CSE 127: Intro to Computer Security

WI24

Lecture 18 - Cybersecurity History

Slide Credit: Kirill Levchenko, Stefan Savage, Deian Stefan, Nadia Heninger, Earlence Fernandes, Imani Munyaka

Announcements

HW/PA 5 out

Discussion Friday @ 2pm

Lecture outline

CS History

TLS

Same Origin Policy

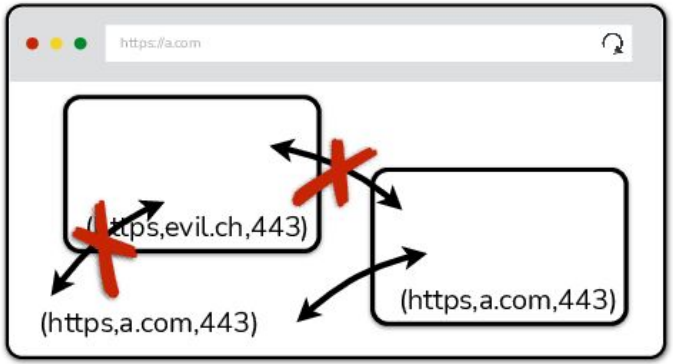
DNS


the grio
KEEPING
BLACK
HISTORY
ALIVE


RSA
Conference
2018

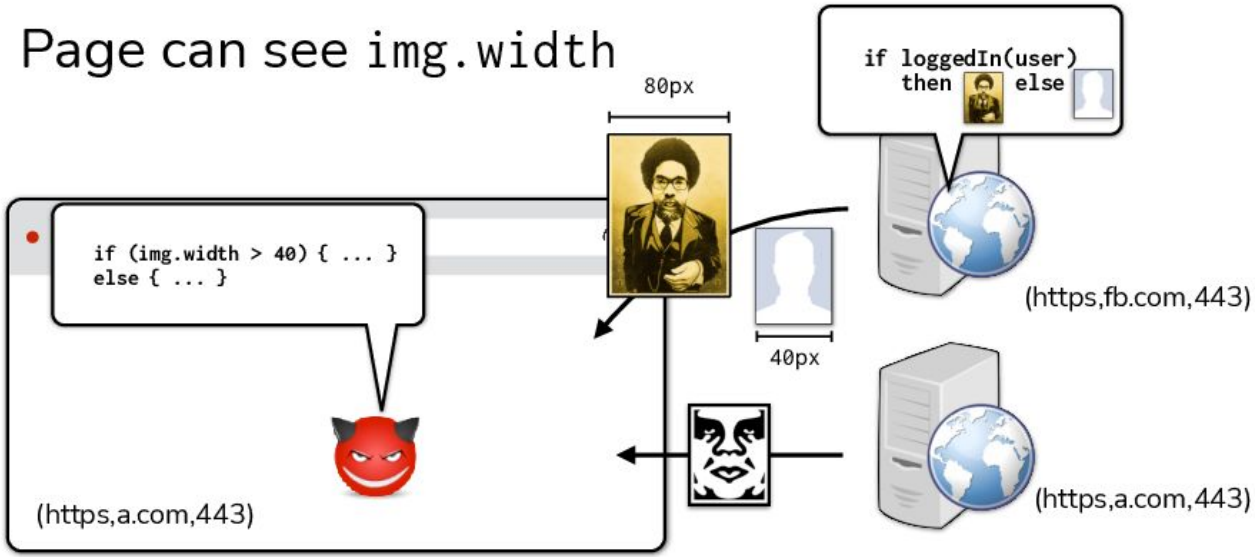
SOP for the DOM

- Each frame in a window has its own origin
- Frame can only access data with the same origin
  - DOM tree, local storage, cookies, etc.



Images

- Browser renders cross-origin images, but SOP prevents page from inspecting individual pixels
- Page can see `img.width`



SOP for fonts and CSS are similar.

Application layer threats: DNS spoofing

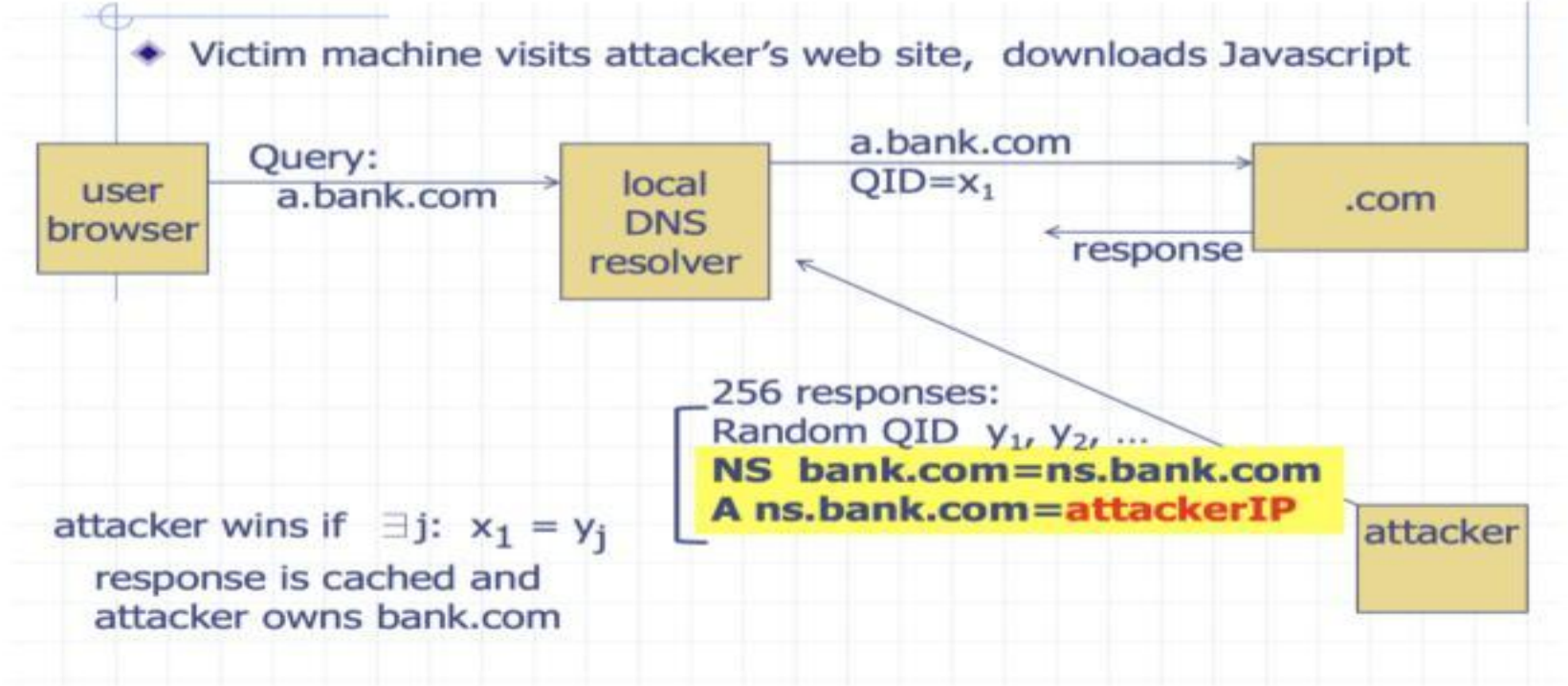
Recall:

- DNS maps between domain names and IP addresses.
- Responses cached to avoid query times.

DNS Threat Models:

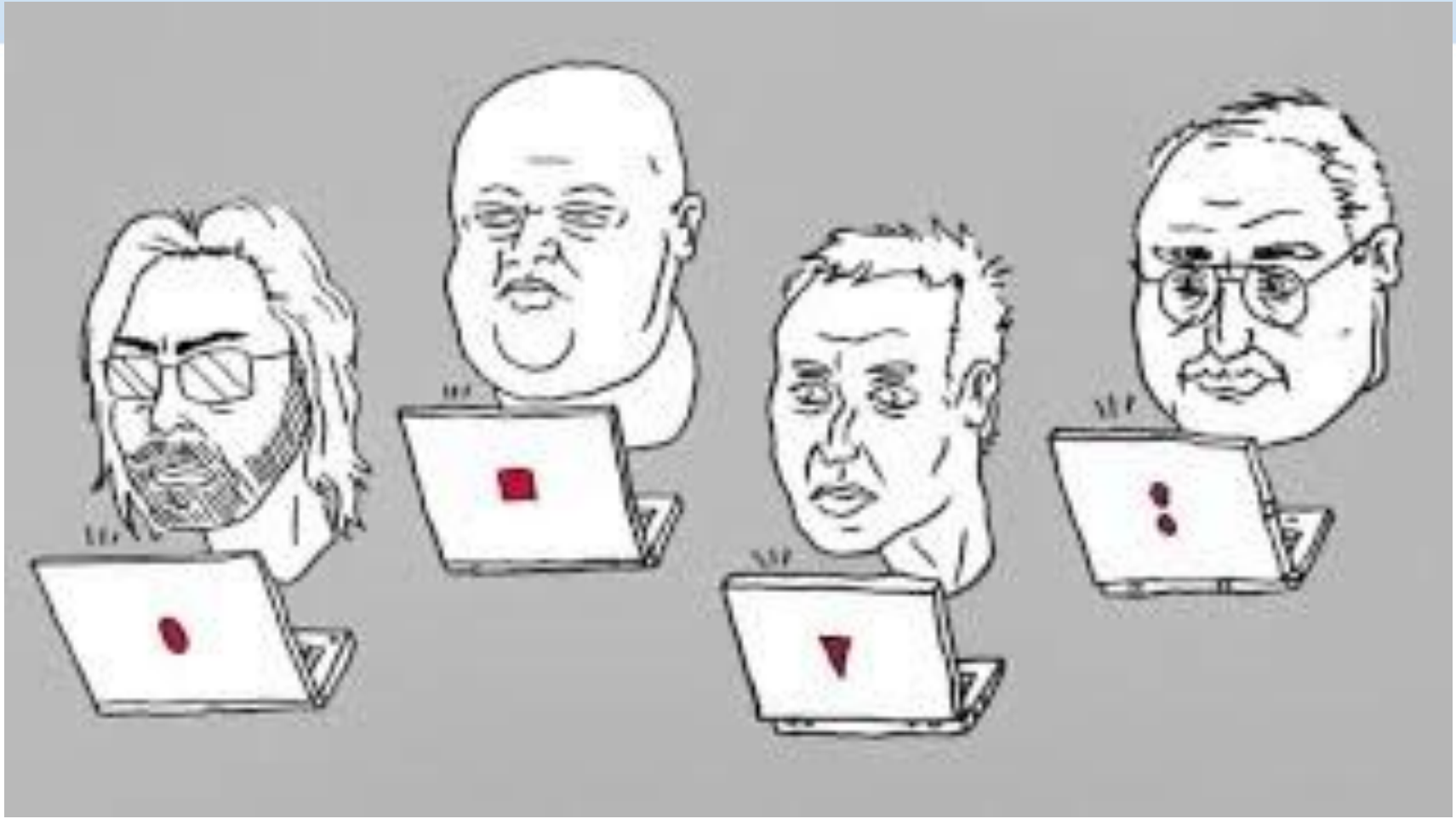
- **Malicious DNS server:** Any DNS server in query chain can lie about responses.
- **Local/on-path attacker**: Can impersonate DNS server and send a fake response.
- **Off-path attacker:** Can try to forge response: needs to match 16-bit query ID.
    - Original spec: query ID increments with each request.
    - How can you attack this?

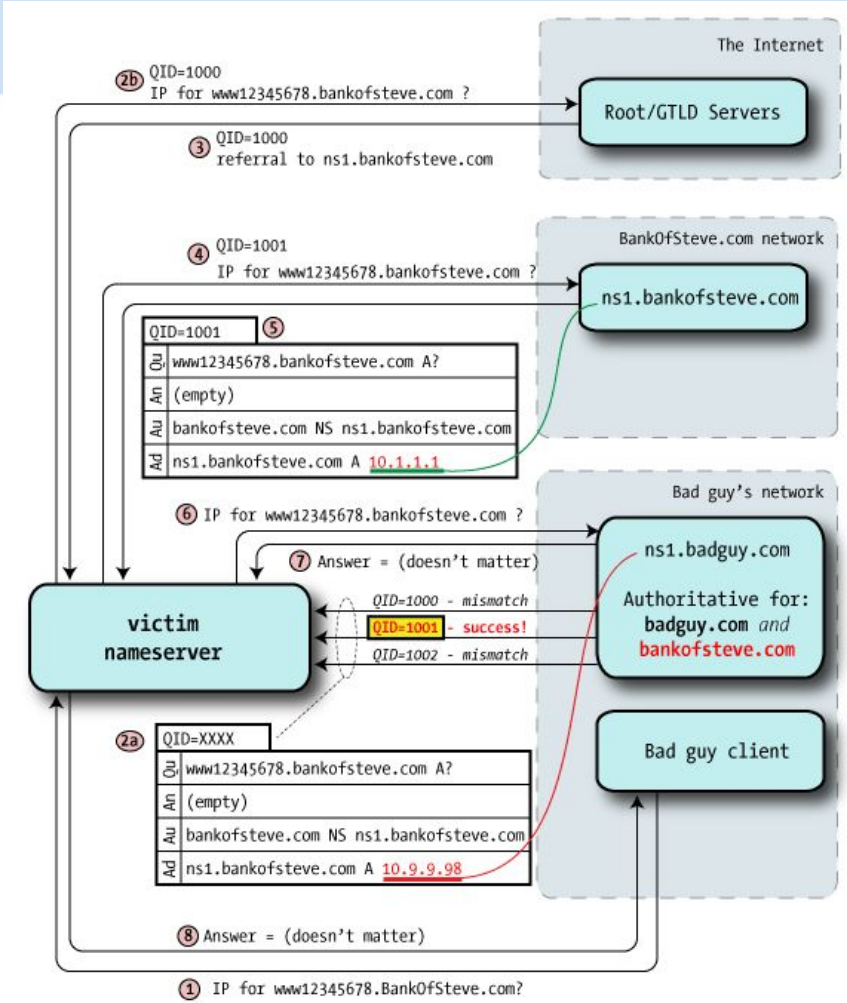
DNS spoofing: 2008 Kaminsky attack



Birthday bound: attacker expects to succeed after $2^8 = 256$ lookups

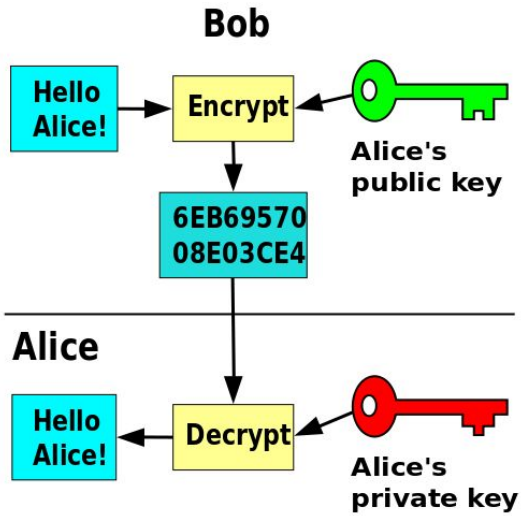
Mitigation: randomize source port



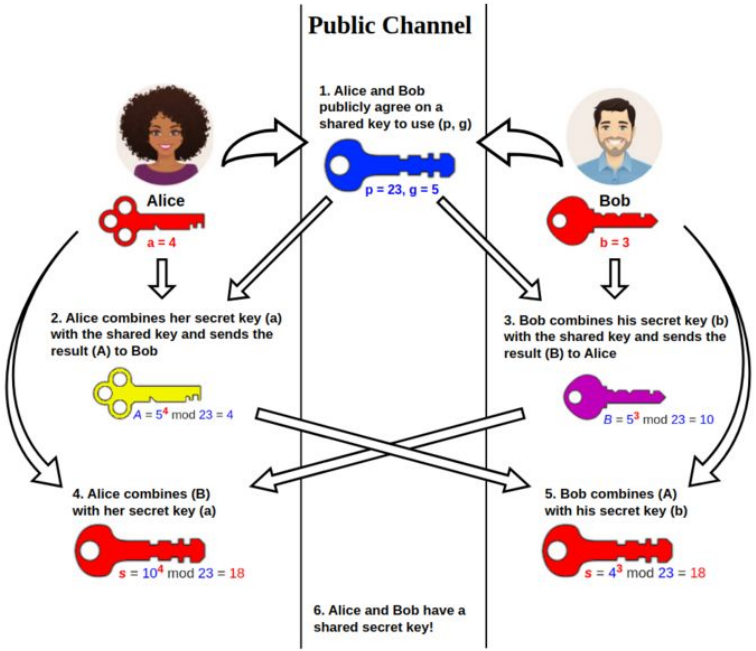

The Internet
Root/GTLD Servers
2b QID=1000
IP for www12345678.bankofsteve.com ?
3 QID=1000
referral to ns1.bankofsteve.com
BankOfSteve.com network
ns1.bankofsteve.com
4 QID=1001
IP for www12345678.bankofsteve.com ?
QID=1001
5
Qu www12345678.bankofsteve.com A?
An (empty)
Au bankofsteve.com NS ns1.bankofsteve.com
Ad ns1.bankofsteve.com A 10.1.1.1
Bad guy's network
6 IP for www12345678.bankofsteve.com ?
7 Answer = (doesn't matter)
ns1.badguy.com
Authoritative for:
badguy.com and
bankofsteve.com
victim
nameserver
QID=1000 - mismatch
QID=1001 - success!
QID=1002 - mismatch
Bad guy client
2a QID=XXXX
Qu www12345678.bankofsteve.com A?
An (empty)
Au bankofsteve.com NS ns1.bankofsteve.com
Ad ns1.bankofsteve.com A 10.9.9.98
8 Answer = (doesn't matter)
1 IP for www12345678.BankOfSteve.com?

TLS Review

TLS - Transport Layer Security



RSA
wikimedia.org



Diffie Hellman
wikipedia.org-

TLS 1.2 with Diffie-Hellman Key Exchange

Step 8: The client and server can now send encrypted application data (e.g. HTTP) using their secure channel.

client hello: client random

[list of cipher suites]

server hello: server random, [cipher suite]

certificate = public RSA key + CA signatures

server kex: p, g, g^a, $Sign_{RSAkey}(p, g, g^a)$

client kex: g^b

client finished: MAC_k (dialog)

server finished: MAC_{kms} (dialog)

Enc_{ke} (request)

KDF(g^{ab}, random) → k_{m_c}, k_{m_s} , k_e

KDF(g^{ab}, random) → k_{m_c} , k_{m_s} , k_e

TLS 1.2 with RSA Key Exchange

TLS versions prior to 1.3 also supported using RSA public key encryption to share the premaster secret (shared secret master key).

client hello: client random

[list of cipher suites]

server hello: server random, [RSA cipher suite]

certificate = **RSA pubkey** k_{2048} + CA signatures

client key exchange: $RSAenc_{k^{2048}}(pms)$

client finished: $Auth_{k^{mc}}(dialog)$

server finished: $Auth_{k^{ms}}(dialog)$

Enc_{ke} (request)

KDF(pms, random) → k_{mc} , k_{ms} , k_e

KDF(pms, random) → k_{mc} , k_{ms} , k_e

TLS 1.2 with Diffie-Hellman Key Exchange

Step 8: The client and server can now send encrypted application data (e.g. HTTP) using their secure channel.

server kex: p, g, g^a, $Sign_{RSAkey}(p, g, g^a)$

client kex: g^b

KDF(g^{ab}, random) → k_{m_c}, k_{m_s}, k_e

KDF(g^{ab}, random) → k_{m_c}, k_{m_s}, k_e

TLS 1.2 with RSA Key Exchange

TLS versions prior to 1.3 also supported using RSA public key encryption to share the premaster secret (shared secret master key).

certificate = **RSA pubkey** k_{2048} + CA signatures

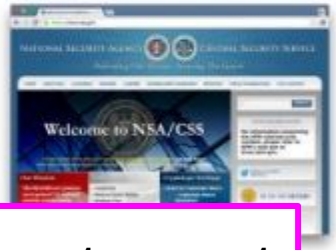
client key exchange: $RSAenc_{k^{2048}}(pms)$

KDF(pms, random) → k_{m_c} , k_{m_s} , k_e

KDF(pms, random) → k_{m_c} , k_{m_s} , k_e

TLS 1.2 with RSA Key Exchange

TLS versions prior to 1.3 also supported using RSA public key encryption to share the premaster secret (shared secret master key).



certificate = **RSA pubkey** k_{2048} + CA signatures

client key exchange: $RSAenc_{k^{2048}}(pms)$

Pms = premaster secret

KDF(pms, random) → k_{m_c} , k_{m_s} , k_e

KDF(pms, random) → k_{m_c} , k_{m_s} , k_e

How TLS achieves its security goals

What happens if a passive eavesdropper watches all the traffic?

- The application-layer traffic is encrypted, and Diffie-Hellman and RSA are secure against a passive eavesdropper so the attacker cannot discover the keys.

- The eavesdropper can see all the IP and TCP-layer packet headers.

- The eavesdropper can also see the initial handshake and metadata (which includes the server certificate)

How TLS achieves its security goals

What happens if an active attacker tries to man-in-the-middle the connection?

- For Diffie-Hellman, the key exchange is digitally signed by the private key corresponding to the public key in the server's certificate and the attacker doesn't know the server's key, so they cannot forge the signature. The client will not accept the key exchange.

- For RSA, the attacker does not know the private key corresponding to the public key in the server's certificate, so cannot learn the client's choice of premaster secret to learn the session keys.

How TLS achieves its security goals

What happens if a network attacker tries to impersonate the server?

• For Diffie-Hellman, the attacker does not know the private key corresponding to the public key in the server's certificate, so they cannot generate a valid signature on their Diffie-Hellman key exchange that will be accepted by the client.

• For RSA the attacker does not know the server's private key so cannot decrypt the client's encrypted premaster secret message.

What if a private key gets stolen or compromised?

If an adversary obtains a server certificate private key:

- With Diffie-Hellman key exchange, the adversary can:
  - actively man-in-the-middle a connection.
  - impersonate the server to anyone.
- With RSA key exchange, the adversary can:
  - impersonate the server to anyone.
  - decrypt any traffic from now and any point in the past.