COMP435: *SECURITY CONCEPTS!*

Lecture 24: SQL Injection, Other Web-Based Attacks

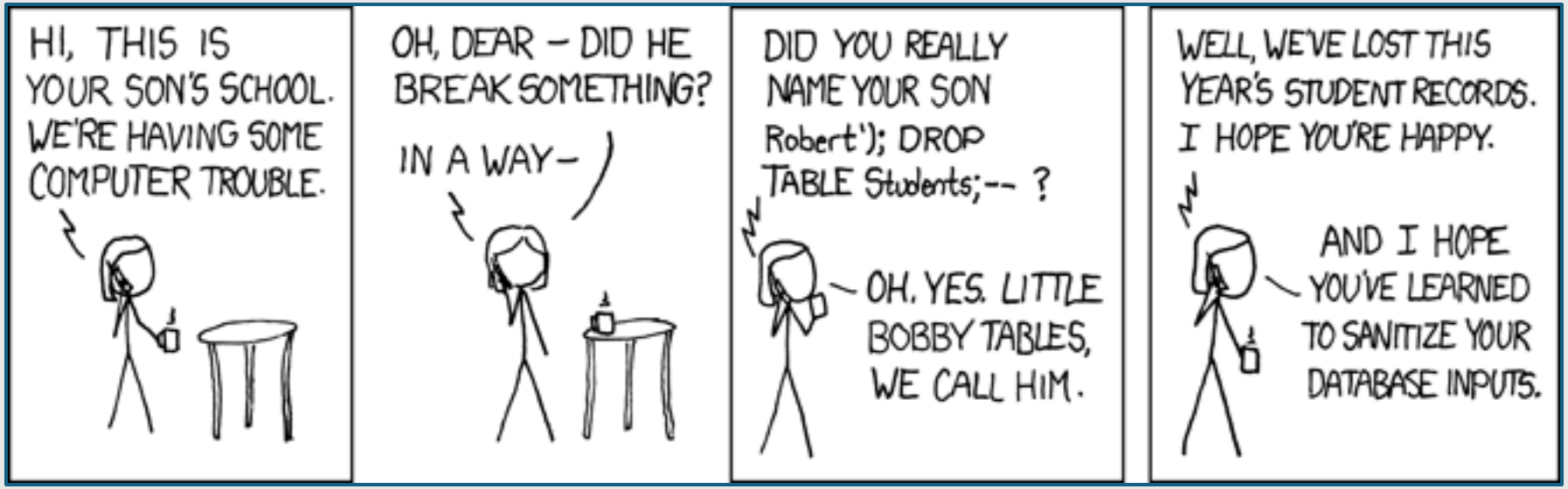
tinyurl.com/comp435-fa25

SQL INJECTION

Structured Query language

SQL Injection

Def'n: user-provided data is used in a database query, but is interpreted as code rather than data



https://xkcd.com/327/

Web Architecture with SQL Database

client

web server

database server

Web server front-end

SQL database back-end

Ideally, HTTPS so password is not sent in the clear... ☺

Server receives the request + decrypts it

Server has the information!

HTTP request

client

web server

database server

POST http://www.... HTTP/1.1

....

name=Alice&passwd=cheshire

Web server constructs a query dynamically using:

- data from the user
- cookies
- Variables/data stored on the server about this particular user

client

HTTP request

web server

database server

```
query = "SELECT * FROM pswdtab
WHERE username='" + name + "' AND
password='" + passwd + "'"
```

```
query = "SELECT * FROM pswdtab
WHERE username='" + name + "' AND
password='" + passwd + "'"
```

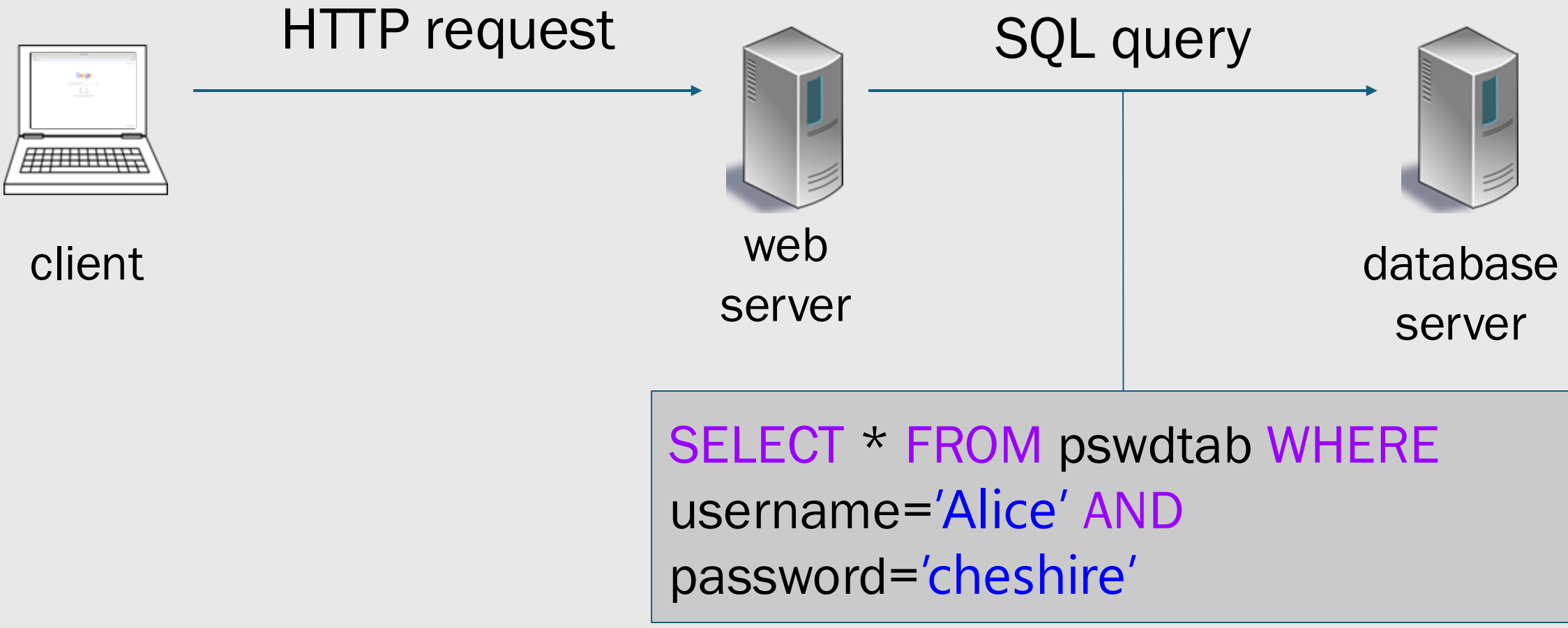
name = Alice

passwd = 1234

SELECT * from pswdtab

WHERE username='Alice' AND

password='1234'


HTTP request
SQL query
client
web server
database server
SELECT * FROM pswdtab WHERE
username='Alice' AND
password='cheshire'

DB retrieves the records (rows) in DB which match the query

client

HTTP request

web server

SQL query

database server

executes query

client

web server

results

database server

name=Alice W;
ID=4512

Server uses information to create the response page

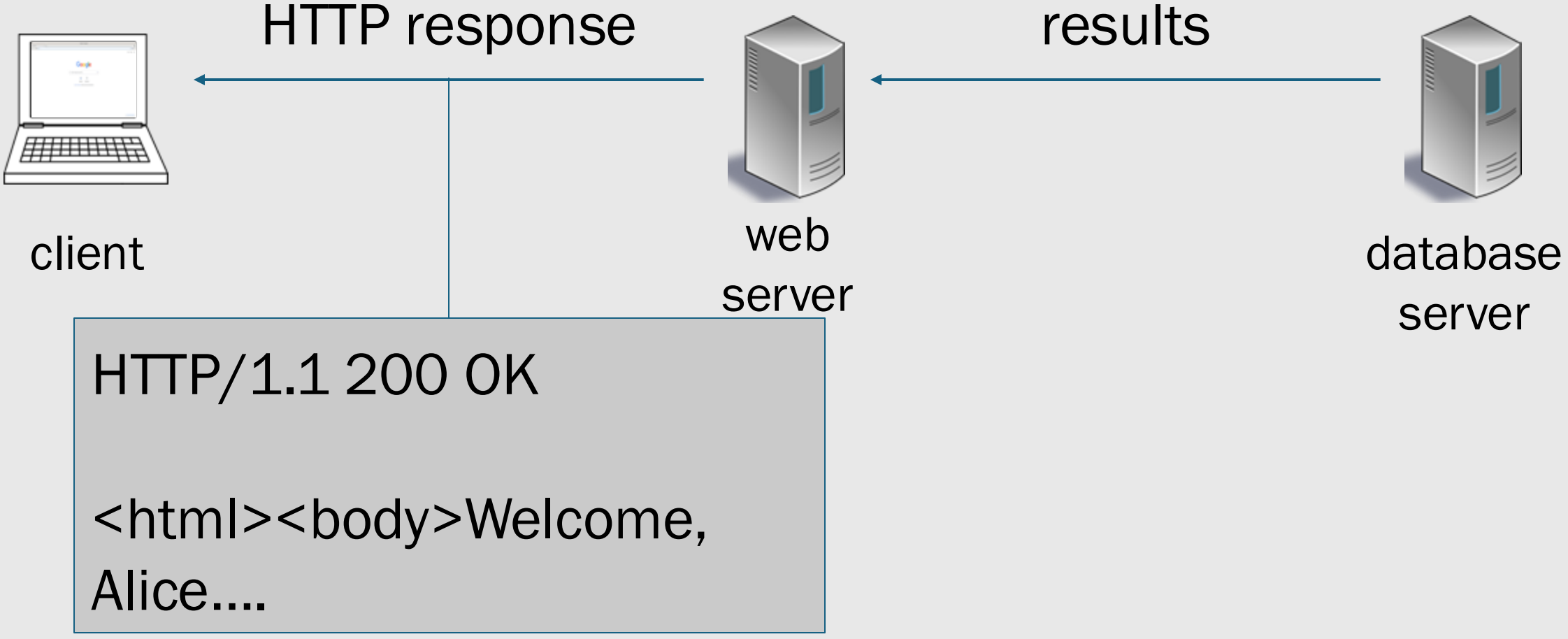
client

web server

results

database server

creates response


HTTP response
results
client
web server
database server
HTTP/1.1 200 OK
<html><body>Welcome, Alice....

SQL Injection

Client is the attacker!
They type the string shown in blue (no password)

client

HTTP request

web server

database server

```
POST http://www....  HTTP/1.1
....
name=root' --&passwd=
```

Recall: web server constructs query dynamically

'name' variable gets replaced with string the attacker supplied

client

HTTP request

web server

database server

```
query = "SELECT * FROM pswdtab
WHERE username='" + name + "' AND
password='" + passwd + "'"
```

root' --

Here is the final query to the DB.

In SQL, everything after the two dashes is treated as a comment!

client

HTTP request

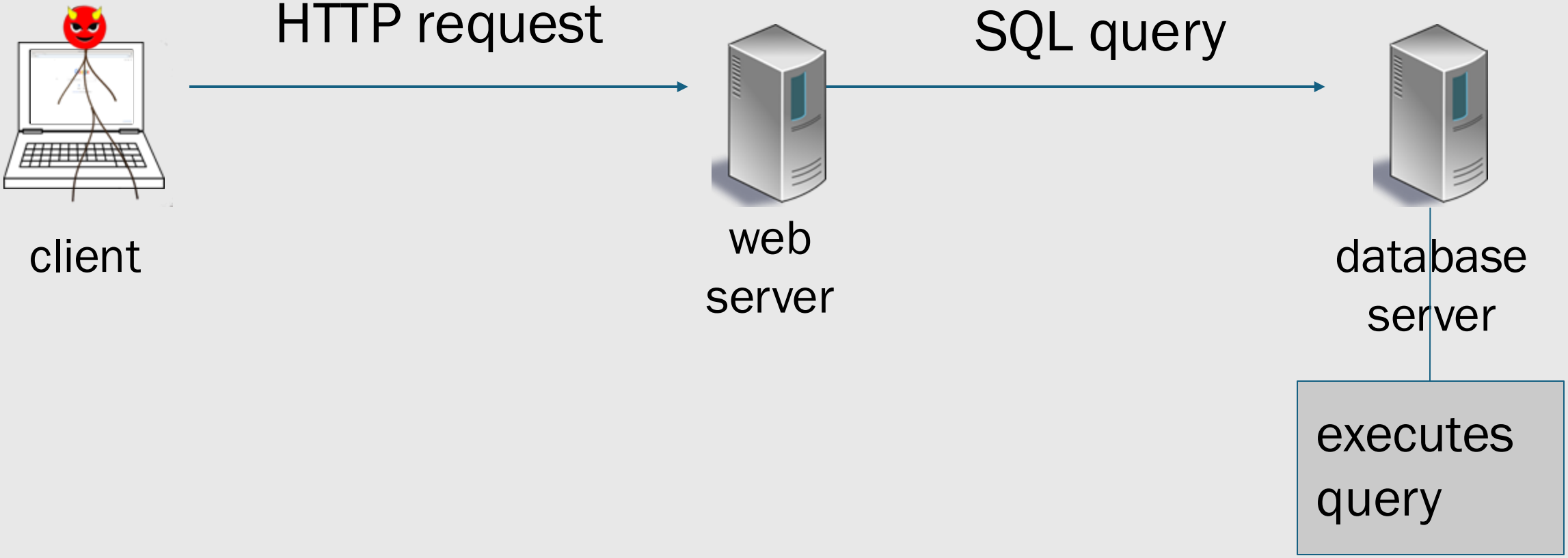
web server

SQL query

database server

No constraints that the password field be some value...
just that username == root

```
SELECT * FROM pswdtab WHERE
username='root' -- ' AND password=''
```


HTTP request
SQL query
client
web server
database server
executes query

client

HTTP request

web server

SQL query

database server

name=root;
ID=1111

as far as the web server is concerned, the client has authenticated as root.

Another Example

client

HTTP request

web server

database server

```
POST http://www....  HTTP/1.1
....
name=' OR 1=1 --&passwd=
```

client

HTTP request

web server

database server

```
query = "SELECT * FROM pswdtab
WHERE username='" + name + "' AND
password='" + passwd + "'"
```

query is for every entry in the passwd table where username = empty string OR 1 = 1… always true!

Every entry will be returned.

client

HTTP request

web server

SQL query

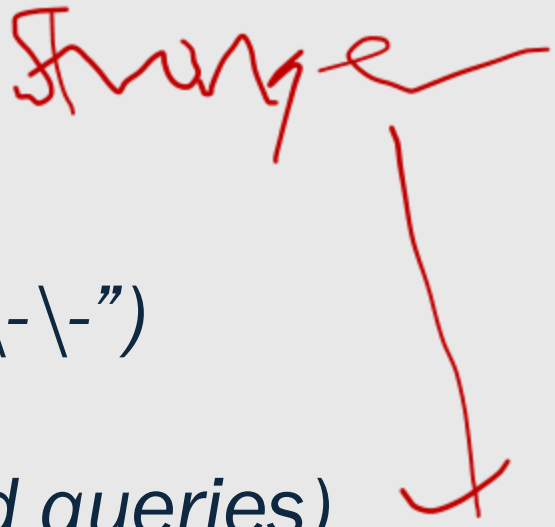
database server

```
SELECT * FROM pswdtab WHERE
username=" OR 1=1 -- ' AND
password="
```

SQL Injection

Input Sanitization: cleaning and validating user-provided data to remove or reject any potentially harmful or unexpected input before it is processed by a program.

- A form of code injection
- Data (user's input) and control (server's query) share a channel
- Input Sanitization is needed
- Defenses:
  - *blocking known-bad input (e.g., "drop")*
  - *escaping control characters (e.g., "--" → "\-\-")*
  - *defining allowed input templates*
  - *using prepared statements (parameterized queries)*



"SELECT * FROM pswtbl
WHERE username = 'Alice' AND
password = '1234'; DROP TABLE
passwd --'))

WS Questions 1-2

DROP: ⟹ deletes!

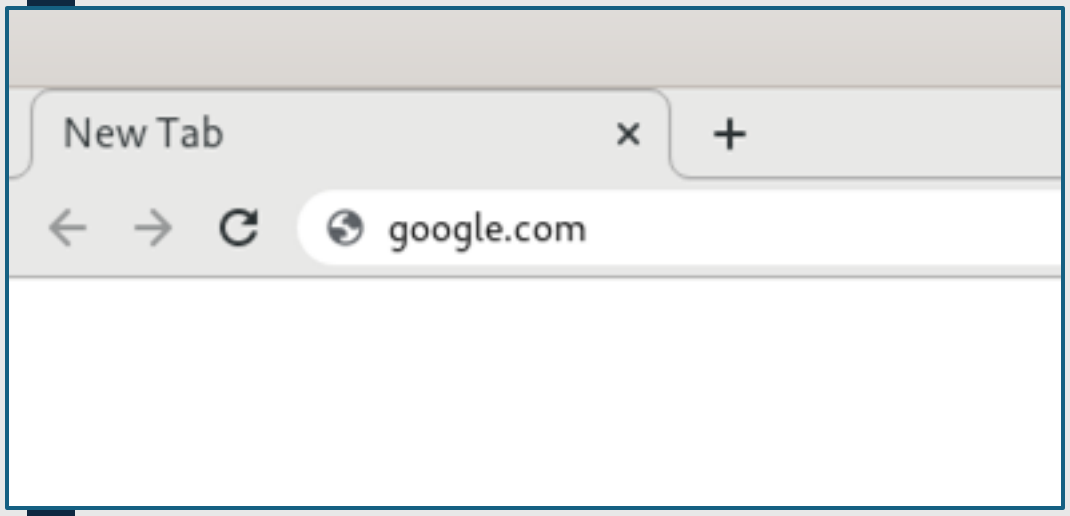
OTHER WEB-BASED ATTACKS

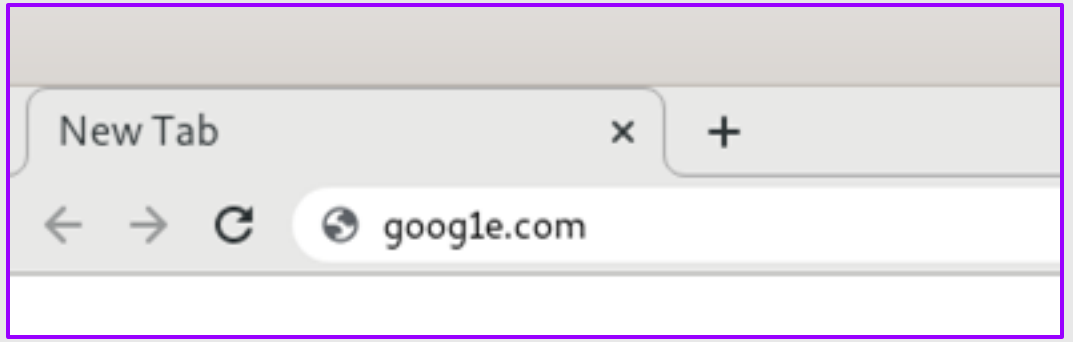
URL Hijacking

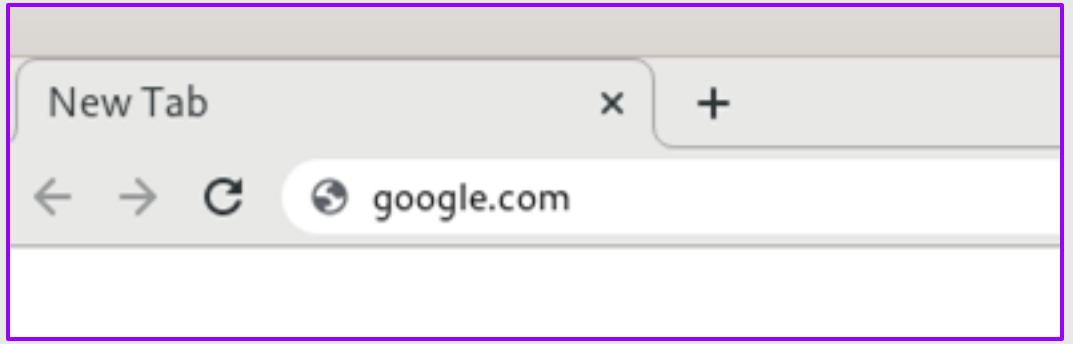
Def’n: attackers create sites with look-alike URLs that mimic well known sites

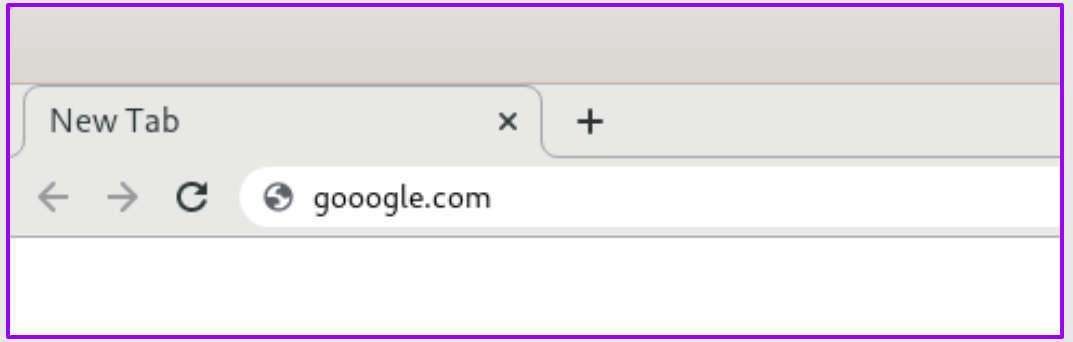
a.k.a. typosquatting

URL Hijacking

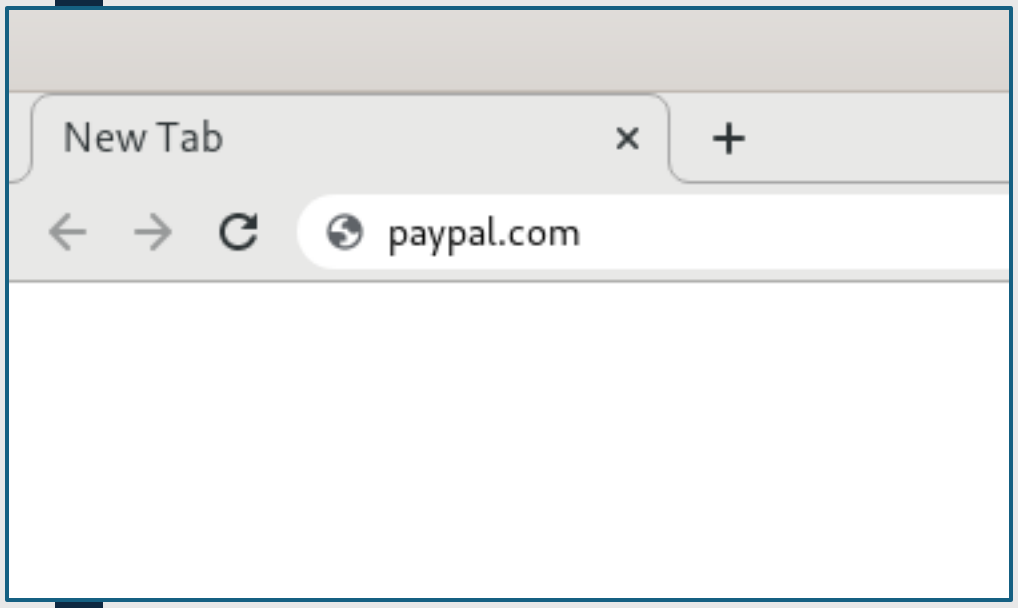


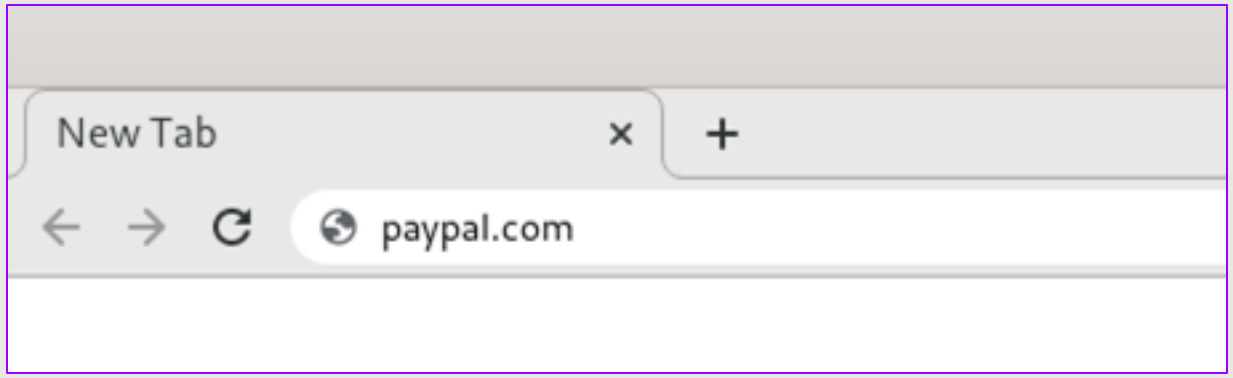


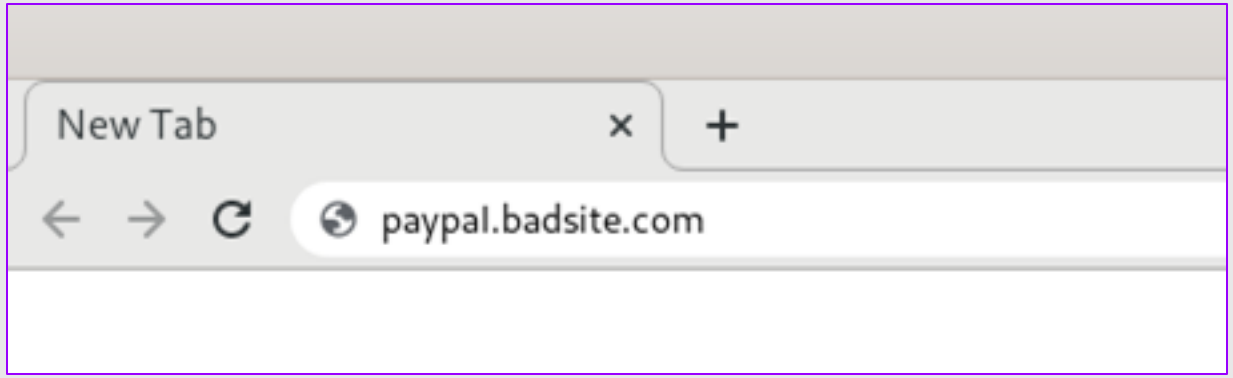


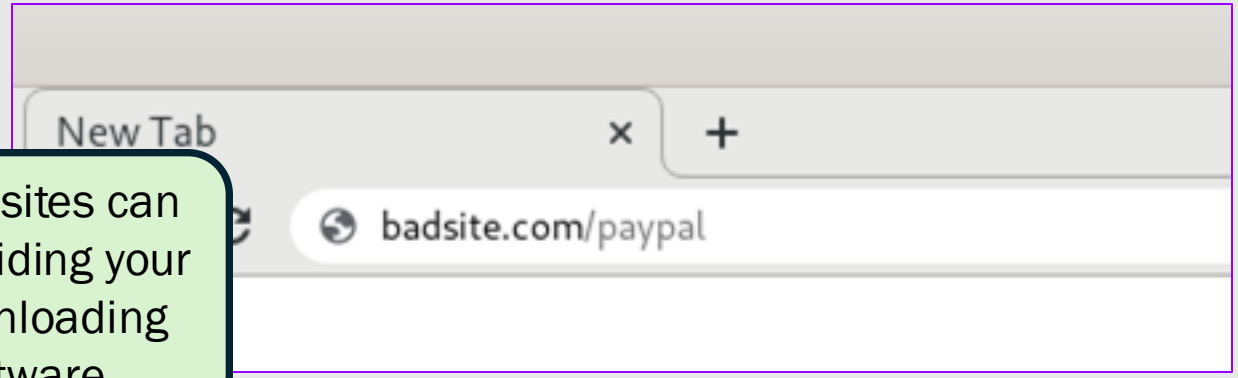


URL Hijacking







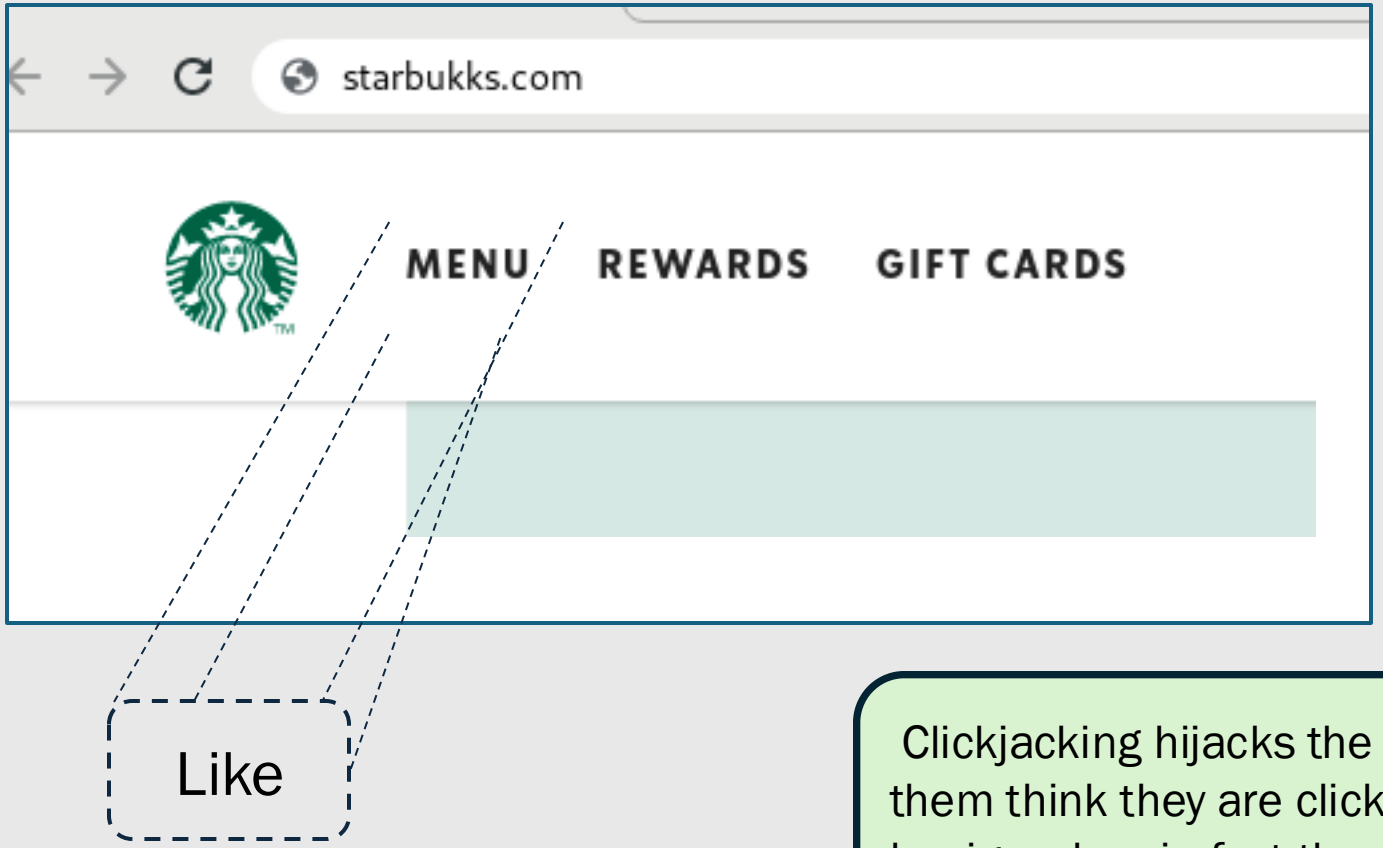


these URLs could be used w/spam email or user-provided links on other sites

once there these sites can trick you into providing your credentials, downloading malicious software

Clickjacking

Clickjacking is a way of tricking users into providing desired input. The attacker makes the input dialog transparent and places an image with an enticement below the transparent dialog



Clickjacking hijacks the user's click… makes them think they are clicking on something benign when in fact they are confirming something dangerous.

Dot Dot Slash

```
http://yoursite.com/webhits.htw?CiWebHits&File=../../../../../winnt/system32/autoexec.nt
```

An example of a directory traversal attack. Moving up folders in a server's file system → attacker can potentially read or execute system files if input is not sanitized.

Third Party Cookies

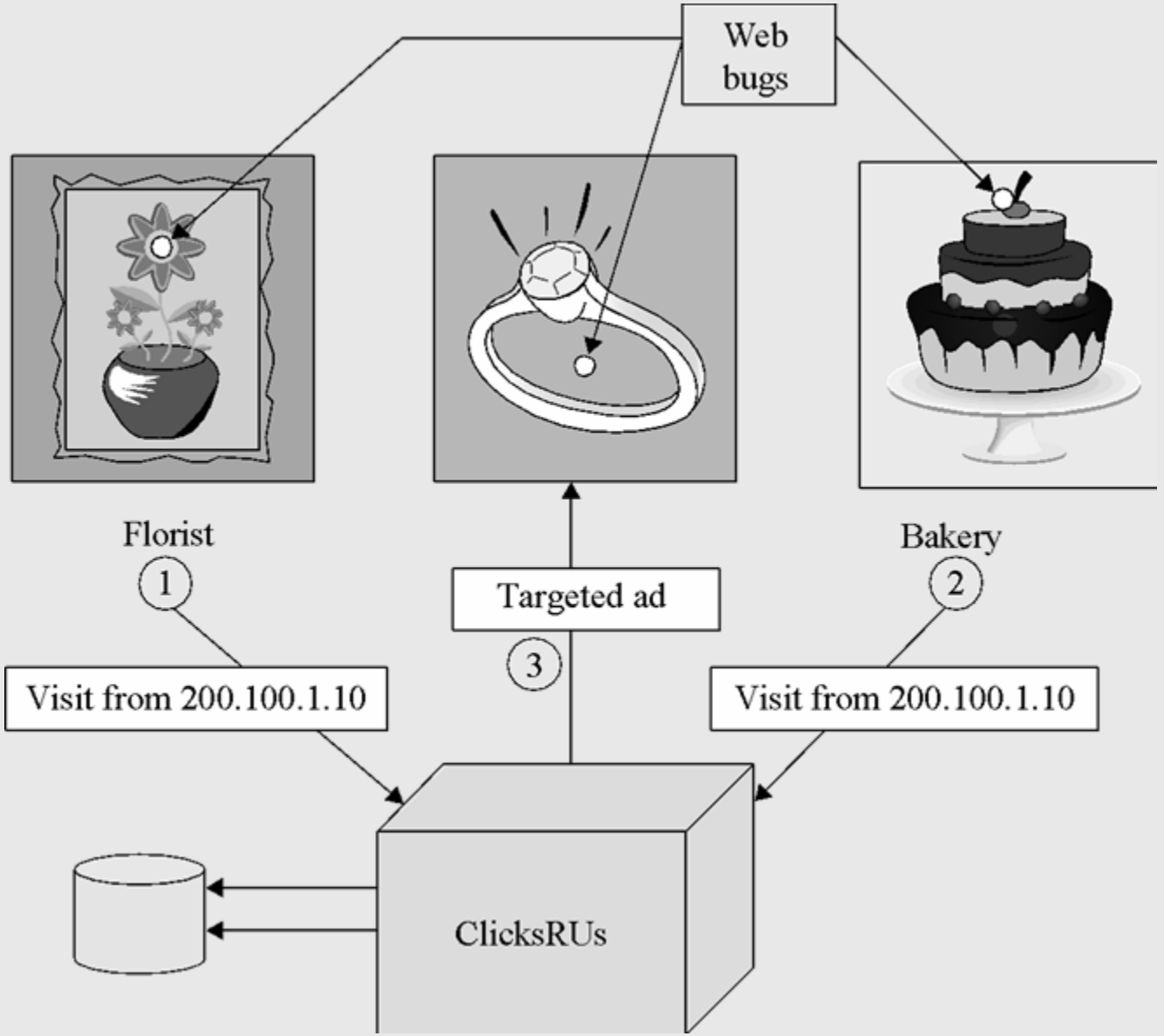
a.k.a. web bugs

Track users across the web!

Each site (florist, bakery), loads an image or ad from the same third-party domain (ClicksRUS).

When that resource loads, it sends identifying info (a cookie) back to the advertiser.

The tracker can now correlate visits from the same user, and build a browsing profile.



Server sends

set-cookie header

Rest of WS questions

Countermeasures

Countermeasures

- security indicators in the design so the user is aware!
  - *(ex: the padlock icon in your browser, showing the https prefix in your URL)*
- safe browsing
- security alerts
- using https

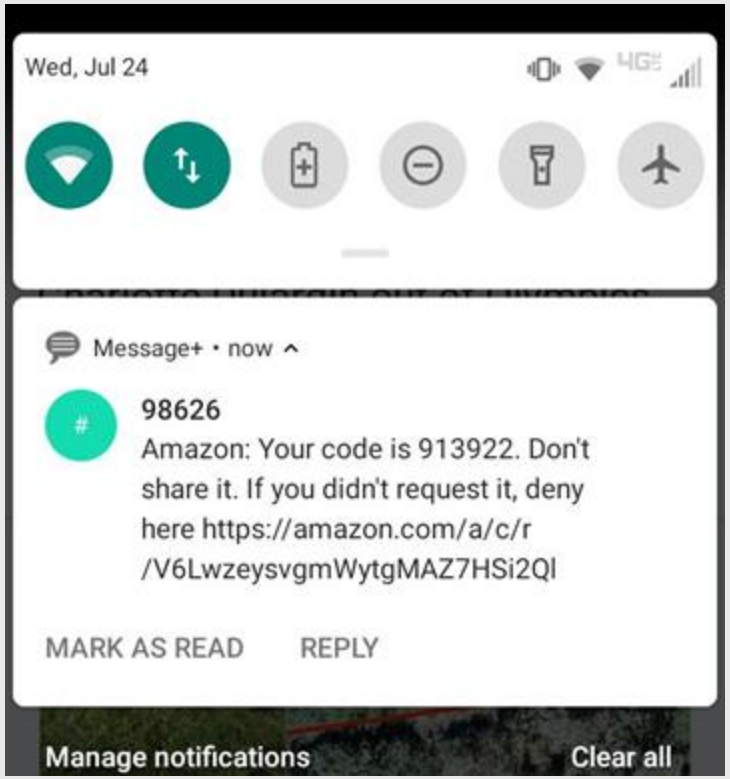
THE SPAM ECONOMY

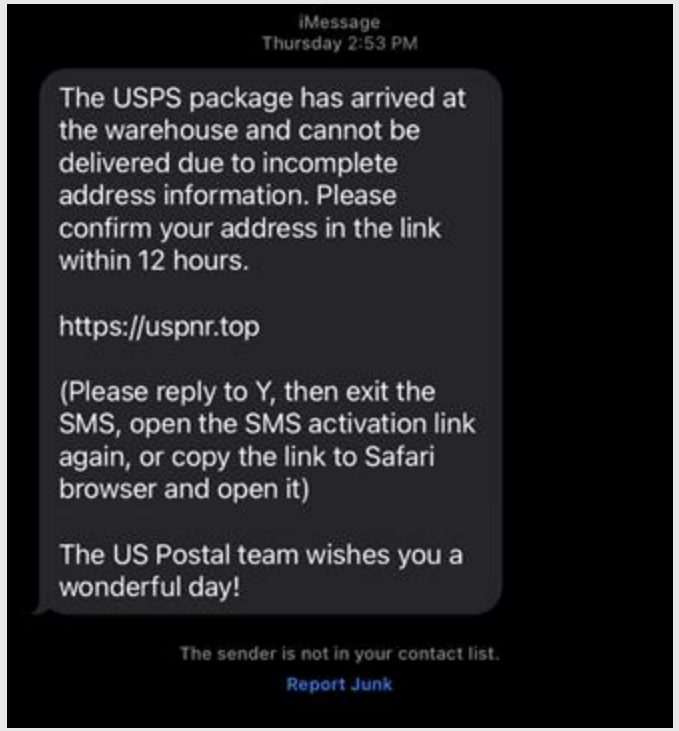
Spam

Def’n: unwanted email, text, or messages sent to advertise, sell, or scam people

Phishing

Def’n: an email or text that tries to trick the user into taking unsafe action





Spear Phishing

Def’n: a targeted email that tries to trick a particular user into taking unsafe action

Attackers might want to target individuals within an organization. Ex: targeting workers at NGOs. Can use social engineering, too

Pig Butchering

Def’n: a long-game scam that establishes a trusting relationship between target and attacker before asking the target for money

- attackers are typically enslaved people serving organized crime lords
- starts with spam text, “hey”

Motivation for Spam: $$

- Advertising
- Phishing
- Scams
- Stock price manipulation
- New bots

Monetizing Spam

- 60% – 90% of all email
- one campaign = 10s of billions messages
- Most spam is never delivered
- Success rates:
  - *1/200,000 (.0005%) malware hit rate*
  - *1/12,000,000 (.0000083%) purchase*

- Revenue: $1.5-2M/year

Kanich et al., 2011. http://www.icir.org/vern/papers/ppair-usesec11.pdf

Spam Countermeasures

- Server blocklists
- Email filtering

Provider

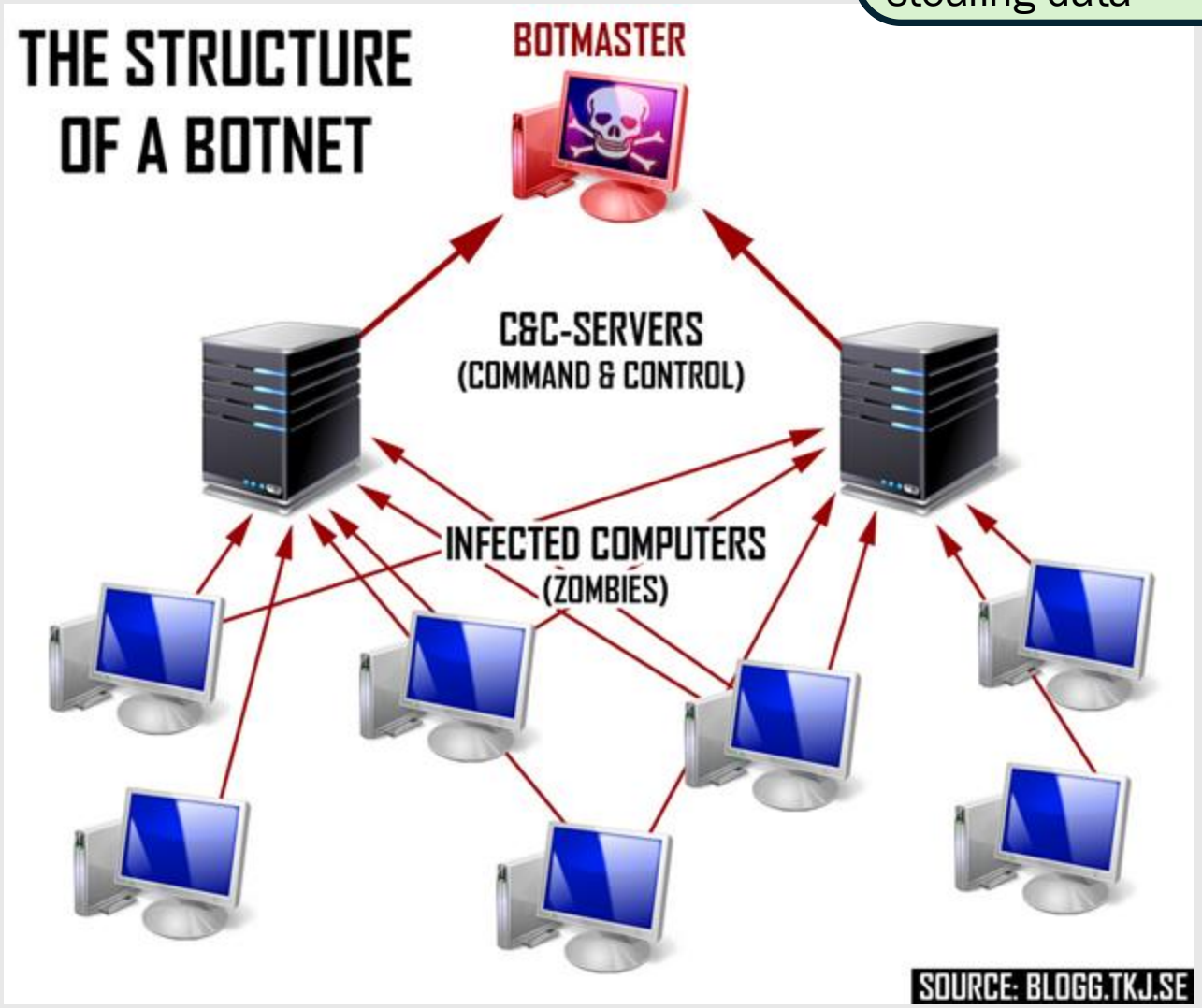
- Education
- Vigilance

User

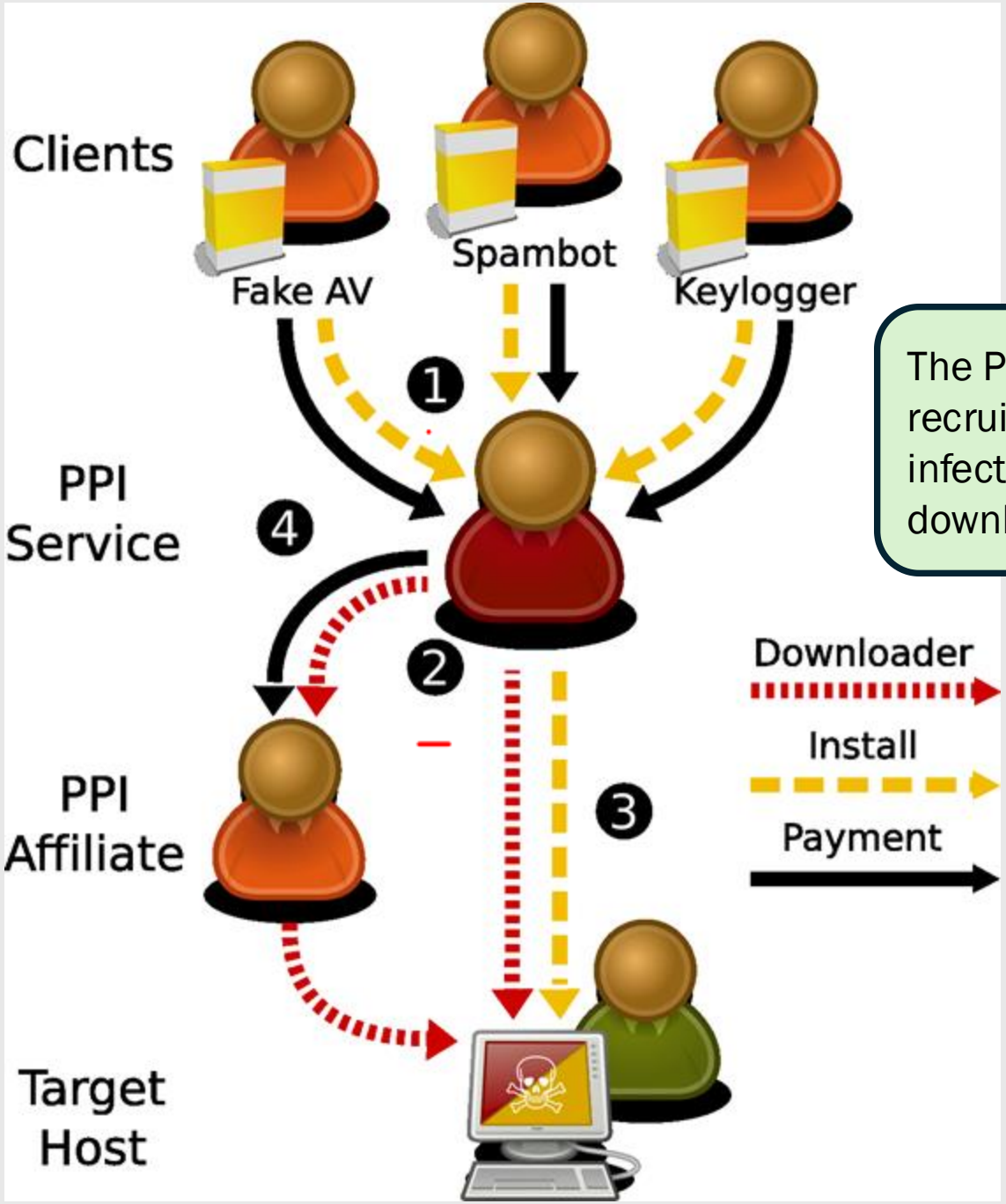
Underground Economy

Botnet

a network of compromised computers controlled remotely by an attacker to perform coordinated tasks like sending spam, launching DDoS attacks, or stealing data



PPI
→
pay per
install



The PPI Service acts as a broker: it recruits affiliates who already control infected systems or distribute downloaders

"Measuring Pay-per-Install: The Commoditization of Malware Distribution," Caballero et al. 2011

Threats to the Underground Economy

- Identification and authentication
- Single point of failure
- No honor among thieves

Threats to the Underground Economy

- Identification and authentication
- Single point of failure
- No honor among thieves

On the other hand

- A service economy