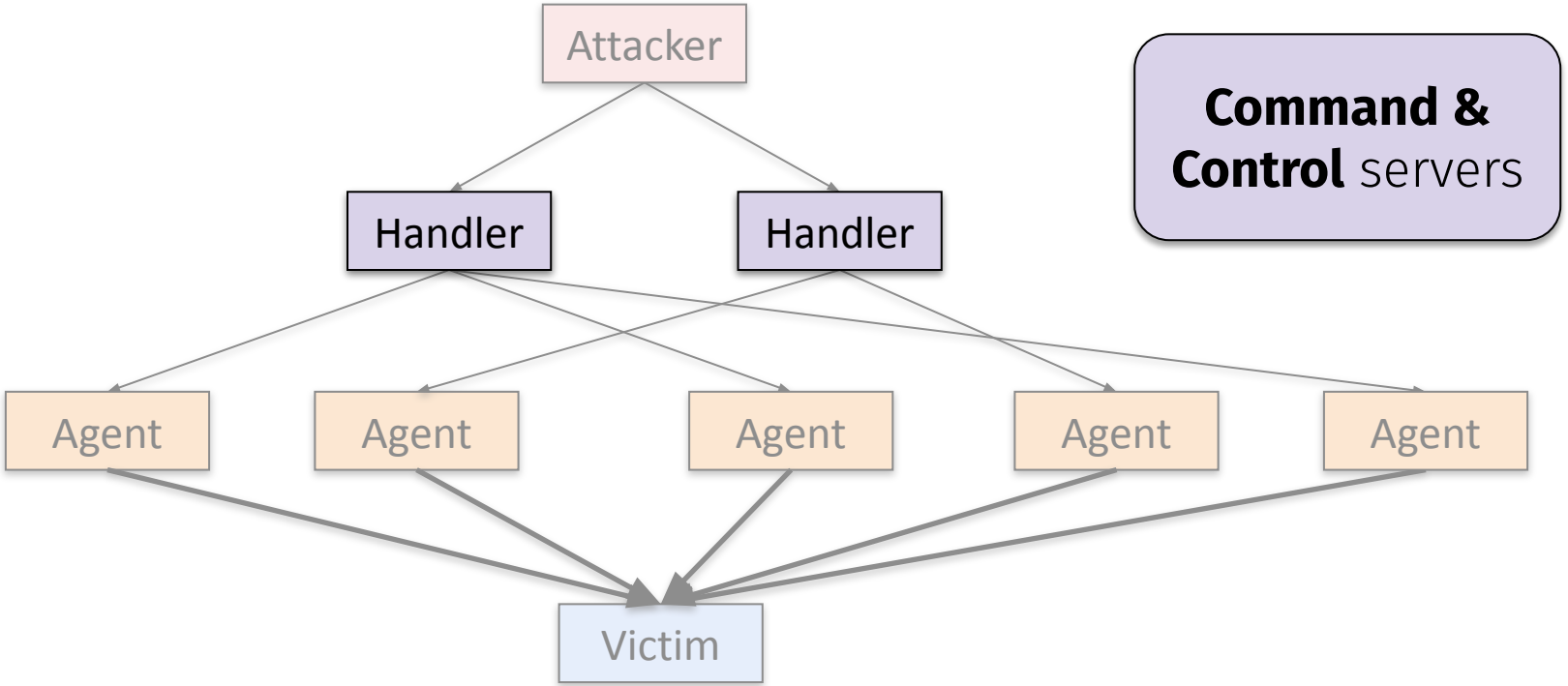
Distributed DoS Attacks (DDoS)



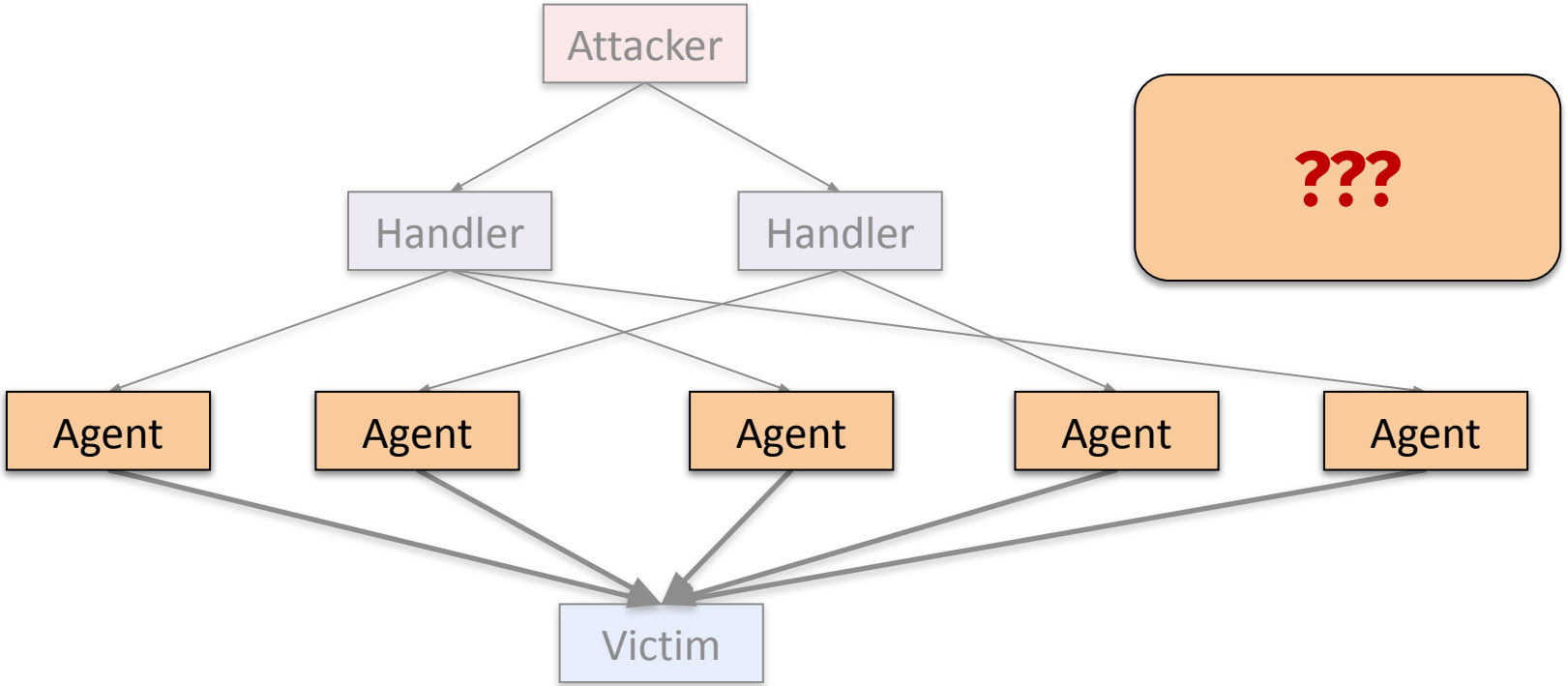
Distributed DoS Attacks (DDoS)



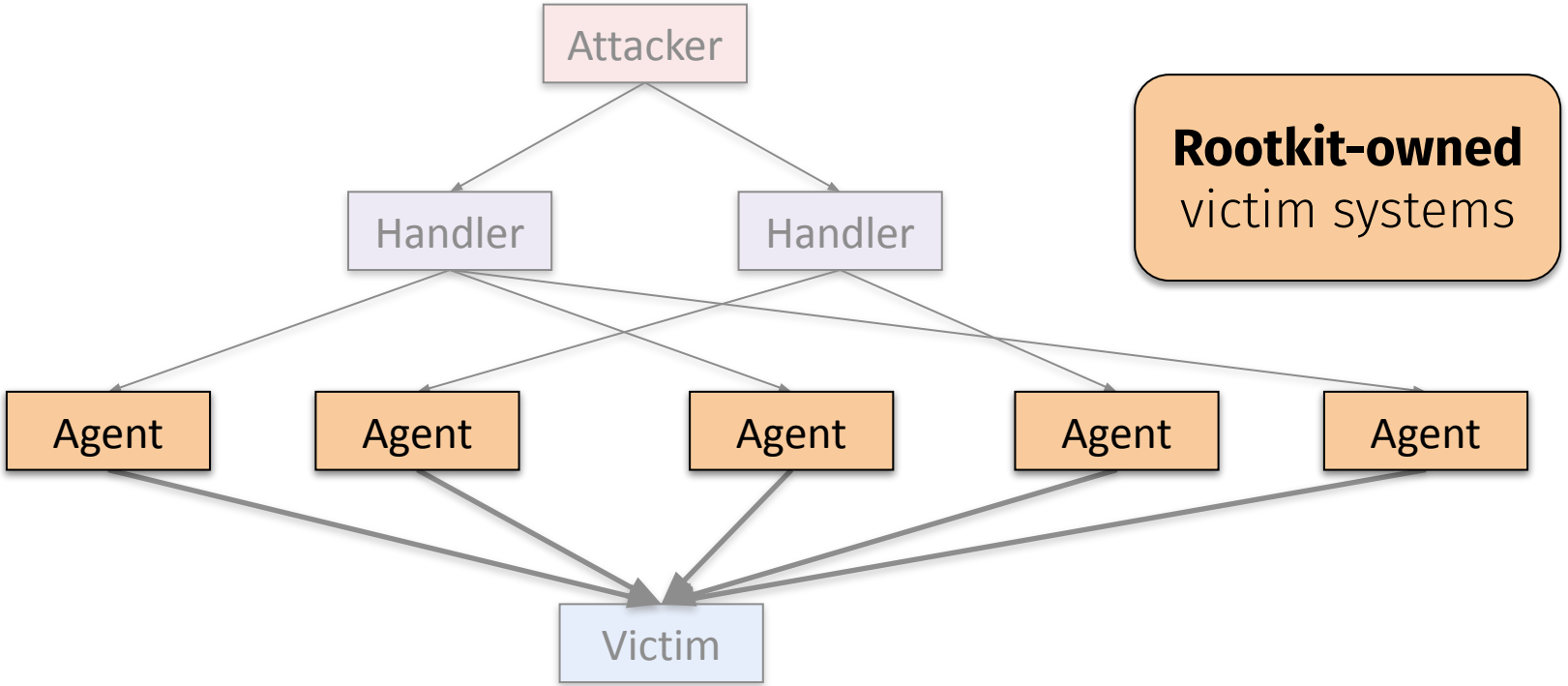
Distributed DoS Attacks (DDoS)



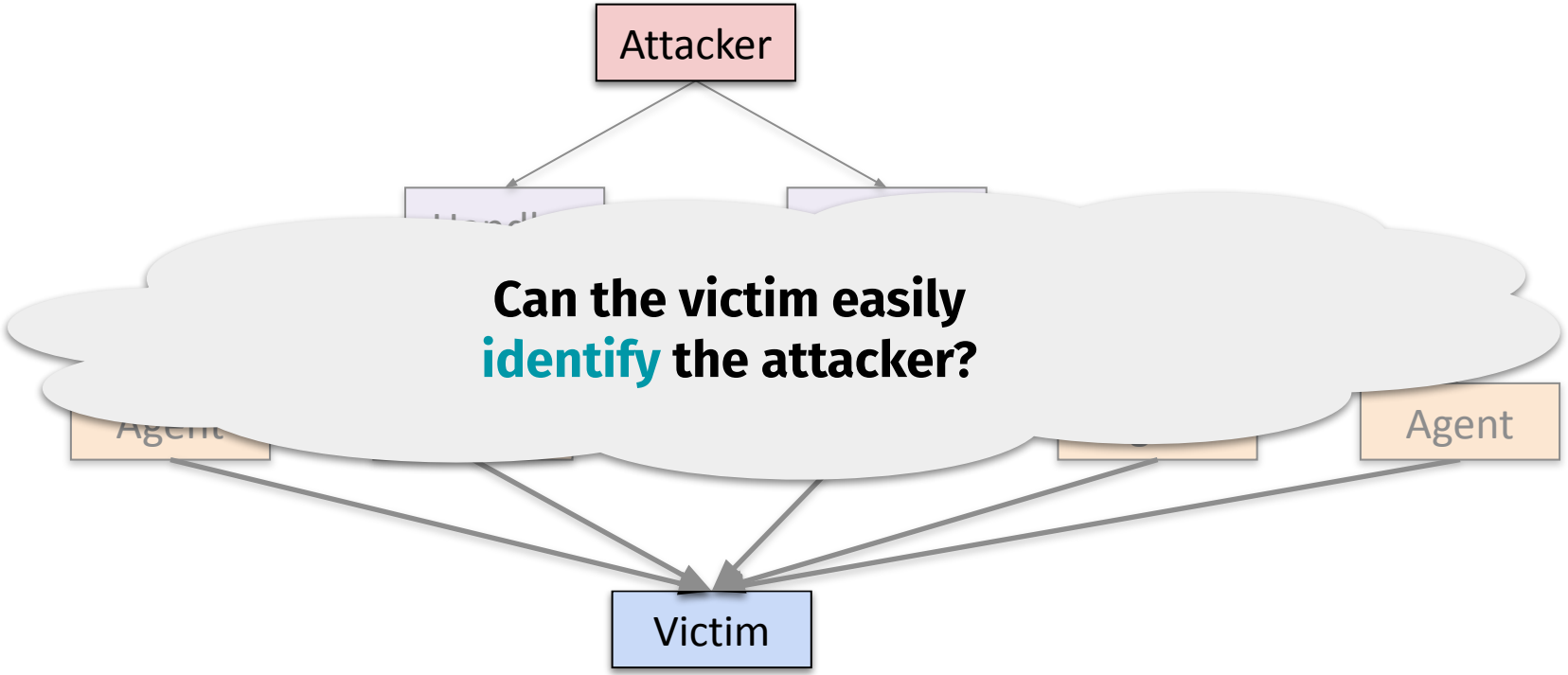
Distributed DoS Attacks (DDoS)



Distributed DoS Attacks (DDoS)

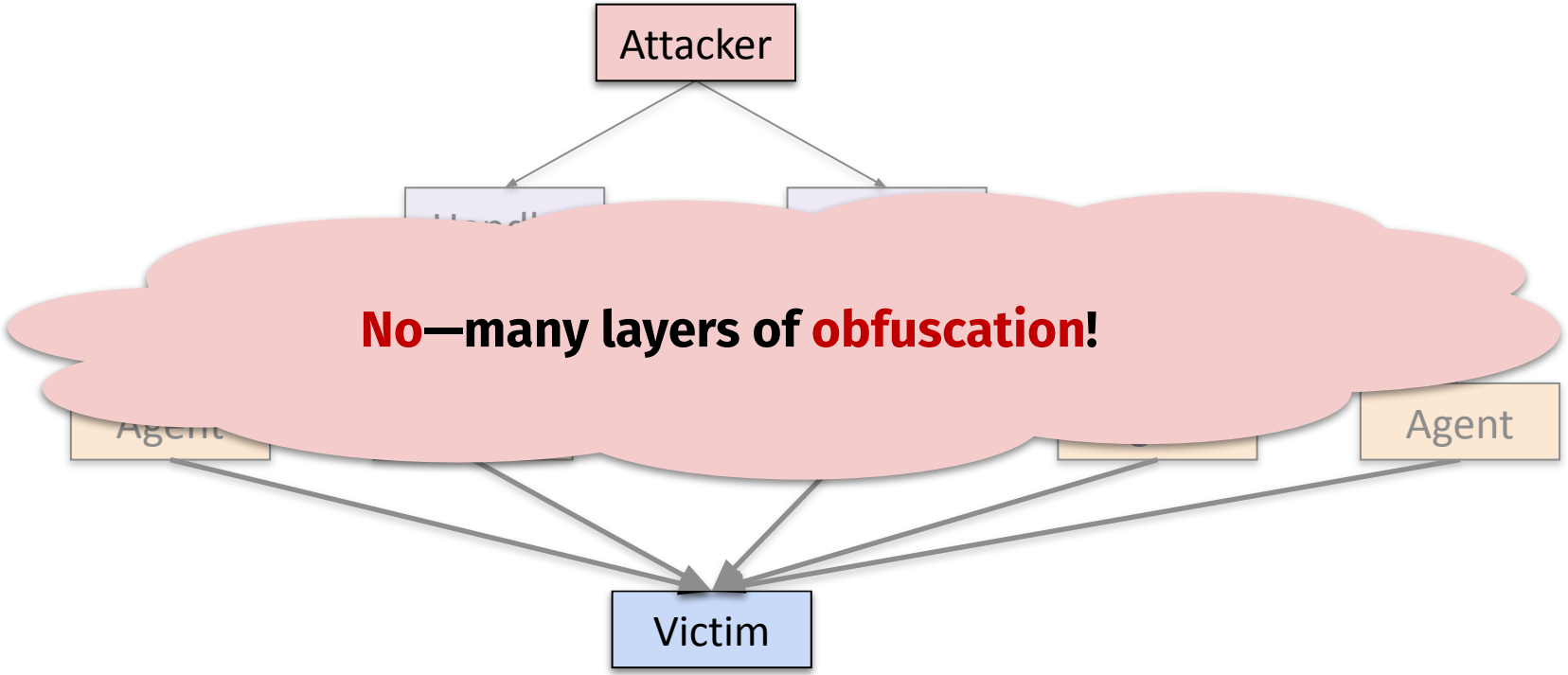


Distributed DoS Attacks (DDoS)



Can the victim easily identify the attacker?

Distributed DoS Attacks (DDoS)



Advanced DoS Strategies

- **Reflection:**
  - ???

Advanced DoS Strategies

- **Reflection:**
  - IP spoofing to redirect response to a victim
- **Amplification:**
  - **???**

Advanced DoS Strategies

- **Reflection:**
  - IP spoofing to redirect response to a victim
- **Amplification:**
  - Technique that increases the amount of traffic or packet size that the victim sees versus what the attacker originally sent
- **How do these make detection harder?**
  - **???**

Advanced DoS Strategies

- **Reflection:**
  - IP spoofing to redirect response to a victim
- **Amplification:**
  - Technique that increases the amount of traffic or packet size that the victim sees versus what the attacker originally sent
- **How do these make detection harder?**
  - Source remains **obfuscated**
  - Source constantly **changes**

DDoS or legitimate traffic?

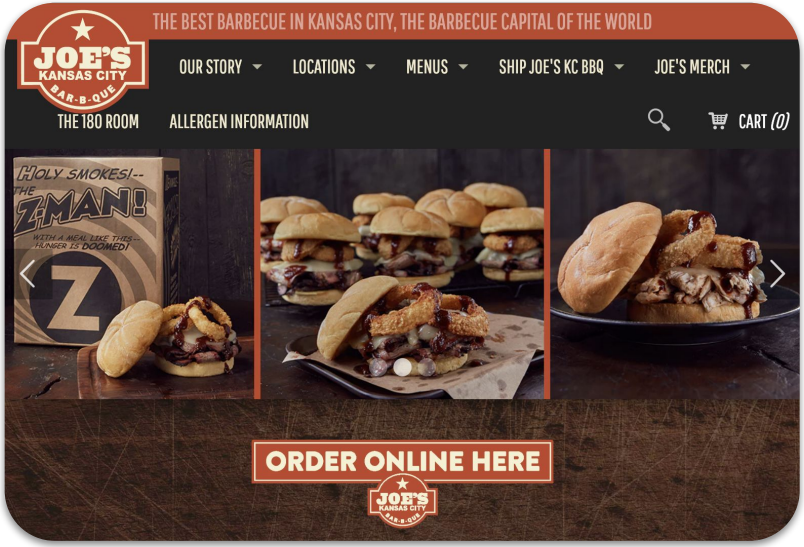
reddit

u/TheRealAndyReid

"OMG... **Joe's in KC** serves the BEST **brisket sandwich**"

"Ooooh!" "Click!" "Order now!" "Hungry!" "Yum!" "I must try!" "I'm from Cali and am clueless about BBQ!"

https://www.joeskc.com/



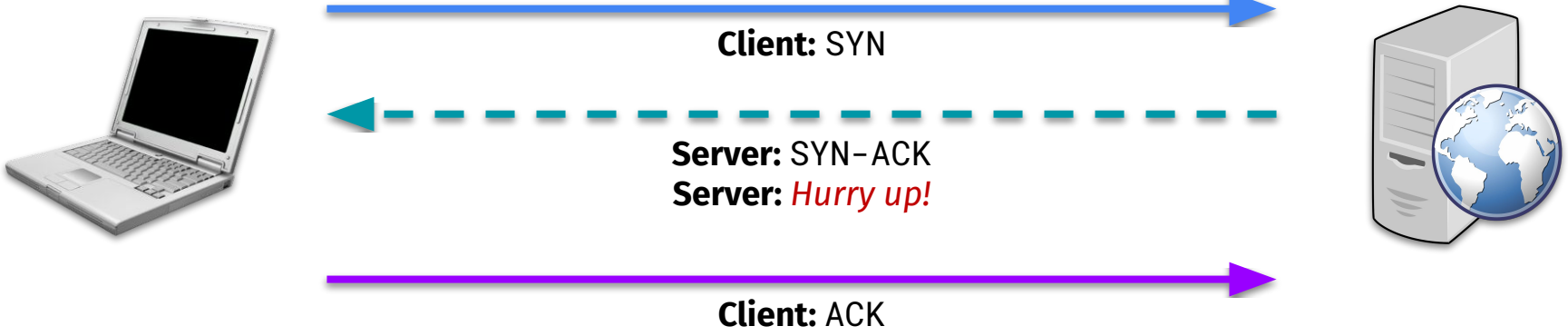
Distinguishing factors of legit traffic?

Nobody has responded yet.

Hang tight! Responses are coming in.

The TCP Three-way Handshake

- **Recall:** TCP is a **connection-oriented** protocol
  - Initiate with three-way "handshake": **SYN**, **SYN-ACK**, **ACK**
  - **Server waits until client responds with ACK**

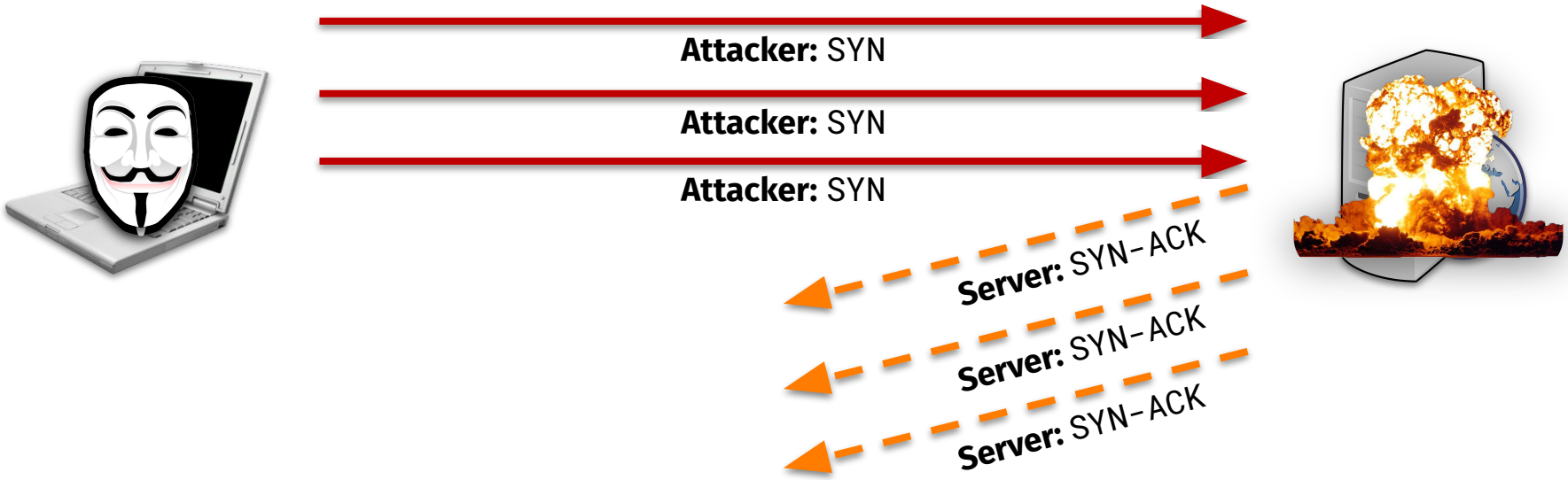


SYN Flooding Attack

- **Attack: ???**

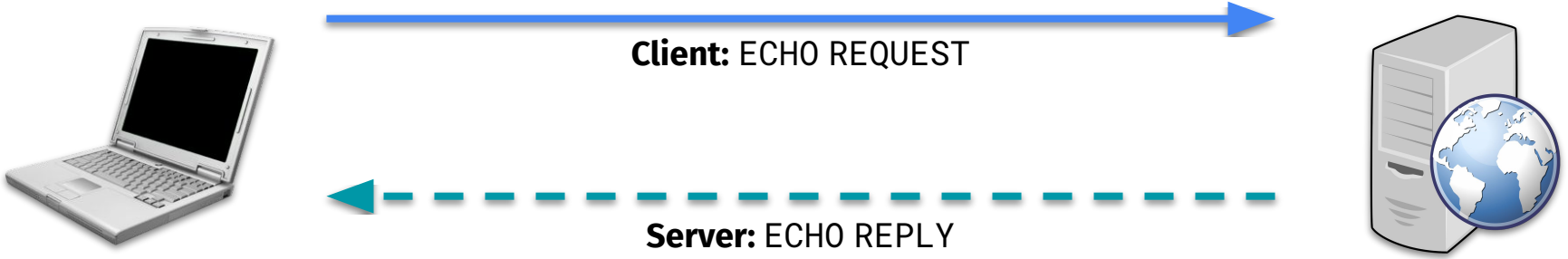
SYN Flooding Attack

- **Attack: spam SYN packets** to server, with **spoofed origin** address
  - Server's resources **completely reserved**—now **can't serve legitimate clients**



ICMP: Internet Control Message Protocol

- **ICMP:** pings to determine whether a system is **connected to the Internet**
  - Analogous to **“Hello, are you still there?”**

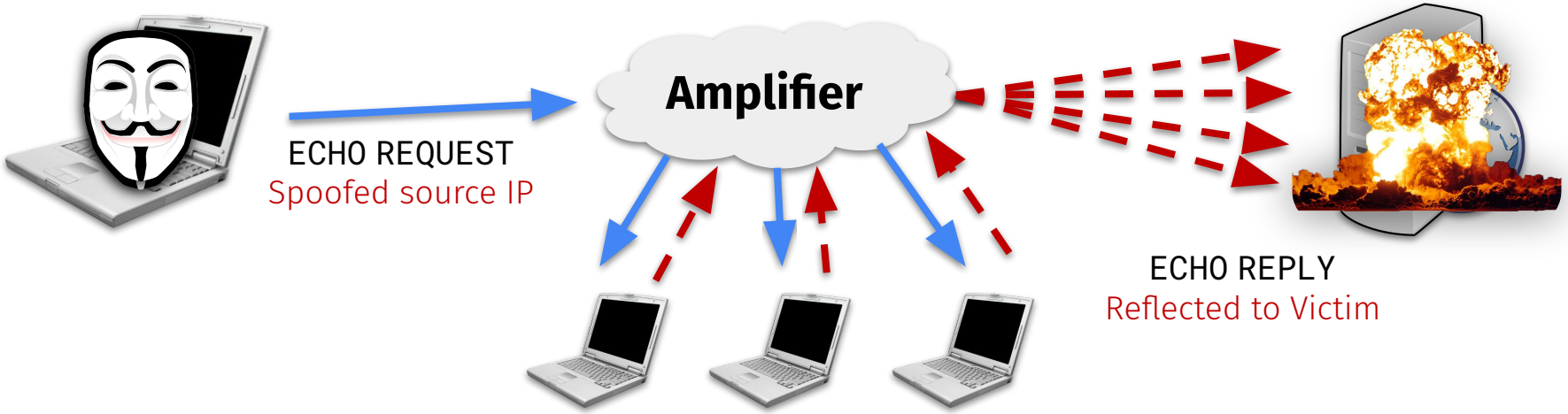


ICMP Smurf Attacks

- **Attack: ???**

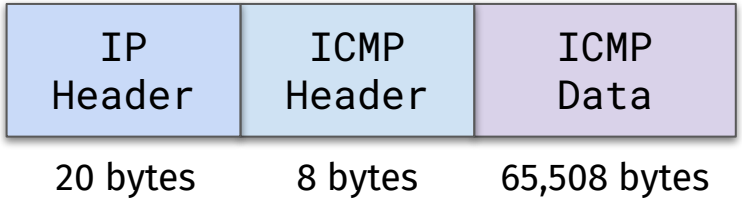
ICMP Smurf Attacks

- **Attack:** takes advantage of **broadcast-enabled hosts** to **amplify** attack
- Attacker spams **spoofed-source** ICMP requests, **reflected to victim's** IP



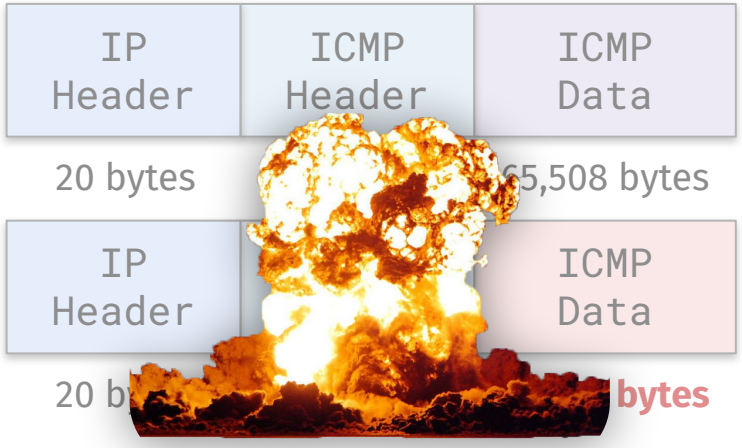
ICMP Ping of Death Attack

- **Internet Protocol:** IPV4 packets should be **less than 65,536 bytes**
  - Packets can be sent in **fragments** and **reassembled** by receiver
- **Attack: ???**



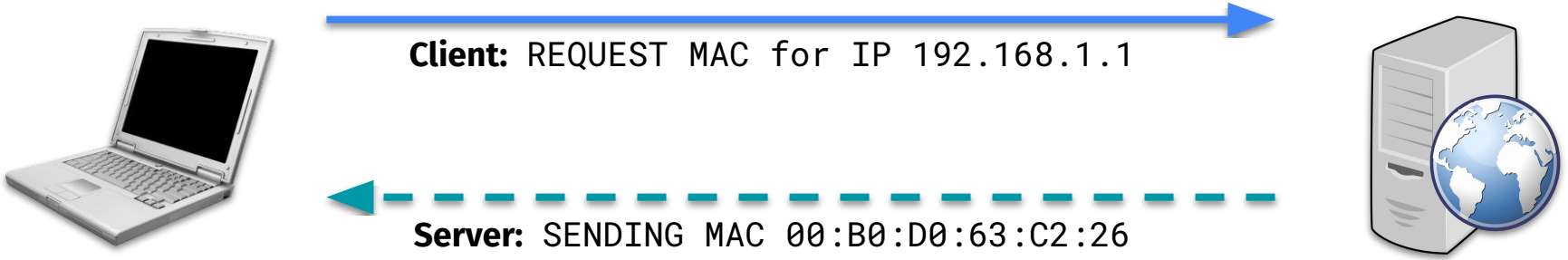
ICMP Ping of Death Attack

- **Internet Protocol:** IPV4 packets should be **less than 65,536 bytes**
  - Packets can be sent in **fragments** and **reassembled** by receiver
- **Attack:** send packet in fragments that **reassemble** to **64K+ bytes**
  - Many historical computer systems **could not handle larger packets**
- **Result:** crash by **buffer overflow**
  - Can't serve clients until restart!



ARP: Address Resolution Protocol

- **ARP:** query to **resolve the MAC address** given a desired host IP
  - How we know which **physical** address to transmit data to from its logical address

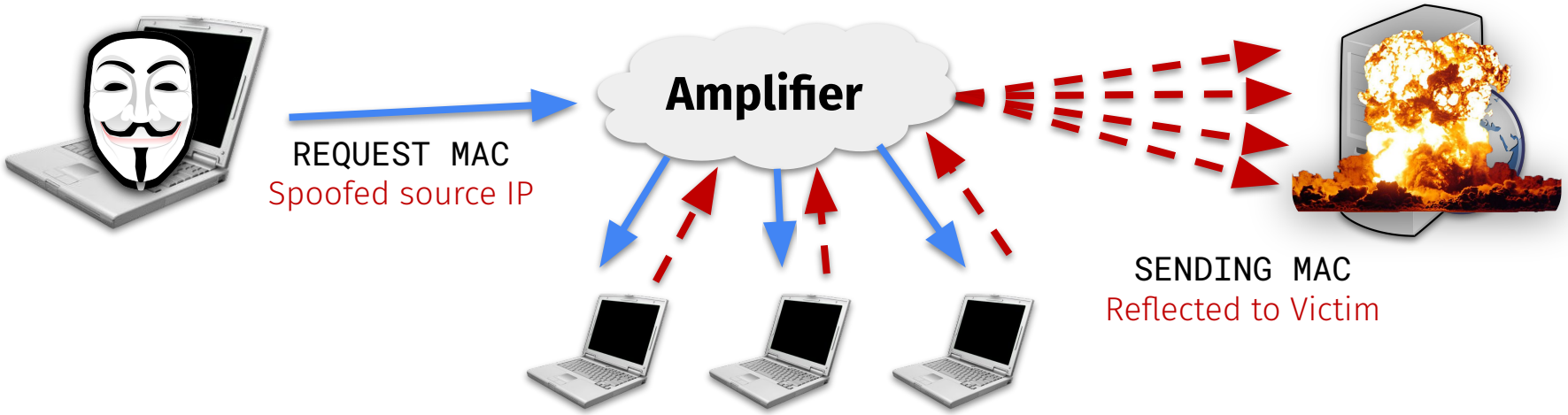


ARP Flooding Attack

- **Attack: ???**

ARP Flooding Attack

- **Attack:** same idea as **ICMP Smurfing**; **spoof source to victim** and spam away!
  - Victim gets overwhelmed by ARP replies and bandwith crashes



Physical Layer DoS

Russian Spy Submarines Are Tampering with Undersea Cables That Make the Internet Work. Should We Be Worried?

A massive cable attack is probably an over-hyped scenario, at least for a country with as many redundant cables as the United States pitted against a limited number of Russian special-operations submarines.

CNN Exclusive: FBI investigation determined Chinese-made Huawei equipment could disrupt US nuclear arsenal communications

Thwarting DoS/DDoS Attacks

- **How?**

Thwarting DoS/DDoS Attacks

- Limit **connection rate**
  - Reduce to *N* total requests
- Detect **anomalous activity**
  - IP geo-filtering
  - Packet similarity detection
- Avoid holding **connection state**
  - Don’t wait on “half-open” connections
- **Don’t be part of the problem!**
  - Disable potential amplifiers
  - Prevent botnet infection

Questions?

This time on CS 4440...

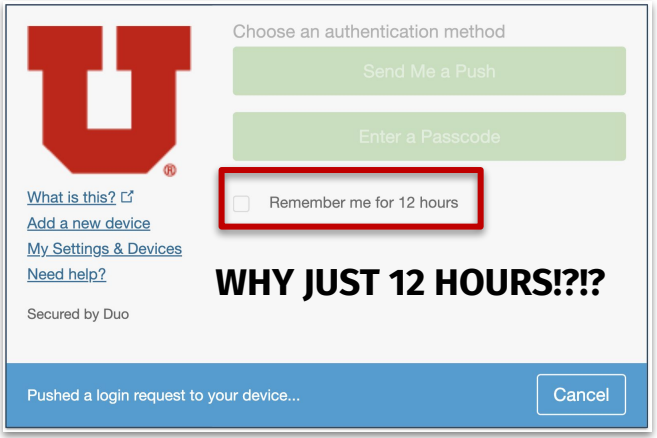
Authentication
Multiple Authentication Factors
One-time PINs
Secure Password Storage

What is authentication?

- **What is it?**

What is authentication?

- **What is it?**
  - That password you re-use for every website
  - An ever-changing set of rules to frustrate you
  - The most annoying thing about attending UofU



What is authentication?

- **Goal: ???**

What is authentication?

- **Goal:** establish trust in the **identity** of another communicating party
- **Problem:** ???

What is authentication?

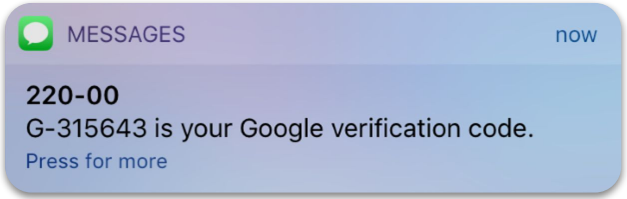
- **Goal:** establish trust in the **identity** of another communicating party
- **Problem:** **cannot directly interact** with them to verify their identity
  - Must be performed **remotely**
- **Challenge:** how can someone prove they are who they say they are?

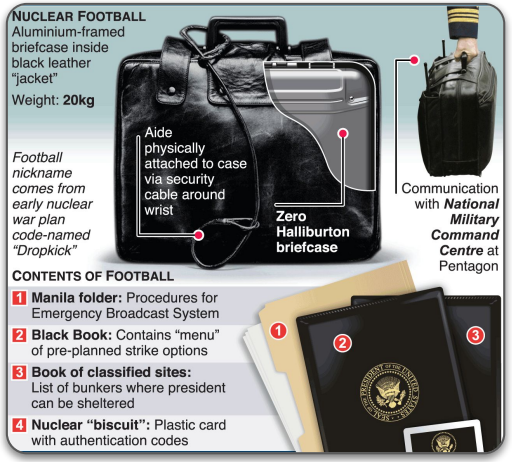
The Three Factors of Authentication

- Something you **???**
- Something you **???**
- Something you **???**

The Three Factors of Authentication

- Something you **have**
  - Smartphone
  - Laptop
  - Email account
- Something you **are**
  - Your fingerprint
  - Your DNA
  - Your iris, retina
- Something you **know**
  - Account password, banking PIN number
  - Nuclear strike challenge-response code





Single- vs. Multi-factor Authentication

- ***N*-factor authentication:** how many factors are used to authenticate
  - **Password-only login** is a single-factor authentication
- What are the **trade-offs**?
  - **???**

Single- vs. Multi-factor Authentication

- ***N*-factor authentication:** how many factors are used to authenticate
  - **Password-only login** is a single-factor authentication

- What are the **trade-offs**?
  - **Fewer** factors = **worse** security
    - Compromise of one factor is total authentication violation
  - **More** factors = **increased** security
    - To fully violate authentication, attacker must compromise all
  - **Trade-off:** more annoying for user
    - Who cares? **Security >> UX**

Nowadays, most authentication is **at least 2-factor**

Questions?

One-time PINs

Proof of Possession

- How can you prove—**remotely**—that you **possess something**?

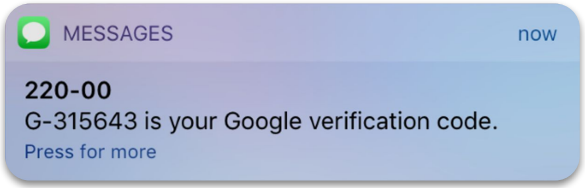
Proof of possession: make the user perform some **object-specific** **action** that requires their **physical interaction**

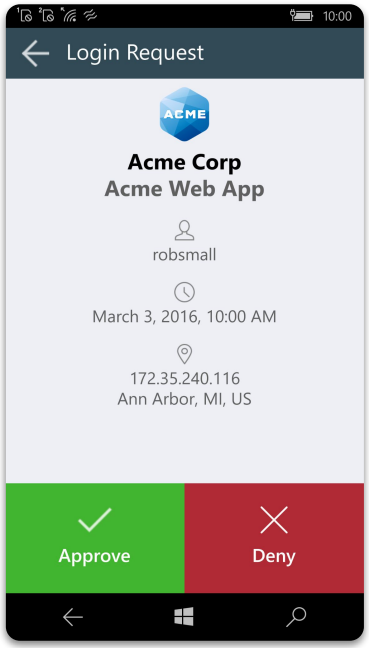
One-time PINs

- **One-time PINs / Passwords:**
  - Password valid for only **one** login session or transaction
- **Delivering** One-time PINs:
  - **???**

One-time PINs

- **One-time PINs / Passwords:**
  - Password valid for only **one** login session or transaction
- **Delivering** One-time PINs:
  - **SMS**
    - Phone call
    - Text message
  - **Hardware**
    - Yubico YubiKey
    - RSA SecureID
  - **Application**
    - DUO Mobile
    - Google authenticator

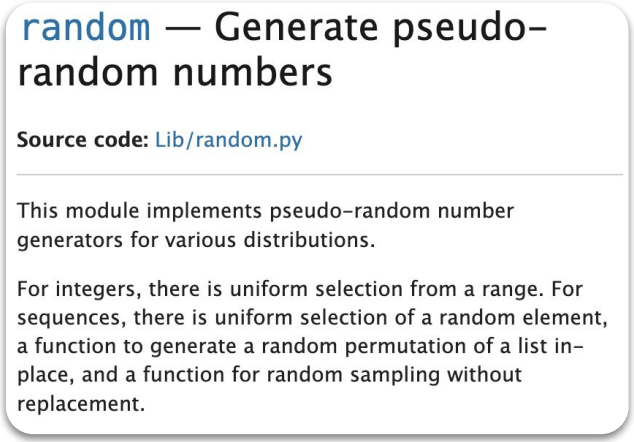




Implementing OTPs

- **Idea:** call an API (e.g., `math.random`), send **random** to user, user re-enters it

Downsides?



`random` — Generate pseudo-random numbers

Source code: Lib/random.py

This module implements pseudo-random number generators for various distributions.

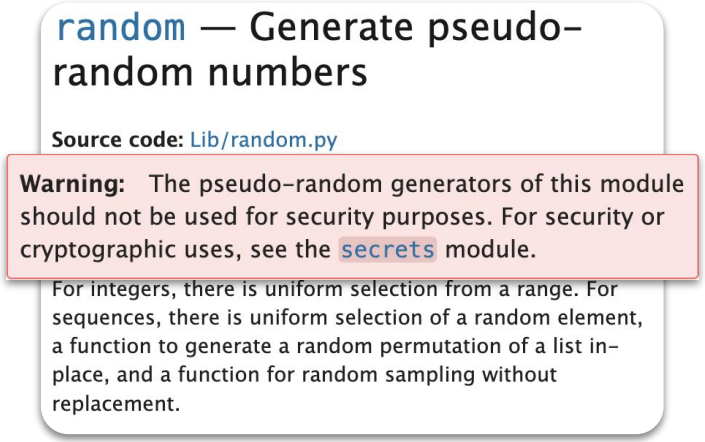
For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement.

Implementing OTPs

- **Idea:** call an API (e.g., `math.random`), send **random** to user, user re-enters it

- Authentication **offline**? **No!**
  - User needs internet to receive the OTP code
  - Without a connection, they can't authenticate

- Demonstrably **secure**? **No!**
  - Most "random" APIs have small/predictable seeds
  - Also vulnerable to man-in-the-middle attacks



Attack: SIM Swap

- **SIM: Subscriber Identity Module**
  - A small card inserted into your phone
  - Connects you to your carrier's network

Attack: SIM Swap

- **SIM: Subscriber Identity Module**
  - A small card inserted into your phone
  - Connects you to your carrier's network
- **Social engineering attack:**
  - Learn key info about victim. E.g.:
    - Mothers' maiden name
    - Childhood street address
  - Trick carrier to issue new SIM card
    - "I'm Jeff Bezos, my phone broke!"
    - Attacker "appears to be" victim

Attack: SIM Swap

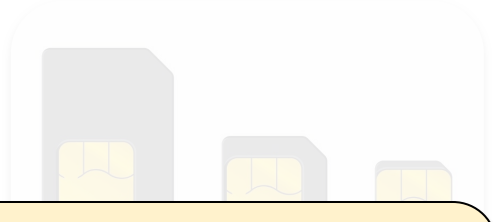
- **SIM: Subscriber Identity Module**
  - A small card inserted into your phone
  - Connects you to your carrier's network
- **Social engineering attack:**
  - Learn key info about victim. E.g.:
    - Mothers' maiden name
    - Childhood street address
  - Trick carrier to issue new SIM card
    - "I'm Jeff Bezos, my phone broke!"
    - Attacker "appears to be" victim
- **Result:** attacker is **man-in-the-middle**
  - **Receives any OTPs transmitted by SMS!**

Hackers steal thousands of dollars through victims' cell phones using SIM swap fraud

Hackers Hit Twitter C.E.O. Jack Dorsey in a 'SIM Swap.' You're at Risk, Too.

Attack: SIM Swap

- **SIM: Subscriber Identity Module**
  - A small card inserted into your phone
  - Connects you to your carrier's network
- **Social eng**
  - Learn k
    - M
    - C
  - Trick ca
    - "I'm Jeff Bezos, my phone broke!"
    - Attacker "appears to be" victim
- **Result:** attacker is **man-in-the-middle**
  - **Receives any OTPs transmitted by SMS!**



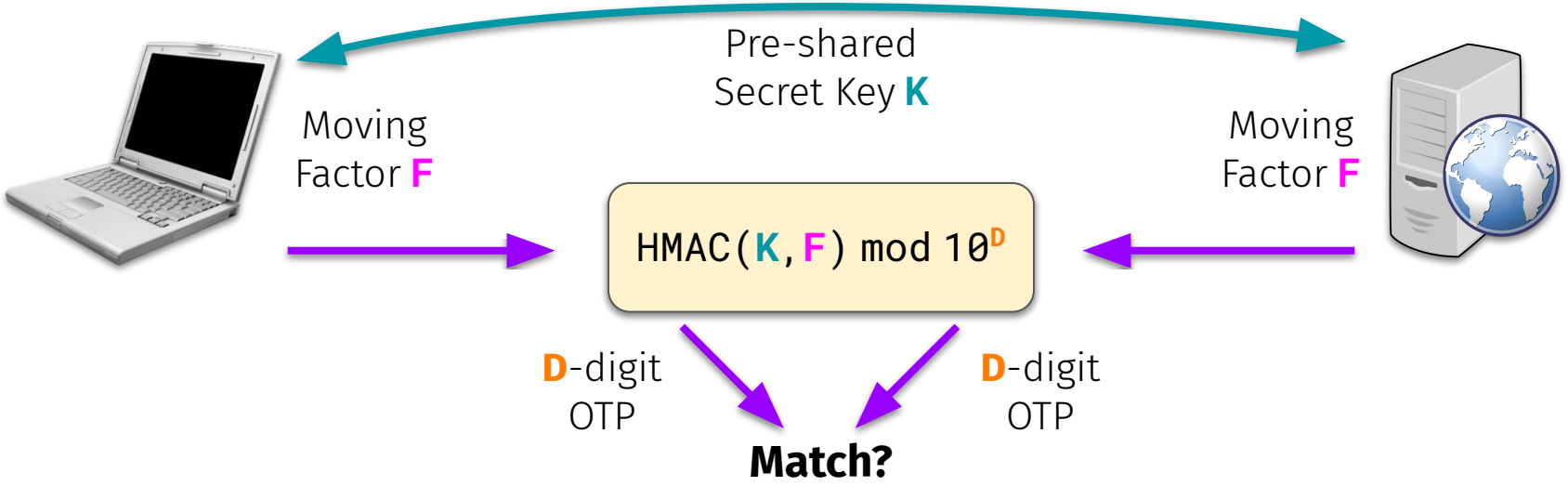
How can we **increase entropy** and eliminate need for **online sharing**?

sands of tims' cell swap fraud

Hackers Hit Twitter C.E.O. Jack Dorsey in a 'SIM Swap.' You're at Risk, Too.

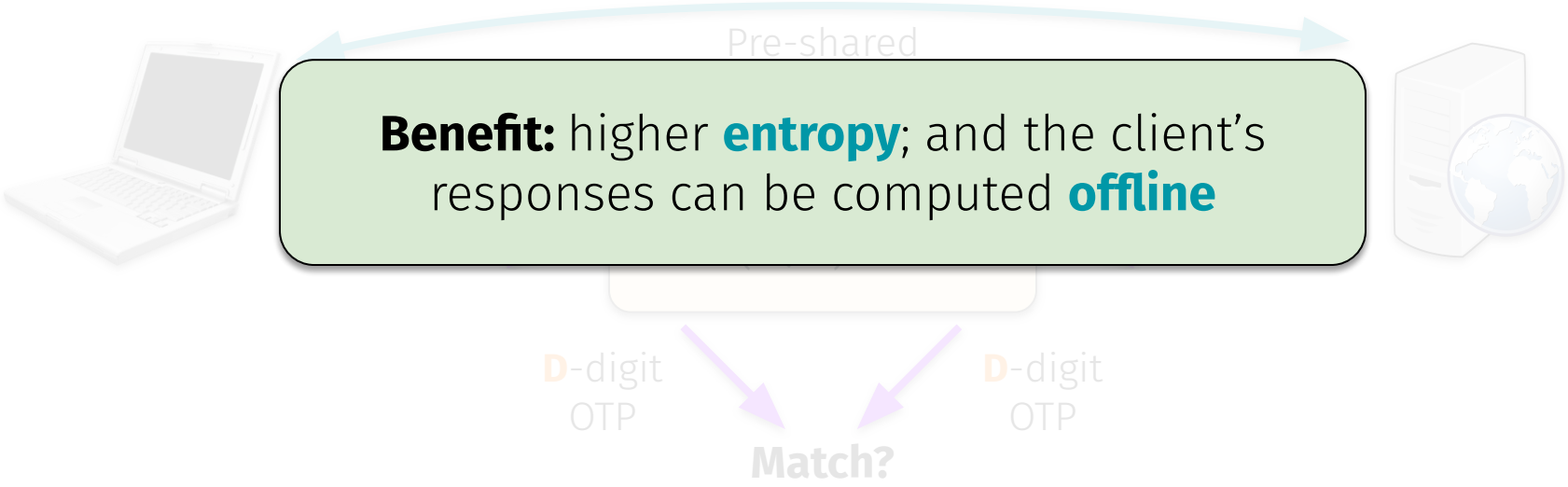
Implementing OTPs

- **Better idea:** independently generate OTP codes based on a **moving factor**
  - E.g., intervals of **time**, unique session **count**, etc.



Implementing OTPs

- **Better idea:** independently generate OTP codes based on a **moving factor**
  - E.g., intervals of **time**, unique session **count**, etc.



Benefit: higher **entropy**; and the client's responses can be computed **offline**

Implementing OTPs

- **Better idea:** independently generate OTP codes based on a **moving factor**
  - E.g., intervals of **time**, unique session **count**, etc.

- **Common OTP protocols:**
  - HMAC-based OTP (**HOTP**)
    - Use **session count** as factor
  - Time-based OTP (**TOTP**)
    - Use **time interval** as factor

Implementing OTPs

- **Better idea:** independently generate OTP codes based on a **moving factor**
  - E.g., intervals of **time**, unique session **count**, etc.

- **Common OTP protocols:**
  - HMAC-based OTP (**HOTP**)
    - Use **session count** as factor
  - Time-based OTP (**TOTP**)
    - Use **time interval** as factor

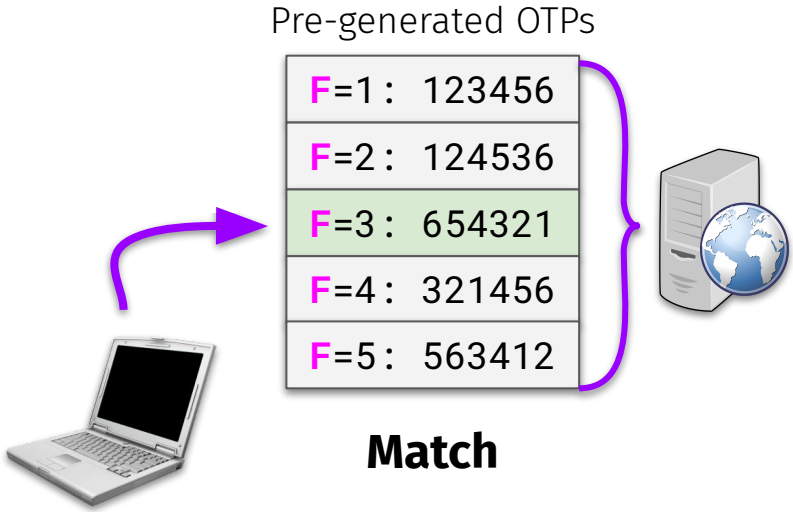
- **Problem: desynchronization**
  - E.g., user hits “login” one too many times

Implementing OTPs

- **Better idea:** independently generate OTP codes based on a **moving factor**
  - E.g., intervals of **time**, unique session **count**, etc.

- **Common OTP protocols:**
  - HMAC-based OTP (**HOTP**)
    - Use **session count** as factor
  - Time-based OTP (**TOTP**)
    - Use **time interval** as factor

- **Problem: desynchronization**
  - E.g., user hits "login" one too many times
  - **Solution:** make a few OTPs; user matches once

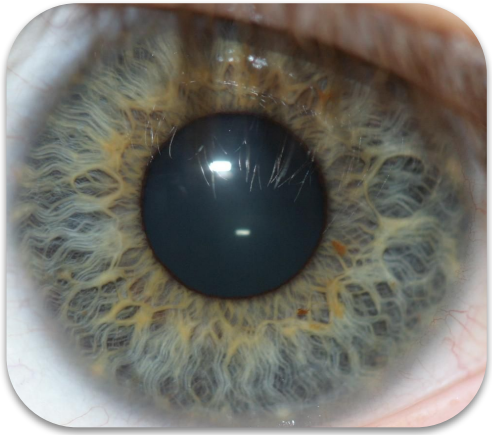


Questions?

Biometrics

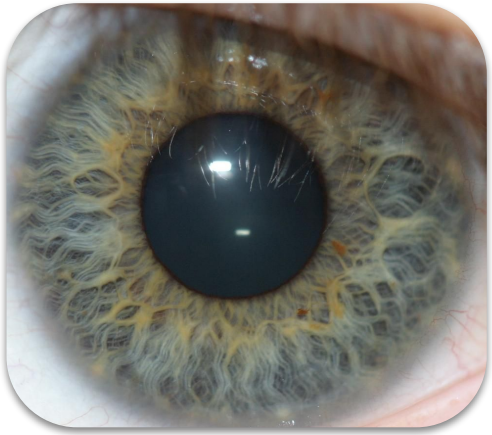
Biometrics

- **Provides proof of ???**



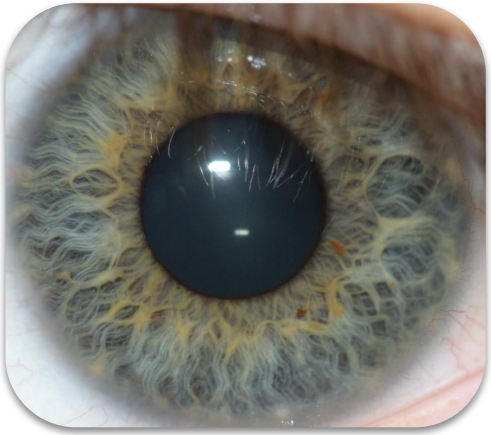
Biometrics

- **Provides proof of physical identity**



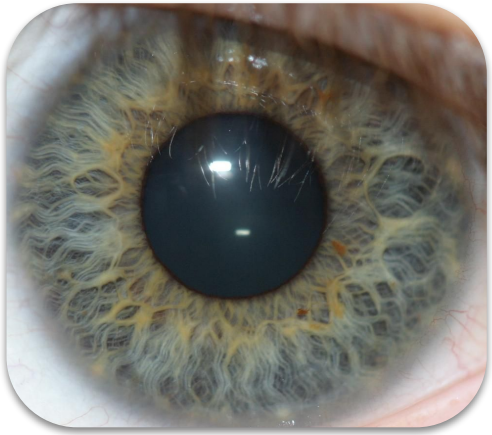
Biometrics

- **Provides proof of physical identity**
- **Something unique to you** (hopefully)
  - Fingerprint, iris, retina, DNA
- Security = **unlikely match probability**
  - Fingerprint match chance: **???**
  - Iris pattern match chance: **???**



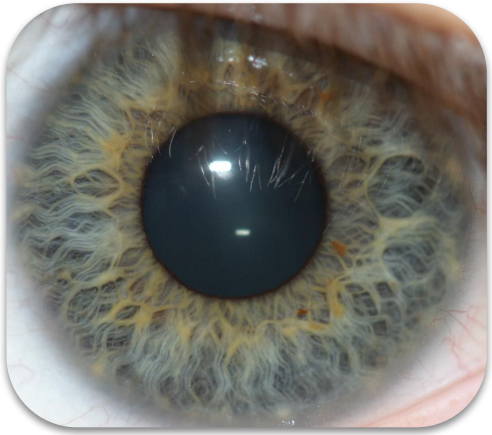
Biometrics

- **Provides proof of physical identity**
- **Something unique to you** (hopefully)
  - Fingerprint, iris, retina, DNA
- Security = **unlikely match probability**
  - Fingerprint match chance: **1 in $64 * 10^{13}$**
  - Iris pattern match chance: **1 in 10^{78}**
- **Trade-offs?**
  - **???**



Biometrics

- **Provides proof of physical identity**
- **Something unique to you** (hopefully)
  - Fingerprint, iris, retina, DNA
- Security = **unlikely match probability**
  - Fingerprint match chance: **1 in $64 * 10^{13}$**
  - Iris pattern match chance: **1 in 10^{78}**
- **Trade-offs?**
  - Engineering effort, storage size, privacy concerns



Biometric Challenges

Downsides?

Biometric Challenges

- **Replay attacks**
  - Spoofs an enrolled user
- **Poisoning attacks**
  - Alter enrollment template
  - Alter one user's enrollment
- **Noisy sensors**
  - Gives attackers "leeway" in crafting adversarial inputs
- **Change / loss of biometric**
  - **Change:** cataracts surgery
  - **Loss:** losing your finger

After an initial analysis, the Indian and American scientists used three iris sensors and two commercial iris biometric matchers to check if the new irises passed biometric authentication. They found that the iris sensors' success rate dropped to 75% after surgery. The biometric matchers did better authenticating 93% of the irises.

Crane horror *Reg* reader uses his severed finger to unlock Samsung Galaxy phone

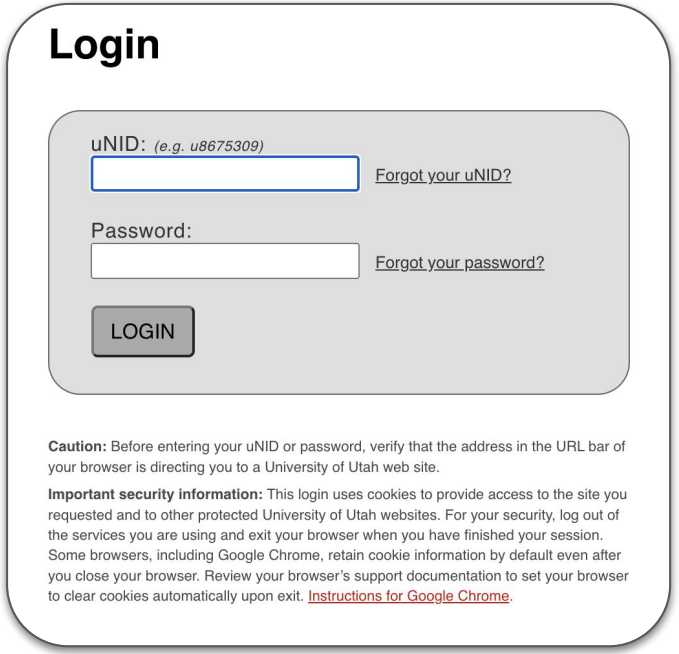
On the other hand he was fine

Questions?

Passwords

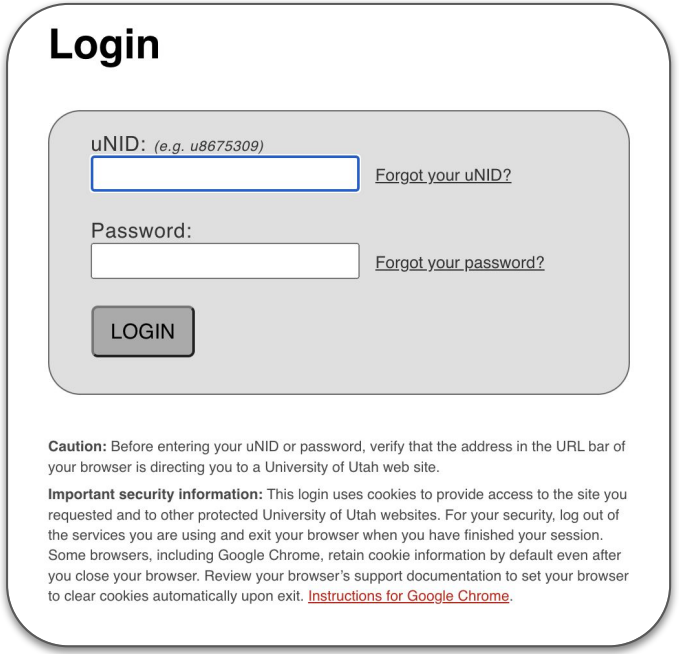
Passwords

- **Something that you ???**



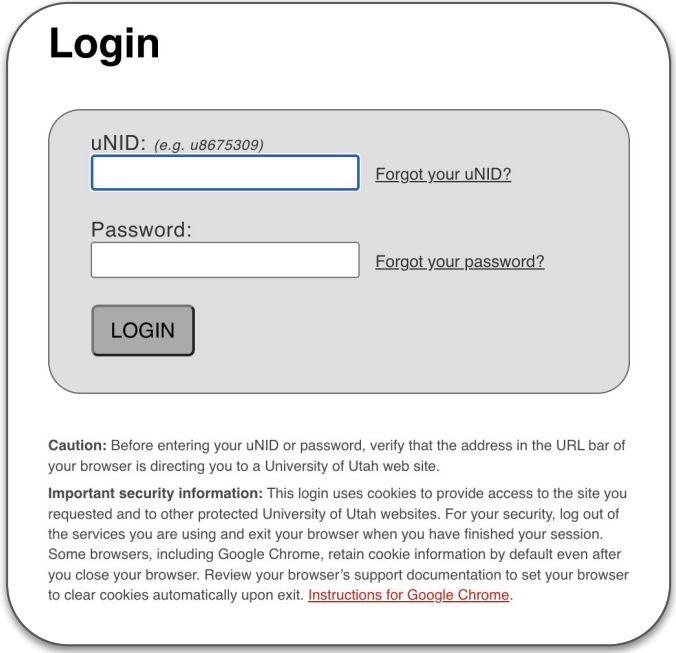
Passwords

- **Something that you know**
  - Something that you forget?
- A **secret** string of data that confirms a user's identity



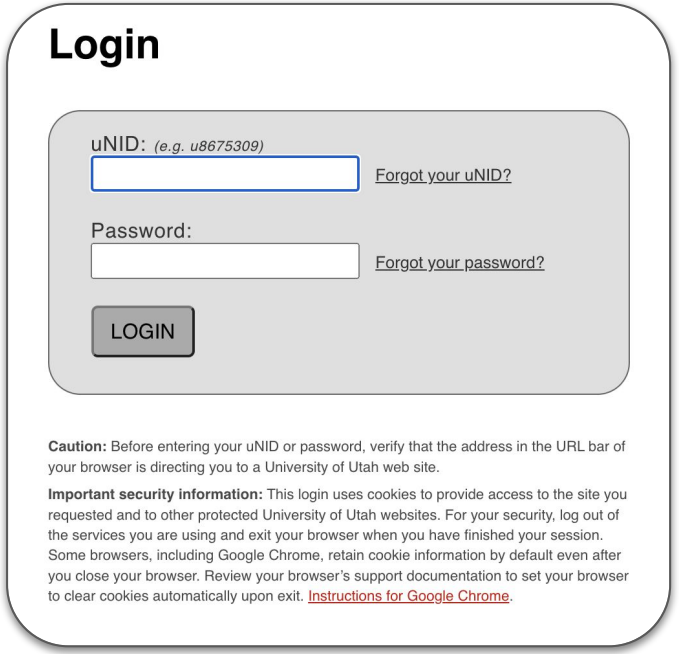
Passwords

- **Something that you know**
  - Something that you forget?
- A **secret** string of data that confirms a user's identity
  - **Letters** (ABCDEFGH)
  - **Digits** (0123456789)
  - **Other symbols** ($#%-_!)



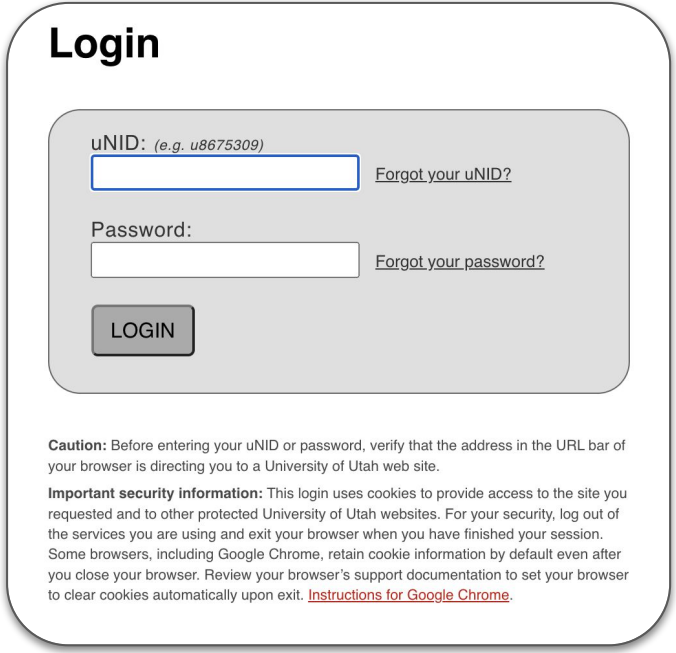
Passwords

- **Something that you know**
  - Something that you forget?
- A **secret** string of data that confirms a user's identity
  - **Letters** (ABCDEFGH)
  - **Digits** (0123456789)
  - **Other symbols** ($#%-_!)
- **Cryptographically secure?**



Passwords

- **Something that you know**
  - Something that you forget?
- A **secret** string of data that confirms a user's identity
  - **Letters** (ABCDEFGH)
  - **Digits** (0123456789)
  - **Other symbols** ($#%-_!)
- **Cryptographically secure?**
  - **Not at all!**



Why aren’t passwords cryptographically secure?

- **Cryptographically Secure** = **???**

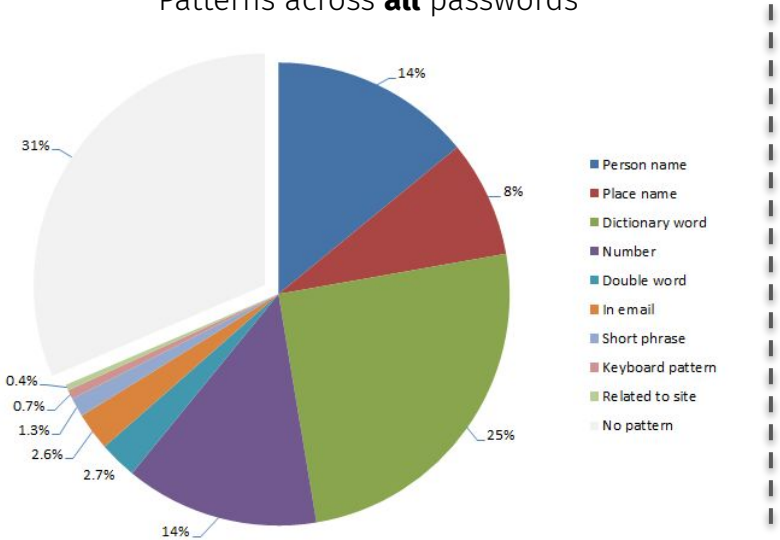
Why aren't passwords cryptographically secure?

- **Cryptographically Secure** = unbiased output, cannot be **predicted**
  - E.g., a cryptographically-secure pseudo-random number generator

Why aren’t passwords cryptographically secure?

- **Are most passwords biased or predictable?**
  - Analysis of Sony and Gawker breached passwords:

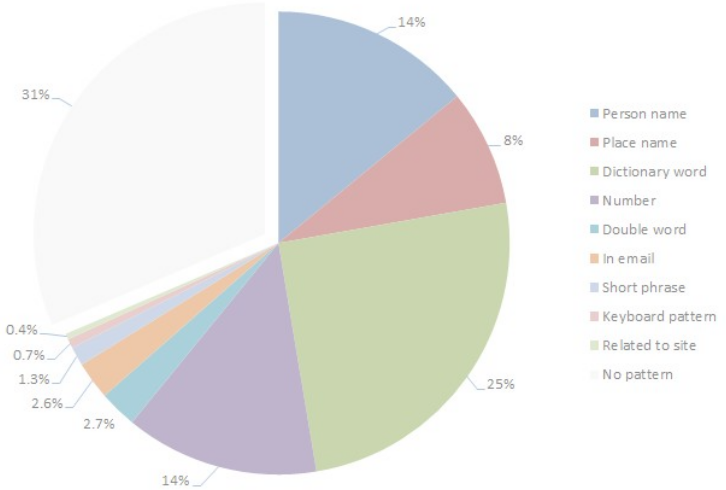
Patterns across **all** passwords



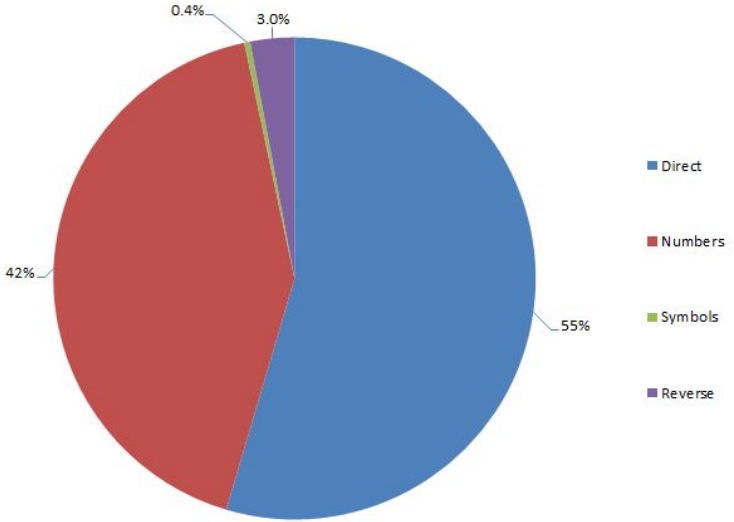
Why aren't passwords cryptographically secure?

- **Are most passwords biased or predictable?**
  - Analysis of Sony and Gawker breached passwords:

Patterns across **all** passwords



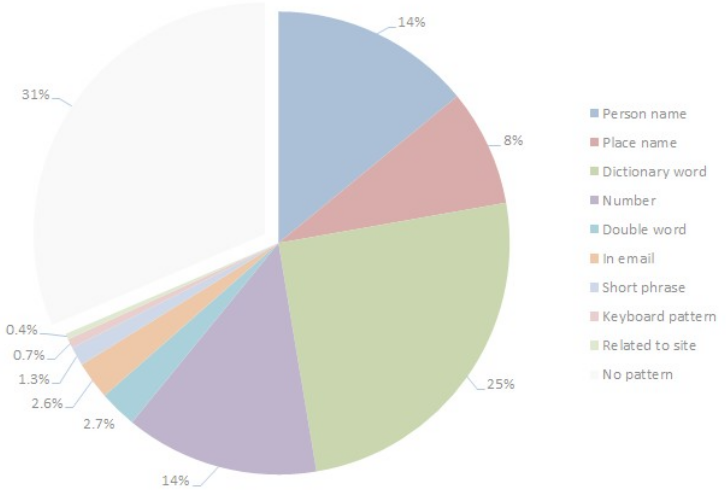
Passwords derived from **people names**



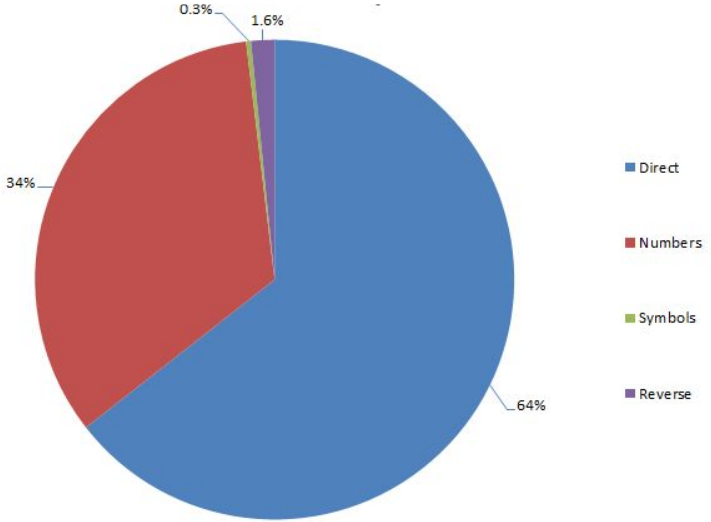
Why aren't passwords cryptographically secure?

- **Are most passwords biased or predictable?**
  - Analysis of Sony and Gawker breached passwords:

Patterns across **all** passwords



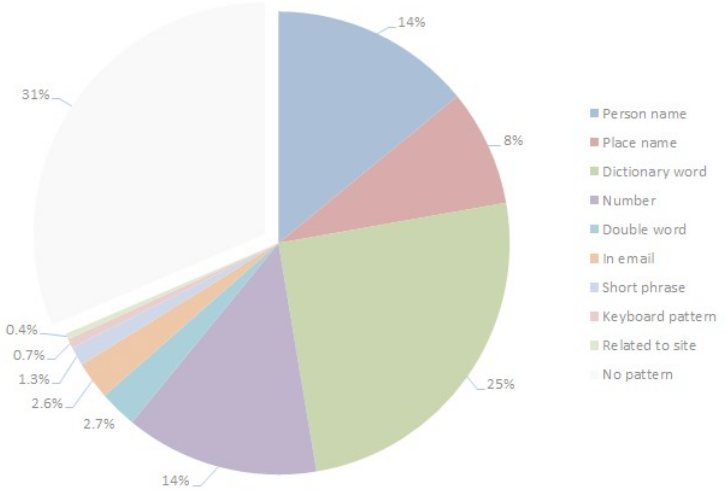
Passwords derived from **location names**



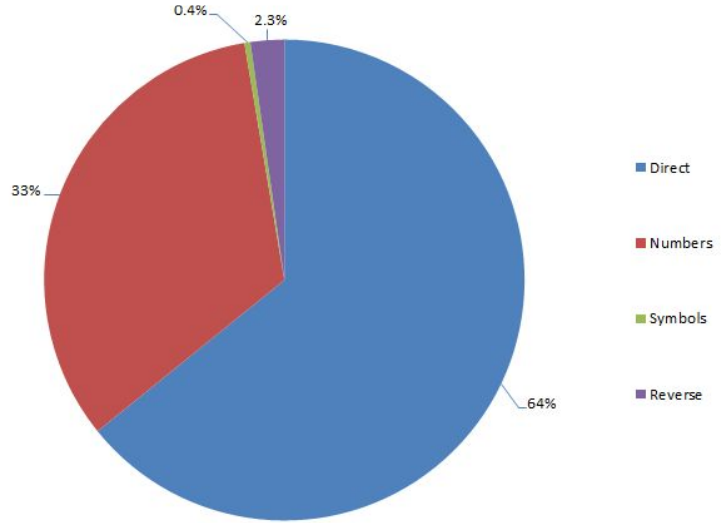
Why aren't passwords cryptographically secure?

- **Are most passwords biased or predictable?**
  - Analysis of Sony and Gawker breached passwords:

Patterns across **all** passwords



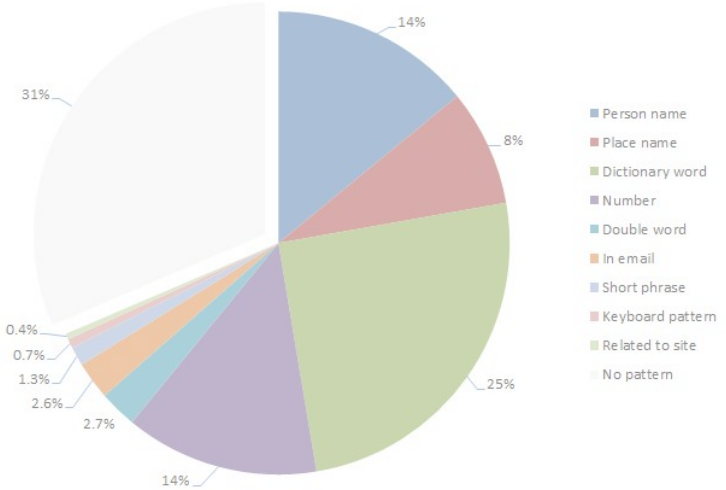
Passwords derived from **dictionary words**



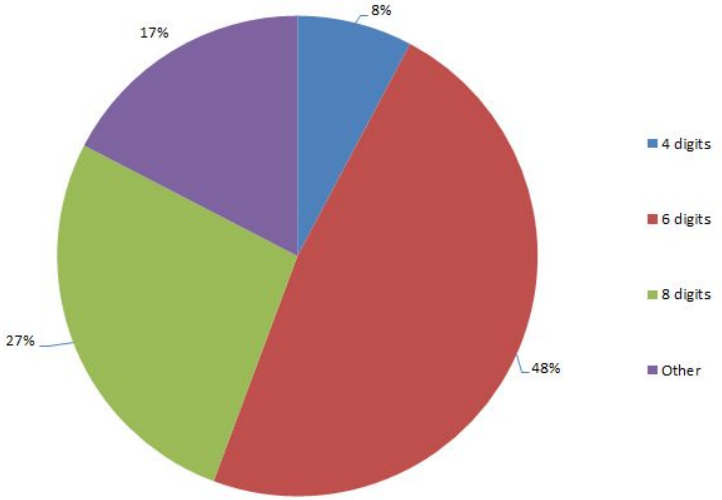
Why aren't passwords cryptographically secure?

- **Are most passwords biased or predictable?**
  - Analysis of Sony and Gawker breached passwords:

Patterns across **all** passwords



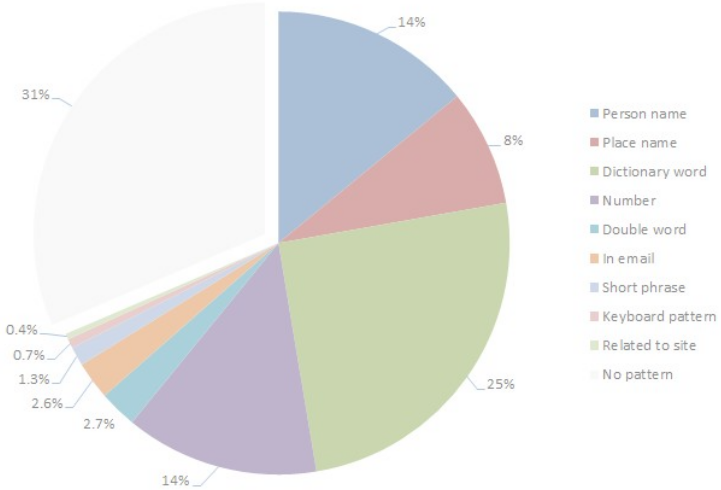
Passwords derived from **numbers**



Why aren't passwords cryptographically secure?

- **Are most passwords biased or predictable?**
  - Analysis of Sony and Gawker breached passwords:

Patterns across **all** passwords



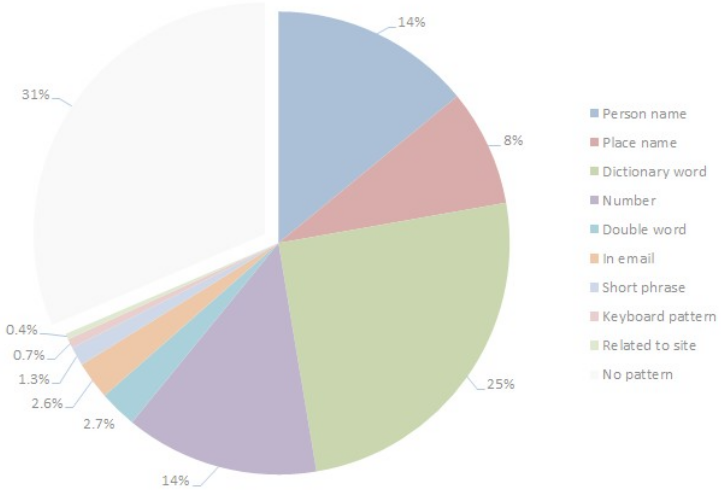
Passwords derived from **keyboard patterns**


```
qwerty

asdfgh

asdf1234
```

Why aren't passwords cryptographically secure?

- **Are most passwords biased or predictable?**
  - Analysis of Sony and Gawker breached passwords:

Patterns across **all** passwords



Passwords derived from **pop culture references**

thx1138

gundam

ncc1701

Attack: Guessing Passwords

- Known **default** passwords:
  - Device manufacturers don't care
  - E.g., `password`, 12345, etc.
  - How Mirai Botnet spread itself

Username	Password
666666	666666
888888	888888
admin	*(none)*
admin	1111
admin	1111111
admin	1234
admin	12345
admin	123456
admin	54321
admin	7ujMko0admin
admin	admin

Attack: Guessing Passwords

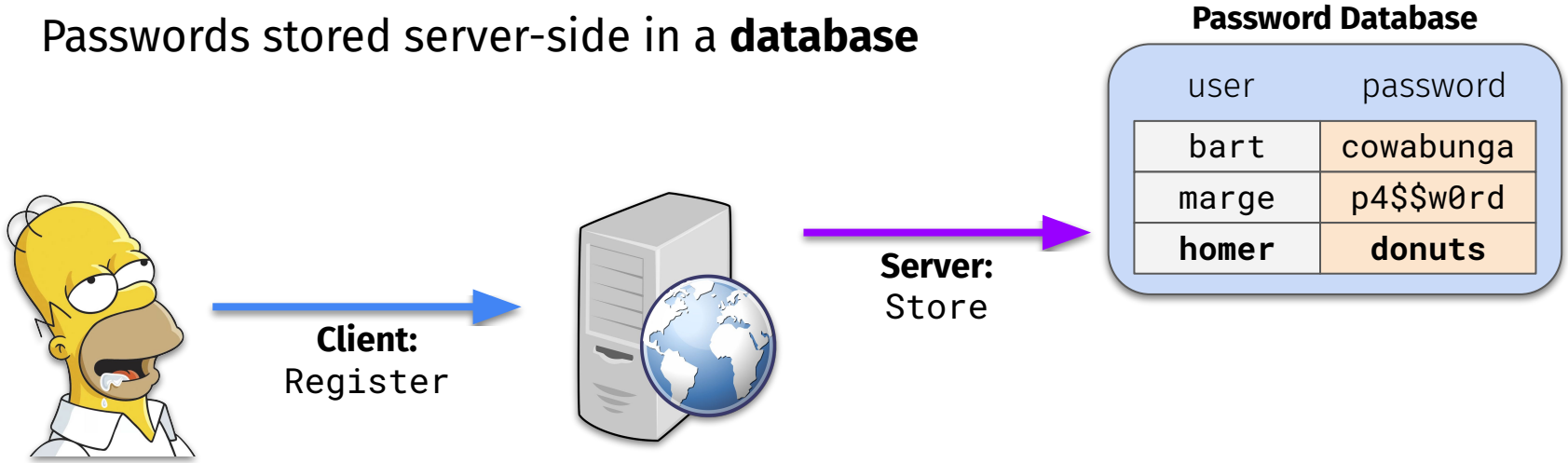
- Known **default** passwords:
  - Device manufacturers don't care
  - E.g., `password`, `12345`, etc.
  - How Mirai Botnet spread itself
- **Social engineering** attacks:
  - Trick victim to revealing key info
    - E.g., date of birth, nickname pet's name, favorite team
  - Try to guess their password
    - E.g., `GoChiefs94`, `Chiefs1994`

Username	Password
666666	666666
888888	888888
admin	*(none)*
admin	1111
admin	1111111
admin	1234
admin	12345
admin	123456
admin	54321
admin	7ujMko0admin
admin	admin

1 in 3 U.S. Pet Parents Have Used Their Pet's Name as Their Password

Server-side Password Storage

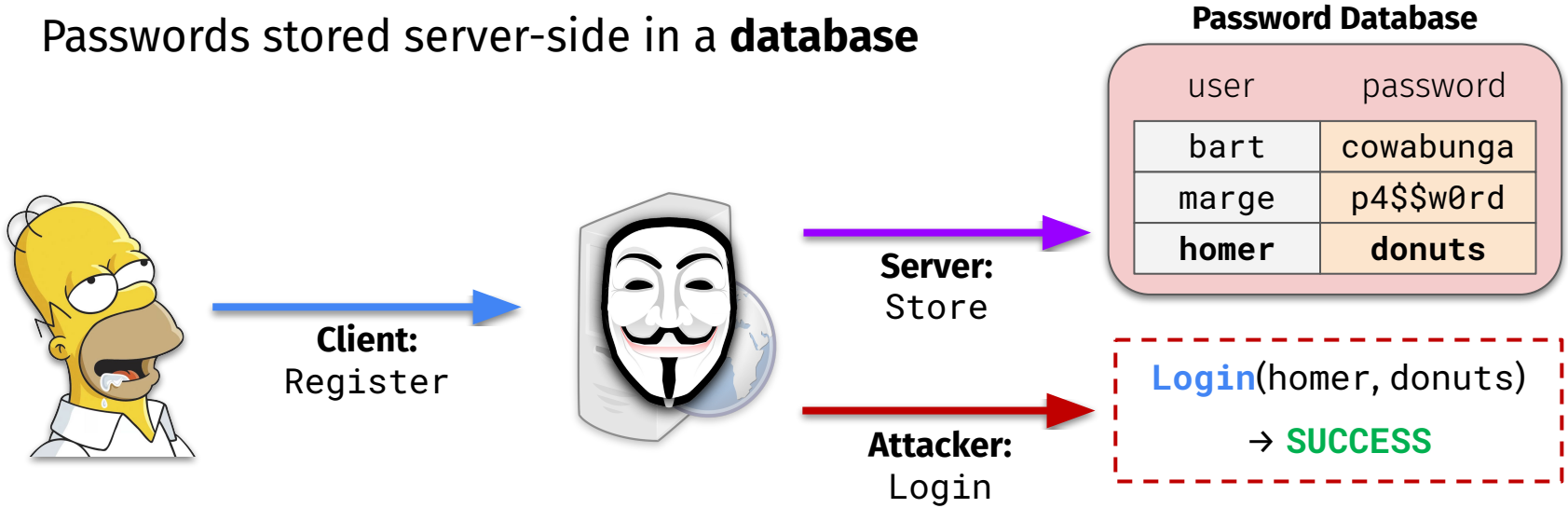
- Passwords stored server-side in a **database**



Why is storing passwords in **plaintext** problematic?

Server-side Password Storage

- Passwords stored server-side in a **database**



If database **breached**, attacker has **all passwords**!

Server-side Password Storage

- Passwords stored server-side in a **database**
  - Increase security by only storing **hashed passwords**

Client:
`Register`

Server:
`Store`

Password Database

user	password
bart	cowabunga
marge	p4$$w0rd
homer	donuts

Server:
`Hash and Store`

Hashed Password Database

user	hash
bart	f0baf06…
marge	b3ea222…
homer	6c493f3…

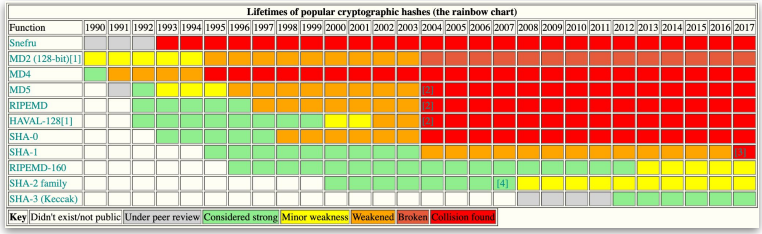
If database **breached**, attacker has **zero plaintext passwords**!

Attacking Stored Passwords

- **Assumption:** attacker has **full access** to our database of **hashed passwords**
  - E.g., SQL injection, other web app attacks

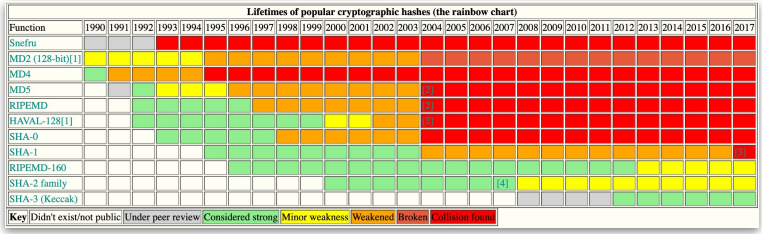
Attacking Stored Passwords

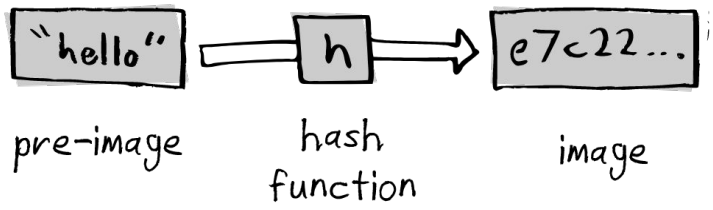
- **Assumption:** attacker has **full access** to our database of **hashed passwords**
  - E.g., SQL injection, other web app attacks
- What if a **weak** hash function is used?
  - **???**



Attacking Stored Passwords

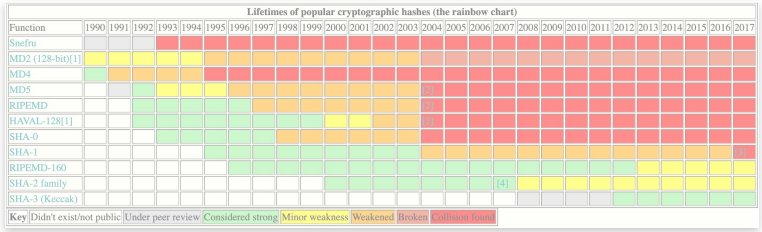
- **Assumption:** attacker has **full access** to our database of **hashed passwords**
  - E.g., SQL injection, other web app attacks
- What if a **weak** hash function is used?
  - **Pre-image attacks:** find the original string
  - **Collision attacks:** find a different string that produces same hash as password

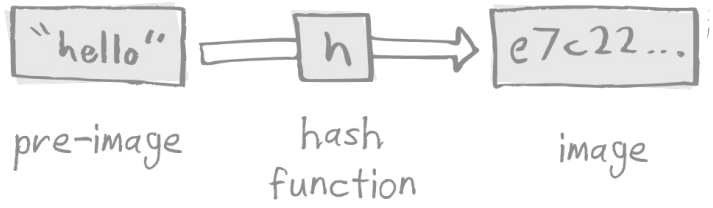




Attacking Stored Passwords

- **Assumption:** attacker has **full access** to our database of **hashed passwords**
  - E.g., SQL injection, other web app attacks
- What if a **weak** hash function is used?
  - **Pre-image attacks:** find the original string
  - **Collision attacks:** find a different string that produces same hash as password
- What if a **fast** hash function is used?



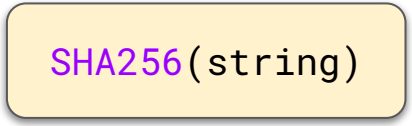


Attacking Stored Passwords

- **Assumption:** attacker has **full access** to our database of **hashed passwords**
  - E.g., SQL injection, other web app attacks
- What if a **weak** hash function is used?
  - **Pre-image attacks:** find the original string
  - **Collision attacks:** find a different string that produces same hash as password
- What if a **fast** hash function is used?
  - Attacker can **quickly pre-generate hashes** for all possible password possibilities

Common Passwords

Same hash function as the server



Attacker's table of pw hash pairs

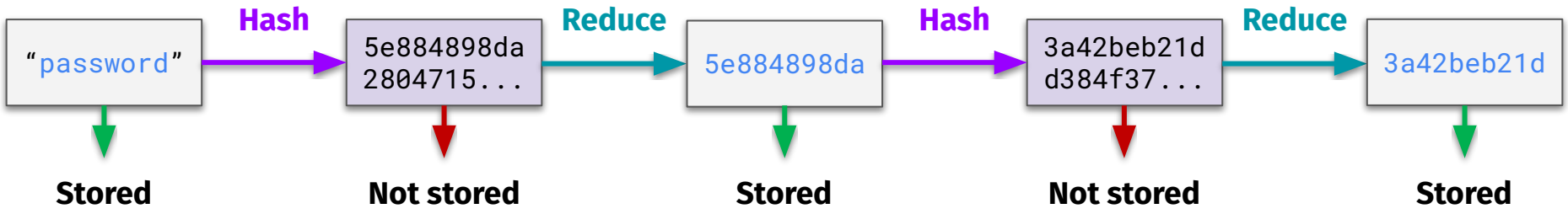
"password"	5e8848…
"123456"	8d969e…
"qwerty"	65e84b…

Attack: Rainbow Tables

- Similar to a **lookup table**—attacker can trade-off **disk space** vs. **CPU time**
  - Attacker wants something that uses **less time, less storage** than a **brute-force attack**

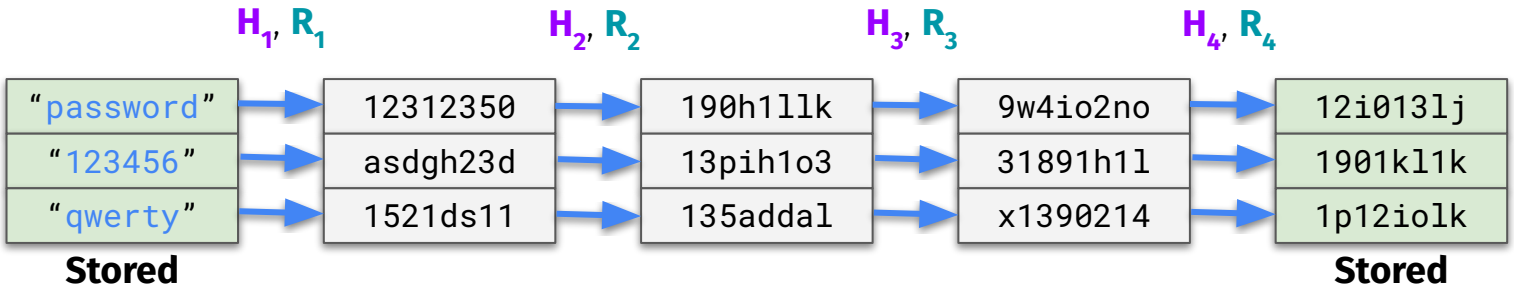
Attack: Rainbow Tables

- Similar to a **lookup table**—attacker can trade-off **disk space** vs. **CPU time**
  - Attacker wants something that uses **less time, less storage** than a **brute-force attack**
- **Idea:** iteratively **hash** and **reduce** to form a connected "**chain**" of hashes
  - **Simple reduction function:** truncate to just the first **10** characters of every hash



Attack: Rainbow Tables

- To find a **password** from its hash, **perform reductions** and check for a match
  - For efficiency, only the **starting** **and** **ending** **links** are stored per each chain



Example: Find password for hash `135addal`

H,**R** (`135addal`) = `x1390214`

H,**R** (`x1390214`) = `1p12iolk` **Found chain!**

Walk the chain from its start (`qwerty`)

= `qwerty` -> `1521ds11` -> `135addal`

Found original string!

Better Password Generation

- **Why is reusing the same password bad practice?**

Better Password Generation

- **Why is reusing the same password bad practice?**
  - If a breached server stores it in **plaintext**, your credentials are now stolen!

Better Server-side Password Storage

- **Slower hash functions**
  - ???

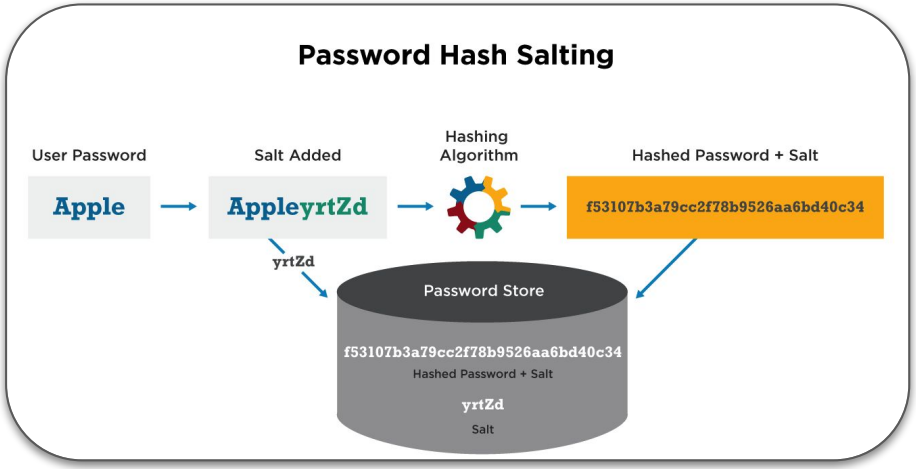
Better Server-side Password Storage

- **Slower hash functions**
  - Makes rainbow table generation **more computationally expensive** for attackers!
  - E.g., **Bcrypt, Scrypt**—perform multiple rounds of hashing (**much slower**)

Better Server-side Password Storage

- **Slower hash functions**
  - Makes rainbow table generation **more computationally expensive** for attackers!
  - E.g., **Bcrypt, Scrypt**—perform multiple rounds of hashing (**much slower**)

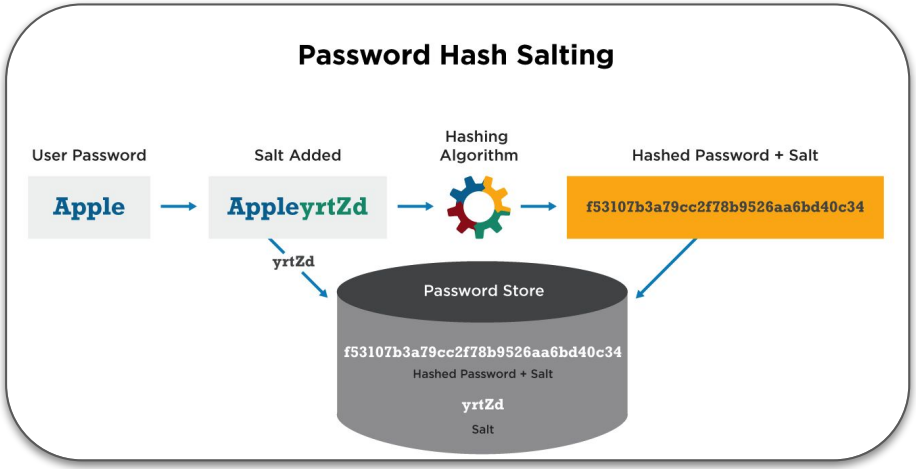
- **Salted passwords:**
  - Add **extra data** when generating hash
  - **Goal:** same input = different output



Better Server-side Password Storage

- **Slower hash functions**
  - Makes rainbow table generation **more computationally expensive** for attackers!
  - E.g., **Bcrypt, Scrypt**—perform multiple rounds of hashing (**much slower**)

- **Salted passwords:**
  - Add **extra data** when generating hash
  - **Goal:** same input = different output
  - Salting considerations:
    - Salt should **not be short**
    - Should be **unique** per user

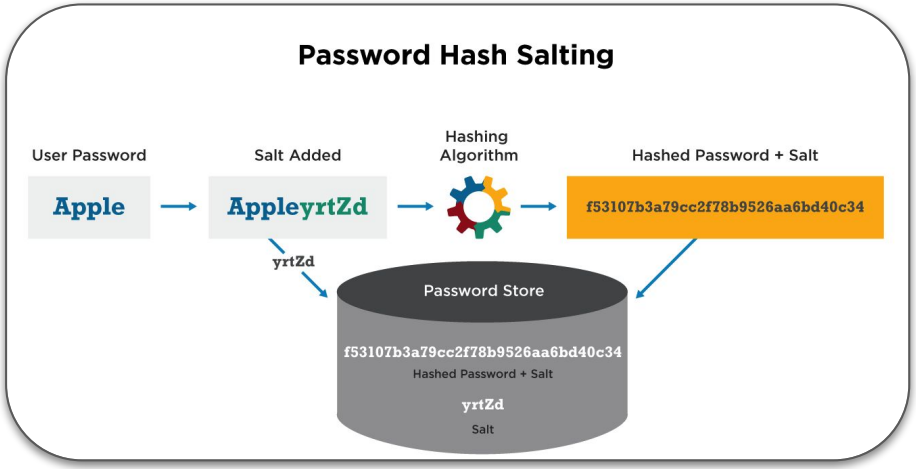


Better Server-side Password Storage

- **Slower hash functions**
  - Makes rainbow table generation **more computationally expensive** for attackers!
  - E.g., **Bcrypt, Scrypt**—perform multiple rounds of hashing (**much slower**)

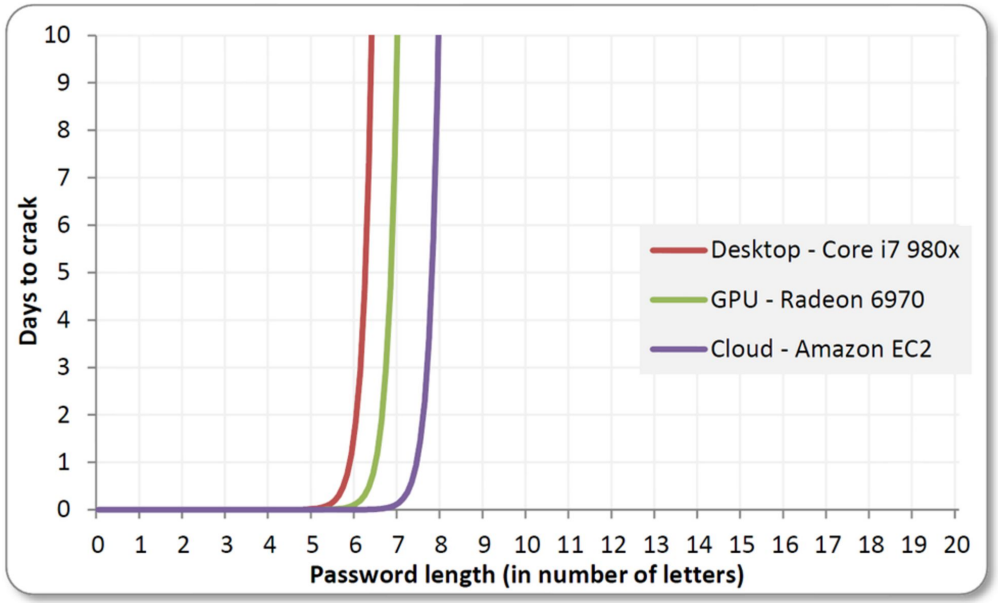
- **Salted passwords:**
  - Add **extra data** when generating hash
  - **Goal:** same input = different output
  - Salting considerations:
    - Salt should **not be short**
    - Should be **unique** per user

- **Better: salting + slow hashing!**



Attack: Password Cracking

- Assume attacker knows hash function and wants to **find a single password**
  - Rapidly **becoming more doable** with advances in hardware!



Attack: Client-side Password Theft

- **How?**

Attack: Client-side Password Theft

- **How?**
  - Keyloggers, unencrypted transit, phishing, angry ex-partner

```
Hypertext Transfer Protocol
› GET /libs/qimessaging/1.0/qimessaging.js?v=1.2.0 HTTP/1.1\r\n
  Host: 10.0.0.6\r\n
  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101
  Accept: */*\r\n
  Accept-Language: en-US,en;q=0.5\r\n
  Accept-Encoding: gzip, deflate\r\n
  Referer: http://10.0.0.6/\r\n
  Connection: keep-alive\r\n
› Authorization: Basic bmFvOmNhcmVzc2VzLTIwMDE=\r\n
    Credentials: nao:[redacted]
  \r\n
```

From: Bank of America <crvdgi@comcast.net>
Subject: Notification Irregular Activity
Date: September 23, 2014 3:44:42 PM PDT
To: Undisclosed recipients: ;
Reply-To: crvdgi@comcast.net

Bank of America

Online Banking Alert

Would be capitalized

Dear member:

We detected unusual activity on your Bank of America debit card on **09/22/2014**.
For your protection, please verify this activity so you can continue making debit card transactions without interruption.

Please sign in to your account **at** https://www.bankofamerica.com

http://bit.do/ghsdfhgsd

to review and verify your account activity. After verifying your debit card transactions we will take the necessary steps to protect your account from fraud.

If you do not contact us, certain limitations may be placed on your debit card.

Grammatical Error

© 2014 Bank of America Corporation. All rights reserved.

Forgetting and Recovering Passwords

- Security questions:
  - What's your childhood pet?
- Password recovery email
  - Click here to reset your password!
- Send in plaintext to email
  - Your password is "in$3cur3"

Good security?

Forgetting and Recovering Passwords

- Security questions:
  - What's your childhood pet?
- Password recovery email
  - Click here to reset your password!
- Send in plaintext to email
  - Your password is "in$3cur3"

Bad security! Attacker might have control of the victim's **email**!

Forgetting and Recovering Passwords

- Security questions:
  - What's your childhood pet?
- Password recovery email
  - Click here to reset your password!
- Send in plaintext to email
  - Your password is "in$3cur3"
- Other approaches:
  - Phone call
  - Session-specific PIN

Bad security! Attacker might have control of the victim's **email**!

Trade-offs?

Questions?

Next time on CS 4440...

Security in Practice:
TOR—The Onion Router