https://brown-csci1660.github.io

CS1660: Intro to Computer Systems Security
Spring 2025

Lecture 9: Web Security

Co-Instructor: **Nikos Triandopoulos**

February 25, 2025

BROWN

CS1660: Announcements

- Course updates
  - Project 2 is going out today
  - Homework 1 is due soon (Thu, Feb 27)
  - Where we are
    - **Part I: Crypto**
    - **Part II: Web**
    - Part III: OS
    - Part IV: Network
    - Part V: Extras

Today

- Web security
  - Web Security Models
  - Browser Security
  - Web Technologies and Protocols

Crypto recap through Discrepancies…

Discrepancies

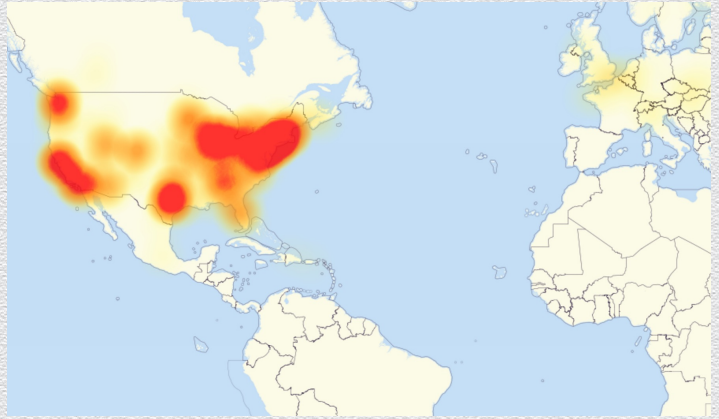
- Security Vs. cryptography
- Guarantees Vs. threat model
- Confidentiality Vs. integrity
- Prevention Vs. detection
- Old Vs. modern cryptography
- Perfect Vs. computational security
- Modelled Vs. practical attacker
- Crypto Vs. non-crypto security
- Truly Vs. pseudo random
- Secret Vs. public
- Theory Vs. practice
- Ideal model Vs. implementation
- Open Vs. closed design
- Symmetric Vs. asymmetric crypto
- Block Vs. all-length designs
- Data Vs. user authentication
- Set-up Vs. real-world assumptions
- Good hygiene Vs. arbitrary practices
- Random Vs. non-random

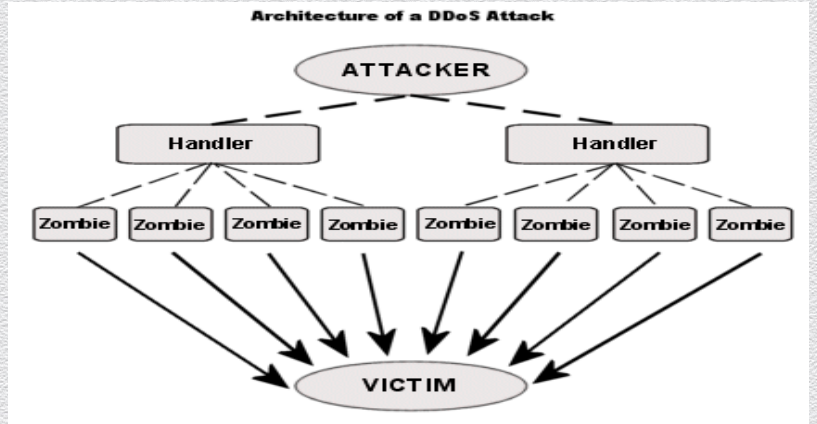
The Dyn DDoS attack

It’s unfair! – I had no class but couldn’t watch my Netflix series!

On October 21, 2016, a large-scale cyber was launched

- it affected globally the entire Internet but particularly hit U.S. east coast
- during most of the day, no one could access a long list of major Internet platforms and services, e.g., Netflix, CNN, Airbnb, PayPal, Zillow, …
- this was a **Distributed Denial-of-Service (DDoS)** attack

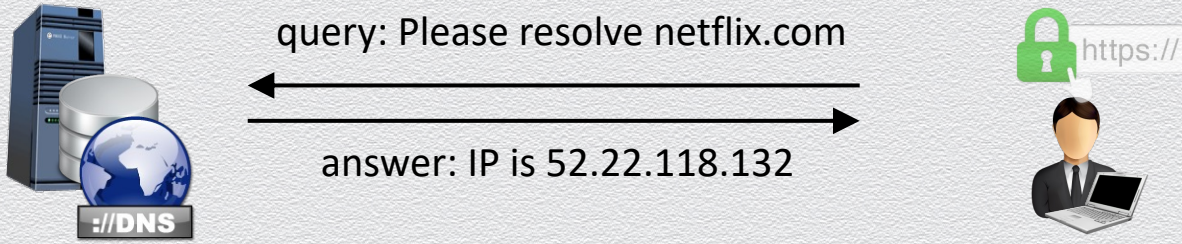




Domain Name Service (DNS) protocol

Resolving domain names to IP addresses

- when you type a URL in your Web browser, its IP address must be found
  - larger websites have multiple IP responses for redundancy to distributing load
- at the heart of Internet addressing is a protocol called DNS
  - a database translating Internet names to addresses



DNS: Hierarchical search

Search is performed recursively and hierarchically across different type of DNS resolvers

- Untrusted recursive DNS servers: query other resolvers and cache recent results
- Trusted TLD (top-level domain) servers: control TLD zones such as .com, .org, .net, etc.

DNS entries:
<netflix.com, 52.22.118.132>

primary

subset of cached queried entries
(or information of other resolvers)

secondary

netflix.com

52.22.118.132
(or “non-existent”)

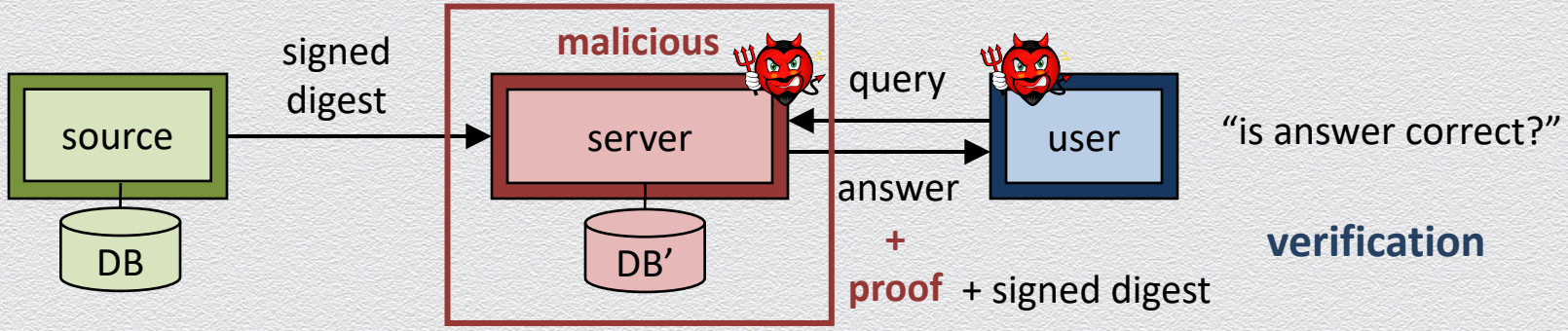
locally cached IP addresses
(at Web browser and OS)

DNS: A critical asset to attack…

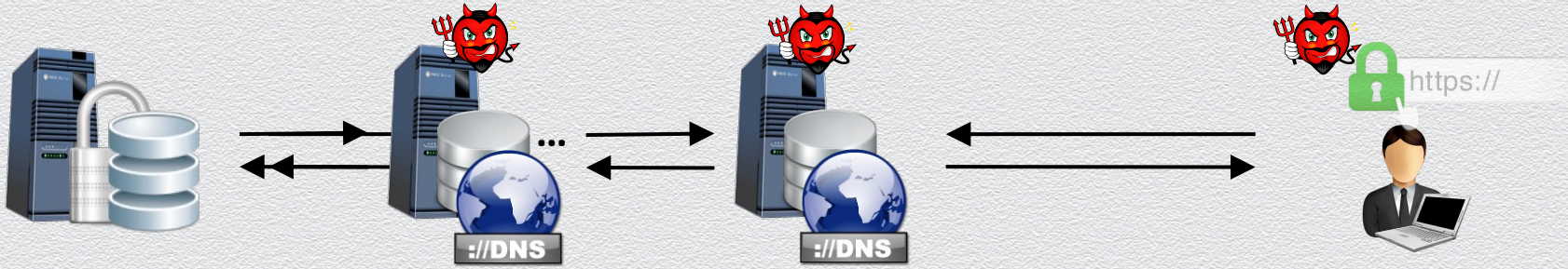
What main security properties must be preserved in such an important service?

- all properties in CIA triad are relevant!
- resolving domain names to IP addresses is a service that
  - must critically be available during all times – availability
  - must critically be trustworthy – integrity
  - must also protect database entries that are not queried – confidentiality

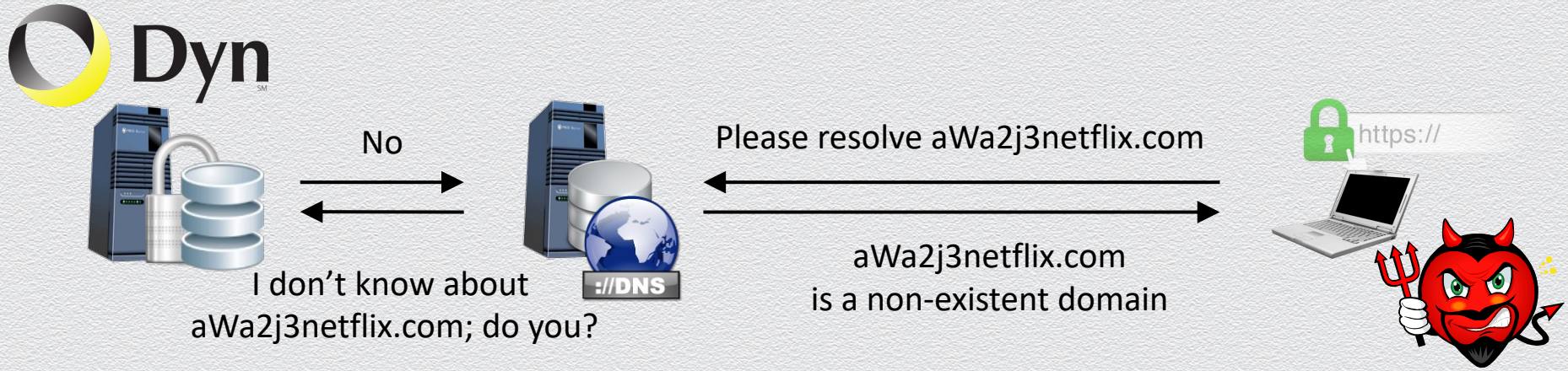
DNS: A critical asset to attack… (cont.)



integrity **availability / confidentiality**



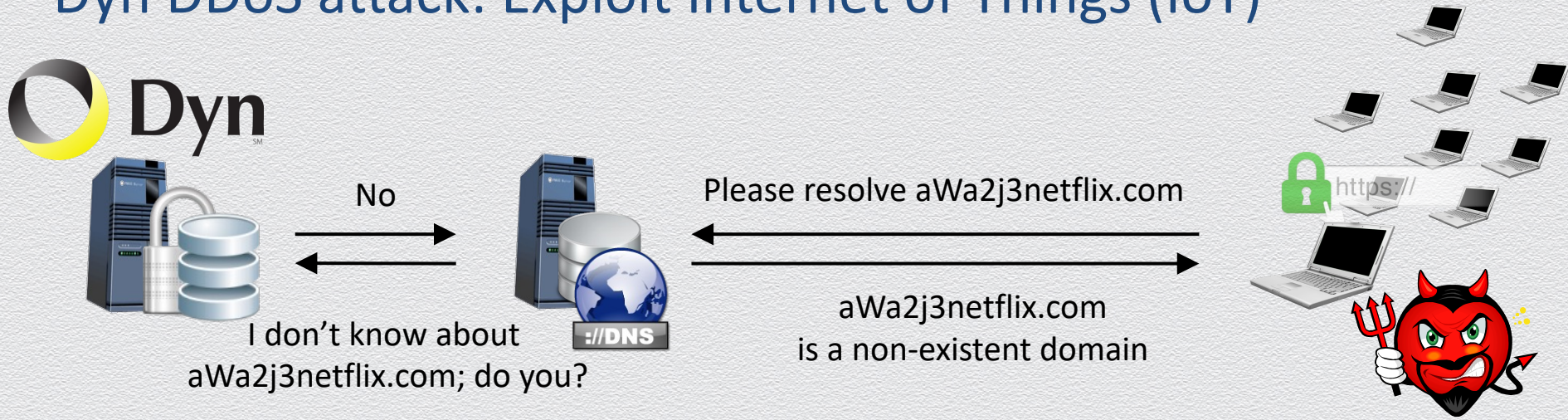
Dyn DDoS attack



Attack:

- from a compromised machine ask for domain names that do not exist
- query is forwarded to fewer primary Dyn servers, i.e., defeating benefits of distribution
- use a botnet to ask **A LOT** of such queries to bring down the Dyn DNS service!

Dyn DDoS attack: Exploit Internet of Things (IoT)



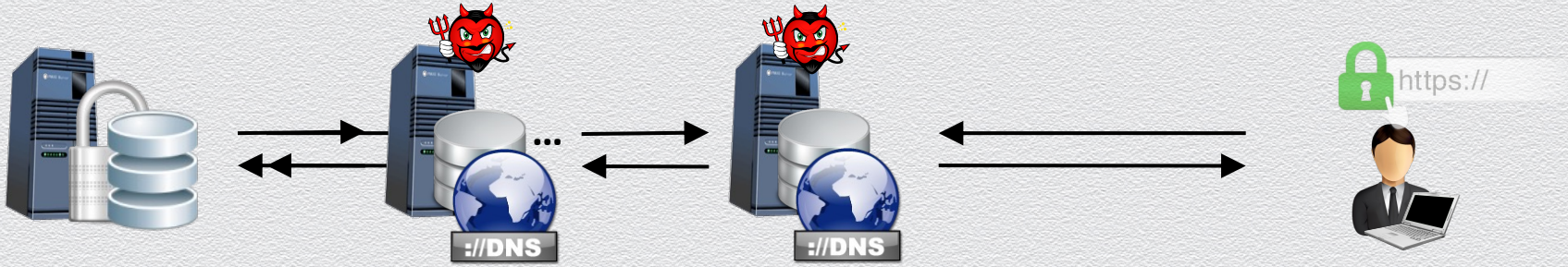
Create a botnet:

- compromise easy targets: IoT "thin" devices, e.g., printers, cameras, home routers, …
- how? find a vulnerability on these devices...
- all such devices used an OS with a static, hard-wired, thus known, admin password...!

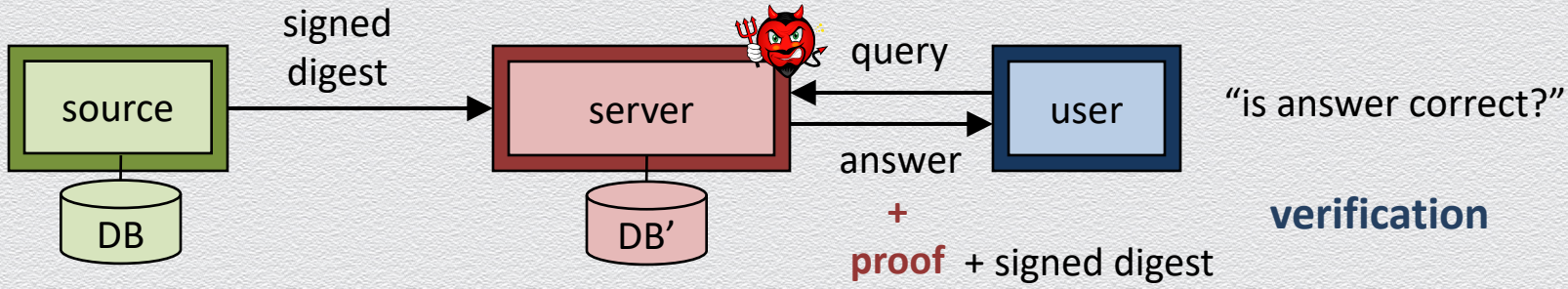
DNSSEC & NSEC

Security extensions of DNS protocol to protect integrity of DNS data

- correct resolution, origin authentication, authenticated denial of existence
- specifications made by Internet Engineering Task Force (IETF) via RFCs
  - an RFC (request for comments) is a suggested solution under peer review
- challenges: backward-compatible, simplicity, confidentiality, who signs
  - DNSSEC/NSEC: extension that provide proofs of existence/denial of existence



DNSSEC & NSEC: core idea



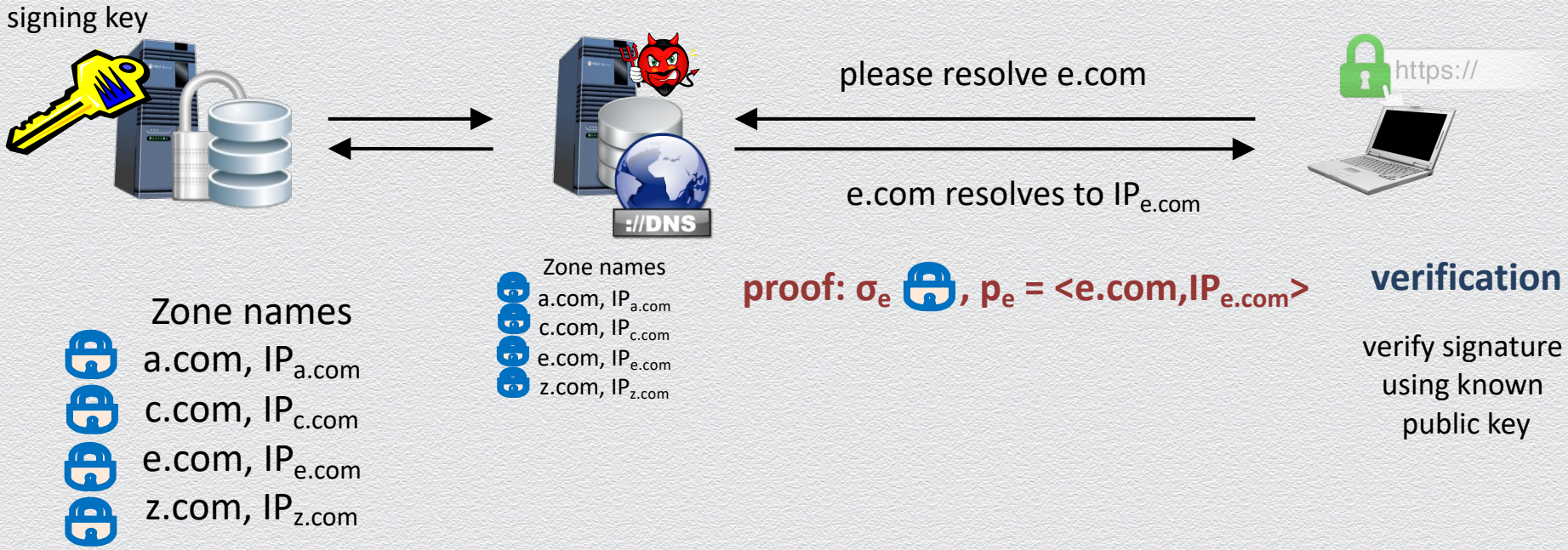
DNSSEC protocol: each DNS entry is pre-signed by primary name server

NSEC protocol:

- domain names are lexicographically ordered and then each pair of neighboring existing domain names is pre-signed by the primary name server
- non-existing names, e.g., aWa2j3netflix.com are proved by providing this pair "containing" missed query name, e.g., <awa.com, awb.com>

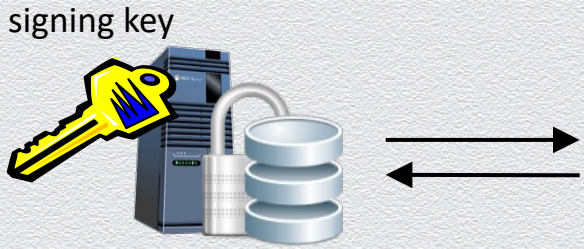
DNSSEC: example

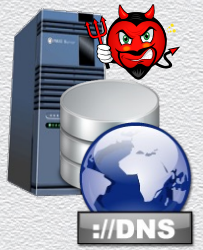
Each entry <domain name, IP address> in the database is individually signed by a primary DNS server and uploaded to secondary DNS servers in signed form



NSEC: example

Additionally, pairs of consecutive (in alphabetical order) domain names are individually signed by a primary DNS server and uploaded to secondary DNS servers in signed form





please resolve b.com

domain name b.com doesn't exist

Zone names

a.com
c.com
e.com
z.com
a.com

σ_1 σ_2 σ_3 σ_4

proof: σ_1, p_1 = <a.com, c.com>

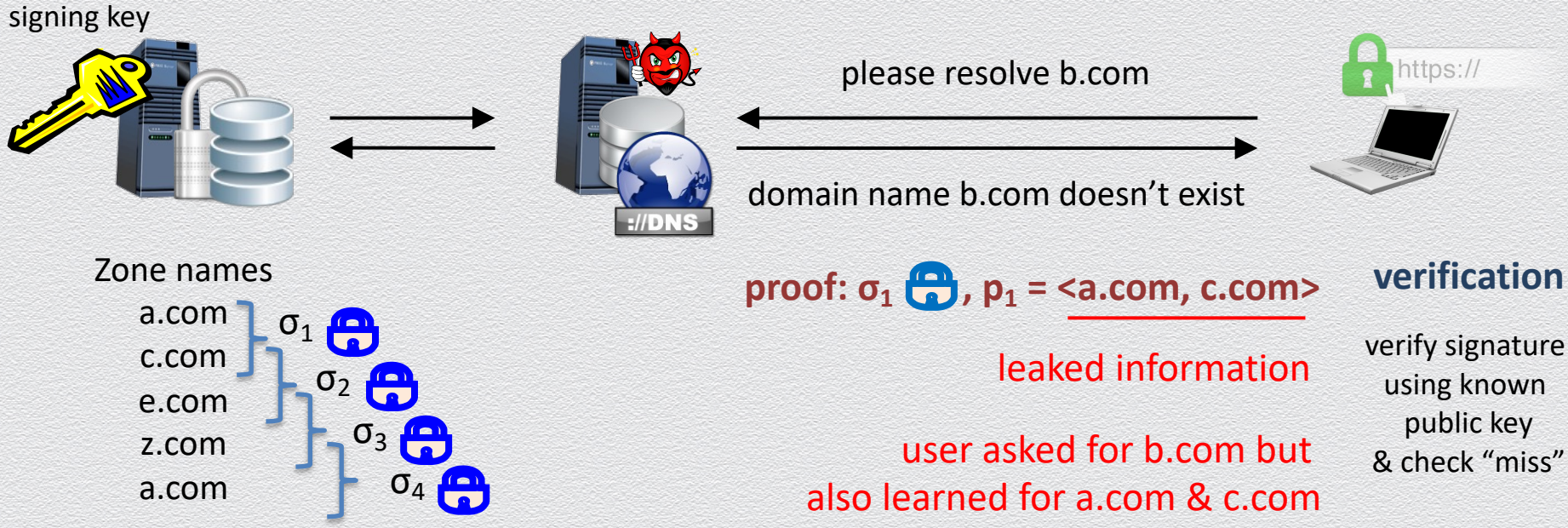
verification

verify signature using known public key & check "miss"

NSEC: Vulnerability

exploit the “leak-domain-names” vulnerability of NSEC to learn the domain names of an entire zone

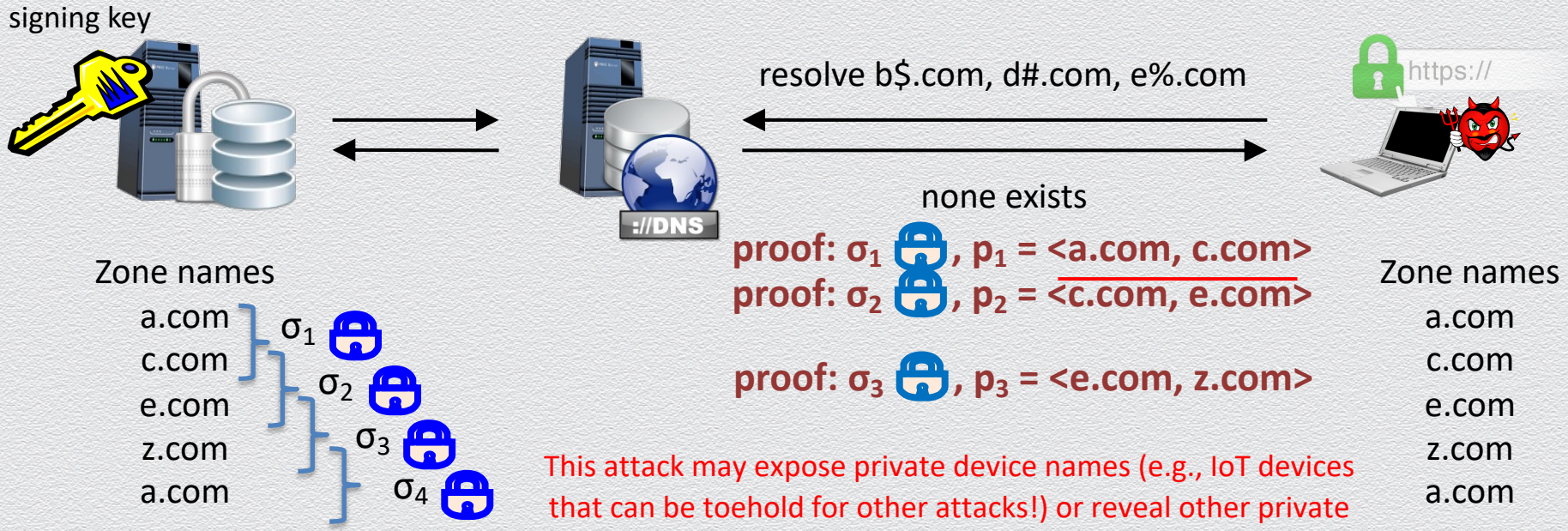
Proofs of non-existing names leak information about other unknown domain names



Zone enumeration attack

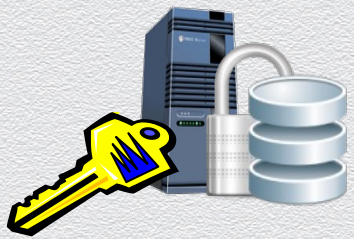
ask for non-existing names
to get all possible proofs

An attacker can simply act as a “querier” to learn target organization’s network structure!



This attack may expose private device names (e.g., IoT devices that can be toehold for other attacks!) or reveal other private DNS data that many registries have legal obligations to protect

NSEC3: NSEC in the hash domain



please resolve b.com

b.com is a non-existent domain

proof: σ_3 , p_3 = <dde45,zrit5>

Zone names

a.com
c.com
e.com
z.com

hash h →

a1bb5
23ced
zrit5
dde45

sort →

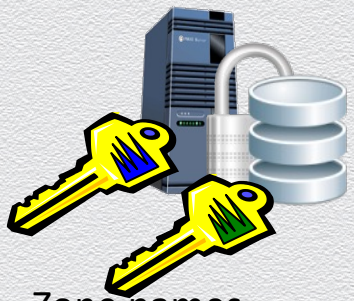
23ced
a1bb5
dde45
zrit5
23ced

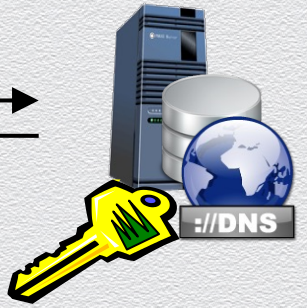
σ_1
σ_2
σ_3
σ_4

asked for b.com but
learned h(e.com) & h(z.com)

h(b.com) = ntwo4
e.g., h is SHA-256

NSEC5: A secure solution





please resolve b.com

b.com is a non-existent domain

proof: σ_3 , p_3 = <dde45,zrit5>

h'(x), RSA-signature of f(b.com)

Zone names

a.com
c.com
e.com
z.com

hash h' →

a1bb5
23ced
zrit5
dde45

sort →

23ced
a1bb5
dde45
zrit5
23ced

σ_1
σ_2
σ_3
σ_4

asked for b.com but
learned h'(e.com) & h'(z.com)

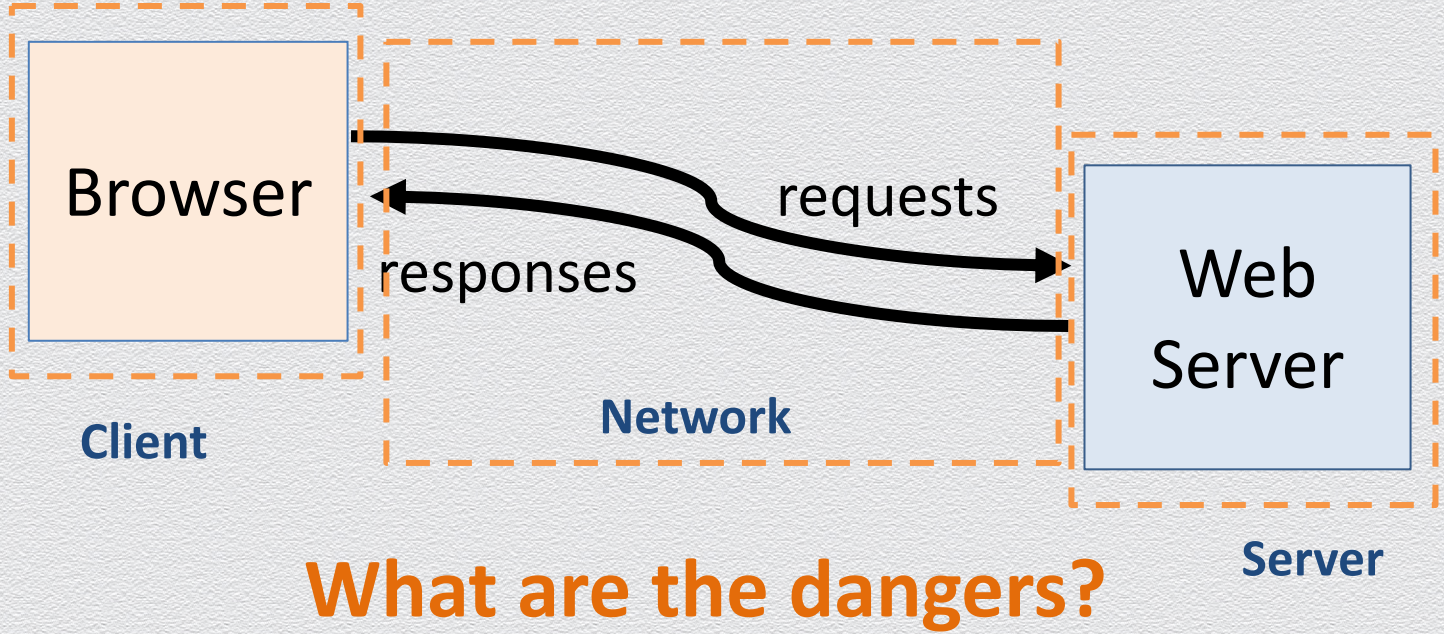
h'(b.com) = ntwo4

h: as in NSEC3
f: "message transformation" hash

h'(x) = h(RSA-Sign( , f(x)))

Web security model

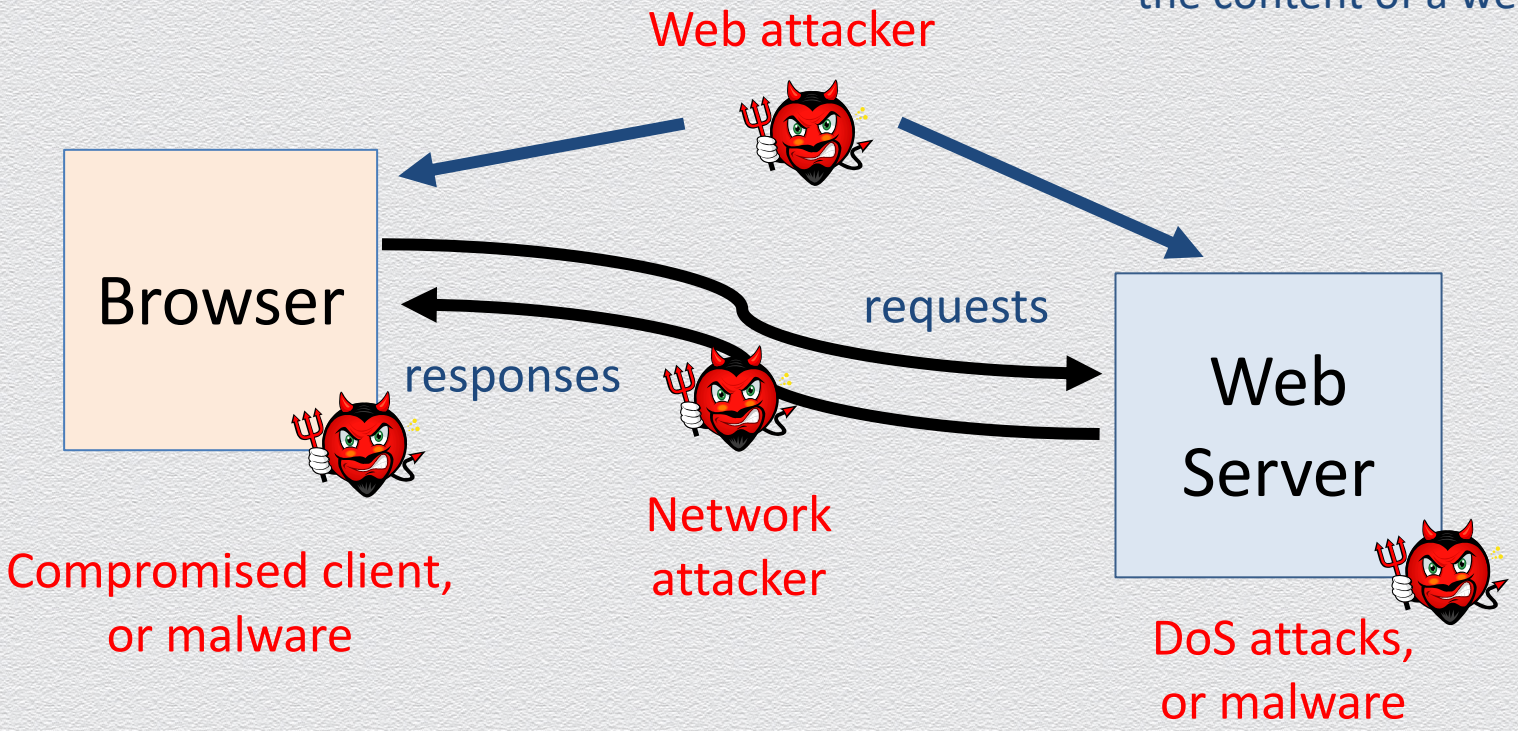
Web applications



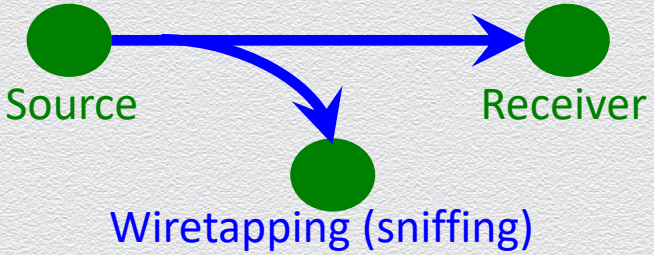
What are the dangers?

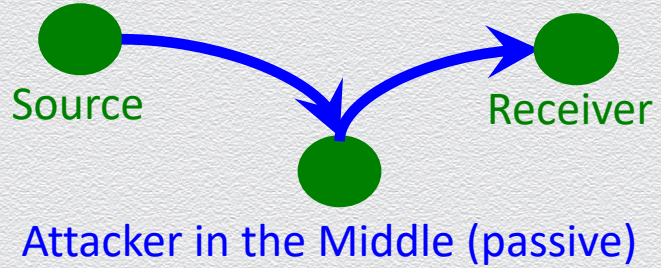
Threat models

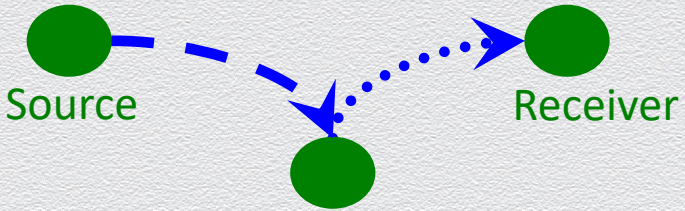
The main vector of attack is via the content of a website

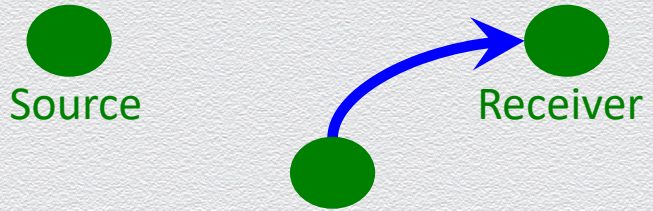


Network attacks









Web Attacker Capabilities

- Attacker controls a malicious website
  - website might look professional, legitimate, etc.
  - attacker can get users to visit website (how?)
- A benign website is compromised by attacker
  - attacker inserts malicious content into website
  - attacker steals sensitive data from website
- Attacker does not have direct access to user's machine

Potential Damage

- An attacker gets you to visit a malicious website…
  - Can they perform actions on other websites impersonating you?
  - Can they run evil code on your OS?
- Ideally, none of these exploits are possible ...

Attack Vectors

- Web browser (focus of this lecture)
  - Renders web content (HTML pages, scripts)
  - Responsible for confining web content
  - **Note:** Browser implementations dictate what websites can do
- Web applications
  - Server code (PHP, Ruby, Python, …)
  - Client-side code (JavaScript)
  - Many potential bugs (e.g., see Project 2)

Browser Security: Sandbox

Goal: protect local computer from web attacker

- Safely execute code on a website, without the code
  - accessing your files, tampering with your network, or accessing other sites

High stakes

- $40K bounty for Google Chrome
  - www.google.com/about/appsecurity/chrome-rewards/

We won't address attacks that break the sandbox

- But they happen check the CVE list
  - https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=sandbox
  - https://support.apple.com/en-us/HT213635

Domains, HTML, HTTP

URL and FQDN

URL: Uniform Resource Locator

https://cs.brown.edu/about/contacts.html

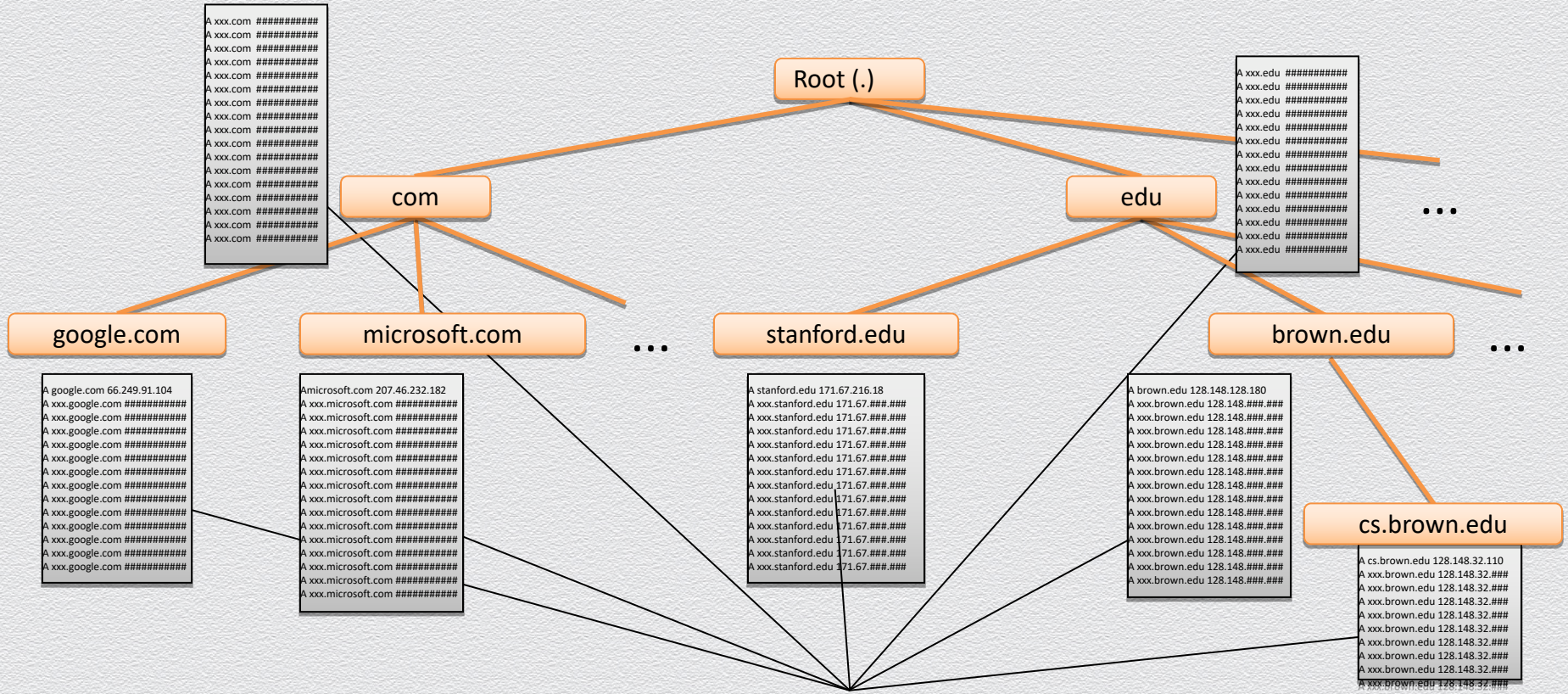
- a protocol
  - e.g. https
- a FQDN
  - e.g. cs.brown.edu
- a path and file name
  - e.g. /about/contacts.html

FQDN: Fully Qualified Domain Name

[Host name].[Domain].[TLD].[Root]

- Two or more labels, separated by dots
  - e.g., cs.brown.edu
- Root name server
  - a "." at the end of the FQDN
- Top-level domain (TLD)
  - generic (gTLD): .com, .org, .net,
  - country-code (ccTLD): .ca, .it, , .gr …

Domain hierarchy



resource records

HTML

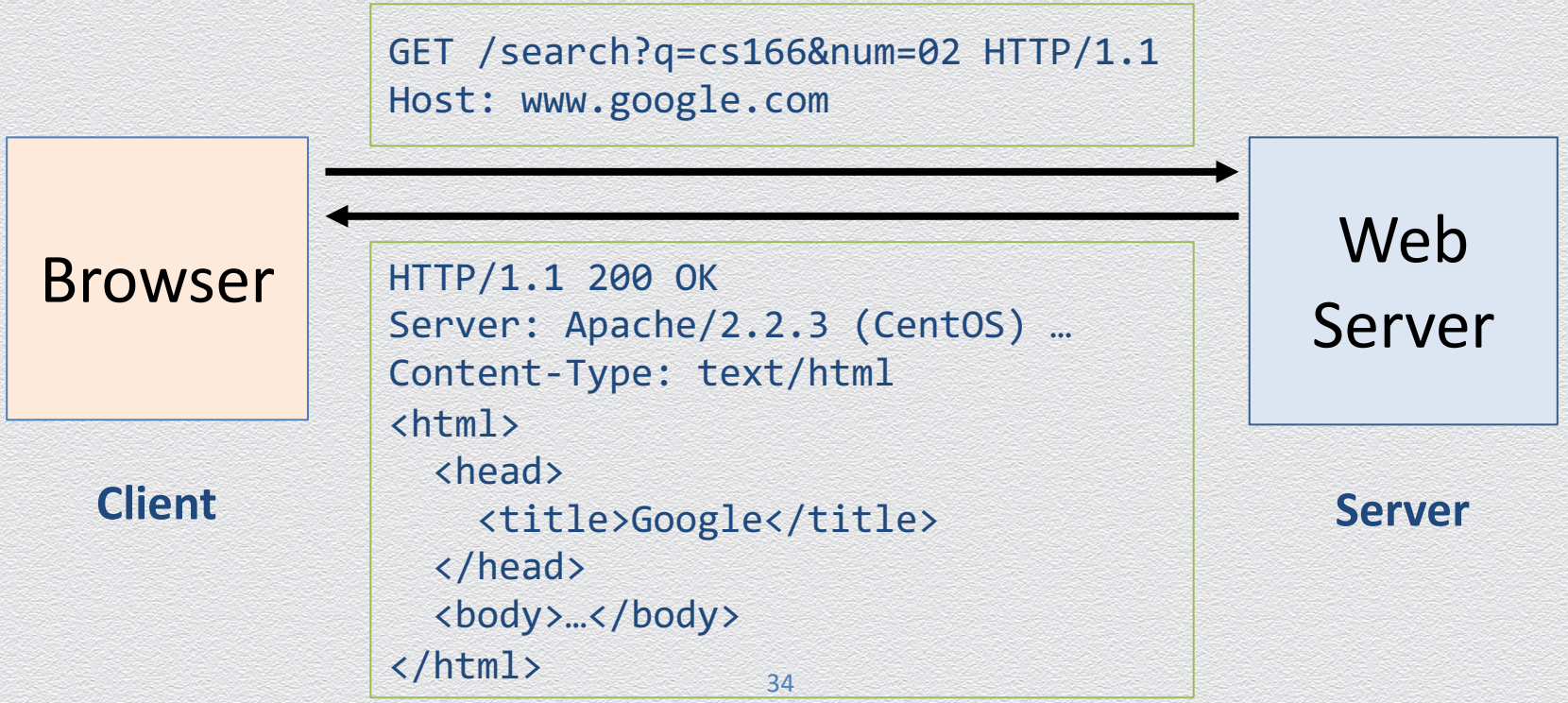
Hypertext markup language (HTML)

- allows linking to other pages (href)
- supports embedding of images, scripts, other pages (script, iframe)
- user input accepted in forms

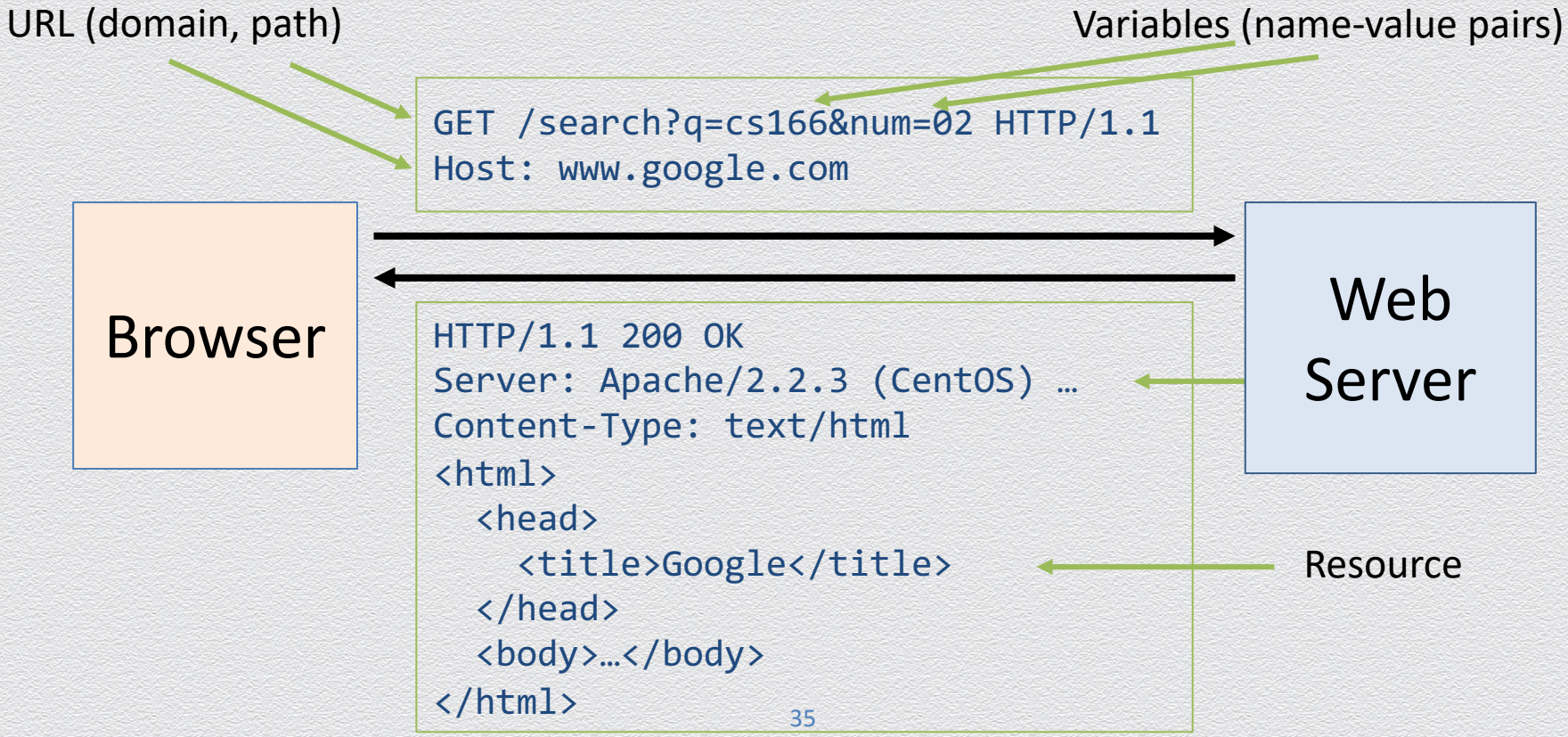
```
<html>
  <head>
    <title>Google</title>
  </head>
  <body>
    <p>Welcome to my page.</p>
    <script>alert(“Hello world”);
    </script>
    <iframe src=“http://example.com”>
    </iframe
  </body>
</html>
```

HTTP (Hypertext Transport Protocol)

Communication protocol between client and server



What's in a request (or response)?



Variables

Key-value pairs obtained from user input into forms & submitted to server

- Submit variables in HTTP via GET or POST
- GET request: variables within HTTP URL
  - e.g., http://www.google.com/search?q=cs166&num=02
- POST request: variables within HTTP body
  - POST / HTTP/1.1
  - Host: example.com
  - Content-Type: application/x-www-form-urlencoded
  - Content-Length: 18
  - month=5&year=2024

Semantics: GET Vs. POST

GET

- Request target resource
- Read-only method
- Submitted variables may specify target resource and/or its format

POST

- Request processing of target resource
- Read/write/create method
- Submitted variables may specify how resource is processed
  - e.g., content of resource to be created, updated, or executed

GET Vs. POST

	GET	POST
Browser history	✓	X
Browser bookmarking	✓	X
Browser caching	✓	X
Server logs	✓	X
Reloading page	immediate	warning
Variable values	Restricted	arbitrary

Web-application security

Client-side controls

- Web security problems arises because clients can submit arbitrary input
- What about using client-side controls to check the input?
- Which kind of controls?

Client-side controls (cont.)

A standard application may rely on client-side controls

- They restrict user input in two general ways
  - Transmitting data via the client component using a mechanism that should prevent the user from modifying that data
  - Implementing measures on the client side

- In this threat model
  - Server does not trust the Client

Bypassing client-side controls

- In general, a security flaw because it is easy to bypass
- The user
  - has a full control over the client and the data it submits
  - can bypass any controls that are client-side and not replicated on the server
- Why these controls are still useful?
  - For load balancing or usability
  - Often we can suppose that the vast majority of users are honest

Transmitting data via the client

- A common developer bad habit is passing data to the client in a form that the end user cannot directly see or modify
- Why is it so common?
  - It removes or reduces the amount of data to store server side per-session
  - In multi-server applications, it removes the need to synchronize the session data among different servers
  - The use of third-party components on the server may be difficult or impossible to integrate
- Transmitting data via the client is often the easy solution
  - But unfortunately it is not secure

Common mechanisms

- HTML Hidden fields
  - A field flagged hidden is not displayed on-screen
- HTTP Cookies
  - Not displayed on-screen, and the user cannot modify directly
- Referrer Header
  - An optional field in the http request that it indicates the URL of the page from which the current request originated
- If you use the proper tool you can tamper the data on the client-side

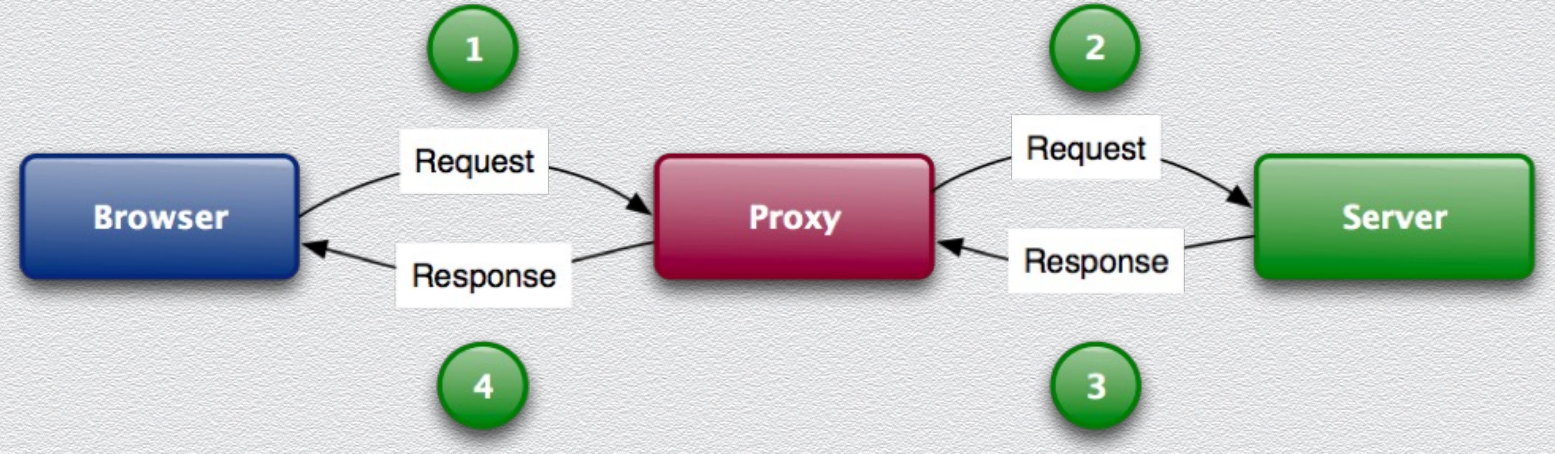
Web client tool

- Web inspection tool:
  - Firefox or Chrome web developer:
    - powerful tools that allow you to edit HTML, CSS and view the coding behind any website: CSS, HTML, DOM and JavaScript
- Web Proxy:
  - Burp, OWASP ZAP, etc.
    - Allow to modify GET or POST requests

HTTP proxy

An intercepting Proxy:

- inspect and modify traffic between your browser and the target application
  - Burp Intruder, OWASP ZAP, etc.



Browser security

In BROWSER we trust…

- Most of our trust on web security relies on information stored in the Browser
  - a Browser should be updated since Bugs in the browser implementation can lead to various attacks
  - e.g., https://us-cert.cisa.gov/ncas/current-activity/2023/02/14/mozilla-releases-security-updates-firefox-110-and-firefox-esr
- Add-ons too are dangerous
  - Hacking Team flash exploits - goo.gl/syVwiD
  - github.com/greatsuspender/thegreatsuspender/issues/1263
- Executing a browser with low privileges helps

Browser Security: Same-Origin Policy (SOP)

Very simple idea: “Content from different origins should be isolated”

- Website origin defined over tuple (protocol, domain, port)

Very difficult to execute in practice…

- Messy number of cases to worry about…

  HTML elements, Navigating Links, Browser cookies, JavaScript capabilities, iframes, … etc.

- Browsers didn’t always get this correct…

Browser Security: Same-Origin Policy (SPO) (cont.)

Goal: Protect and isolate web content from other web content

- Content from different origins should be isolated, e.g., mal.com should not interact with bank.com in unexpected ways
- What about cs.brown.edu vs brown.edu or mail.google.com vs drive.google.com?
- Lots of subtleties

SOP example: http://store.company.com/dir/page.html

(protocol, domain, port)

URL	Outcome	Reason
`http://store.company.com/dir2/other.html`	Same origin	Only the path differs
`http://store.company.com/dir/inner/another.html`	Same origin	Only the path differs
`https://store.company.com/page.html`	Failure	Different protocol
`http://store.company.com:81/dir/page.html`	Failure	Different port (`http://` is port 80 by default)
`http://news.company.com/dir/page.html`	Failure	Different host