CSEP 564: Computer Security and Privacy

Web Security

Authentication [start]

Fall 2022

David Kohlbrenner

dkohlbre@cs

Thanks to Franzi Roesner, Dan Boneh, Dieter Gollmann, Dan Halperin, David Kohlbrenner, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Logistics

- Lab 2 is out
  - Signup form gets you access to the webpage
  - You only need to do *some* of the problems!
  - You'll need/want to host some PHP on your UW homedir

- Lab 1 grades will be out soon
  - If there are scores that don't make sense to you, let us know
  - EC problems are worth 5pts, assigned ones are 10pts

Paper Discussion Time!

SOP
Same Origin Policy

"Pixel Perfect Timing Attacks with HTML5"

Paul Stone

Canvas fingerprinting

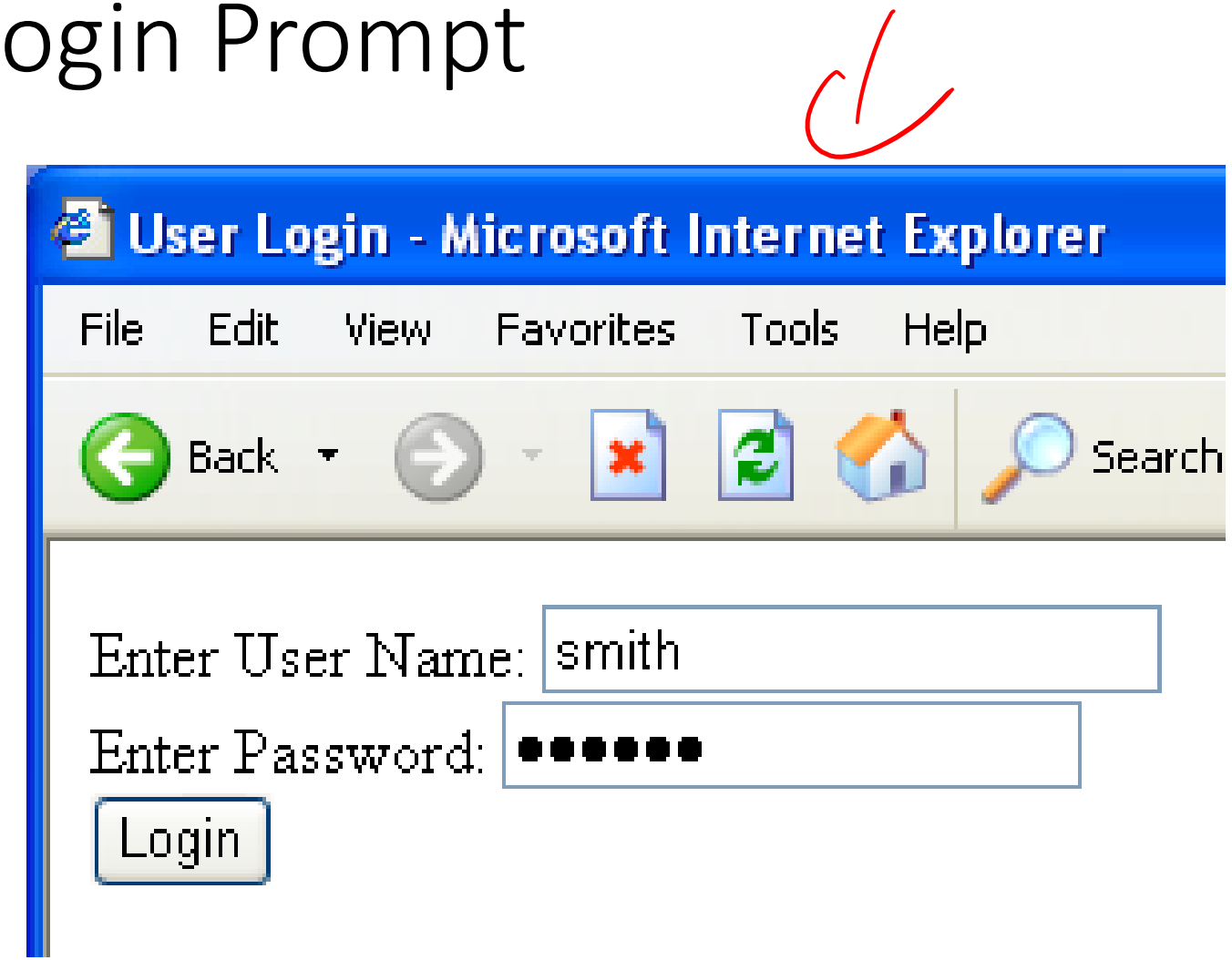
- Choose one/more and discuss with neighbors:
  - What is browser history sniffing?
  - Does timing link painting violate the SOP?
  - What are CSS/SVG filters?
  - What is the root cause of timing variation in the filters?
  - Is applying a filter to an iframe a violation of the SOP?
  - How did view-source matter for attacks?

re:visited

if
else

SQL Injection

Typical Login Prompt

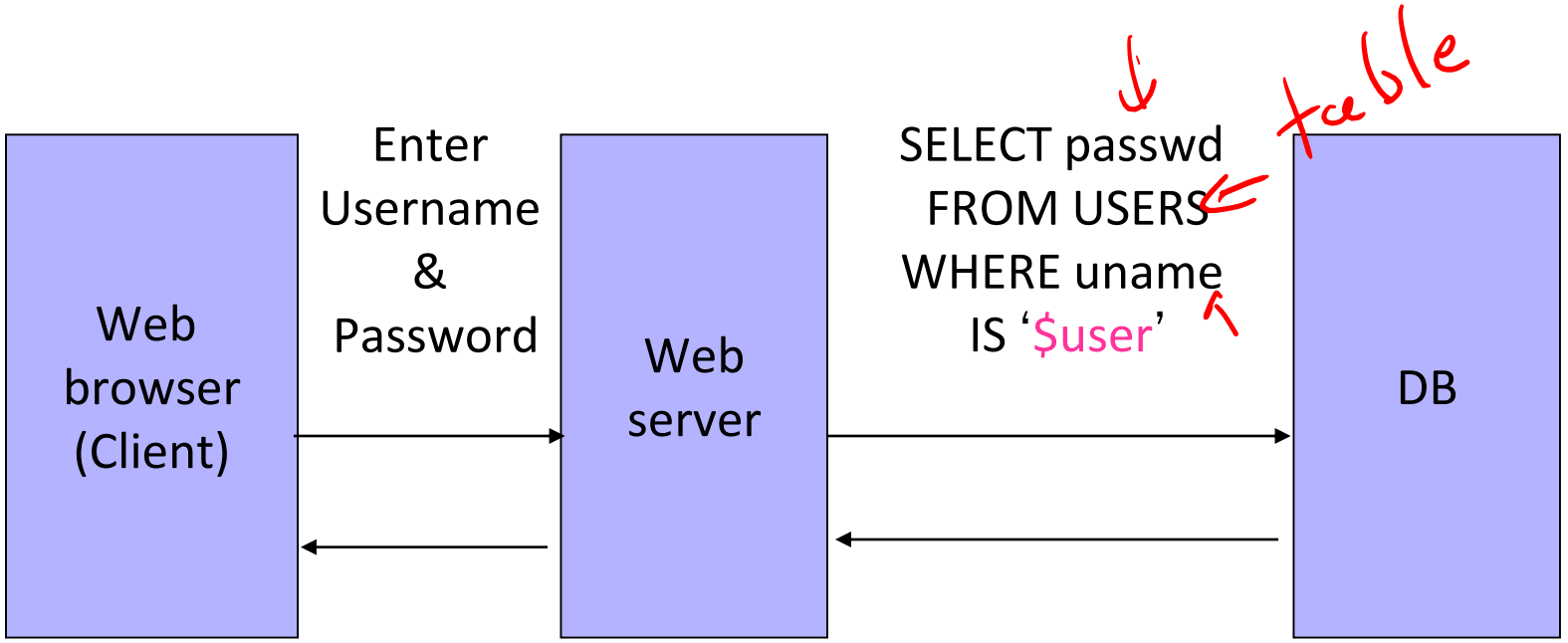


Typical Query Generation Code

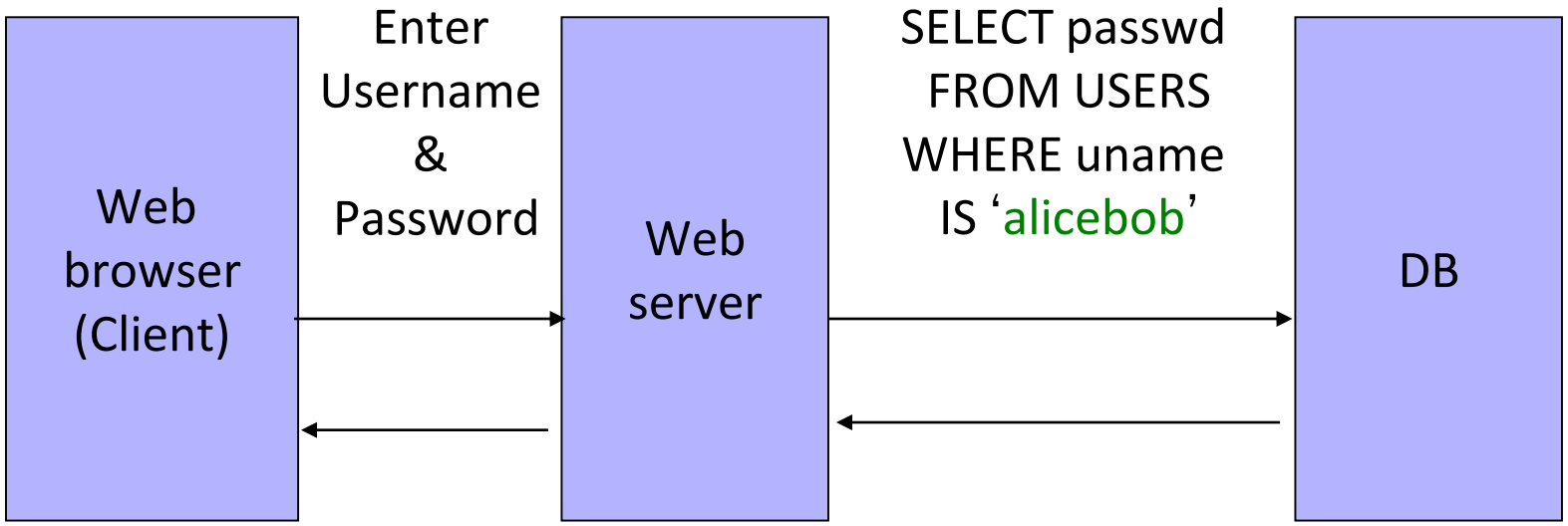
```
$selecteduser = $_GET['user'];
$sql = "SELECT Username, Key FROM Key " .
       "WHERE Username='$selecteduser'";
$rs = $db->executeQuery($sql);
```

What if **'user'** is a malicious string that changes the meaning of the query?

User Input Becomes Part of Query

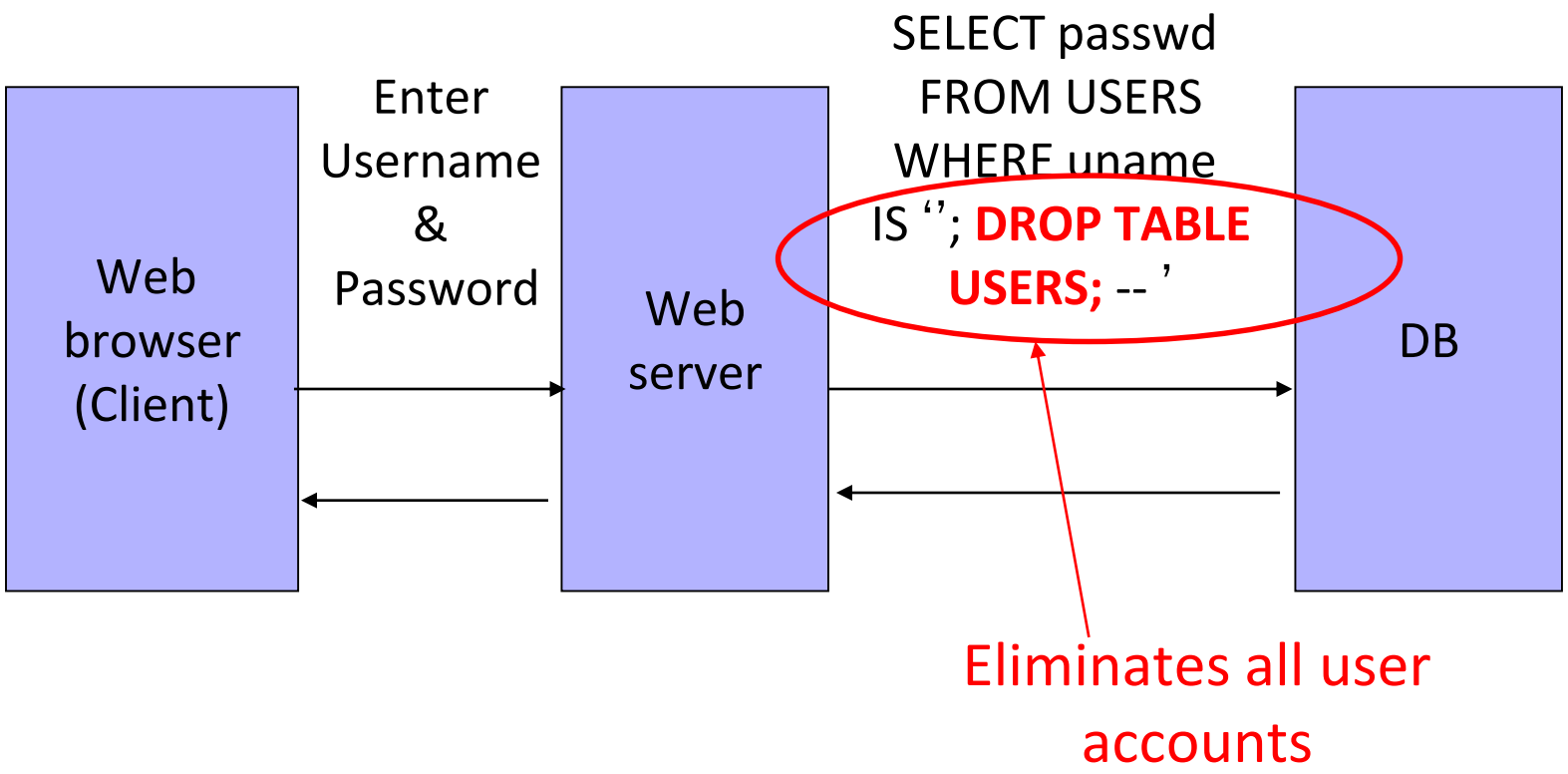


Normal Login

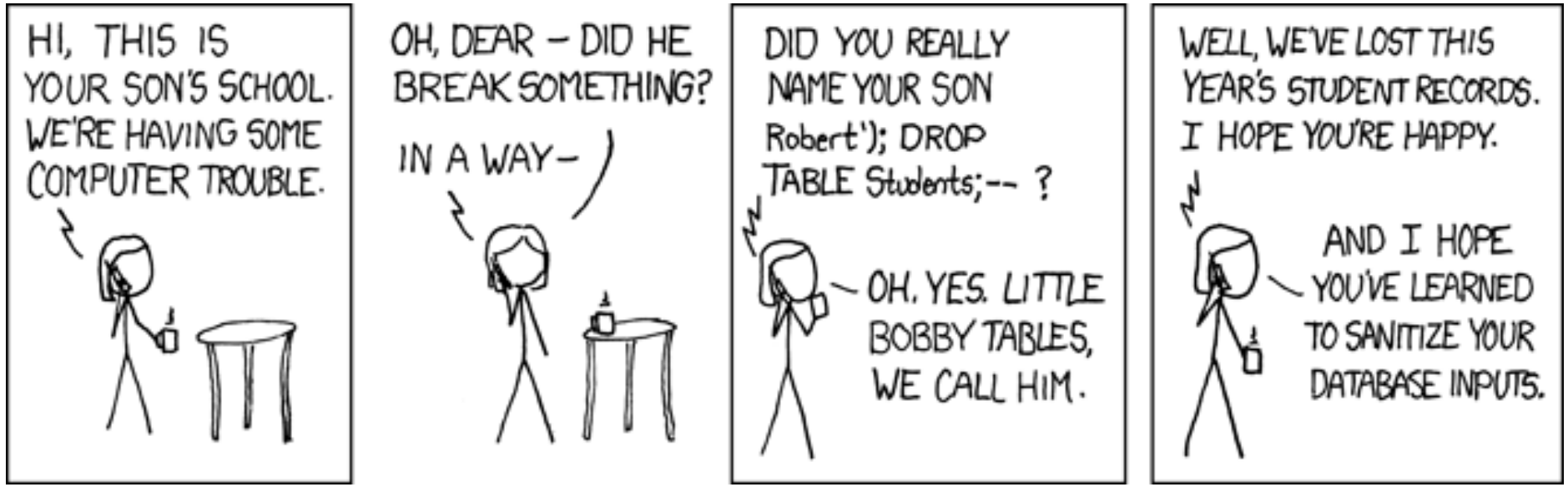


Malicious User Input

SQL Injection Attack

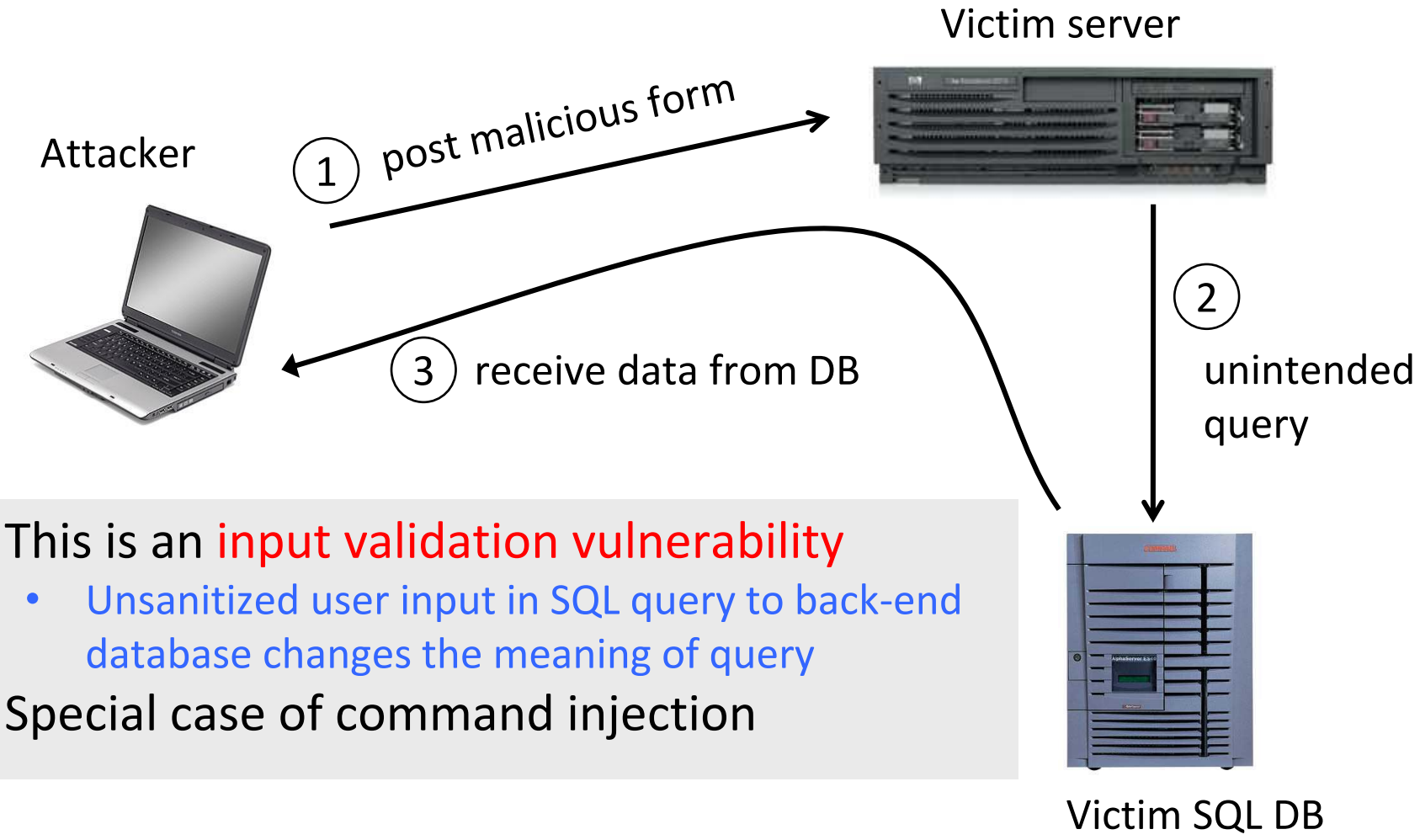


XKCD



http://xkcd.com/327/

SQL Injection: Basic Idea

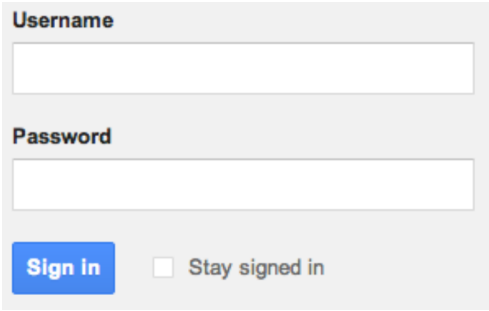


- This is an input validation vulnerability
  - Unsanitized user input in SQL query to back-end database changes the meaning of query
- Special case of command injection

Authentication with Backend DB

(*) remember to hash passwords for real authentication scheme

```
set UserFound = execute(
    "SELECT * FROM UserTable WHERE
    username= ' " & form("user") & " ' AND
    password= ' " & form("pwd") & " ' ");
```



User supplies username and password, this SQL query checks if user/password combination is in the database

```
If not UserFound.EOF
    Authentication correct
else Fail
```

Only true if the result of SQL query is not empty, i.e., user/pwd is in the database

Using SQL Injection to Log In

- User gives username **’ OR 1=1 --**
- Web server executes query

```
set UserFound=execute(
    SELECT * FROM UserTable WHERE
    username= ‘ ’ OR 1=1 -- … );
```

Always true!

Everything after -- is ignored!

- Now all records match the query, so the result is not empty ⇒ correct “authentication”!

“Blind SQL Injection” https://owasp.org/www-community/attacks/Blind_SQL_Injection

CERT

- SQL injection attack where attacker asks database series of true or false questions
- Used when
  - the database does not output data to the web page
  - the web shows generic error messages, but has not mitigated the code that is vulnerable to SQL injection.
- SQL Injection vulnerability more difficult to exploit, but not impossible.

Sqlmap

Preventing SQL Injection

- Validate all inputs
  - Filter out any character that has special meaning
    - Apostrophes, semicolons, percent, hyphens, underscores, …
    - Use escape characters to prevent special characters form becoming part of the query code
      - E.g.: escape(O'Connor) = O\'Connor
  - Check the data type (e.g., input must be an integer)
- Same issue as with XSS: is there anything accidentally not checked / escaped?

Prepared Statements

```
PreparedStatement ps =
    db.prepareStatement("SELECT pizza, toppings, quantity, order_day "
                + "FROM orders WHERE userid=? AND order_month=?");
ps.setInt(1, session.getCurrentUserId());
ps.setInt(2, Integer.parseInt(request.getParamenter("month")));
ResultSet res = ps.executeQuery();
```

- Bind variables: placeholders guaranteed to be data (not code)
- Query is parsed without data parameters
- Bind variables are typed (int, string, …)

http://java.sun.com/docs/books/tutorial/jdbc/basics/prepared.html

Wait, why not do that for XSS?

- “Prepared statements for HTML”?

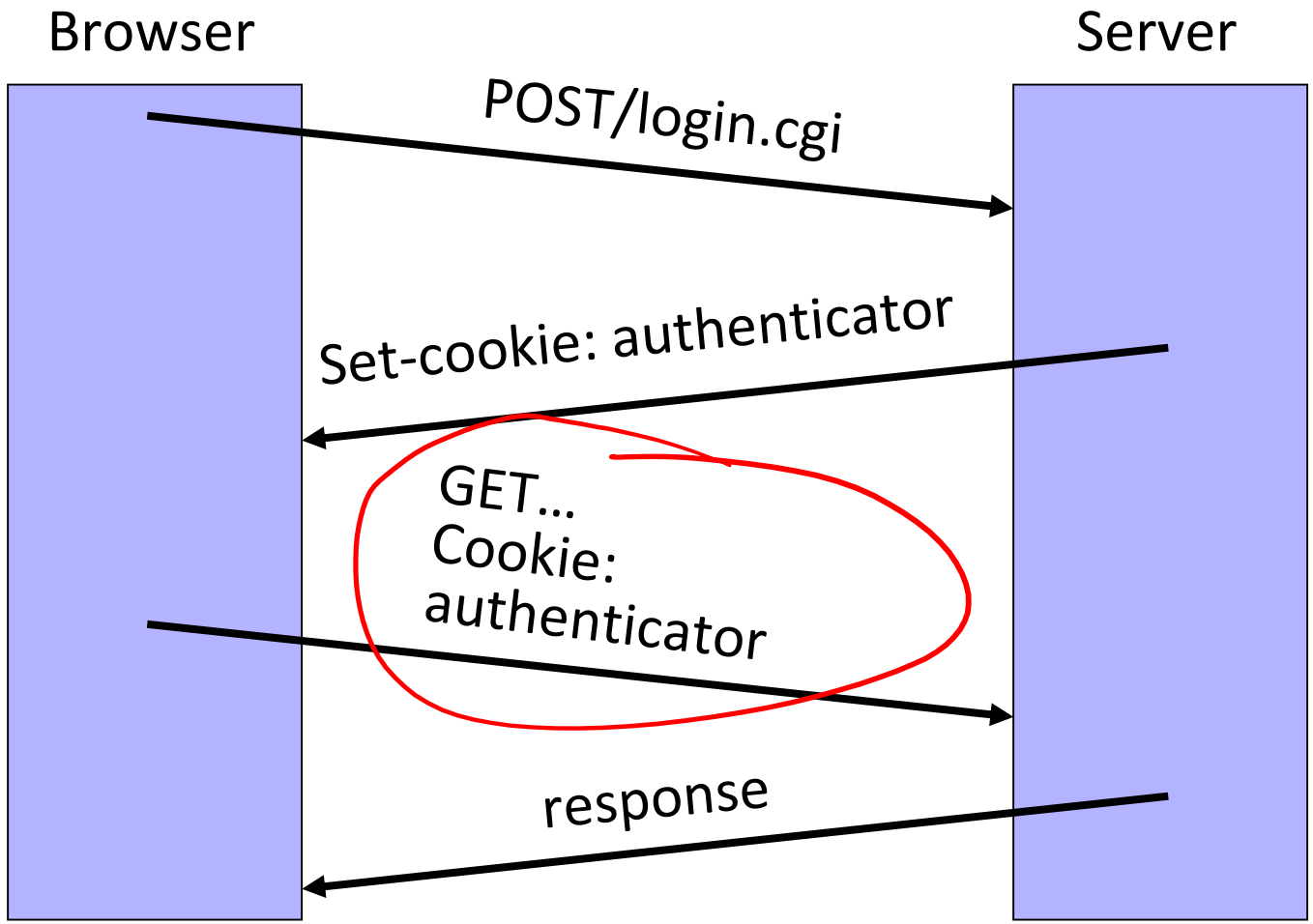
Data-as-code

- XSS
- SQL Injection
- (Like buffer overflows)

7:35

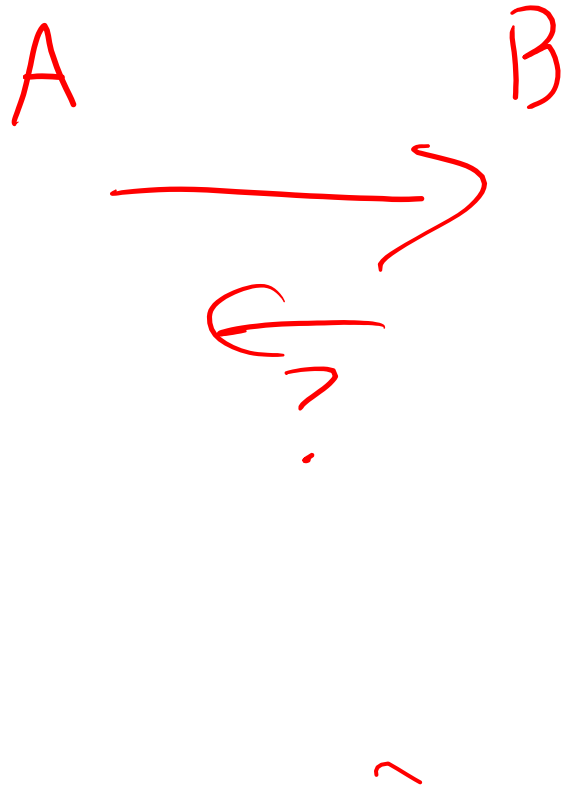
Cross-Site Request Forgery (CSRF/XSRF)

Cookie-Based Authentication Review



Browser Sandbox Review

- Based on the same origin policy (SOP)
- Active content (scripts) can send anywhere!
  - For example, can submit a POST request
  - Some ports inaccessible -- e.g., SMTP (email)
- Can only *read* response from the *same origin*
  - … but you can do a lot with just sending!



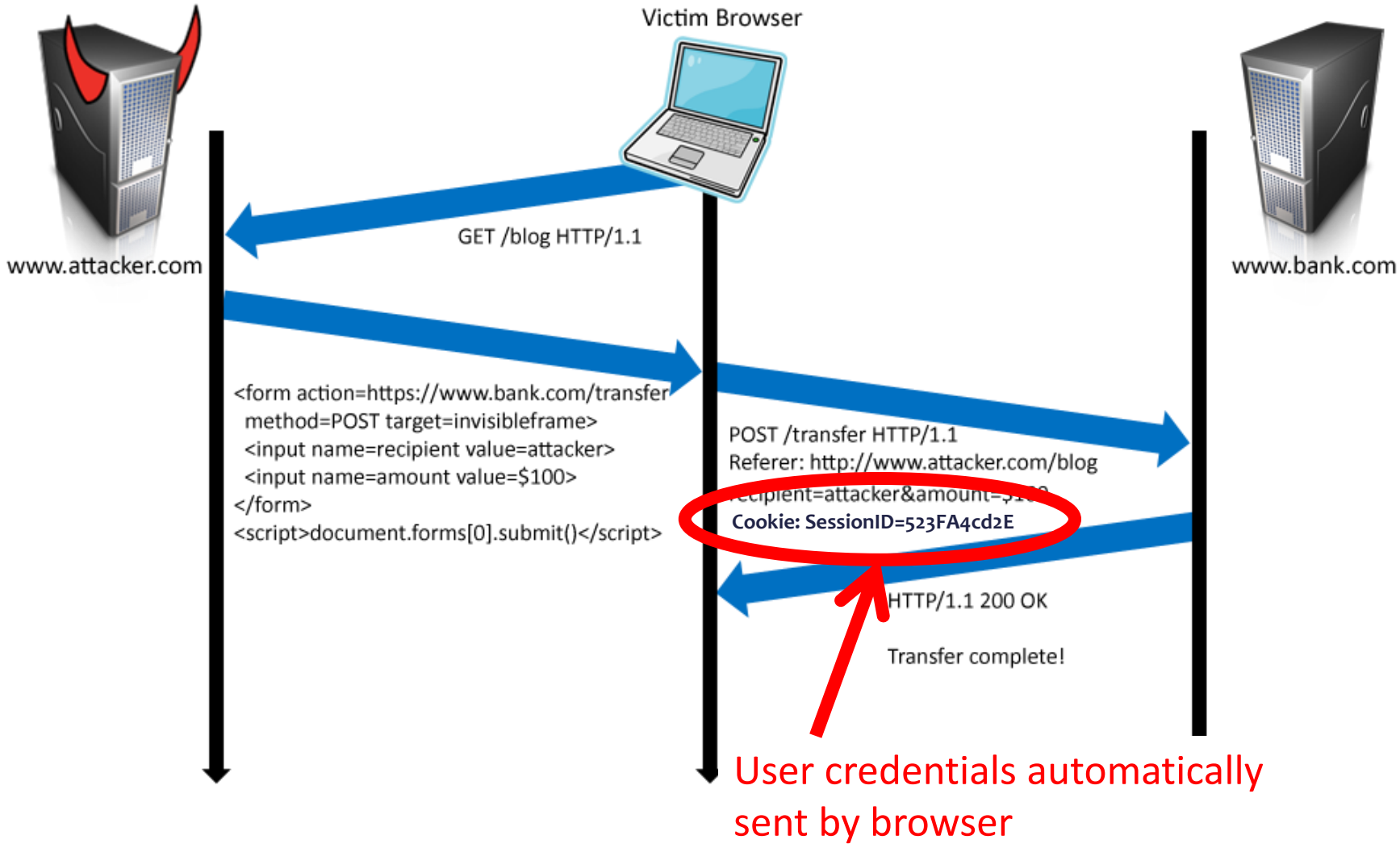
Cross-Site Request Forgery

- Users logs into bank.com, forgets to sign off
  - Session cookie remains in browser state
- User then visits a malicious website containing

```
<form  name=BillPayForm
action=http://bank.com/BillPay.php>
<input  name=recipient  value=attacker> …
<script> document.BillPayForm.submit(); </script>
```

- Browser sends cookie, payment request fulfilled!
- Lesson: cookie authentication is not sufficient when side effects can happen

Cookies in Forged Requests

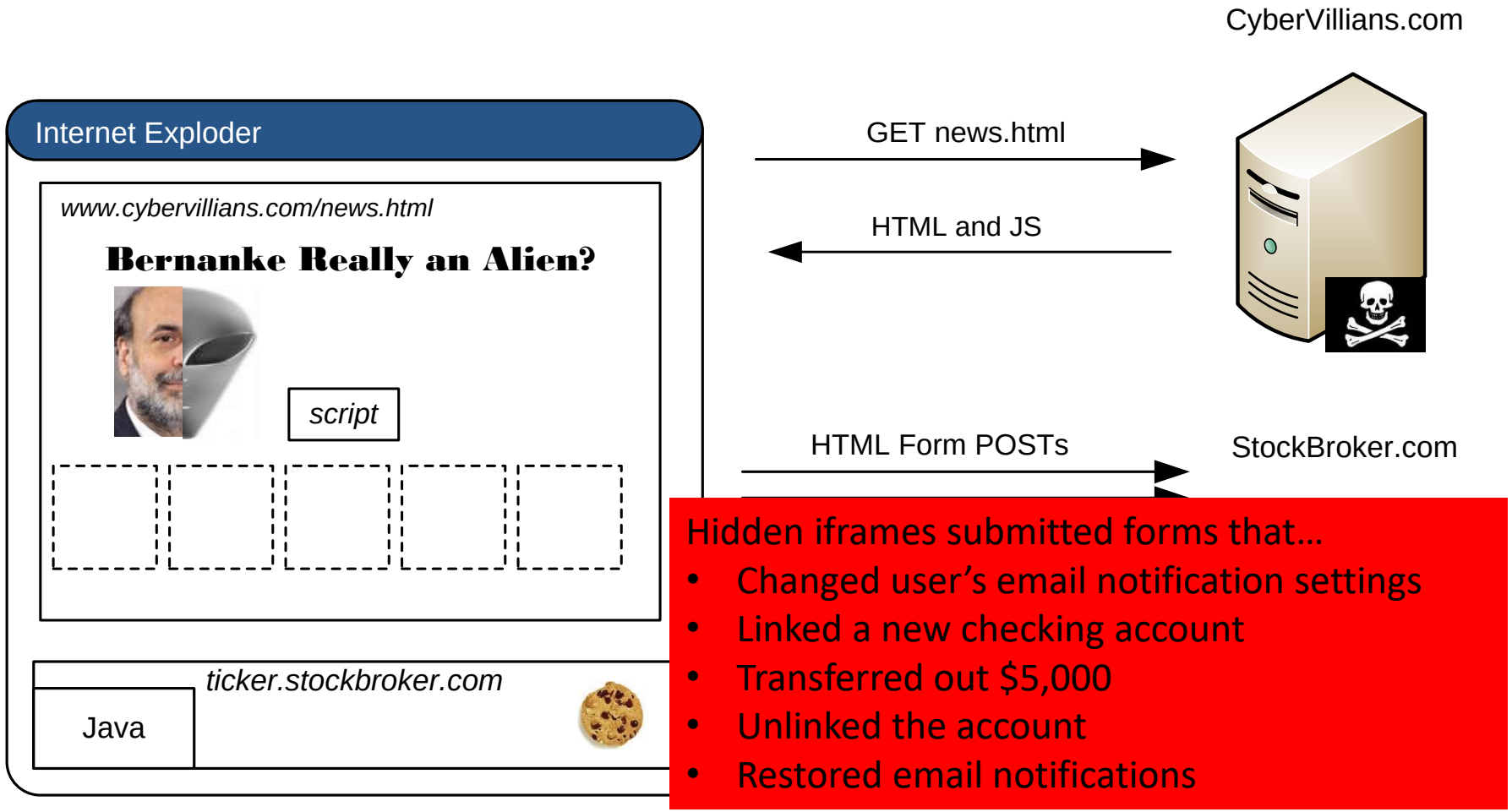


Impact

- Hijack any ongoing session (if no protection)
  - Netflix: change account settings, Gmail: steal contacts, Amazon: one-click purchase
- Reprogram the user's home router
- Login to the *attacker's* account
  - Why?

XSRF True Story

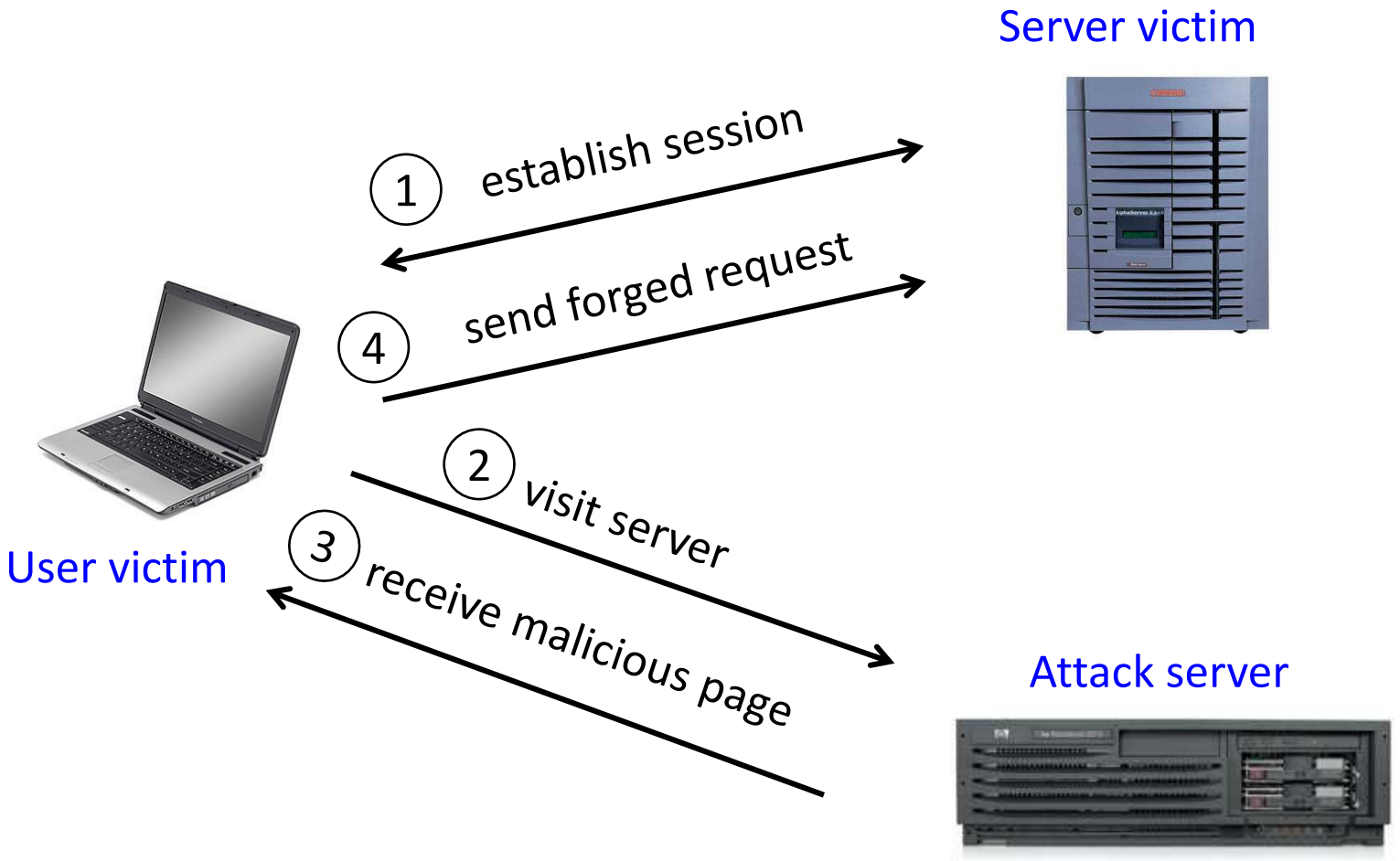
[Alex Stamos]



Hidden iframes submitted forms that…

- Changed user's email notification settings
- Linked a new checking account
- Transferred out $5,000
- Unlinked the account
- Restored email notifications

XSRF (aka CSRF): Summary

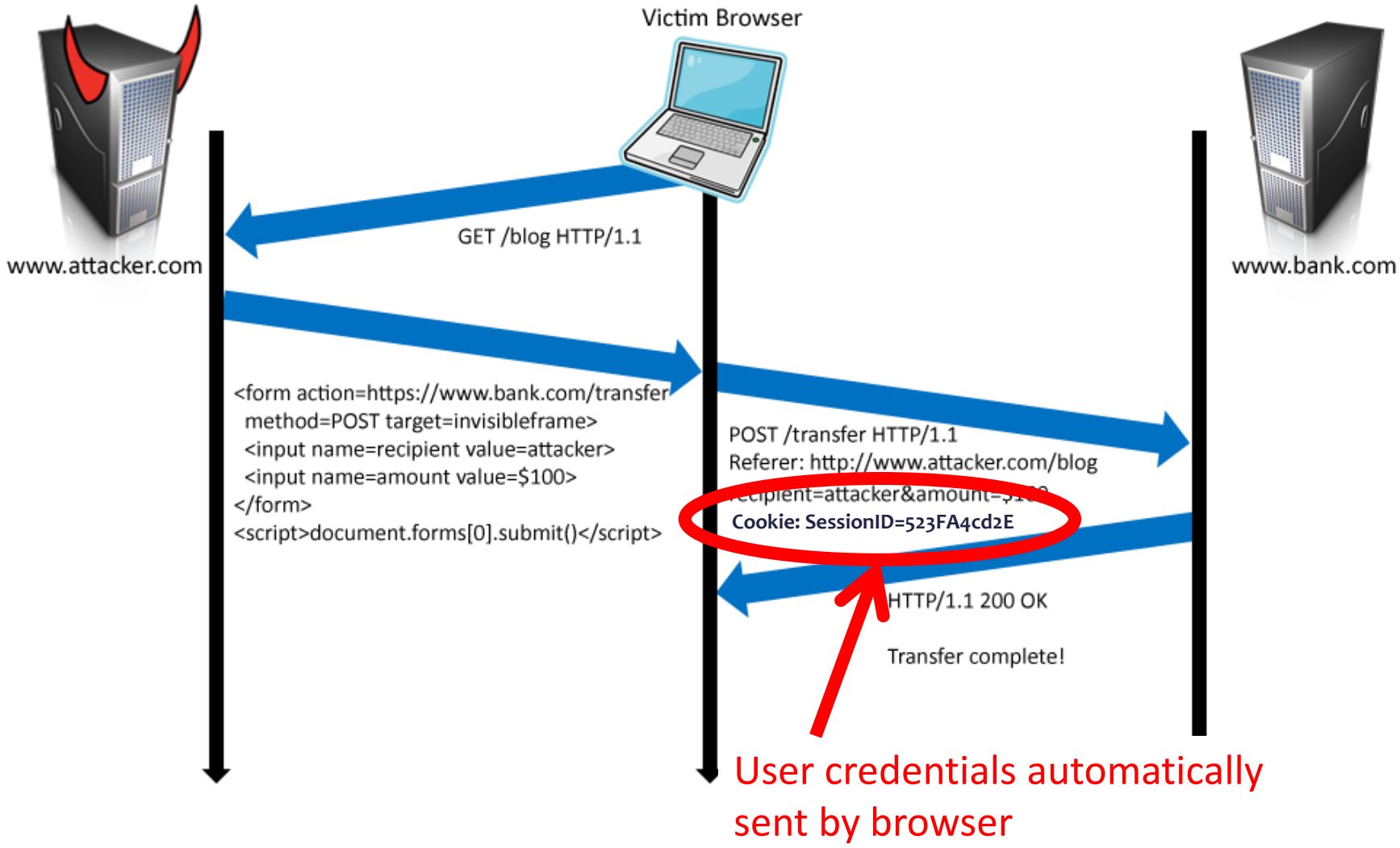


Q: how long do you stay logged on to Gmail? Financial sites?

Broader View of XSRF

- Abuse of cross-site data export
  - SOP does not control data export
  - Malicious webpage can initiates requests from the user's browser to an honest server
  - Server thinks requests are part of the established session between the browser and the server (automatically sends cookies)

How might you protect against XSRF?



XSRF Defenses

- Secret validation token

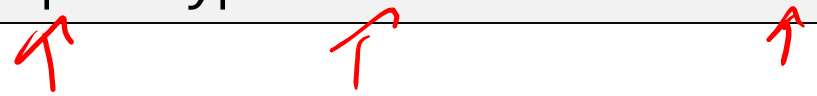
<input type=hidden value=23a3af01b>

- Referer validation

Referer:
http://www.facebook.com/home.php

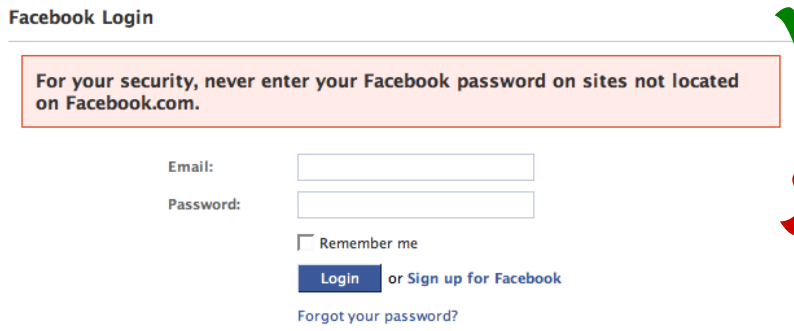
Add Secret Token to Forms

```
<input type=hidden value=23a3af01b>
```



- "Synchronizer Token Pattern"
- Include a secret challenge token as a hidden input in forms
  - Token often based on user's session ID
  - Server must verify correctness of token before executing sensitive operations
- Why does this work?
  - **Same-origin policy:** attacker can't read token out of legitimate forms loaded in user's browser, so can't create fake forms with correct token

Referer Validation



Referer:
http://www.facebook.com/home.php

Referer:
http://www.evil.com/attack.html

Referer:

- Lenient referer checking – header is optional
- Strict referer checking – header is required

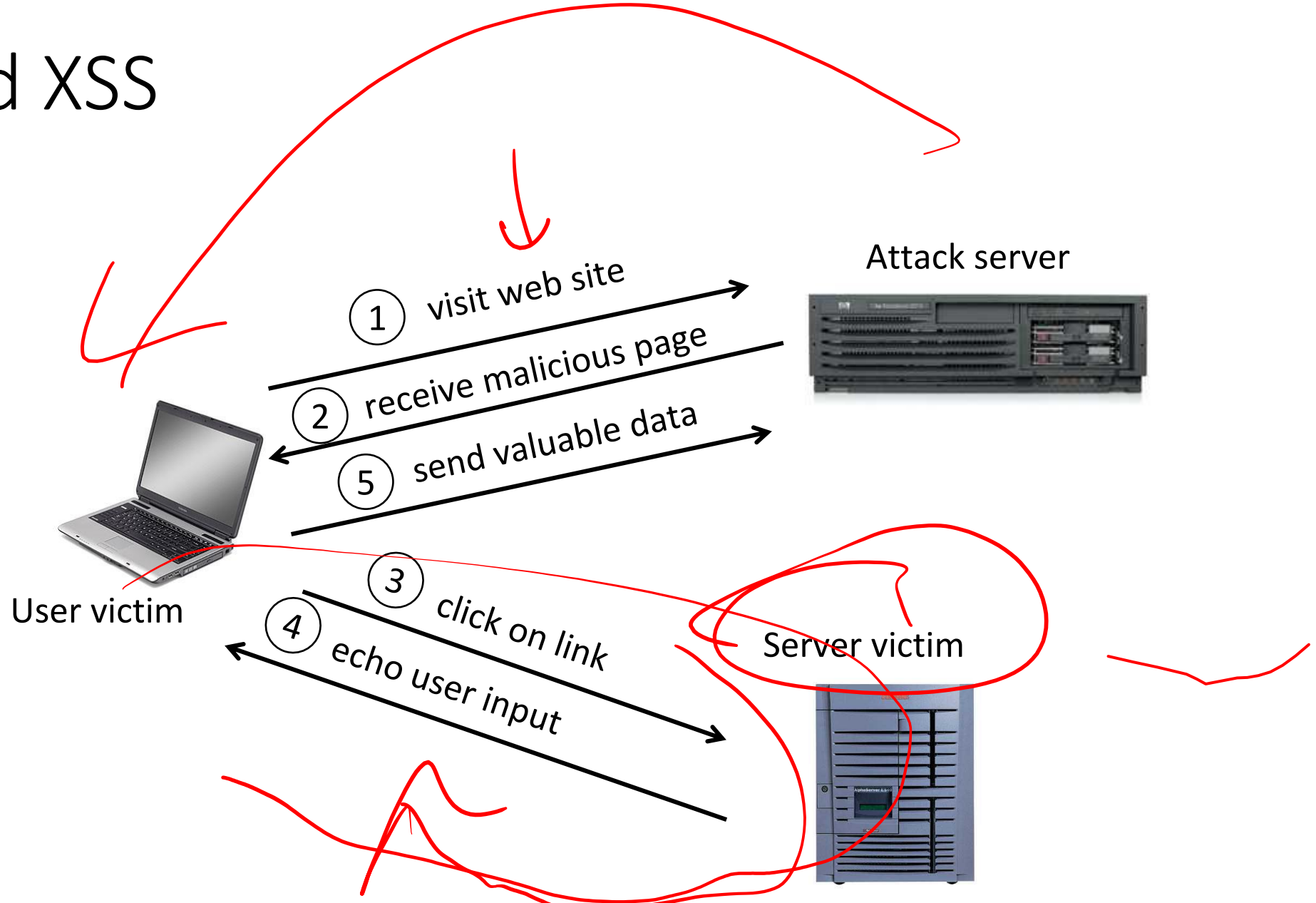
Why Not Always Strict Checking?

- Why might the referer header be suppressed?
  - Stripped by the organization's network filter
  - Stripped by the local machine
  - Stripped by the browser for HTTPS → HTTP transitions
  - User preference in browser
  - Buggy browser
- Web applications can't afford to block these users
- **Many web application frameworks include CSRF defenses today**

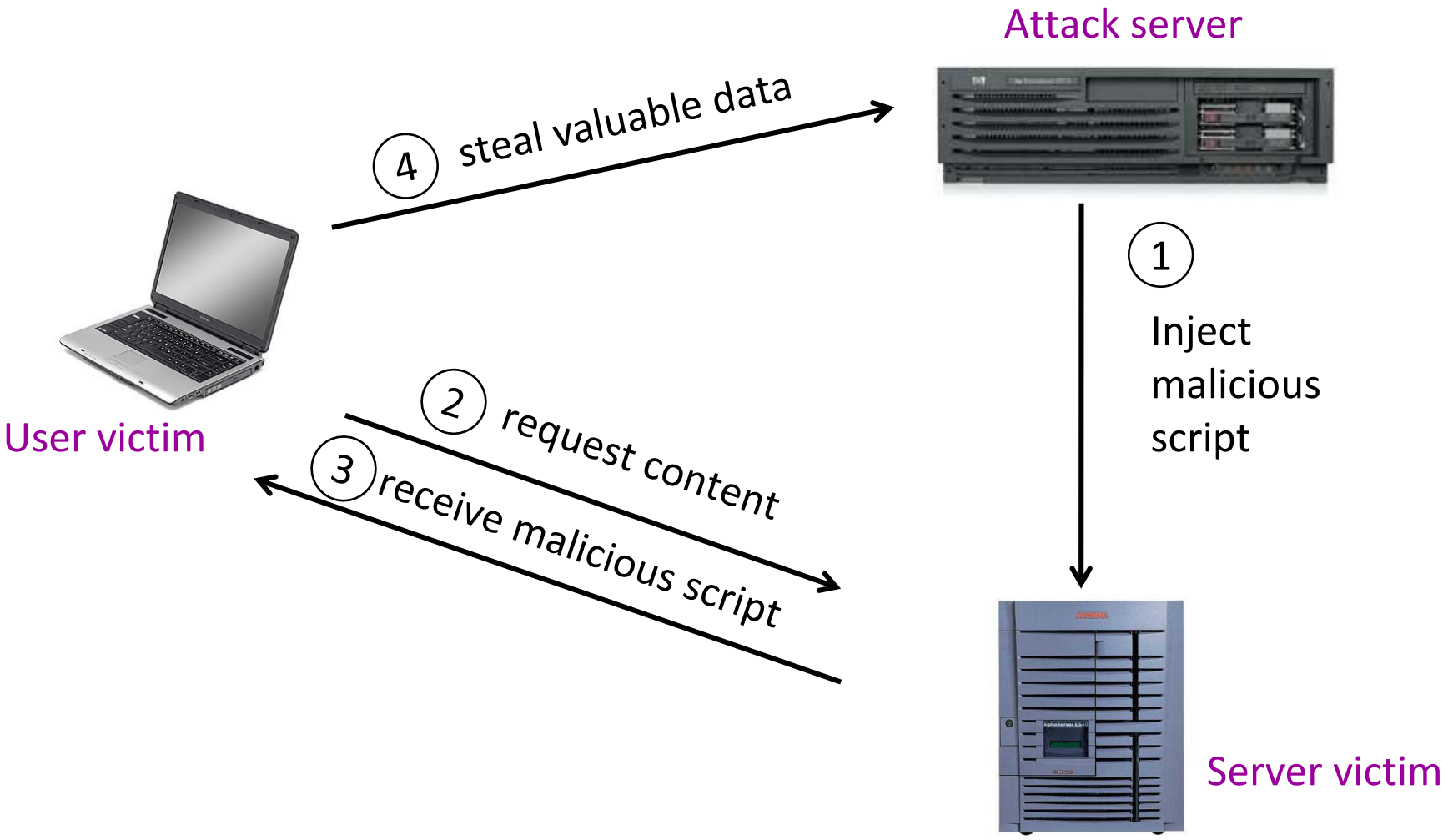
Surprise not-quiz time

XSS again, pollev.com/dkohlbre

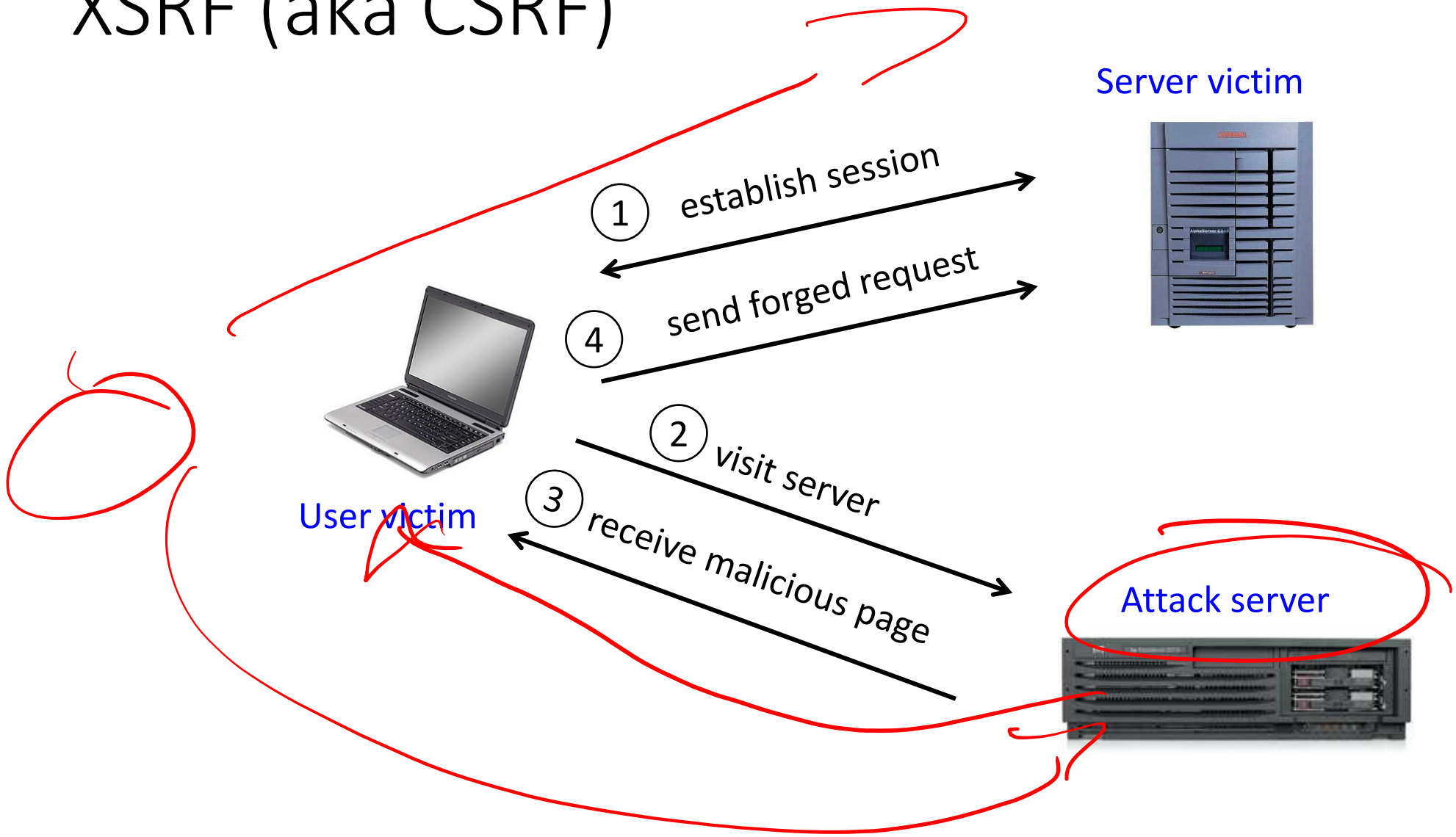
Reflected XSS



Stored XSS


Attack server
4 steal valuable data
1
Inject malicious script
2 request content
3 receive malicious script
User victim
Server victim

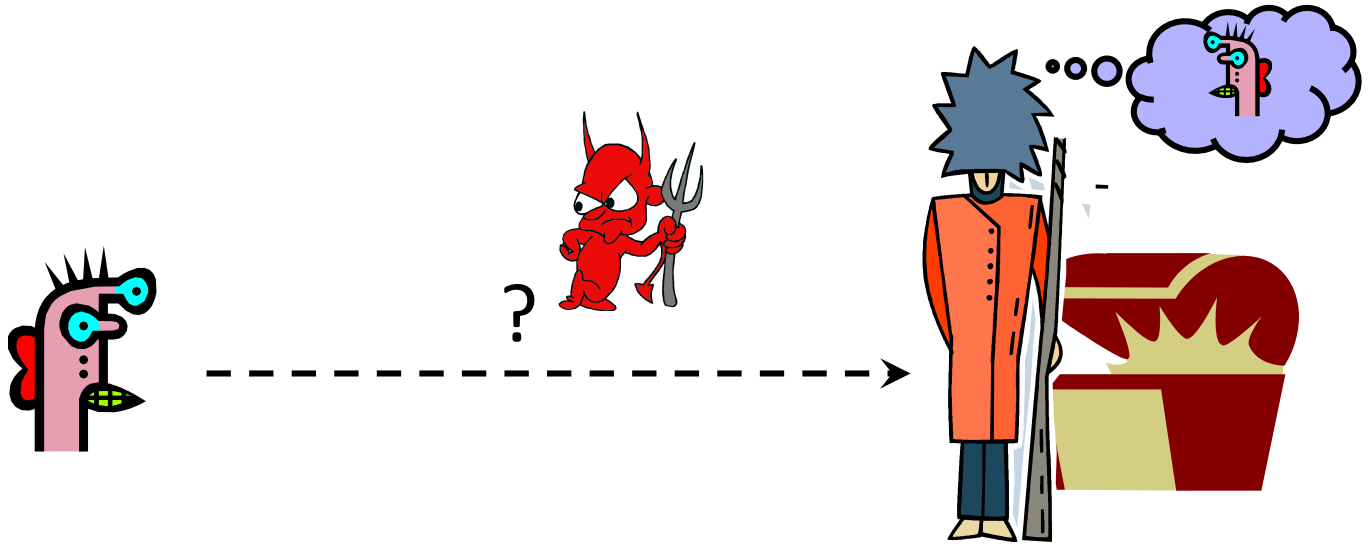
XSRF (aka CSRF)



8:34

Authentication

Basic Problem



How do you prove to someone that
you are who you claim to be?

Any system with access control must solve this problem.

Many Ways to Prove Who You Are

- "Something you know"
  - Passwords
  - Answers to questions that only you know
- "Something you have"
  - Secure tokens, mobile devices
- "Something you are"
  - Biometrics

A slightly more fundamental question

- What are we trying to prove? (pollev)

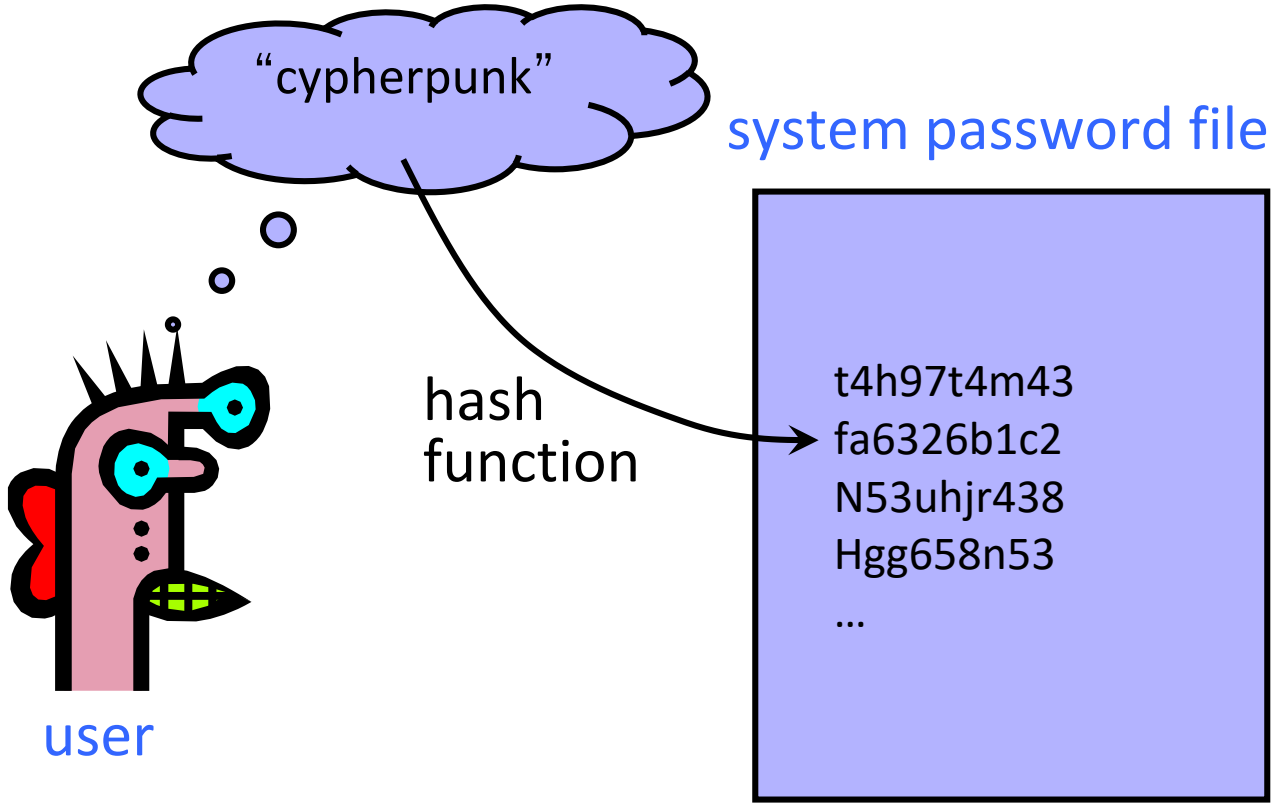
pollev.com/dkohlbre

Passwords and Computer Security

- In 2012, 76% of network intrusions exploited weak or stolen credentials (username/password)
  - Source: Verizon Data Breach Investigations Report
- In Mitnick's "Art of Intrusion" 8 out of 9 exploits involve password stealing and/or cracking
- First step after any successful intrusion: install sniffer or keylogger to steal more passwords
- Second step: run cracking tools on password files
  - Cracking needed because modern systems usually do not store passwords in the clear

UNIX-Style Passwords

- How should we store passwords on a server?
  - In cleartext?
  - Encrypted?
  - Hashed?



Password Hashing

- Instead of user password, store H(password)
- When user enters password, compute its hash and compare with entry in password file
  - System does not store actual passwords!
  - System itself can't easily go from hash to password
    - Which would be possible if the passwords were encrypted

- Hash function H must have some properties
  - One-way: given H(password), hard to find password
    - No known algorithm better than trial and error
  - "Slow" to compute

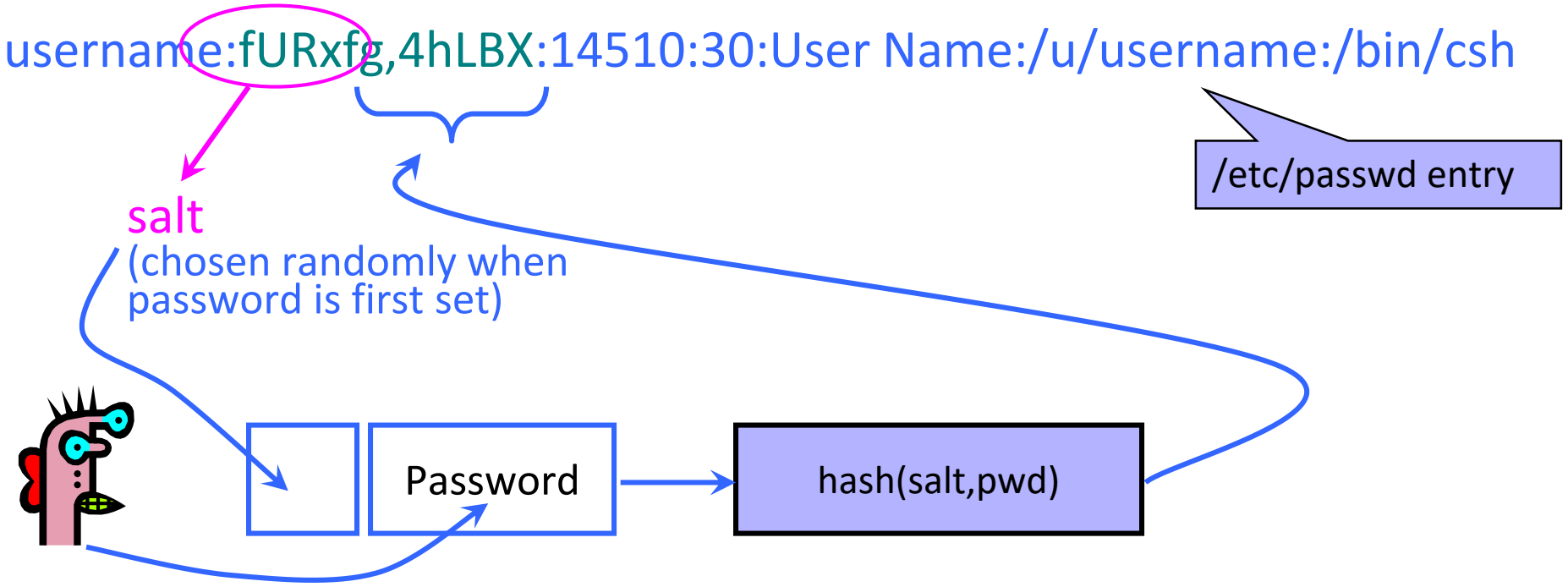
UNIX Password System

- Approach: Hash passwords
- Problem: passwords are not truly random
  - With 52 upper- and lower-case letters, 10 digits and 32 punctuation symbols, there are 94^8 == 6 quadrillion possible 8-character passwords ($\sim 2^{52}$)
  - **BUT:** Humans like to use dictionary words, human and pet names == 1 million common passwords

Dictionary Attack

- Dictionary attack is possible because many passwords come from a small dictionary
  - Attacker can pre-compute H(word) for every word in the dictionary – this only needs to be done once!
    - This is an offline attack
    - Once password file is obtained, cracking is instantaneous
  - Sophisticated password guessing tools are available
    - Take into account freq. of letters, password patterns, etc.

Salt



- Users with the same password have <u>different</u> entries in the password file
- Offline dictionary attack becomes much harder

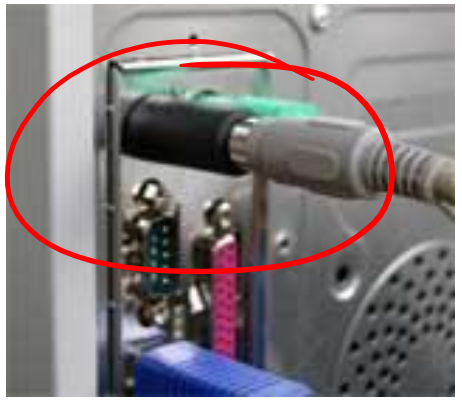
Choose a word

Advantages of Salting

- Without salt, attacker can pre-compute hashes of all dictionary words once for all password entries
  - Same hash function on all UNIX machines
  - Identical passwords hash to identical values; one table of hash values can be used for all password files
- With salt, attacker must compute hashes of all dictionary words once for each password entry
  - With 12-bit random salt, same password can hash to 2^{12} different hash values
  - Attacker must try all dictionary words **for each salt value** in the password file
- Pepper: Secret salt (not stored in password file)

Other Password Security Risks

- Keystroke loggers
  - Hardware
  - Software (spyware)
- Shoulder surfing
- Same password at multiple sites
- Broken implementations
  - Recall TENEX timing attack
- Social engineering



Other Issues

- Usability
  - Hard-to-remember passwords?
  - Carry a physical object all the time?
- Denial of service
  - Attacker tries to authenticate as you, account locked after three failures

Default Passwords

- Examples from Mitnick's "Art of Intrusion"
  - U.S. District Courthouse server: "public" / "public"
  - NY Times employee database: pwd = last 4 SSN digits
- Mirai IoT botnet
  - Weak and default passwords on routers and other devices

Weak Passwords

- RockYou hack
  - "Social gaming" company
  - Database with 32 million user passwords from partner social networks
  - Passwords stored in the clear
  - December 2009: entire database hacked using an SQL injection attack and posted on the Internet
  - One of many such examples!

Weak Passwords

- RockYou hack
  - "
  - D
  - P
  - D
    p

2009

rockyou™

Password Popularity – Top 20

Rank	Password	Number of Users with Password (absolute)
1	123456	290731
2	12345	79078
3	123456789	76790
4	Password	61958
5	iloveyou	51622
6	princess	35231
7	rockyou	22588
8	1234567	21726
9	12345678	20553
10	abc123	17542

Rank	Password	Number of Users with Password (absolute)
11	Nicole	17168
12	Daniel	16409
13	babygirl	16094
14	monkey	15294
15	Jessica	15162
16	Lovely	14950
17	michael	14898
18	Ashley	14329
19	654321	13984
20	Qwerty	13856

[Inglesant and Sasse, “The True Cost of Unusable Password Policies”]

Password Policies

- Old recommendation:
  - 7 or 8 characters, at least 3 out of {digits, upper-case, lower-case, non-alphanumeric}, no dictionary words, change every 4 months, password may not be similar to previous 12 passwords…

Image from http://www.interactivetools.com/staff/dave/damons_office/

[Inglesant and Sasse, "The True Cost of Unusable Password Policies"]

Password Policies

- Old recommendation:
  - 7 or 8 characters, at least 3 out of {digits, upper-case, lower-case, non-alphanumeric}, no dictionary words, change every 4 months, password may not be similar to previous 12 passwords…
- **But** … results in frustrated users and less security
  - Burdens of devising, learning, forgetting passwords
  - Users construct passwords insecurely, write them down
    - Can't use their favorite password construction techniques (small changes to old passwords, etc.)
  - Heavy password re-use across systems
  - (Password managers can help)

“New” (2017) NIST Guidelines ☺

- Remove requirement to periodically change passwords
- Screen for commonly used passwords
- Allow copy-paste into password fields
  - But concern: what apps have access to clipboard?
- Allow but don’t require arbitrary special characters
- Etc.

https://pages.nist.gov/800-63-3/sp800-63b.html

Recovering Passwords

Palin E-Mail Hacker Says It Was Easy

By Kim Zetter ✉ September 18, 2008 | 10:05 am | Categories: Elections, Hacks and Cracks

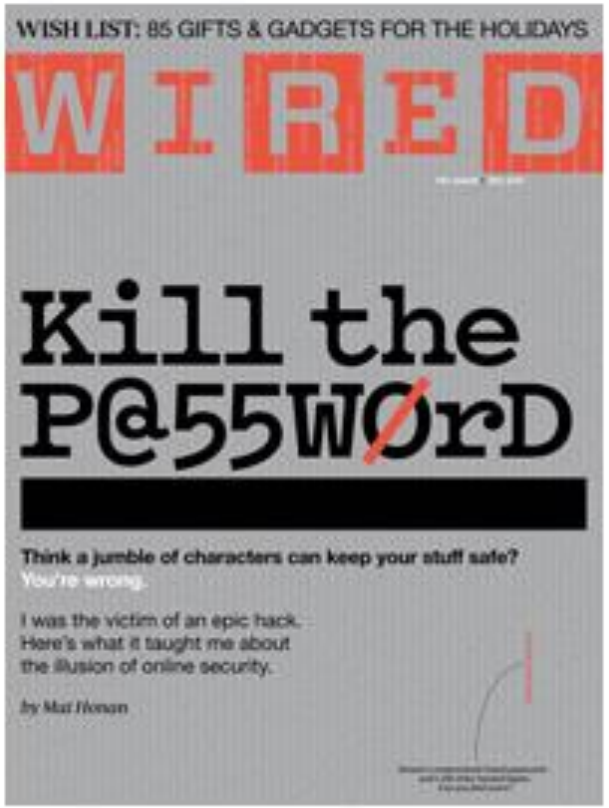
A p
obt
priv
sup
rev
too
Re

after the password recovery was reenabled, it took seriously 45 mins on wikipedia and google to find the info, Birthday? 15 seconds on wikipedia, zip code? well she had always been from wasilla, and it only has 2 zip codes (thanks online postal service!)

the second was somewhat harder, the question was "where did you meet your spouse?" did some research, and apparently she had eloped with mister palin after college, if youll look on some of the screenshits that I took and other fellow anon have so graciously put on photobucket you will see the google search for "palin eloped" or some such in one of the tabs.

I found out later though more research that they met at high school, so I did variations of that, high, high school, eventually hit on "Wasilla high" I promptly changed the password to popcorn and took a cold shower…

Wired Cover Story (Dec 2012)



"This summer, hackers destroyed my entire digital life in the span of an hour. My Apple, Twitter, and Gmail passwords were all robust—seven, 10, and 19 characters, respectively, all alphanumeric, some with symbols thrown in as well—but the three accounts were linked, so once the hackers had conned their way into one, they had them all. They really just wanted my Twitter handle: @mat."

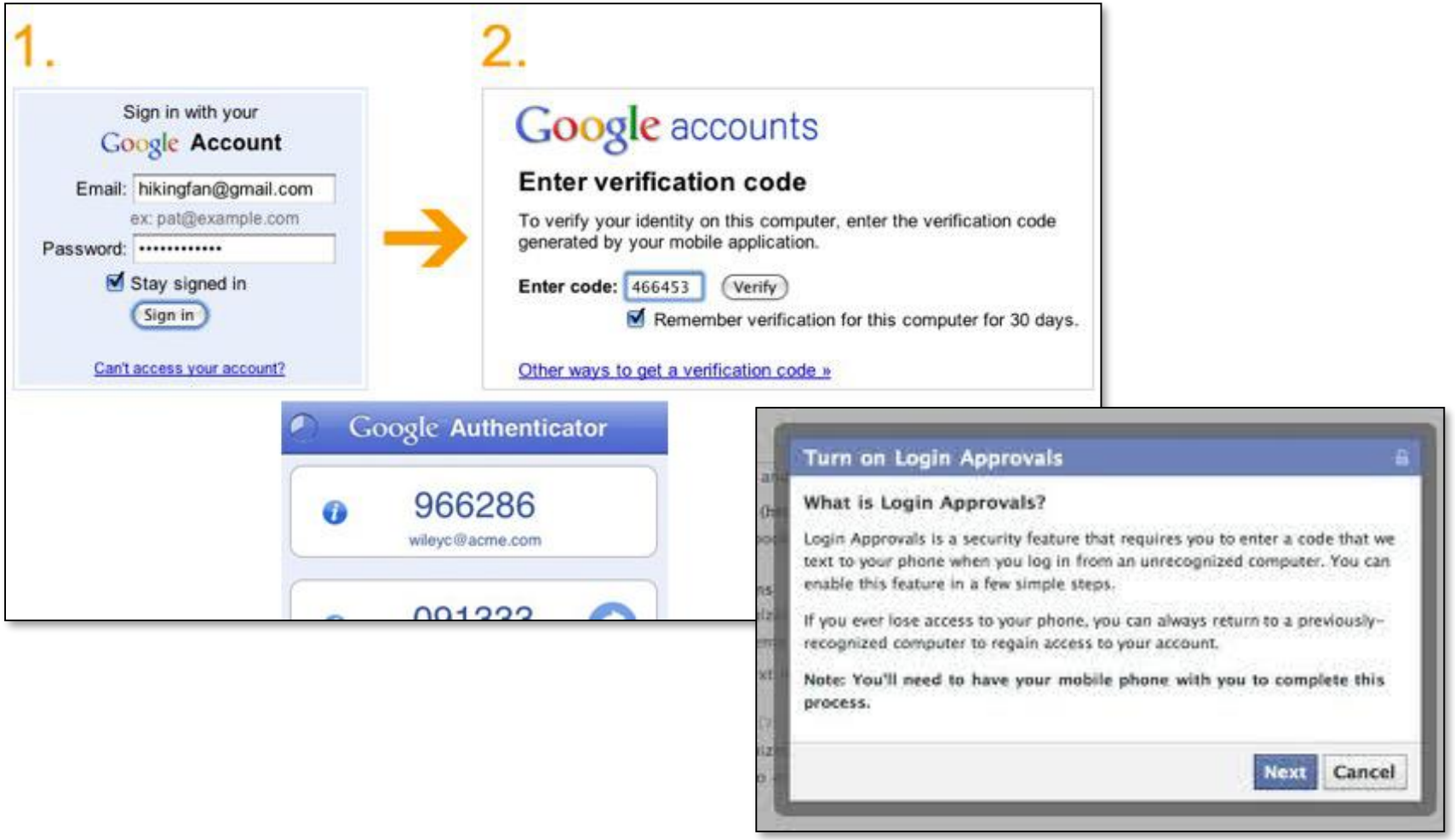
Improving(?) Passwords

- Add biometrics
  - For example, keystroke dynamics or voiceprint
- Graphical passwords
  - Goal: easier to remember? no need to write down?
- Password managers
  - Examples: LastPass, KeePass, built into browsers
  - Can have security vulnerabilities…
- Two-factor authentication
  - Leverage phone (or other device) for authentication

Password managers

- Generation
  - Secure generation of random passwords
- Management
  - Allows for password-per-account
- Safety?
  - Single point of failure
  - Vulnerability?
  - Phishing?

Multi-Factor Authentication



FIDO + Hardware Two Factors

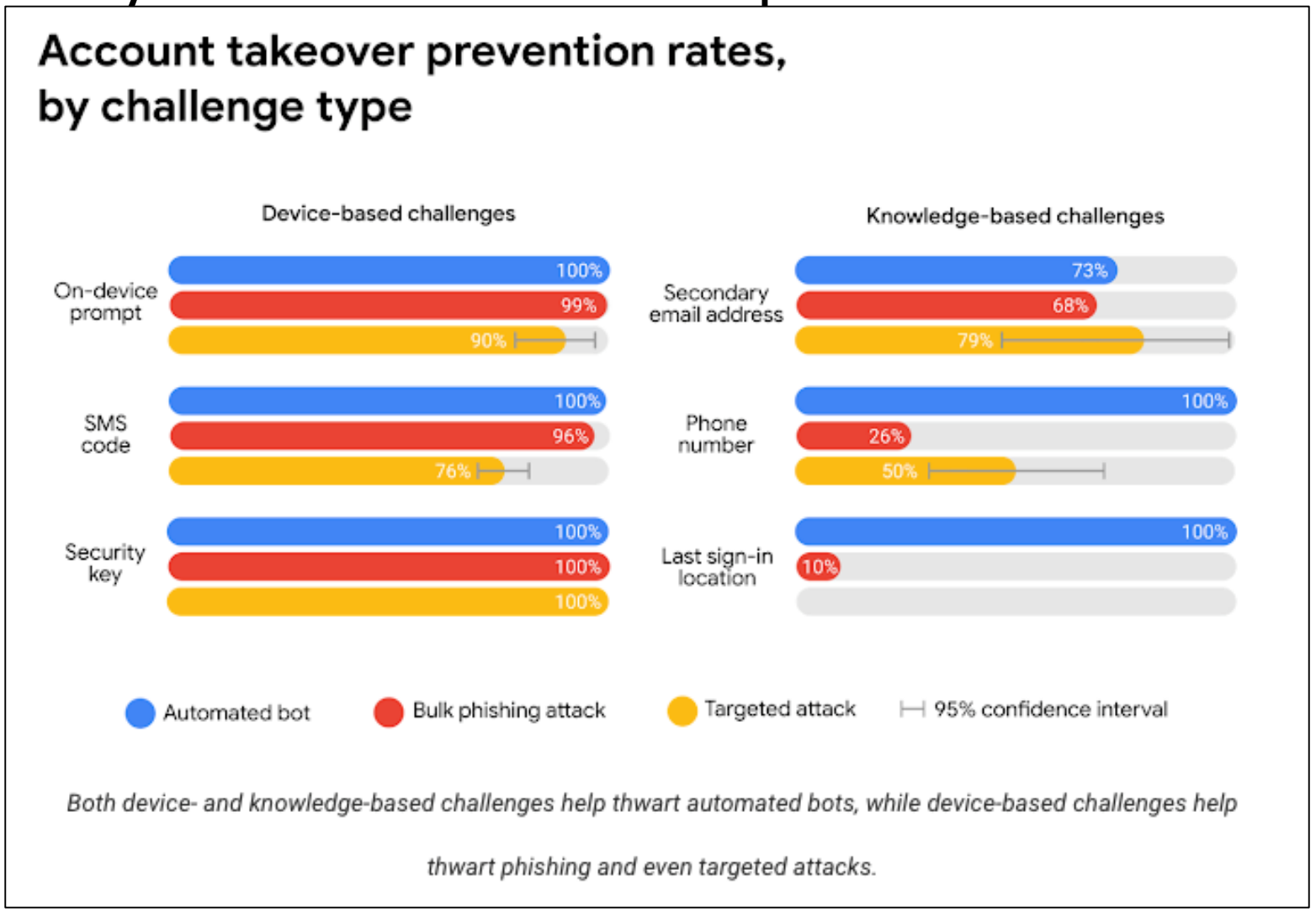
Questions:

Do you use 2-factor auth?
Do you use a password manager?
Why or why not?

How to compromise account protected with hardware second factor?

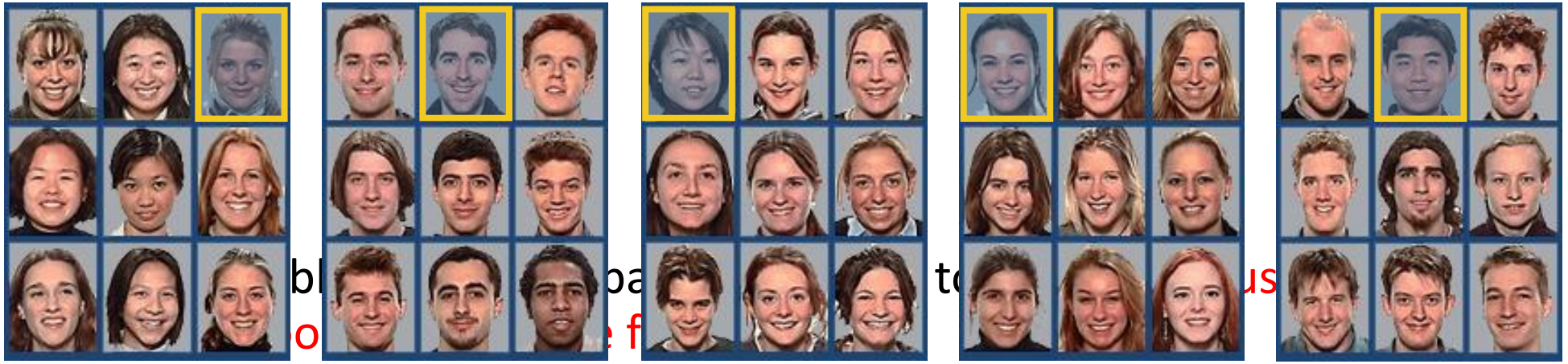
https://security.googleblog.com/2019/05/new-research-how-effective-is-basic.html

Secondary Factors Do Help!



Graphical Passwords

- Many variants… one example: Passfaces
  - Assumption: easy to recall faces

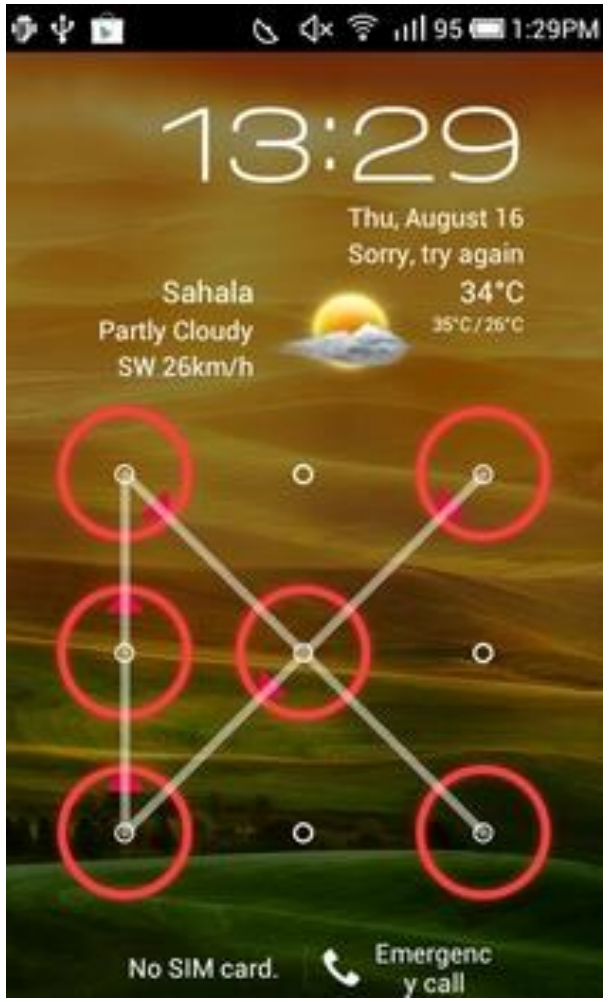


Graphical Passwords

- Another variant: draw on the image (Windows 8)

- Problem: users choose predictable points/lines

Unlock Patterns



- Problems:
  - Predictable patterns (familiar pattern by now)
  - Smear patterns
  - Side channels: apps can use accelerometer and gyroscope to extract pattern!

What About Biometrics?

- Authentication: **What you are**
- Unique identifying characteristics to authenticate user or create credentials
  - Biological and physiological: Fingerprints, iris scan
  - Behaviors characteristics - how perform actions: Handwriting, typing, gait
- Advantages:
  - Nothing to remember
  - Passive
  - Can’t share (generally)
  - With perfect accuracy, could be fairly unique

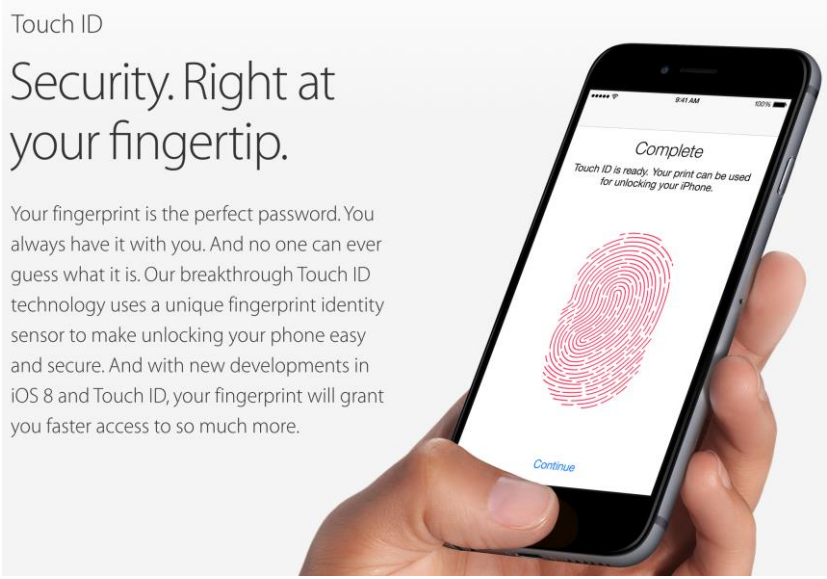
Issues with Biometrics

- Private, but not secret
  - Maybe encoded on the back of an ID card?
  - Maybe encoded on your glass, door handle, ...
  - Sharing between multiple systems?
- Revocation is difficult (impossible?)
  - Sorry, your iris has been compromised, please create a new one...
- Physically identifying
  - Soda machine to cross-reference fingerprint with DMV?
- Birthday paradox
  - With false accept rate of 1 in a million, probability of false match is above 50% with only 1609 samples

Shifting Threat Models…

Attacking Biometrics

- An adversary might try to steal biometric info
  - Malicious fingerprint reader
    - Consider when biometric is used to derive a cryptographic key
  - Residual fingerprint on a glass

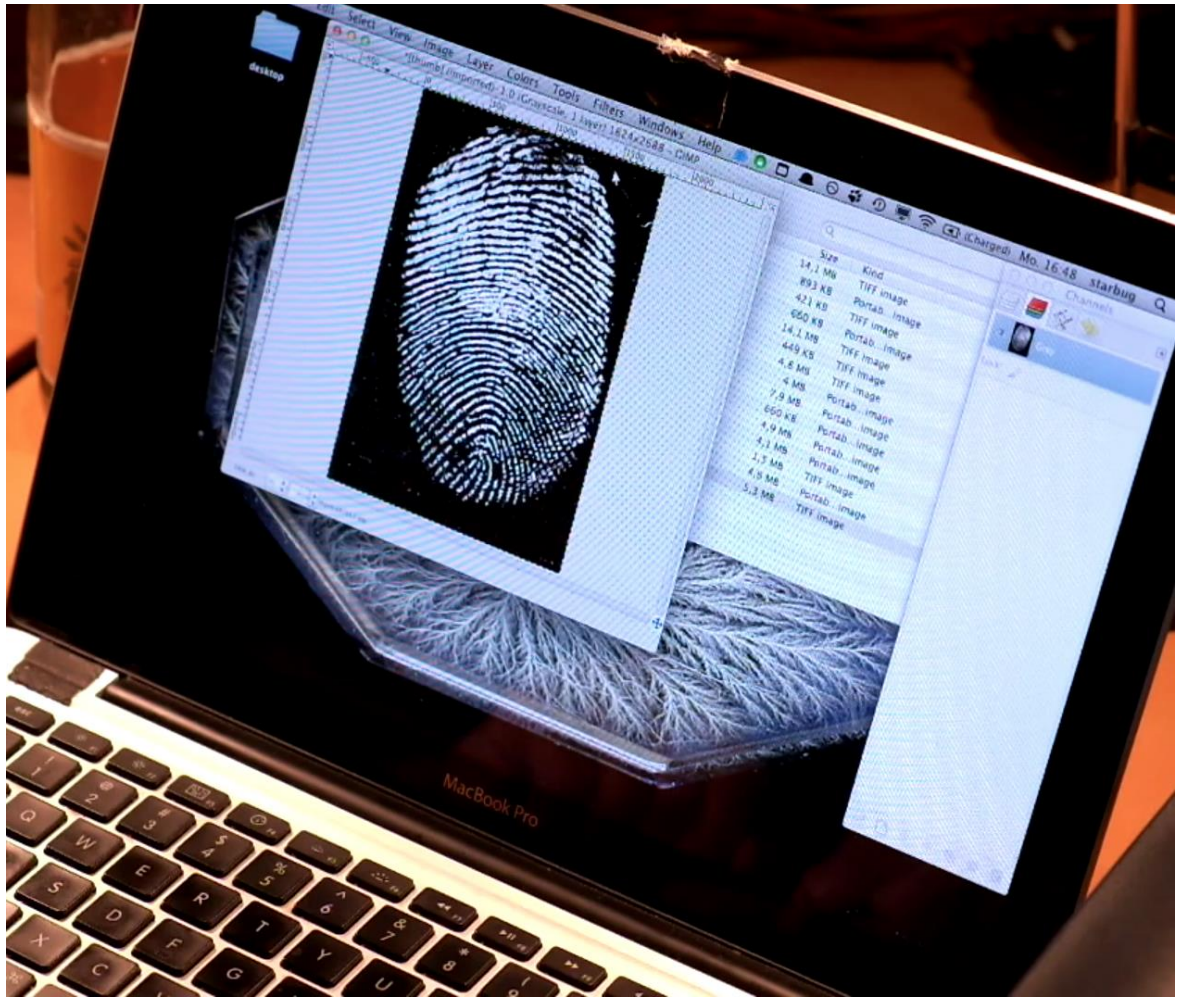


[Starbug -- http://istouchidhackedyet.com/]

Attacking Biometrics

[Starbug -- http://istouchidhackedyet.com/]

Attacking Biometrics



[Starbug -- http://istouchidhackedyet.com/]

Attacking Biometrics

[Starbug -- http://istouchidhackedyet.com/]

Attacking Biometrics

(2022) Passkeys!

- Goal: Replace passwords

- Solution:
  - “something you are” + “something you have” -> generate keys
  - OS managed
  - Keys are pub/private *per-account you login to*
  - Cannot be phished
  - Cannot be ‘lost’ (sort of)

https://arstechnica.com/information-technology/2022/10/passkeys-microsoft-apple-and-googles-password-killer-are-finally-here/

https://developer.apple.com/passkeys/

https://android-developers.googleblog.com/2022/10/bringing-passkeys-to-android-and-chrome.html