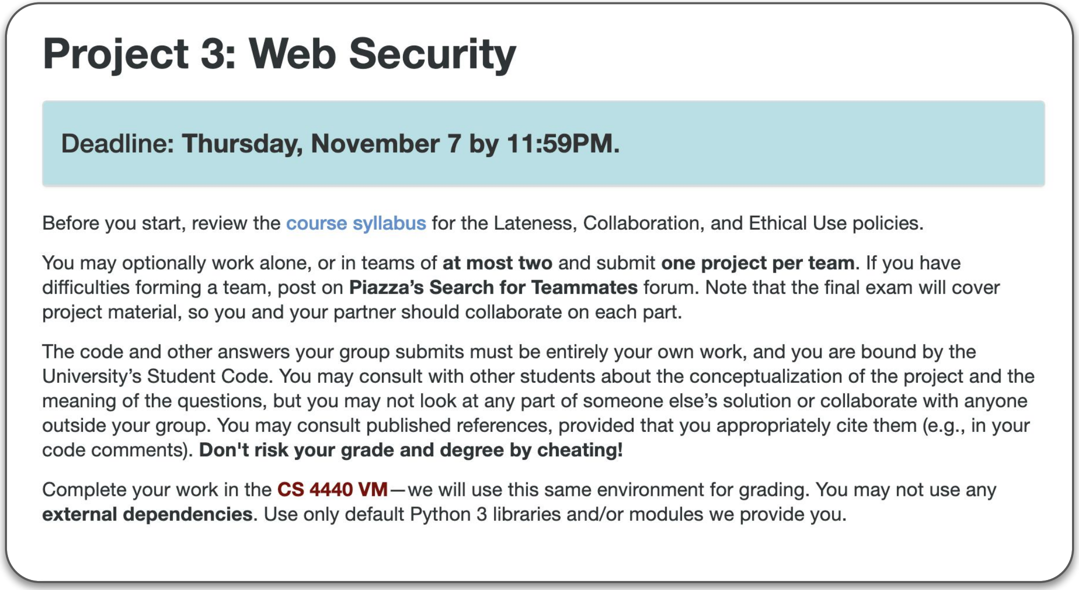
Week 11: Lecture A

Secure Authentication

Tuesday, November 5, 2024

Announcements

- **Project 3: WebSec** released
  - **Deadline: this Thursday**, November 7th by 11:59PM



Project 3: Web Security

Deadline: Thursday, November 7 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!**

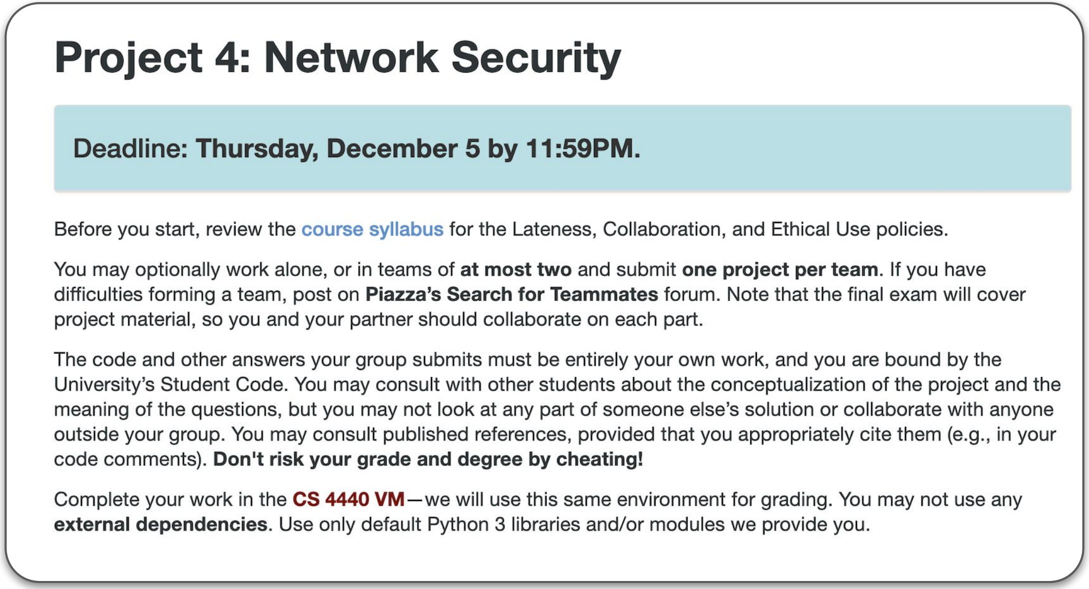
Complete your work in the **CS 4440 VM**—we will use this same environment for grading. You may not use any **external dependencies**. Use only default Python 3 libraries and/or modules we provide you.

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 5th by 11:59PM



Project 4: Network Security

Deadline: Thursday, December 5 by 11:59PM.

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

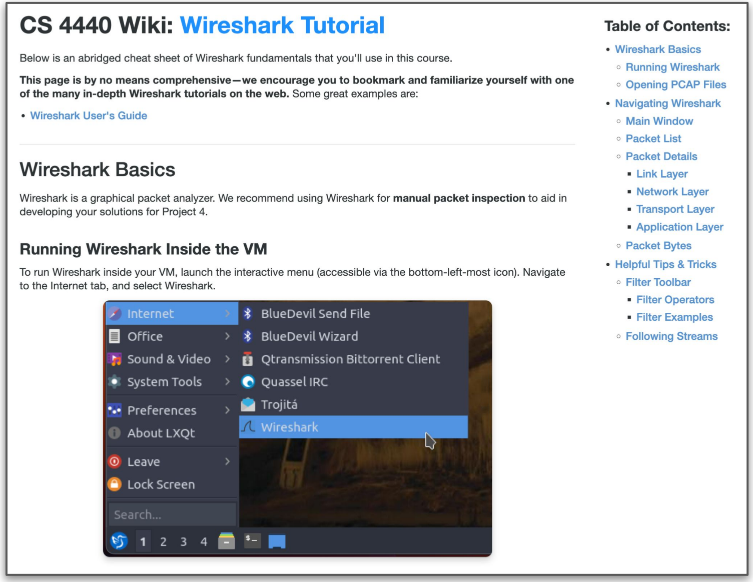
You may optionally work alone, or in teams of **at most two** and submit **one project per team**. If you have difficulties forming a team, post on **Piazza's Search for Teammates** forum. Note that the final exam will cover project material, so you and your partner should collaborate on each part.

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.

Announcements

- New **Wiki pages** to help you on Project 4:



CS 4440 Wiki: Wireshark Tutorial

Below is an abridged cheat sheet of Wireshark fundamentals that you'll use in this course.

This page is by no means comprehensive—we encourage you to bookmark and familiarize yourself with one of the many in-depth Wireshark tutorials on the web. Some great examples are:

- Wireshark User's Guide

Wireshark Basics

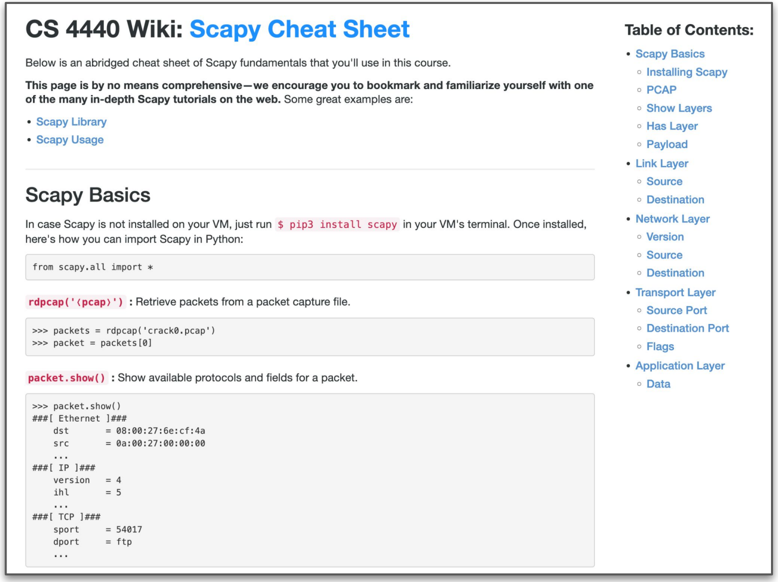
Wireshark is a graphical packet analyzer. We recommend using Wireshark for **manual packet inspection** to aid in developing your solutions for Project 4.

Running Wireshark Inside the VM

To run Wireshark inside your VM, launch the interactive menu (accessible via the bottom-left-most icon). Navigate to the Internet tab, and select Wireshark.

Table of Contents:

- Wireshark Basics
  - Running Wireshark
  - Opening PCAP Files
- Navigating Wireshark
  - Main Window
  - Packet List
  - Packet Details
    - Link Layer
    - Network Layer
    - Transport Layer
    - Application Layer
  - Packet Bytes
- Helpful Tips & Tricks
  - Filter Toolbar
    - Filter Operators
    - Filter Examples
  - Following Streams



CS 4440 Wiki: Scapy Cheat Sheet

Below is an abridged cheat sheet of Scapy fundamentals that you'll use in this course.

This page is by no means comprehensive—we encourage you to bookmark and familiarize yourself with one of the many in-depth Scapy tutorials on the web. Some great examples are:

- Scapy Library
- Scapy Usage

Scapy Basics

In case Scapy is not installed on your VM, just run `$ pip3 install scapy` in your VM's terminal. Once installed, here's how you can import Scapy in Python:

```
from scapy.all import *
```

`rdpcap('(pcap)')` : Retrieve packets from a packet capture file.

```
>>> packets = rdpcap('crack0.pcap')
>>> packet = packets[0]
```

`packet.show()` : Show available protocols and fields for a packet.

```
>>> packet.show()
###[ Ethernet ]###
    dst       = 08:00:27:6e:cf:4a
    src       = 0a:00:27:00:00:00
    ...
###[ IP ]###
    version   = 4
    ihl       = 5
    ...
###[ TCP ]###
    sport     = 54017
    dport     = ftp
    ...
```

Table of Contents:

- Scapy Basics
  - Installing Scapy
  - PCAP
  - Show Layers
  - Has Layer
  - Payload
- Link Layer
  - Source
  - Destination
- Network Layer
  - Version
  - Source
  - Destination
- Transport Layer
  - Source Port
  - Destination Port
  - Flags
- Application Layer
  - Data

Announcements

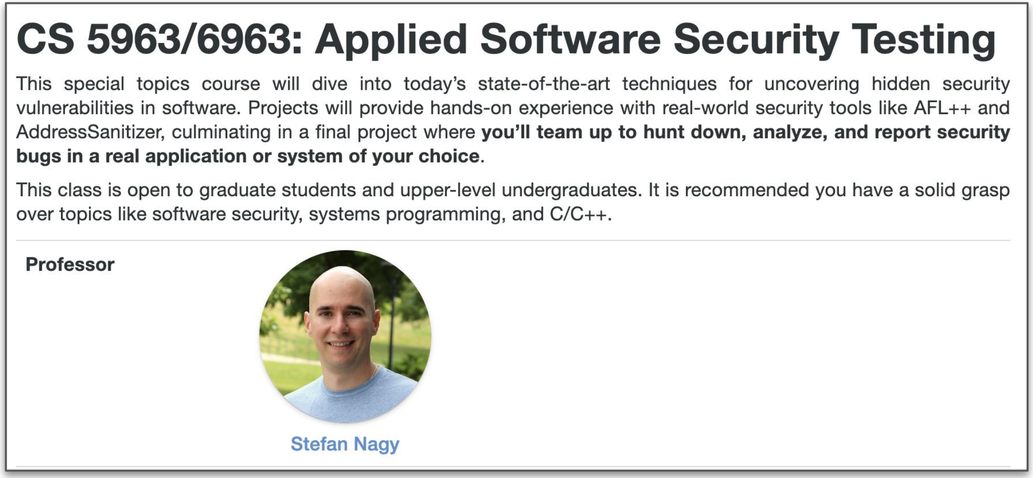
utahsec

See Discord for meeting info!

utahsec.cs.utah.edu

Interested in fuzzing?

- **Spring 2025: CS 5963/6963: 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!
  - https://cs.utah.edu/~snagy/courses/cs5963/



CS 5963/6963: 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. Projects will provide hands-on experience with real-world security tools like AFL++ and AddressSanitizer, culminating in a final project where **you'll team up to hunt down, analyze, and report security bugs in a real application or system 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++.

Professor

Stefan Nagy

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: ???**

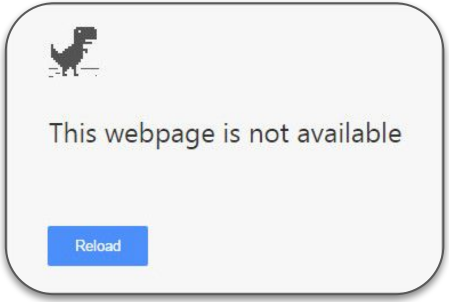
DoS: Denial of Service

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

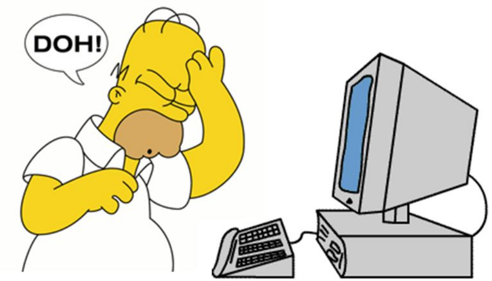
DoS: Denial of Service

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

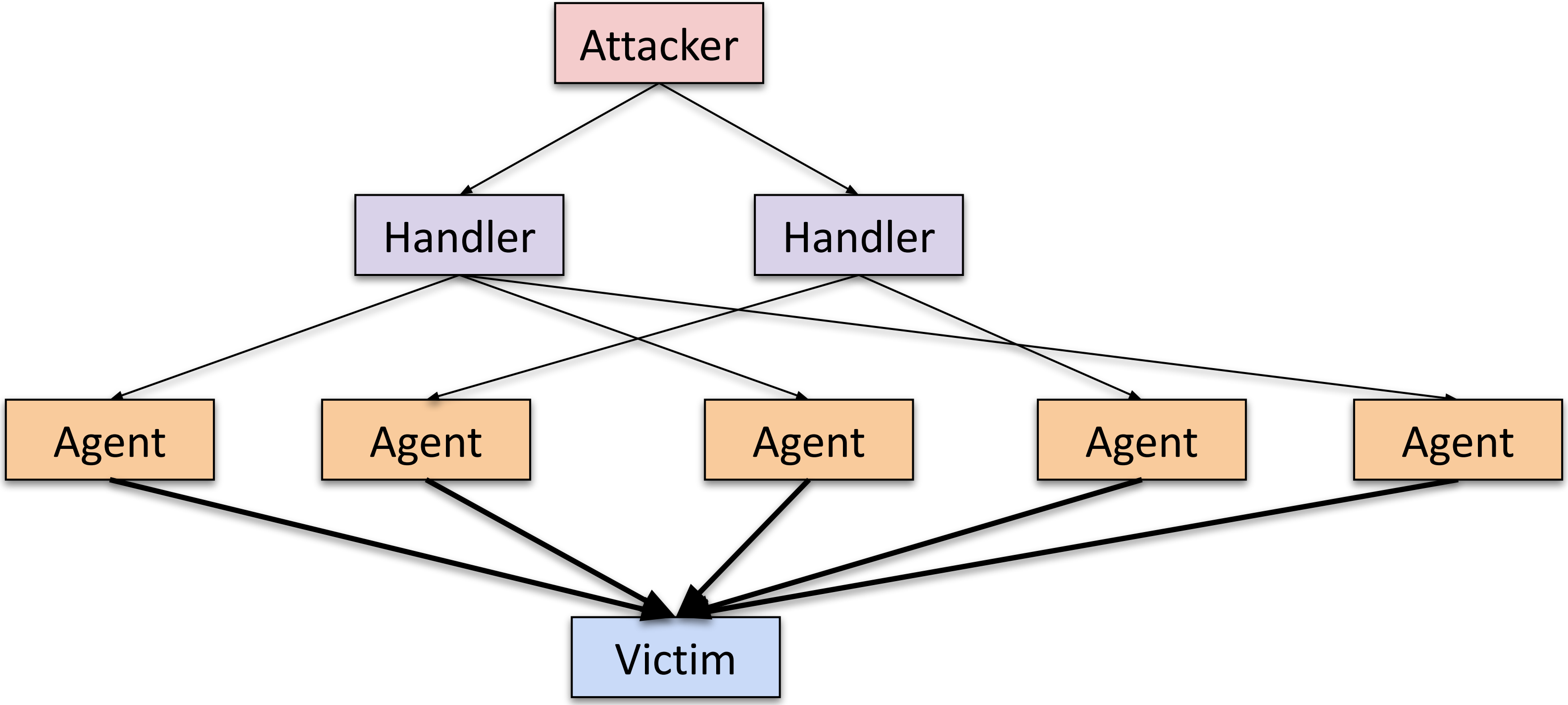
- **How?**
  - Trigger the host to **crash**
    - Application-based DoS
    - Memory corruption
  - Consume host's **resources**
    - TCP SYN floods
    - ICMP ECHO (ping) floods
  - Consume host's **bandwidth**
    - UDP floods
    - ICMP floods



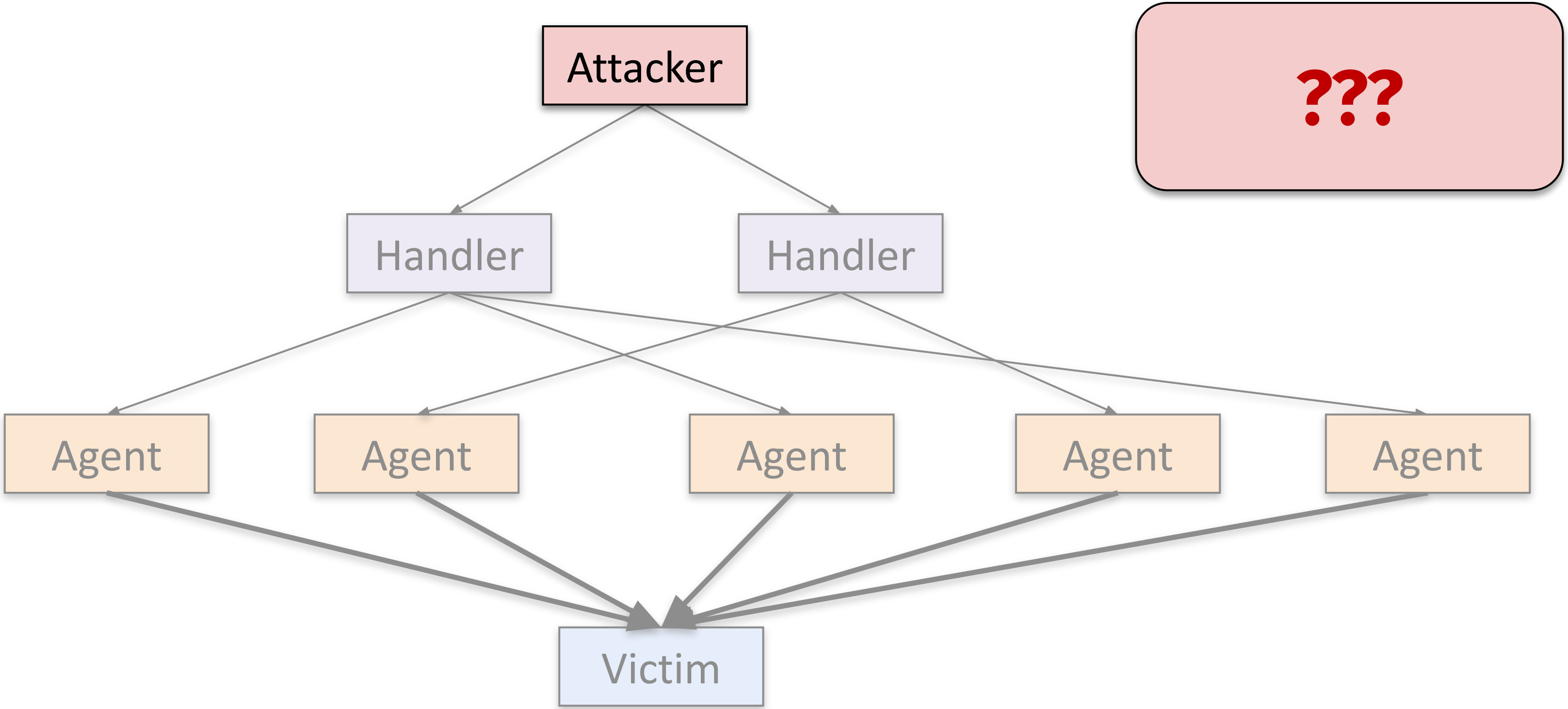
This webpage is not available



Distributed DoS Attacks (DDoS)

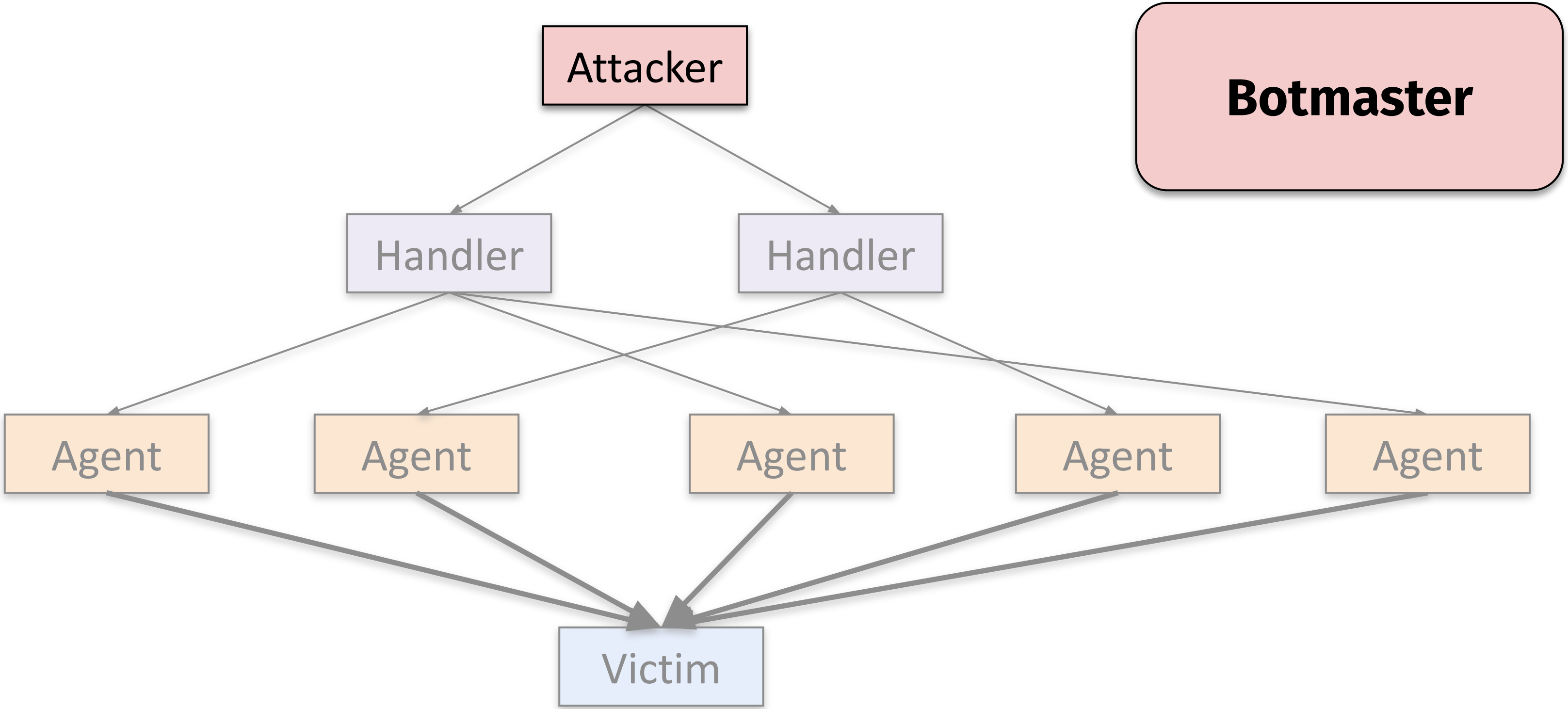


Distributed DoS Attacks (DDoS)



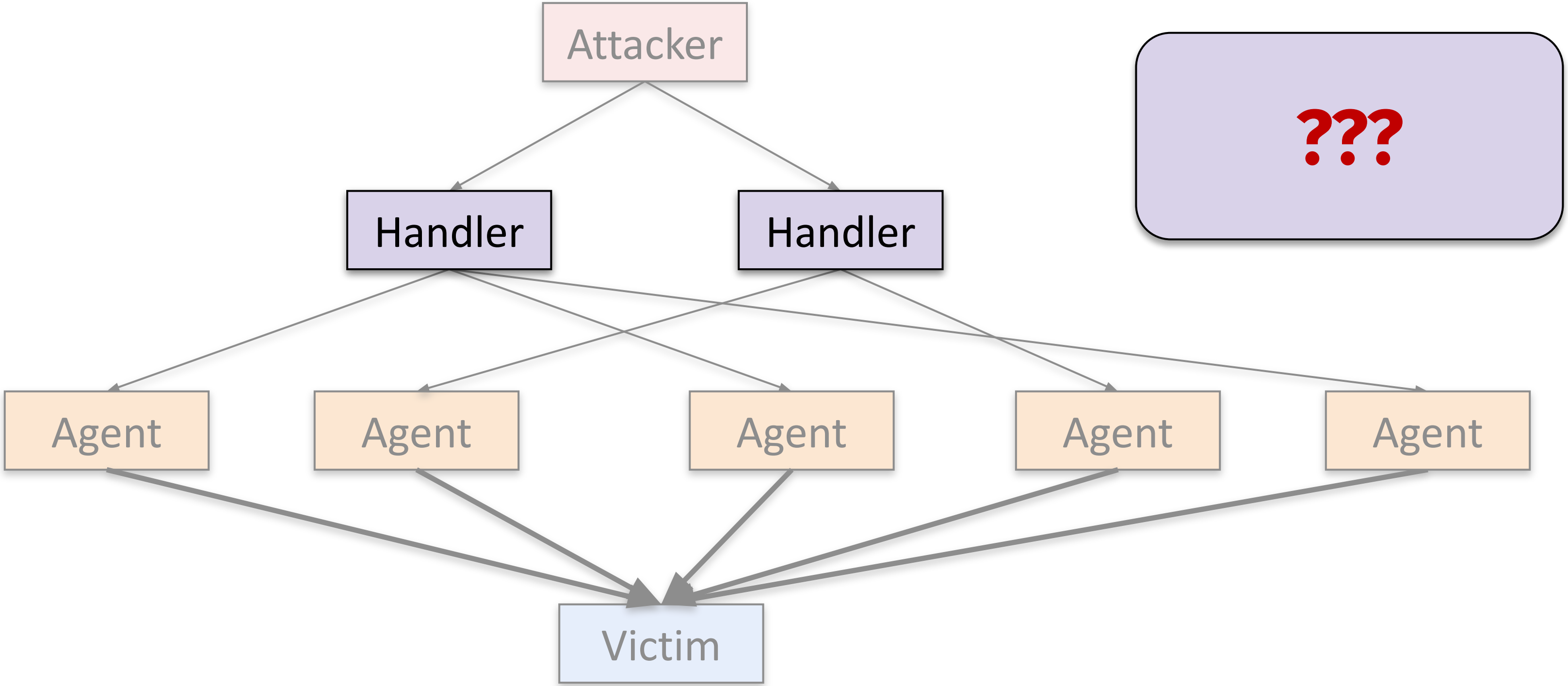
???

Distributed DoS Attacks (DDoS)



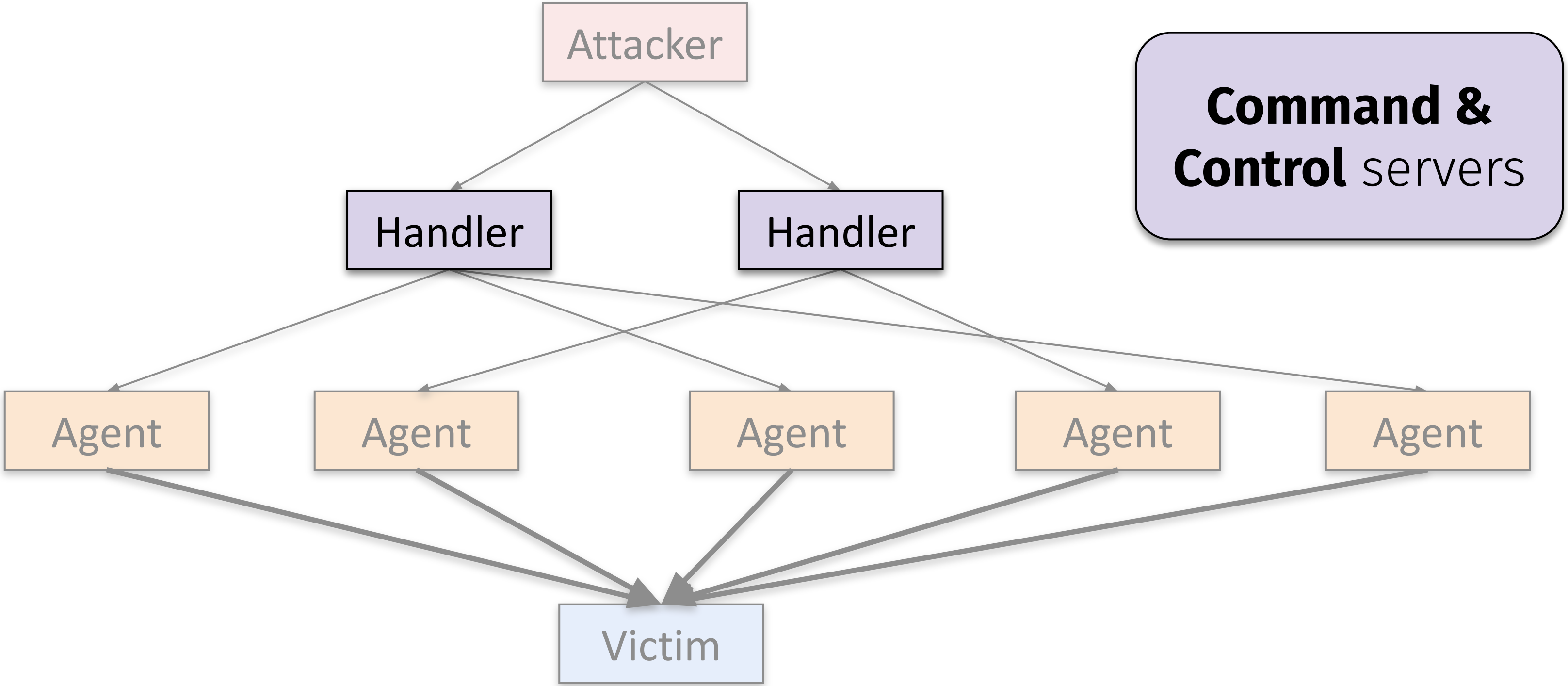
Botmaster

Distributed DoS Attacks (DDoS)



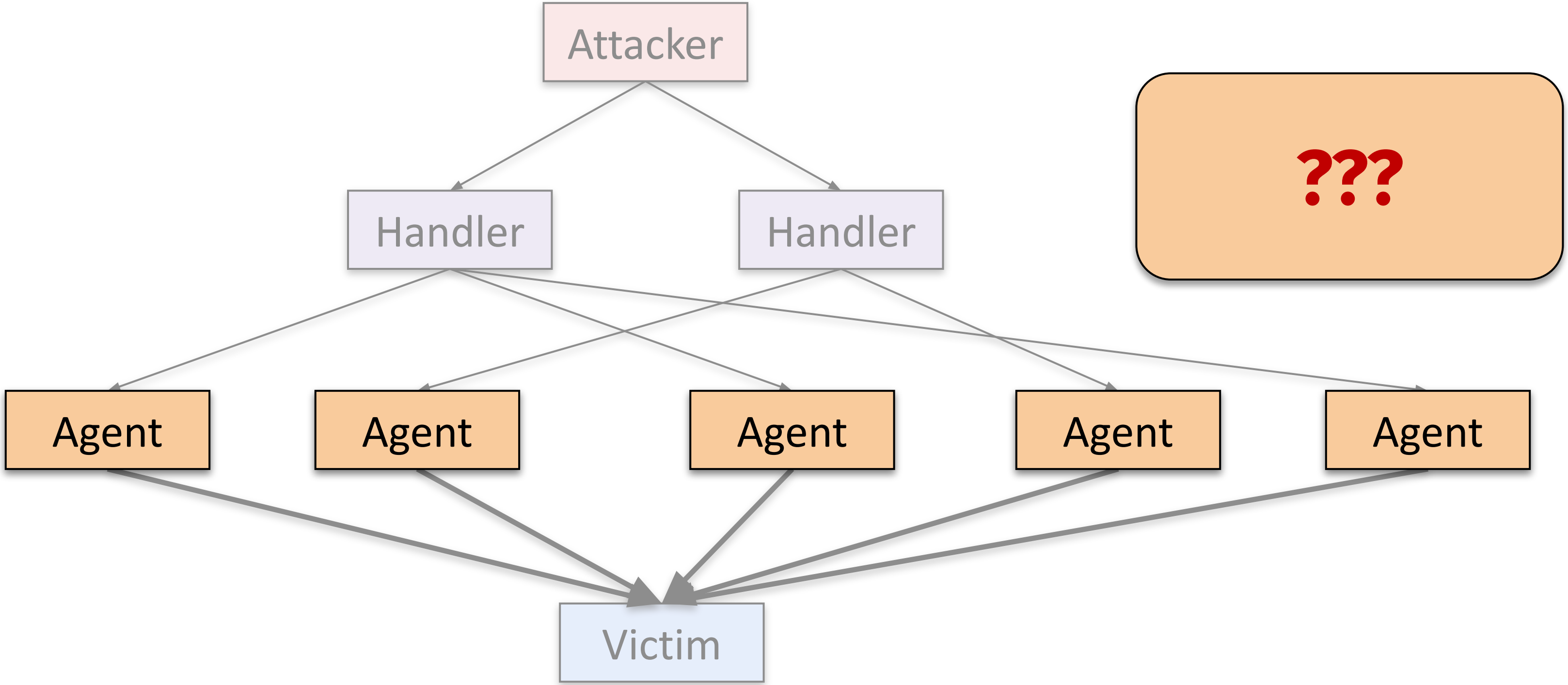
???

Distributed DoS Attacks (DDoS)

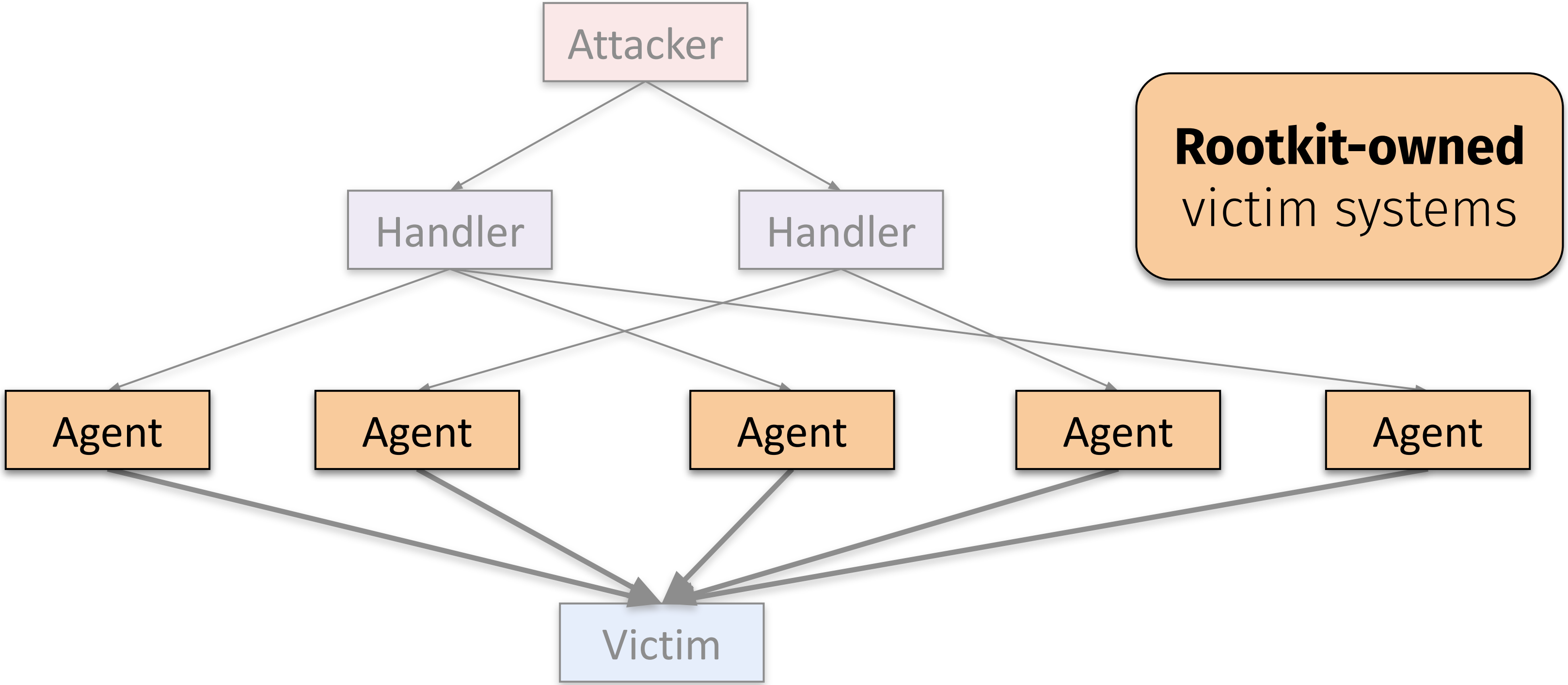


Command & Control servers

Distributed DoS Attacks (DDoS)

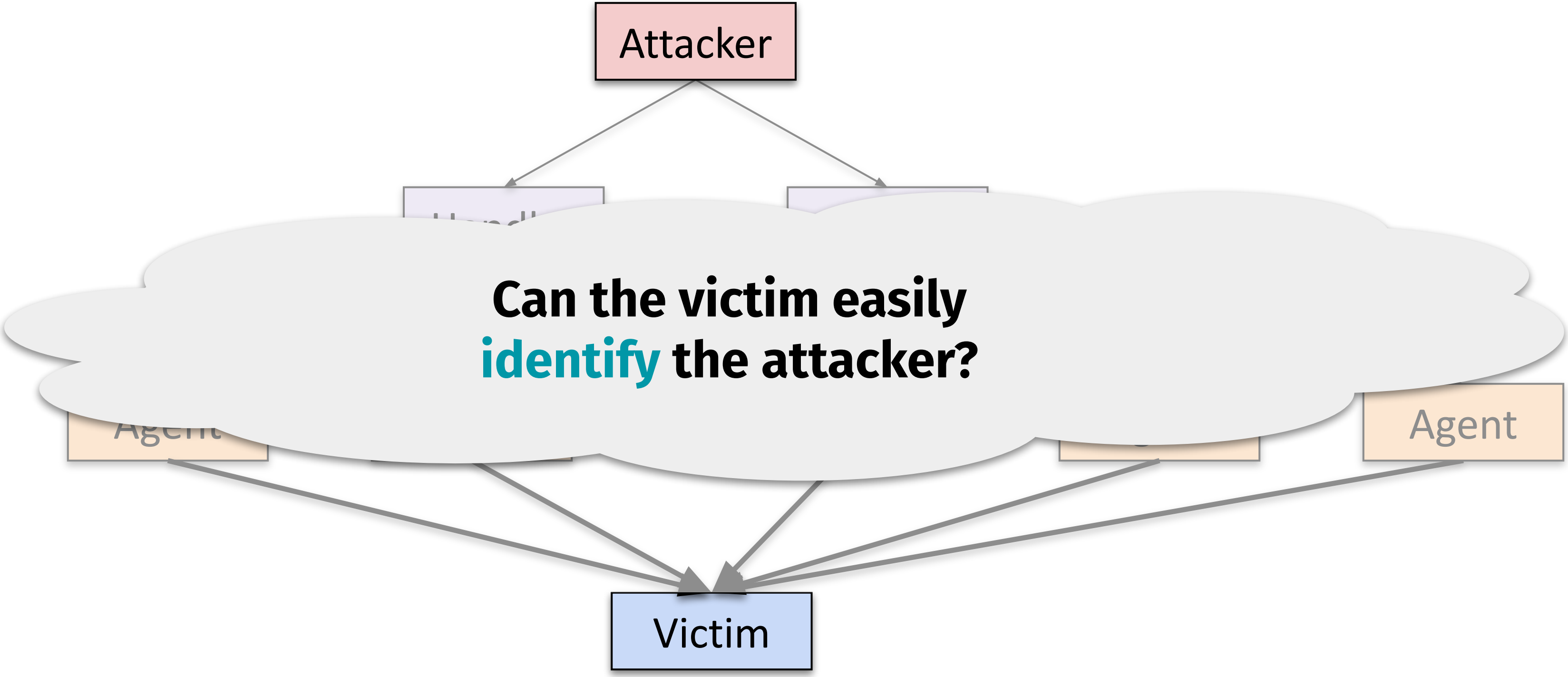


Distributed DoS Attacks (DDoS)



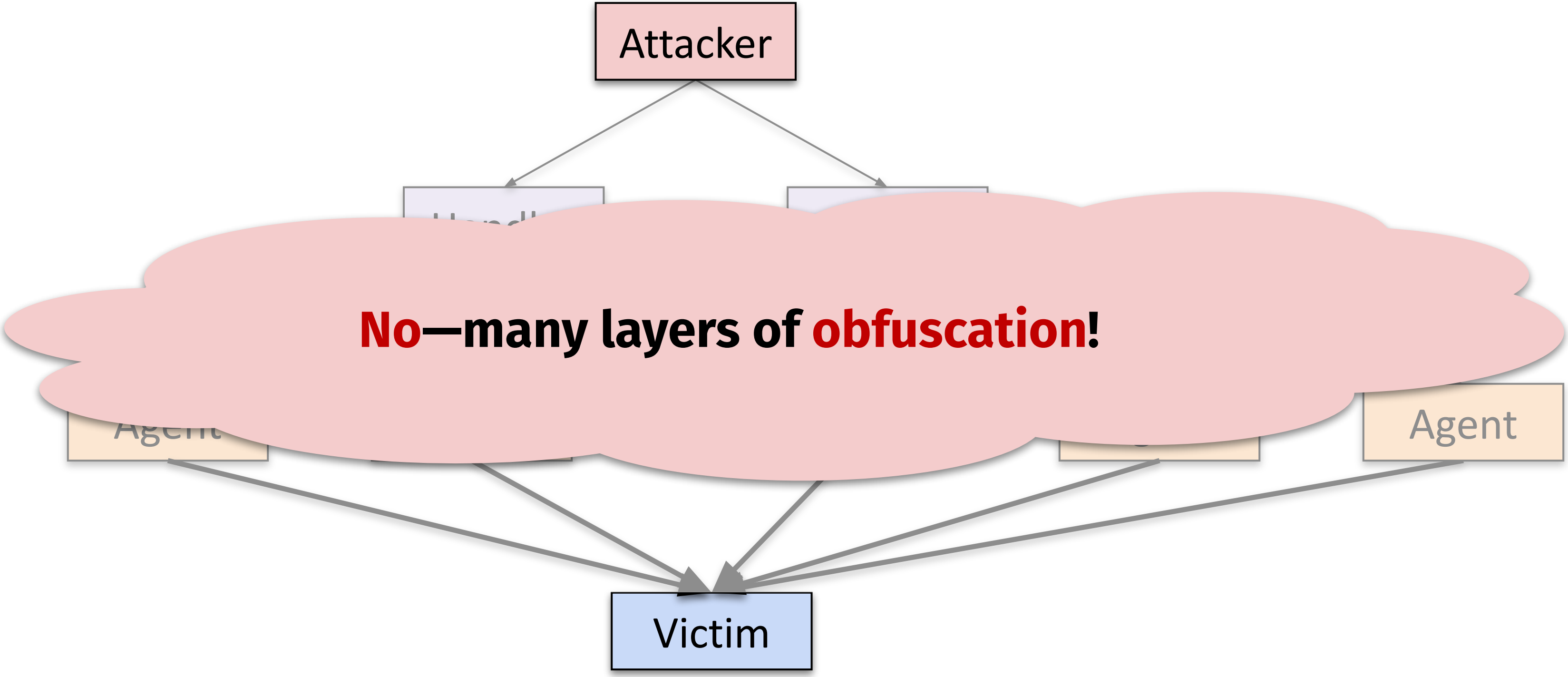
Rootkit-owned victim systems

Distributed DoS Attacks (DDoS)



Can the victim easily identify the attacker?

Distributed DoS Attacks (DDoS)



No—many layers of obfuscation!

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?**
  - **???**

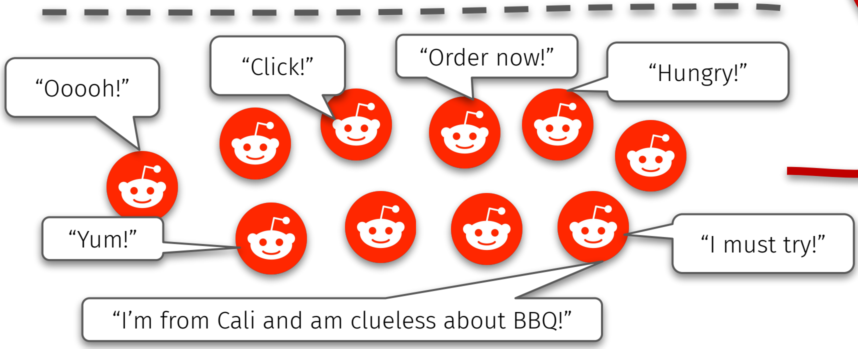
Advanced DoS Strategies

- **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!"

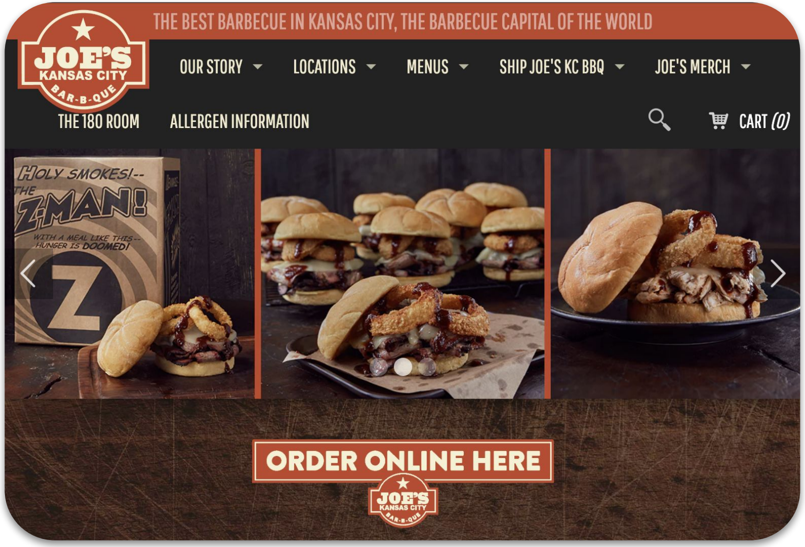
"Hungry!"

"Yum!"

"I must try!"

"I'm from Cali and am clueless about BBQ!"

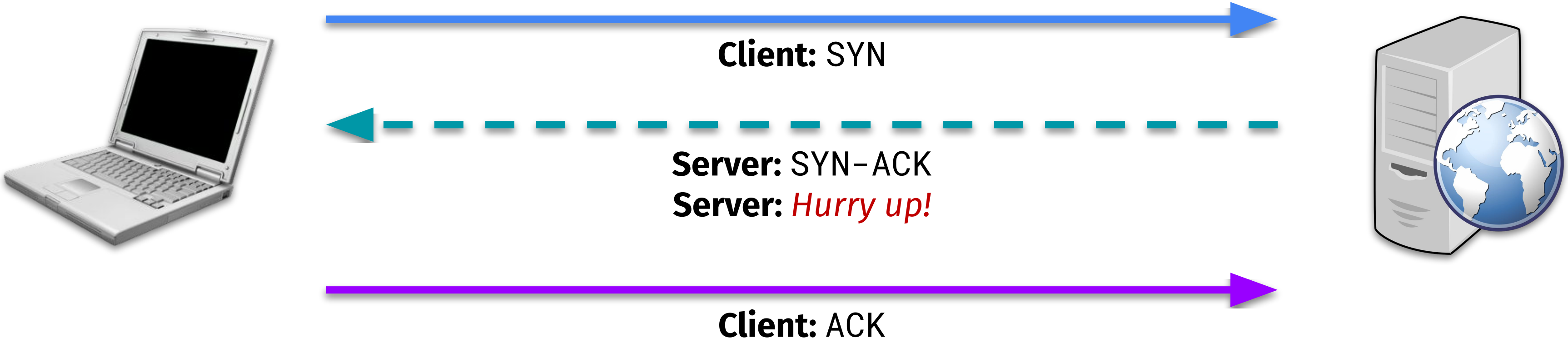
https://www.joeskc.com/



THE BEST BARBECUE IN KANSAS CITY, THE BARBECUE CAPITAL OF THE WORLD

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**



Client: SYN

Server: SYN-ACK
Server: *Hurry up!*

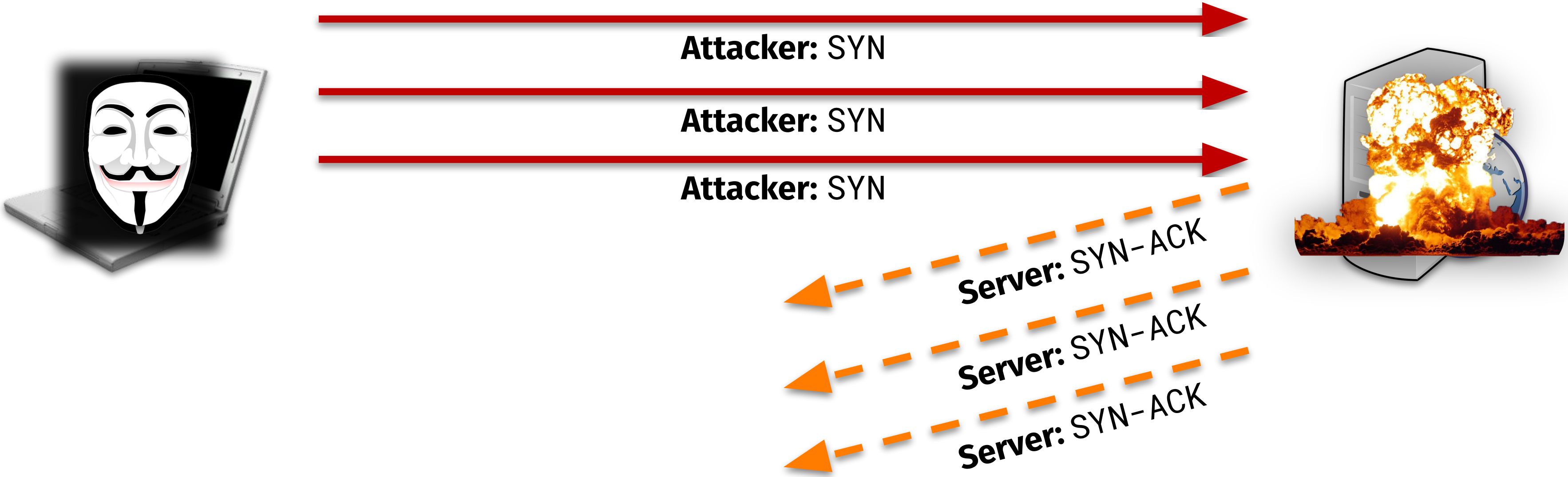
Client: ACK

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**



Attacker: SYN

Attacker: SYN

Attacker: SYN

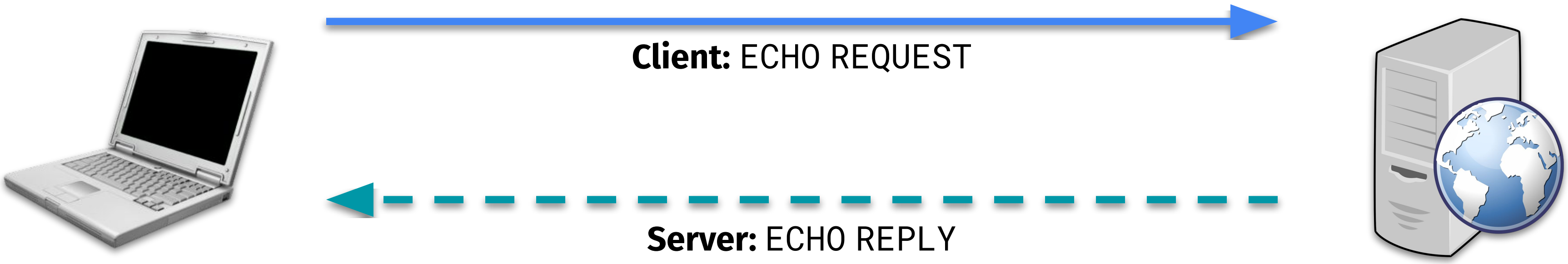
Server: SYN-ACK

Server: SYN-ACK

Server: SYN-ACK

ICMP: Internet Control Message Protocol

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



Client: ECHO REQUEST

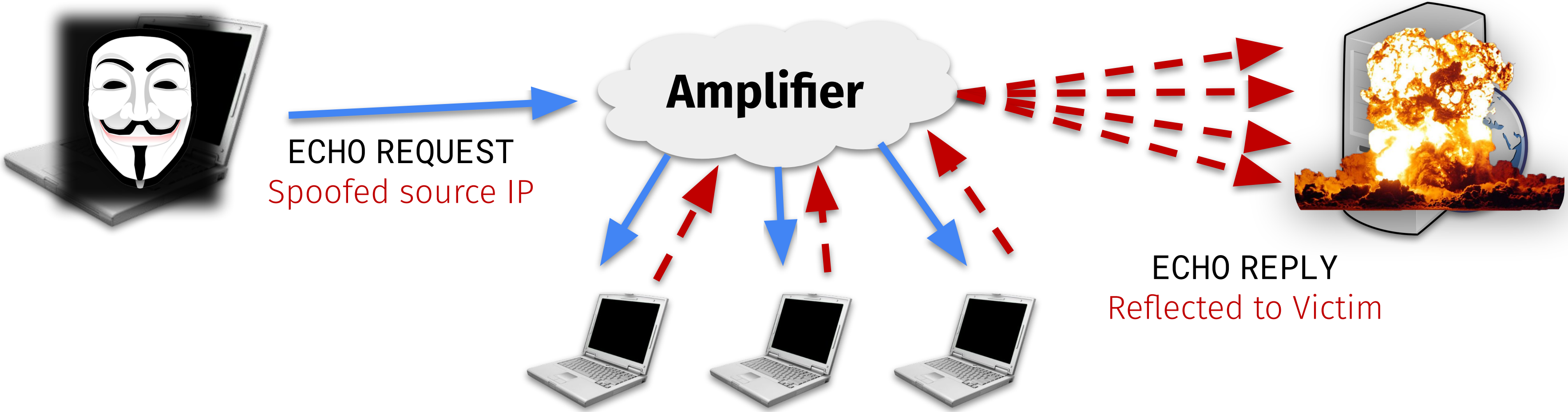
Server: ECHO REPLY

ICMP Smurf Attacks

- **Attack: ???**

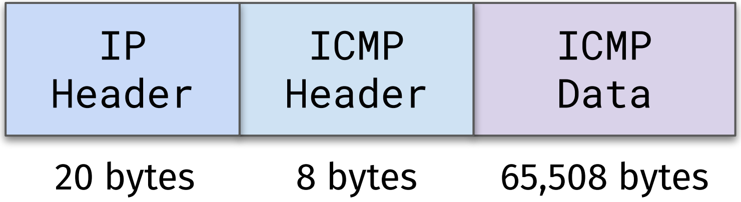
ICMP Smurf Attacks

- **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

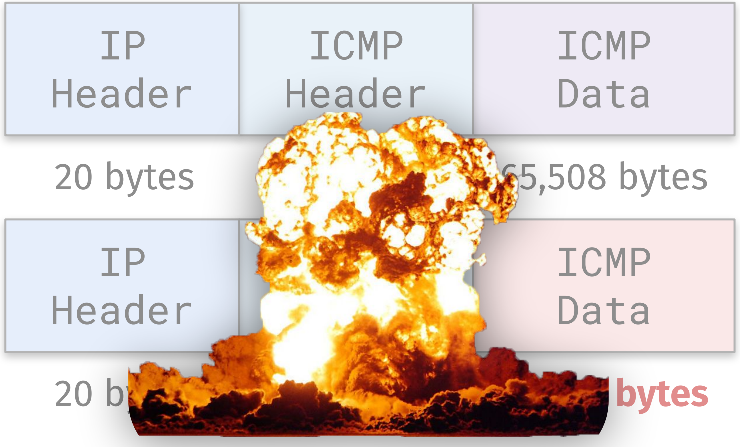
- **Internet Protocol:** IPV4 packets should be **less than 65,536 bytes**
  - Packets can be sent in **fragments** and **reassembled** by receiver
- **Attack: ???**



IP Header	ICMP Header	ICMP Data
20 bytes	8 bytes	65,508 bytes

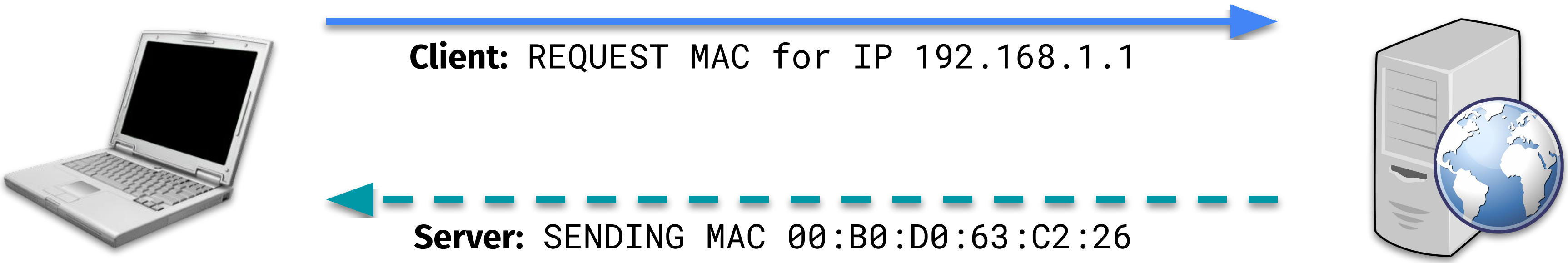
ICMP Ping of Death Attack

- **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



Client: `REQUEST MAC for IP 192.168.1.1`

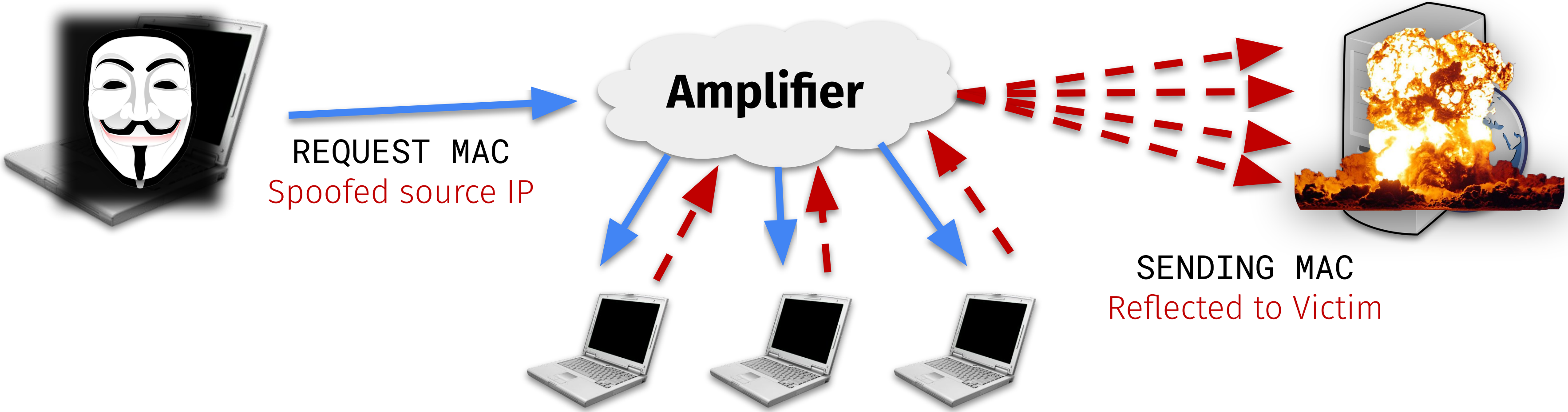
Server: `SENDING MAC 00:B0:D0:63:C2:26`

ARP Flooding Attack

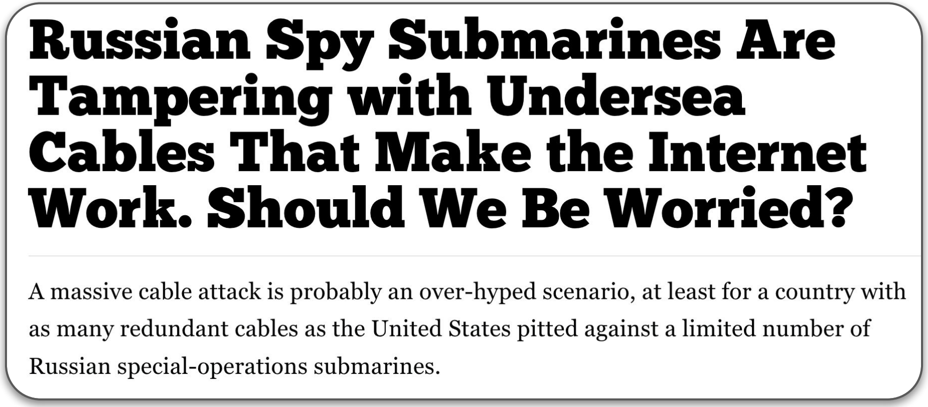
- **Attack: ???**

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

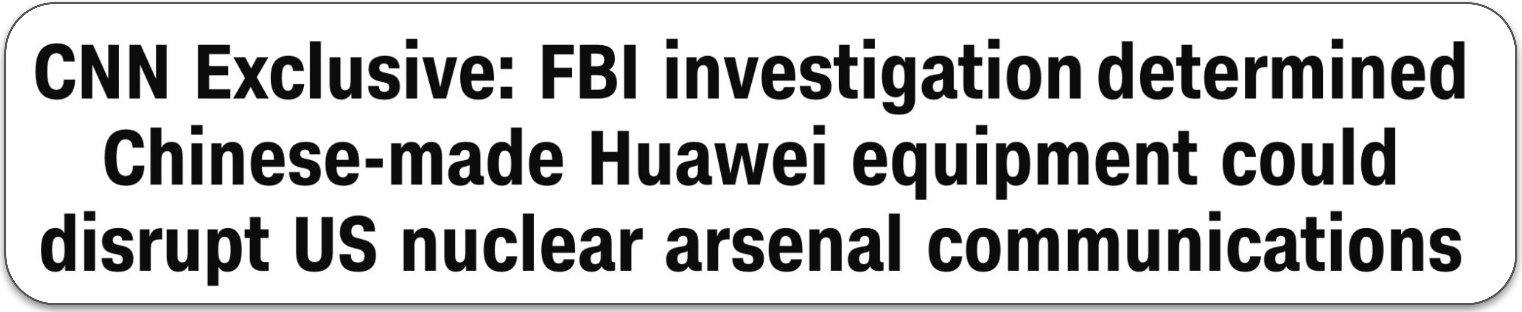


Physical Layer DoS



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

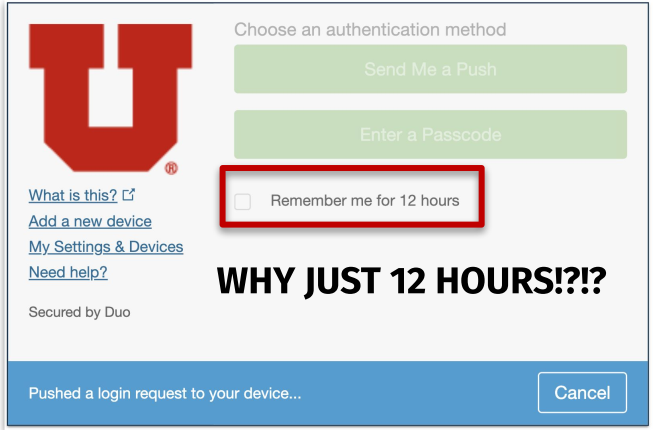
What is authentication?

- **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

Sorry, but your password must contain an uppercase letter, a number, a hieroglyph, a feather from a hawk and the blood of a unicorn.
someecards
user card


Choose an authentication method

□ Remember me for 12 hours

Pushed a login request to your device...

WHY JUST 12 HOURS!?!?

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

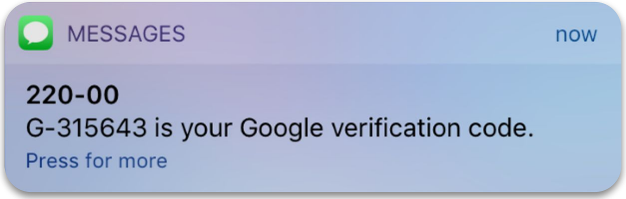
- Something you **???**

- Something you **???**

- Something you **???**

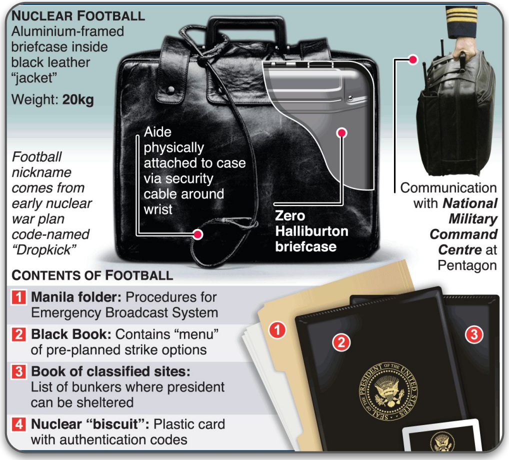
The Three Factors of Authentication

- 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



220-00

G-315643 is your Google verification code.


NUCLEAR FOOTBALL
Aluminium-framed briefcase inside black leather "jacket"
Weight: 20kg
Football nickname comes from early nuclear war plan code-named "Dropkick"
Aide physically attached to case via security cable around wrist
Zero Halliburton briefcase
Communication with National Military Command Centre at Pentagon
CONTENTS OF FOOTBALL
1 Manila folder: Procedures for Emergency Broadcast System
2 Black Book: Contains "menu" of pre-planned strike options
3 Book of classified sites: List of bunkers where president can be sheltered
4 Nuclear "biscuit": Plastic card with authentication codes
1
2
3

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

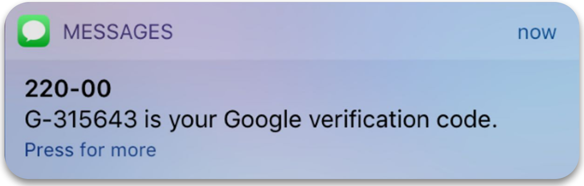
- **One-time PINs / Passwords:**
  - Password valid for only **one** login session or transaction

- **Delivering** One-time PINs:
  - **???**

One-time PINs

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

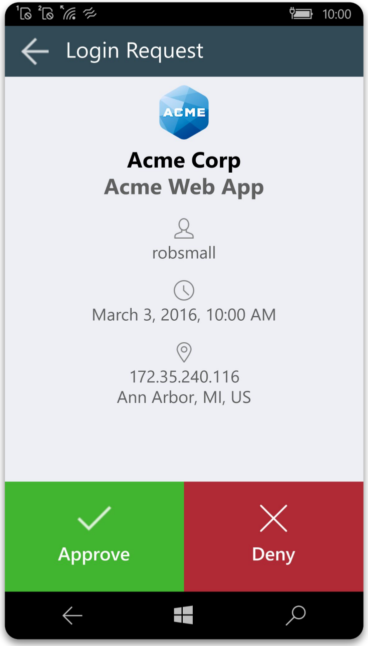
- **Delivering** One-time PINs:
  - **SMS**
    - Phone call
    - Text message
  - **Hardware**
    - Yubico YubiKey
    - RSA SecureID
  - **Application**
    - DUO Mobile
    - Google authenticator


MESSAGES now

220-00

G-315643 is your Google verification code.

Press for more


10:00

Login Request

ACME

Acme Corp

Acme Web App

robsmall

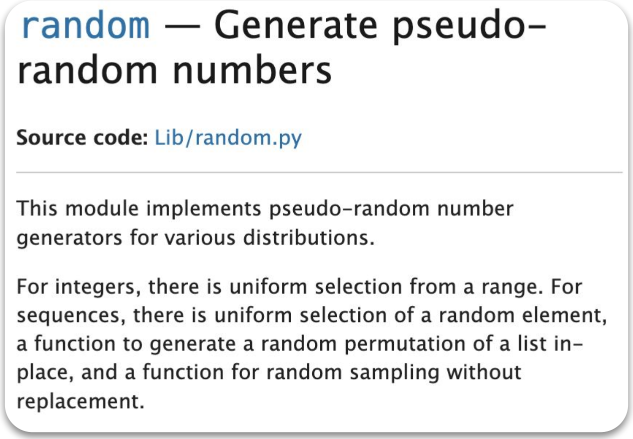
March 3, 2016, 10:00 AM

172.35.240.116
Ann Arbor, MI, US

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.

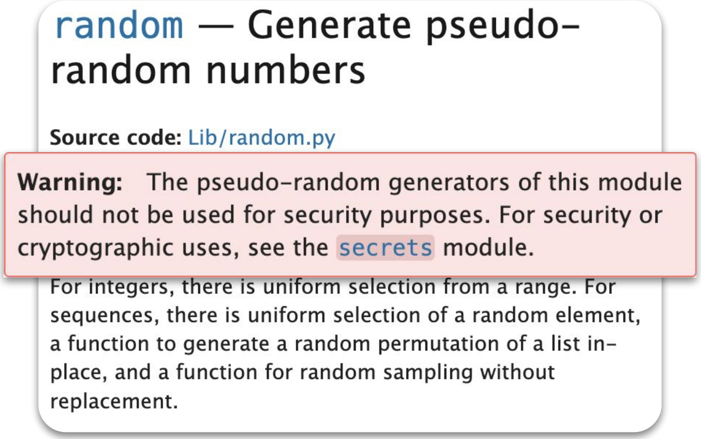
For integers, there is uniform selection from a range. For sequences, there is uniform selection of a random element, a function to generate a random permutation of a list in-place, and a function for random sampling without replacement.

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



`random` — Generate pseudo-random numbers

Source code: Lib/random.py

Warning: The pseudo-random generators of this module should not be used for security purposes. For security or cryptographic uses, see the `secrets` module.

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.

Attack: SIM Swap

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

Attack: SIM Swap

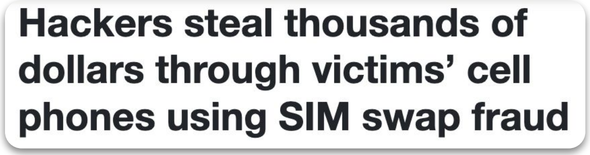
- **SIM: Subscriber Identity Module**
  - A small card inserted into your phone
  - Connects you to your carrier’s network
- **Social engineering attack:**
  - Learn key info about victim. E.g.:
    - Mothers’ maiden name
    - Childhood street address
  - Trick carrier to issue new SIM card
    - “I’m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

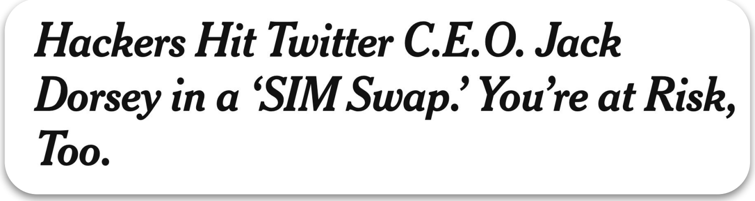
Attack: SIM Swap

- **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

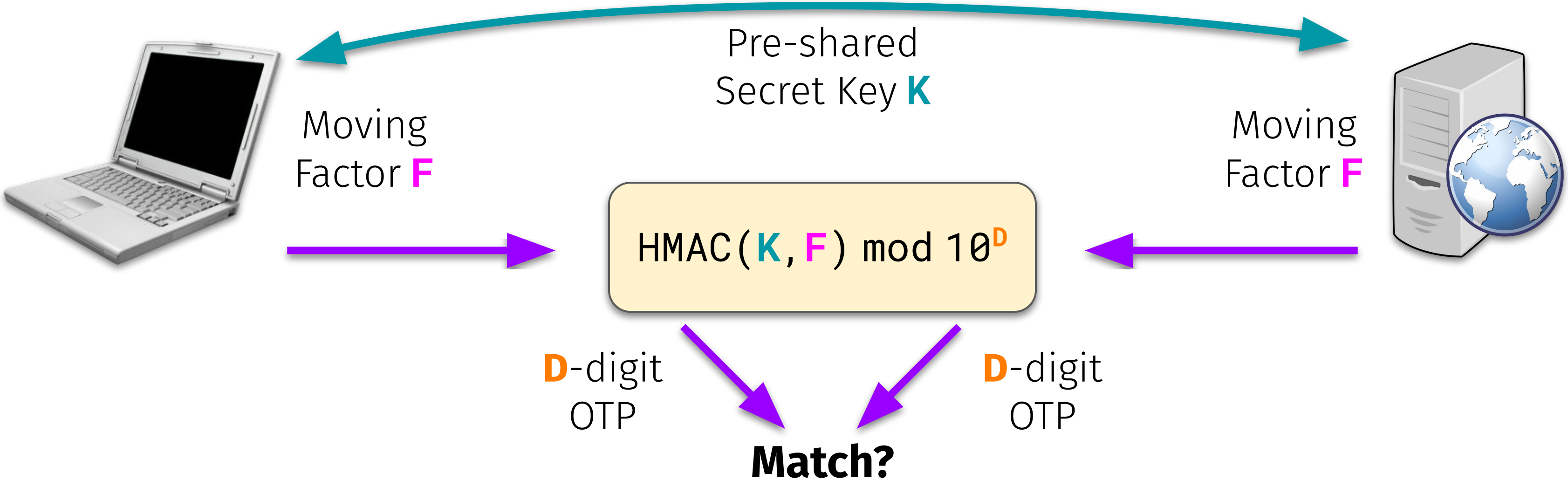
- **Result:** attacker is **man-in-the-middle**
  - **Receives any OTPs transmitted by SMS!**


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.

Implementing OTPs

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



Pre-shared Secret Key **K**

Moving Factor **F**

Moving Factor **F**

HMAC(K,F) mod 10^D

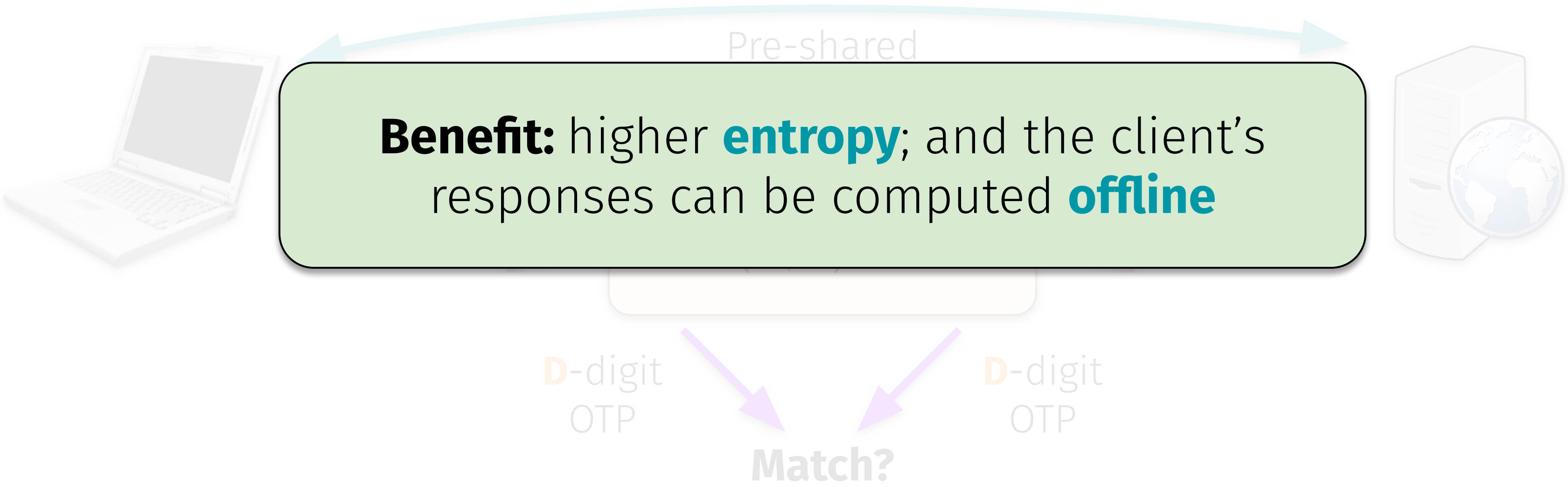
D-digit OTP

D-digit OTP

Match?

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

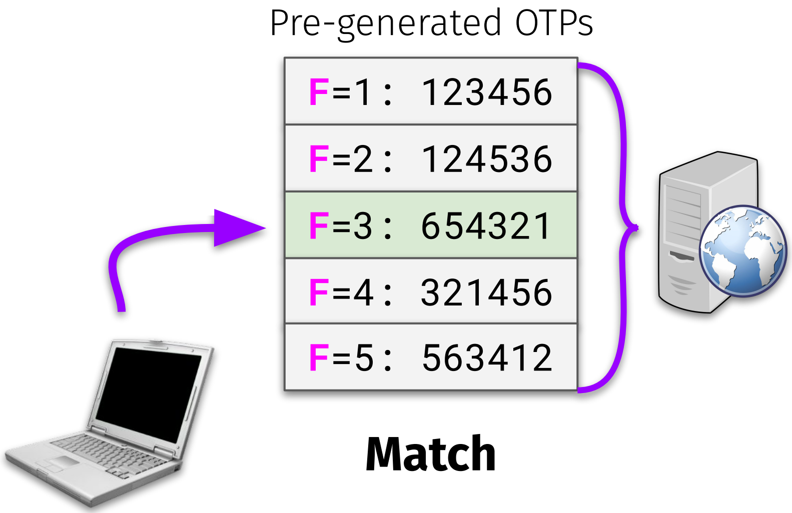
- **Problem: desynchronization**
  - E.g., user hits “login” one too many times

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



Pre-generated OTPs

F=1 :	123456
F=2 :	124536
F=3 :	654321
F=4 :	321456
F=5 :	563412

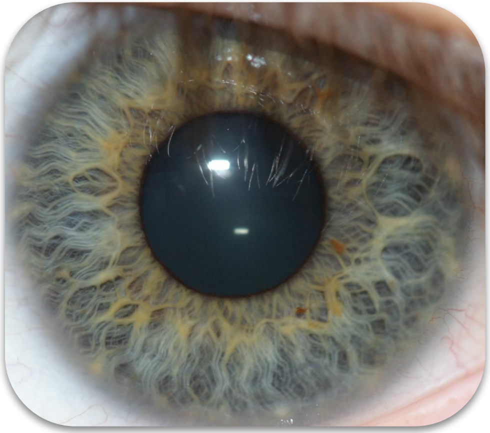
Match

Questions?

Biometrics

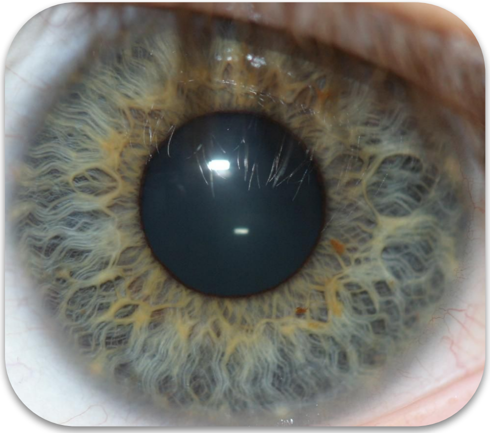
Biometrics

- **Provides proof of ???**



Biometrics

- Provides proof of **physical identity**

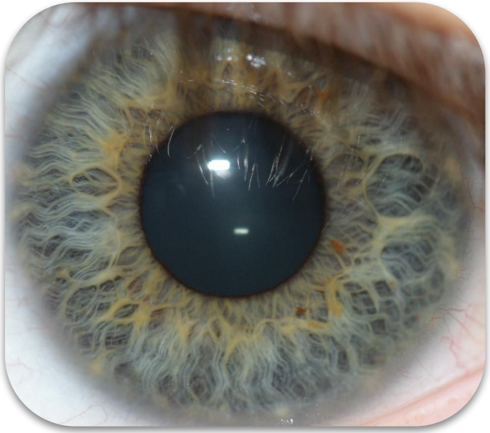


Biometrics

- **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: **???**



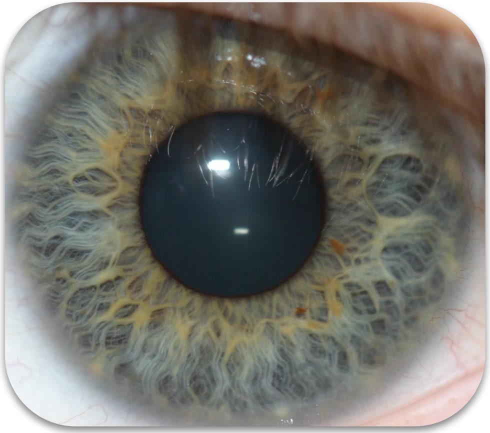
Biometrics

- **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?**
  - **???**



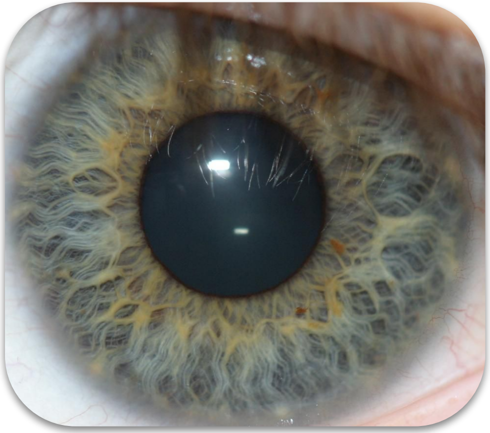
Biometrics

- **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

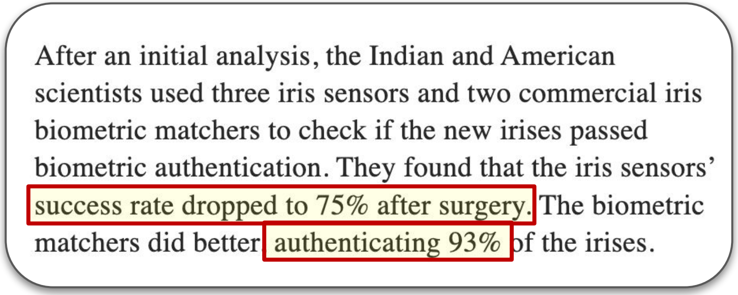


Biometric Challenges

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

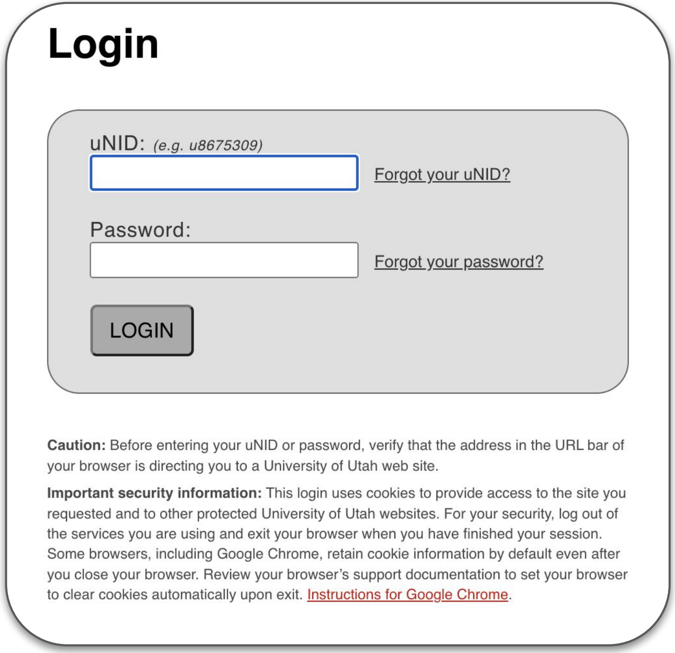
On the other hand he was fine

Questions?

Passwords

Passwords

- **Something that you ???**



Login

uNID: *(e.g. u8675309)*

Forgot your uNID?

Password:

Forgot your password?

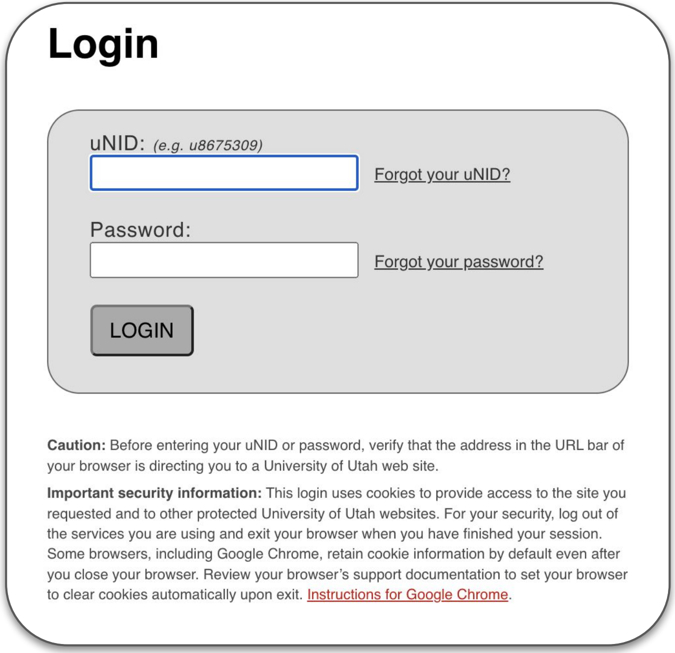
LOGIN

Caution: Before entering your uNID or password, verify that the address in the URL bar of your browser is directing you to a University of Utah web site.

Important security information: This login uses cookies to provide access to the site you requested and to other protected University of Utah websites. For your security, log out of the services you are using and exit your browser when you have finished your session. Some browsers, including Google Chrome, retain cookie information by default even after you close your browser. Review your browser's support documentation to set your browser to clear cookies automatically upon exit. Instructions for Google Chrome.

Passwords

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



Login

uNID: *(e.g. u8675309)*

Forgot your uNID?

Password:

Forgot your password?

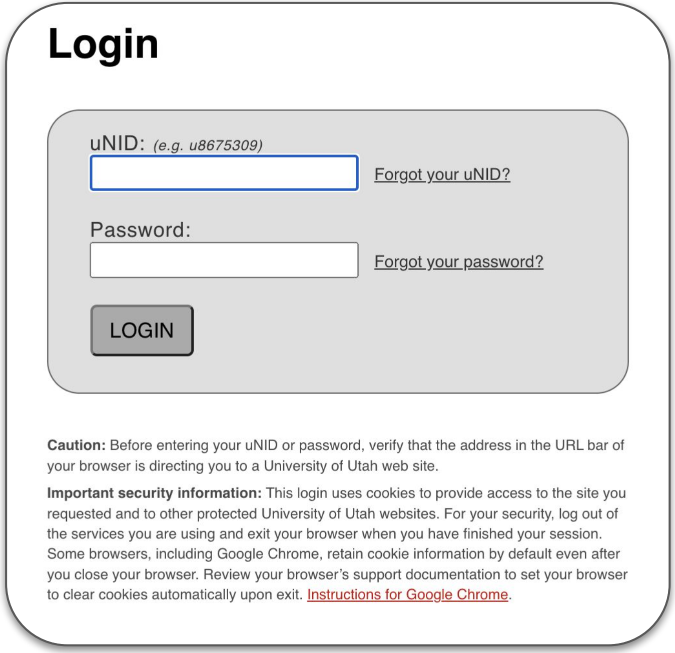
Caution: Before entering your uNID or password, verify that the address in the URL bar of your browser is directing you to a University of Utah web site.

Important security information: This login uses cookies to provide access to the site you requested and to other protected University of Utah websites. For your security, log out of the services you are using and exit your browser when you have finished your session. Some browsers, including Google Chrome, retain cookie information by default even after you close your browser. Review your browser's support documentation to set your browser to clear cookies automatically upon exit. Instructions for Google Chrome.

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** ($#%-_!)



Login

uNID: *(e.g. u8675309)*

Forgot your uNID?

Password:

Forgot your password?

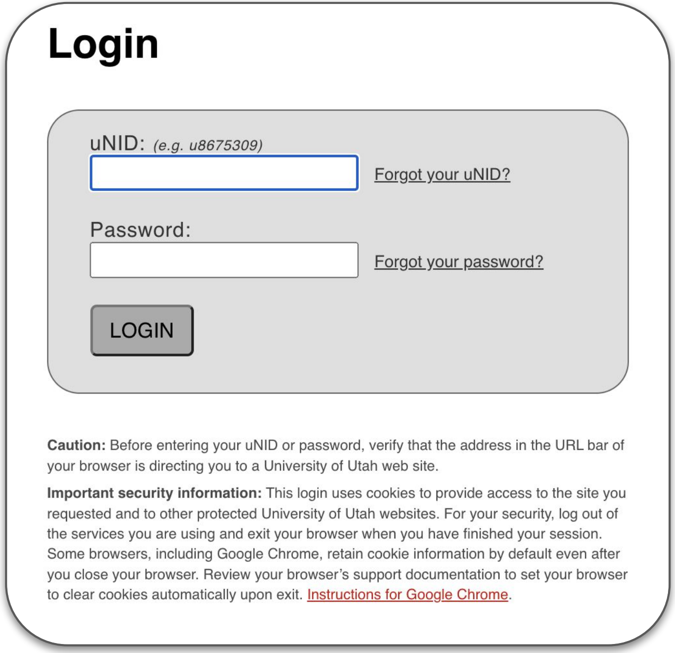
LOGIN

Caution: Before entering your uNID or password, verify that the address in the URL bar of your browser is directing you to a University of Utah web site.

Important security information: This login uses cookies to provide access to the site you requested and to other protected University of Utah websites. For your security, log out of the services you are using and exit your browser when you have finished your session. Some browsers, including Google Chrome, retain cookie information by default even after you close your browser. Review your browser's support documentation to set your browser to clear cookies automatically upon exit. Instructions for Google Chrome.

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?**



Login

uNID: *(e.g. u8675309)*

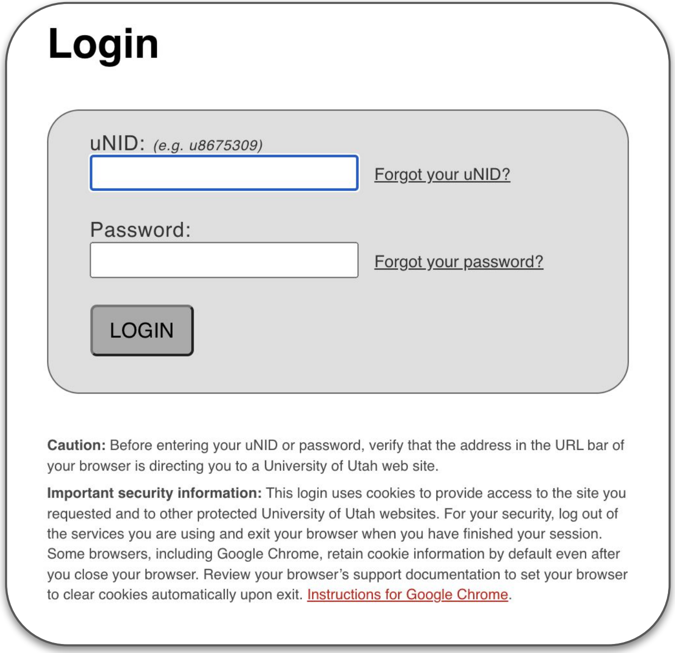
Password:

Caution: Before entering your uNID or password, verify that the address in the URL bar of your browser is directing you to a University of Utah web site.

Important security information: This login uses cookies to provide access to the site you requested and to other protected University of Utah websites. For your security, log out of the services you are using and exit your browser when you have finished your session. Some browsers, including Google Chrome, retain cookie information by default even after you close your browser. Review your browser's support documentation to set your browser to clear cookies automatically upon exit. Instructions for Google Chrome.

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!**



Login

uNID: *(e.g. u8675309)*

Forgot your uNID?

Password:

Forgot your password?

Caution: Before entering your uNID or password, verify that the address in the URL bar of your browser is directing you to a University of Utah web site.

Important security information: This login uses cookies to provide access to the site you requested and to other protected University of Utah websites. For your security, log out of the services you are using and exit your browser when you have finished your session. Some browsers, including Google Chrome, retain cookie information by default even after you close your browser. Review your browser's support documentation to set your browser to clear cookies automatically upon exit. Instructions for Google Chrome.

Why aren’t passwords cryptographically secure?

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

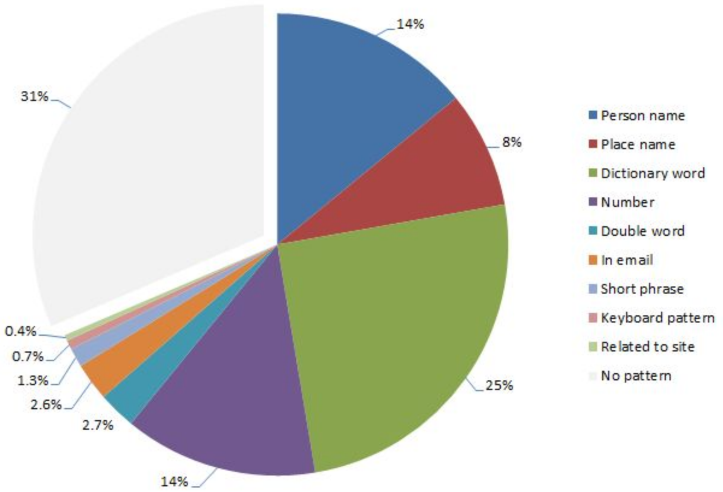
Why aren’t passwords cryptographically secure?

- **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:

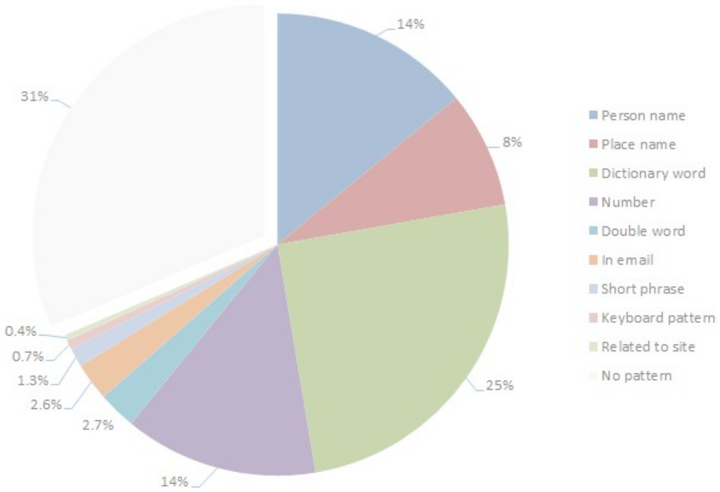
Patterns across **all** passwords



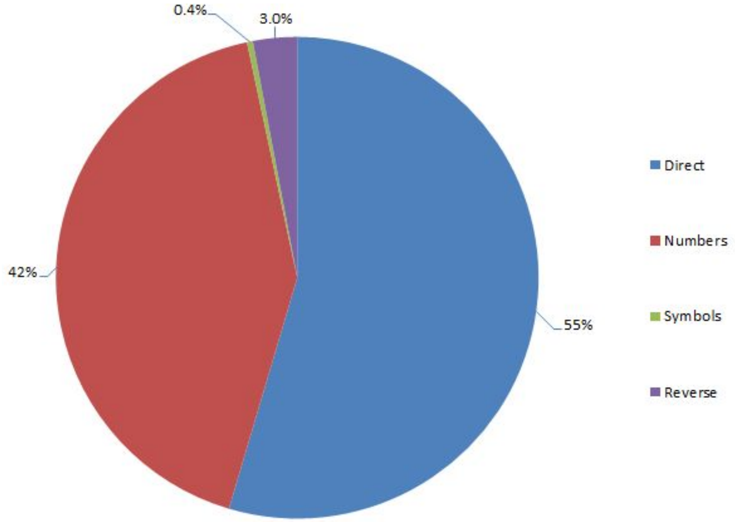
Why aren’t passwords cryptographically secure?

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

Patterns across **all** passwords



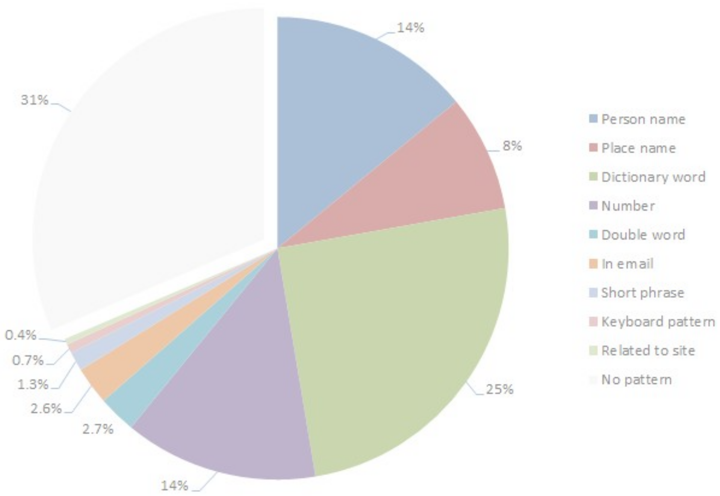
Passwords derived from **people names**



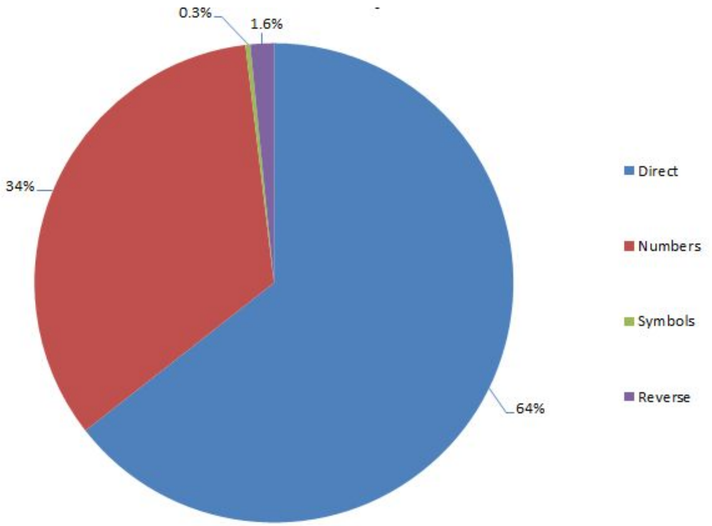
Why aren’t passwords cryptographically secure?

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

Patterns across **all** passwords



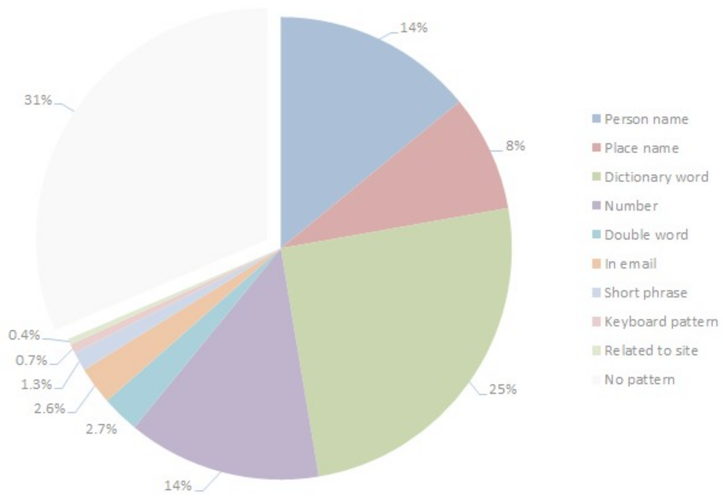
Passwords derived from **location names**



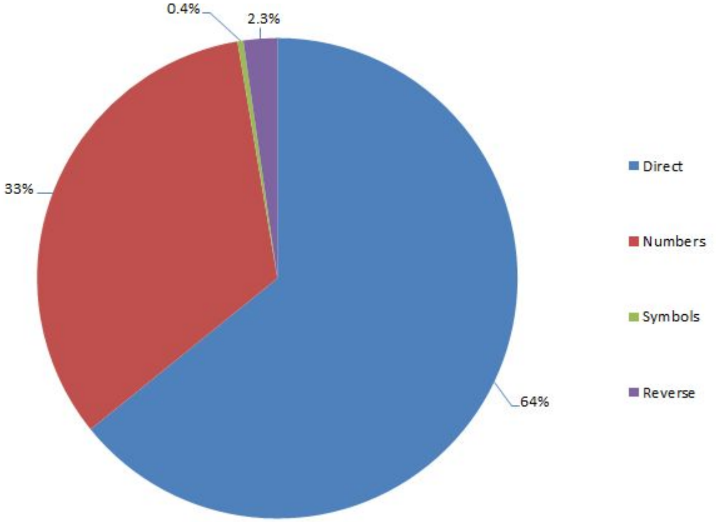
Why aren't passwords cryptographically secure?

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

Patterns across **all** passwords



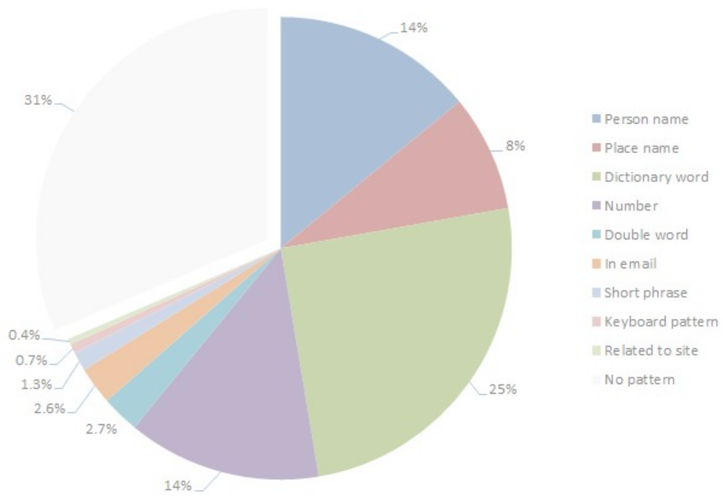
Passwords derived from **dictionary words**



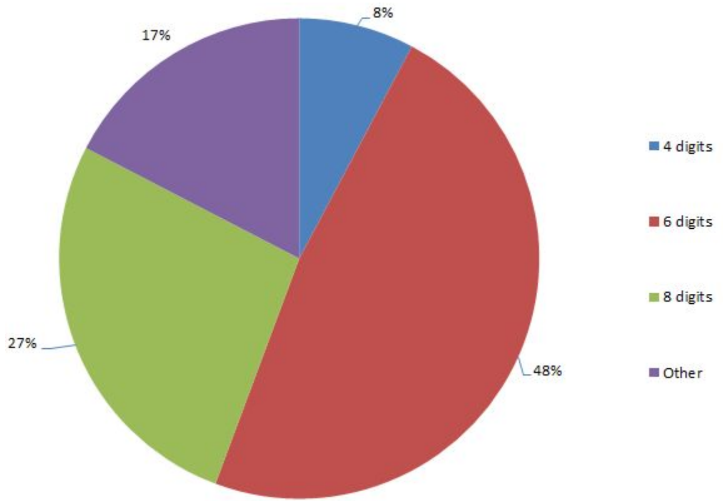
Why aren't passwords cryptographically secure?

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

Patterns across **all** passwords



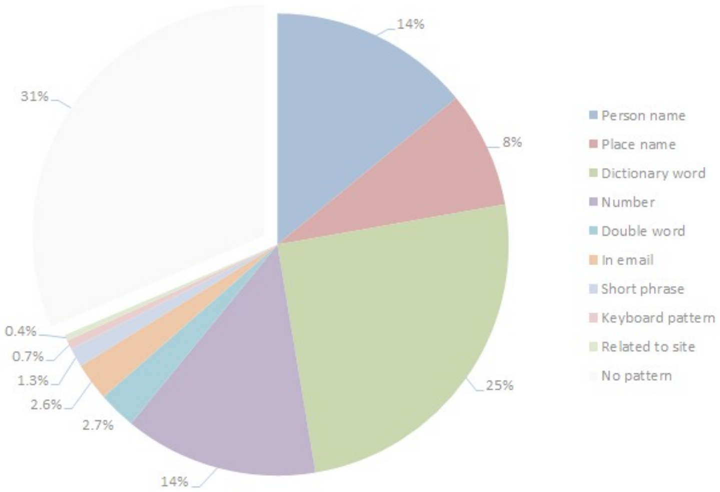
Passwords derived from **numbers**



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

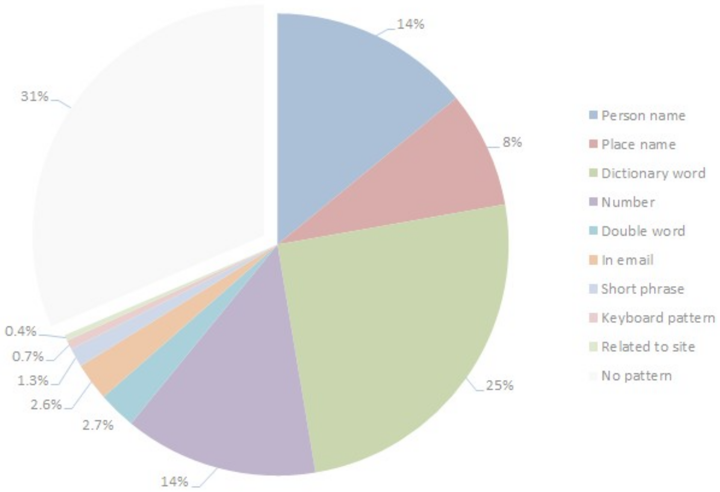
asdfgh

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**

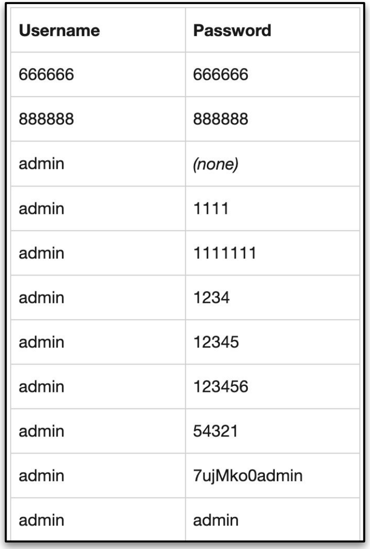
thx1138

gundam

ncc1701

Attack: Guessing Passwords

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

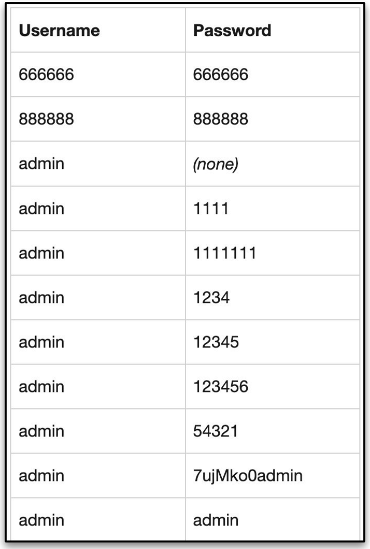


Username	Password
666666	666666
888888	888888
admin	*(none)*
admin	1111
admin	1111111
admin	1234
admin	12345
admin	123456
admin	54321
admin	7ujMko0admin
admin	admin

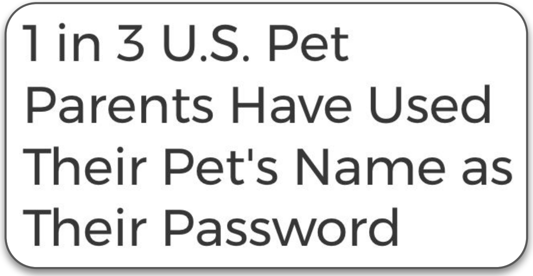
Attack: Guessing Passwords

- 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`



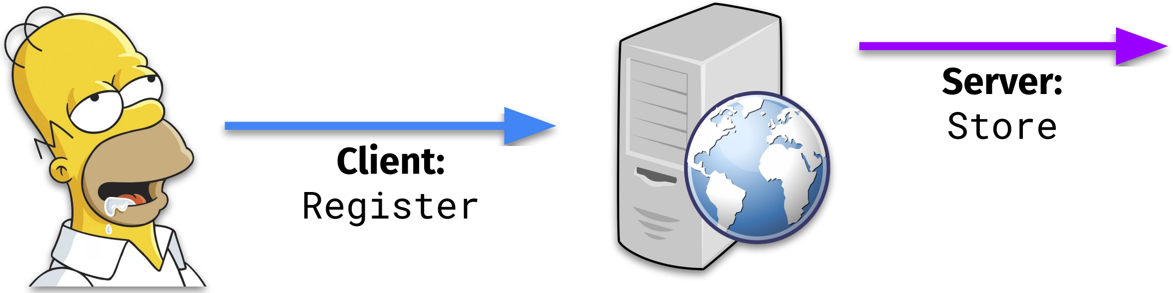
Username	**Password**
666666	666666
888888	888888
admin	*(none)*
admin	1111
admin	1111111
admin	1234
admin	12345
admin	123456
admin	54321
admin	7ujMko0admin
admin	admin



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**



Password Database

user	password
bart	cowabunga
marge	p4$$w0rd
homer	**donuts**

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

Server-side Password Storage

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

Password Database

user	password
bart	cowabunga
marge	p4$$w0rd
homer	**donuts**

Client:
Register

Server:
Store

Attacker:
Login

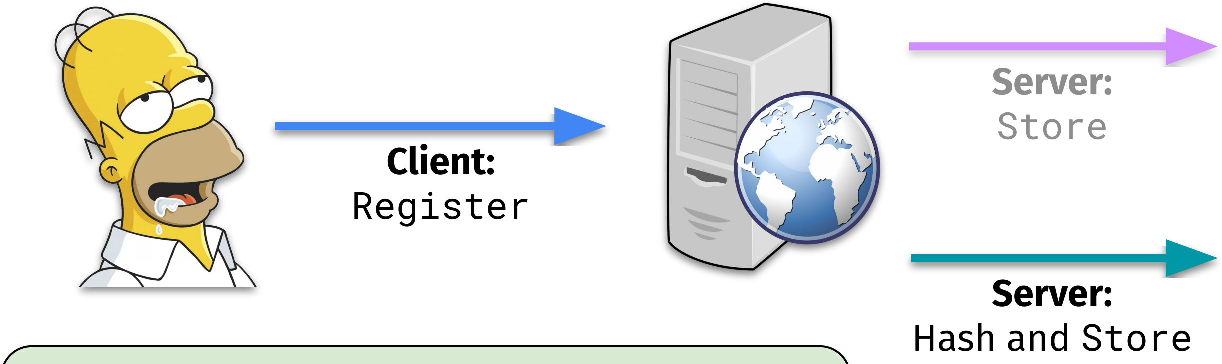
Login(homer, donuts)

→ SUCCESS

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**



Password Database

user	password
bart	cowabunga
marge	p4$$w0rd
homer	**donuts**

Hashed Password Database

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

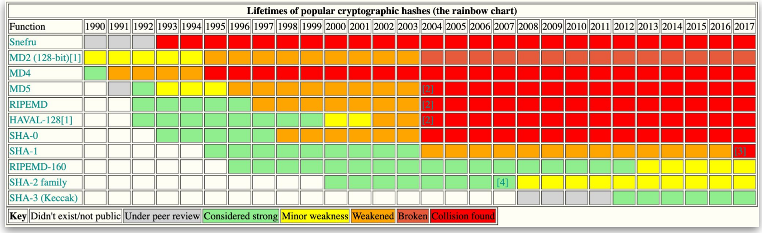
If database **breached**, attacker has **zero** **plaintext passwords**!

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?
  - **???**



Lifetimes of popular cryptographic hashes (the rainbow chart)

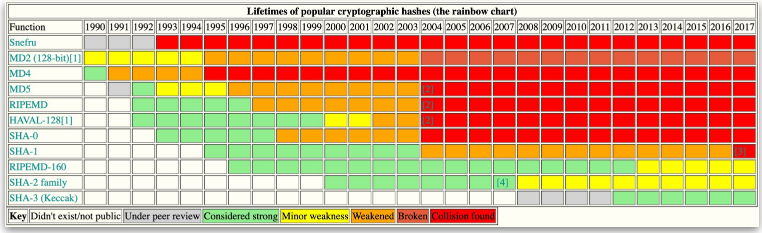
Function	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
Snefru	▨	▨	▨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD2 (128-bit)[1]	🟨	🟨	🟨	🟨	🟨	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫
MD4	🟩	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD5	⬜	▨	🟩	🟨	🟨	🟨	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥 [2]	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
RIPEMD	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥 [2]	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
HAVAL-128[1]	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟧	🟧	🟥 [2]	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
SHA-0	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
SHA-1	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥 [3]
RIPEMD-160	⬜	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟨	🟨	🟨
SHA-2 family	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩 [4]	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨
SHA-3 (Keccak)	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	▨	▨	▨	▨	🟩	🟩	🟩	🟩	🟩	🟩

Key ⬜ Didn't exist/not public ▨ Under peer review 🟩 Considered strong 🟨 Minor weakness 🟧 Weakened 🟫 Broken 🟥 Collision found

Attacking Stored Passwords

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

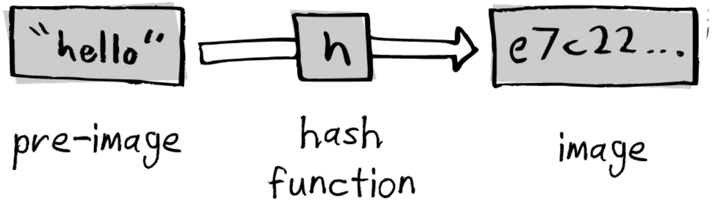
- What if a **weak** hash function is used?
  - **Pre-image attacks:** find the original string
  - **Collision attacks:** find a different string that produces same hash as password



Lifetimes of popular cryptographic hashes (the rainbow chart)

Function	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
Snefru	▨	▨	▨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD2 (128-bit)[1]	🟨	🟨	🟨	🟨	🟨	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫
MD4	🟩	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD5	⬜	▨	🟩	🟨	🟨	🟨	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥 [2]	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
RIPEMD	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥 [2]	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
HAVAL-128[1]	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟧	🟧	🟥 [2]	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
SHA-0	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
SHA-1	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥 [3]
RIPEMD-160	⬜	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟨	🟨	🟨
SHA-2 family	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩 [4]	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨
SHA-3 (Keccak)	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	▨	▨	▨	▨	🟩	🟩	🟩	🟩	🟩	🟩

Key	⬜ Didn't exist/not public	▨ Under peer review	🟩 Considered strong	🟨 Minor weakness	🟧 Weakened	🟫 Broken	🟥 Collision found

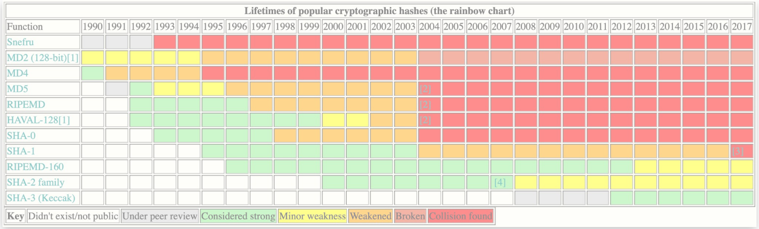


Attacking Stored Passwords

- **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

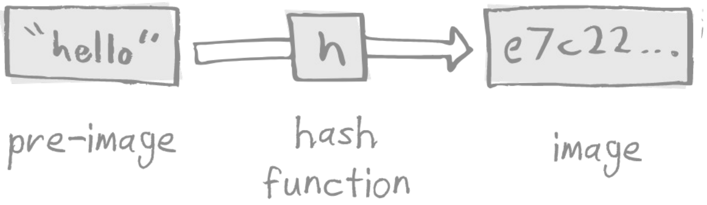
- What if a **fast** hash function is used?



Lifetimes of popular cryptographic hashes (the rainbow chart)

Function	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
Snefru	▨	▨	▨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD2 (128-bit)[1]	🟨	🟨	🟨	🟨	🟨	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫	🟫
MD4	🟩	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
MD5	⬜	▨	🟩	🟨	🟨	🟨	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥 [2]	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
RIPEMD	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥 [2]	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
HAVAL-128[1]	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟧	🟧	🟥 [2]	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
SHA-0	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
SHA-1	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟥 [3]
RIPEMD-160	⬜	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟨	🟨	🟨
SHA-2 family	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩 [4]	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨
SHA-3 (Keccak)	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	⬜	▨	▨	▨	▨	🟩	🟩	🟩	🟩	🟩	🟩

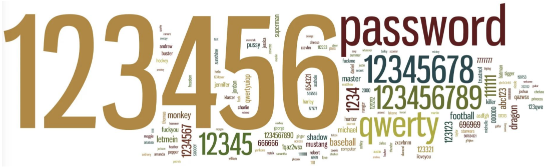
Key	⬜ Didn't exist/not public	▨ Under peer review	🟩 Considered strong	🟨 Minor weakness	🟧 Weakened	🟫 Broken	🟥 Collision found



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

SHA256(string)

Attacker's table of pw hash pairs

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

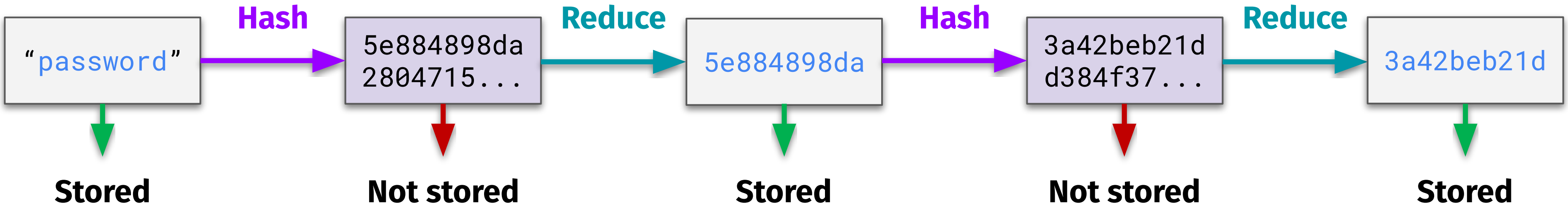
Attack: Rainbow Tables

- 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

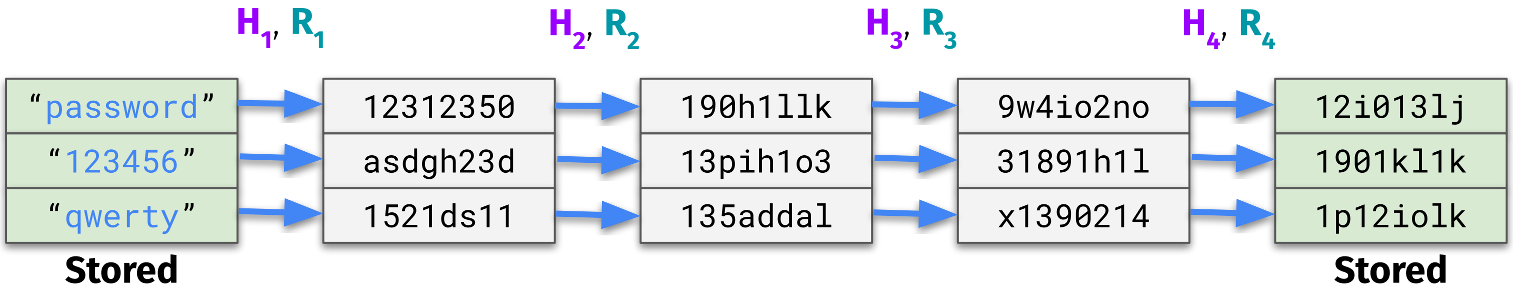
- Similar to a **lookup table**—attacker can trade-off **disk space** vs. **CPU time**
  - Attacker wants something that uses **less time, less storage** than a **brute-force attack**

- **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



"password"	12312350	190h1llk	9w4io2no	12i013lj
"123456"	asdgh23d	13pih1o3	31891h1l	1901kl1k
"qwerty"	1521ds11	135addal	x1390214	1p12iolk
Stored				**Stored**

Example: Find password for hash 135addal

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

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

Walk the chain from its start (qwerty)

= qwerty -> $\boxed{\text{1521ds11}}$ -> 135addal

Found original string!

Better Password Generation

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

I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$W
I SHALL USE STRONG P@$$
I SHALL USE STRONG P@$
I SHALL USE STRONG P@$
I SHALL USE STRONG P@

Sorry, but your password must contain an uppercase letter, a number, a hieroglyph, a feather from a hawk and the blood of a unicorn.

Better Password Generation

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

I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.
I SHALL USE STRONG P@$$WORDS.

Sorry, but your password must contain an uppercase letter, a number, a hieroglyph, a feather from a hawk and the blood of a unicorn.
someecards
user card

Better Server-side Password Storage

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

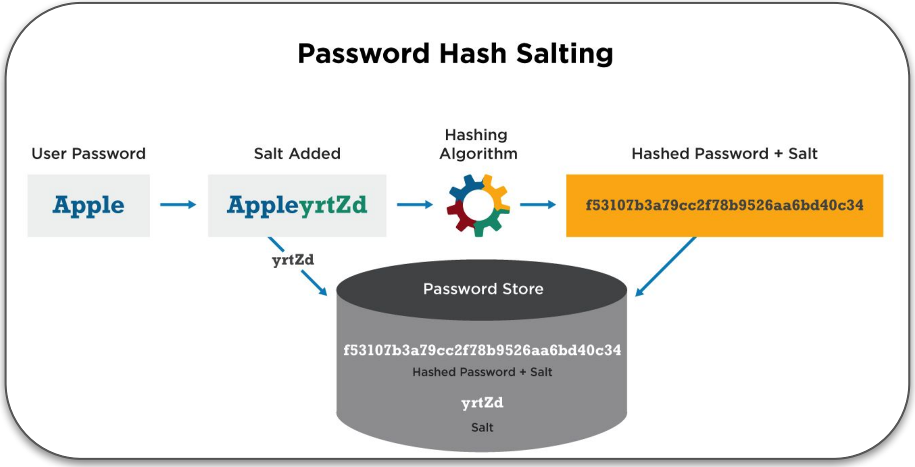
Better Server-side Password Storage

- 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**)

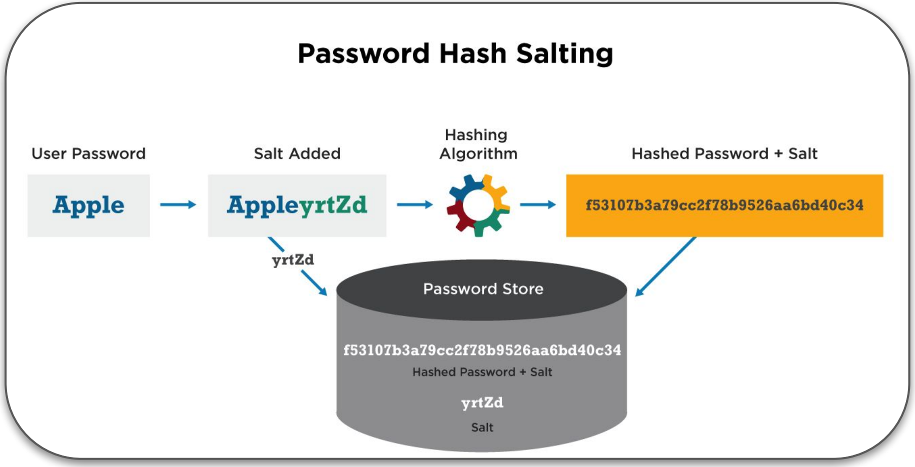
- **Salted passwords:**
  - Add **extra data** when generating hash
  - **Goal:** same input = different output



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

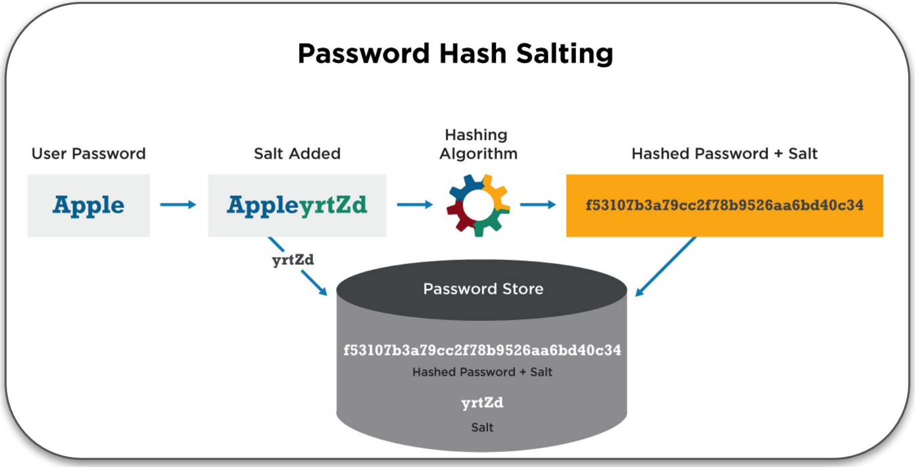


Better Server-side Password Storage

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

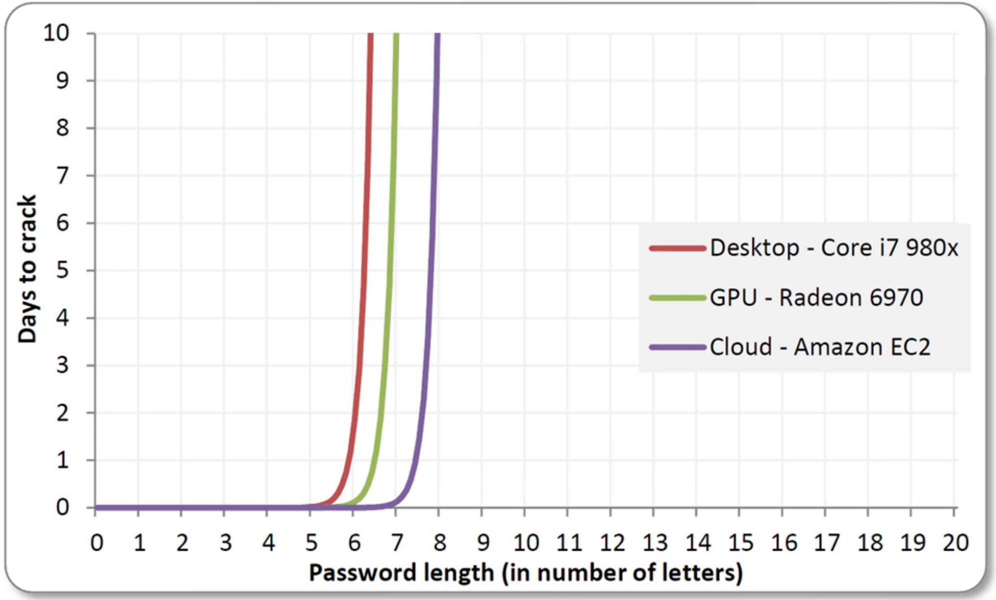
- **Salted passwords:**
  - Add **extra data** when generating hash
  - **Goal:** same input = different output
  - Salting considerations:
    - Salt should **not be short**
    - Should be **unique** per user

- **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!

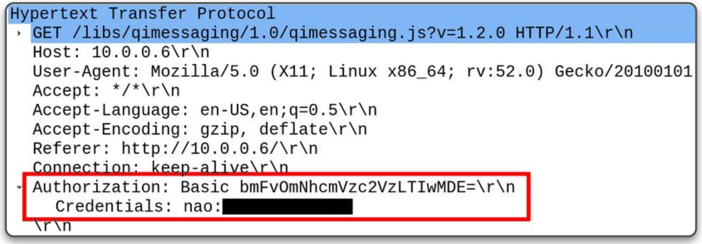


Attack: Client-side Password Theft

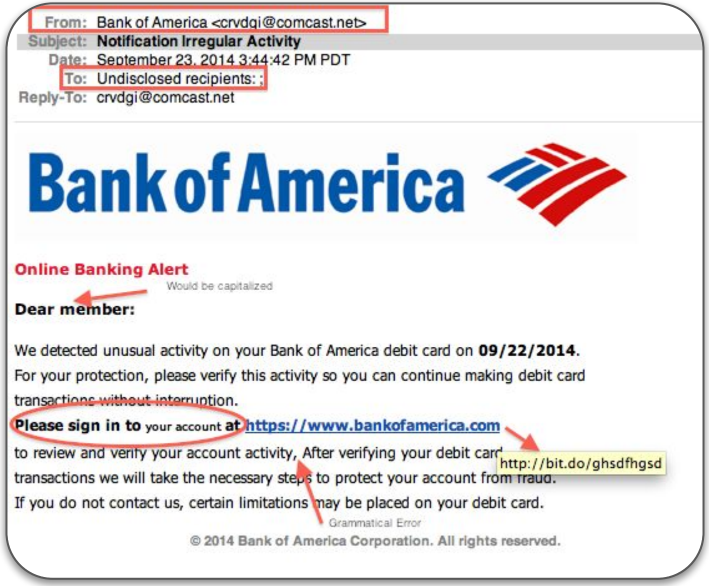
- **How?**

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 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.

http://bit.do/ghsdfhgsd

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…

Tor: The Onion Router
Project 4 Tips