CS 4530 & CS 5500
Software Engineering

Lecture 9.3: Software Engineering & Security Threats

Jonathan Bell, John Boyland, Mitch Wand
Khoury College of Computer Sciences

Learning Objectives for this Lesson

By the end of this lesson, you should be able to…

- Describe that security is a spectrum, and be able to define a realistic threat model for a given system
- Evaluate the tradeoffs between security and costs in software engineering

What does it mean for a system to be secure?

CIA: An overview of security properties

- Confidentiality: is information disclosed to unauthorized individuals?
- Integrity: is code or data tampered with?
- Availability: is the system accessible and usable?

Security isn't (always) free

In software, as in the real world…

- You just moved to a new house, someone just moved out of it. What do you do to protect your belongings/property?
- Do you change the locks?
- Do you buy security cameras?
- Do you hire a security guard?
- Do you even bother locking the door?

Security: Managing Risk

- Security architecture is a set of mechanisms and policies that we build into our system to mitigate risks from threats
- Threat: potential event that could compromise a security requirement
- Attack: realization of a threat
- Vulnerability: a characteristic or flaw in system design or implementation, or in the security procedures, that, if exploited, could result in a security compromise

Costs & Benefits

- Increasing security might:
  - Increase development & maintenance cost
  - Increase infrastructure requirements
  - Degrade performance
- But, if we are attacked, increasing security might also:
  - Decrease financial and intangible losses
- So: How likely do we think we are to be attacked in way **X**?

Threat Models

- What is being defended?
    - What resources are important to defend?
    - What malicious actors exist and what attacks might they employ?
- Who do we trust?
    - What entities or parts of system can be considered secure and trusted
    - Have to trust **something**!
    - Never trust remote users (especially remote users!)

Example: Client/server application

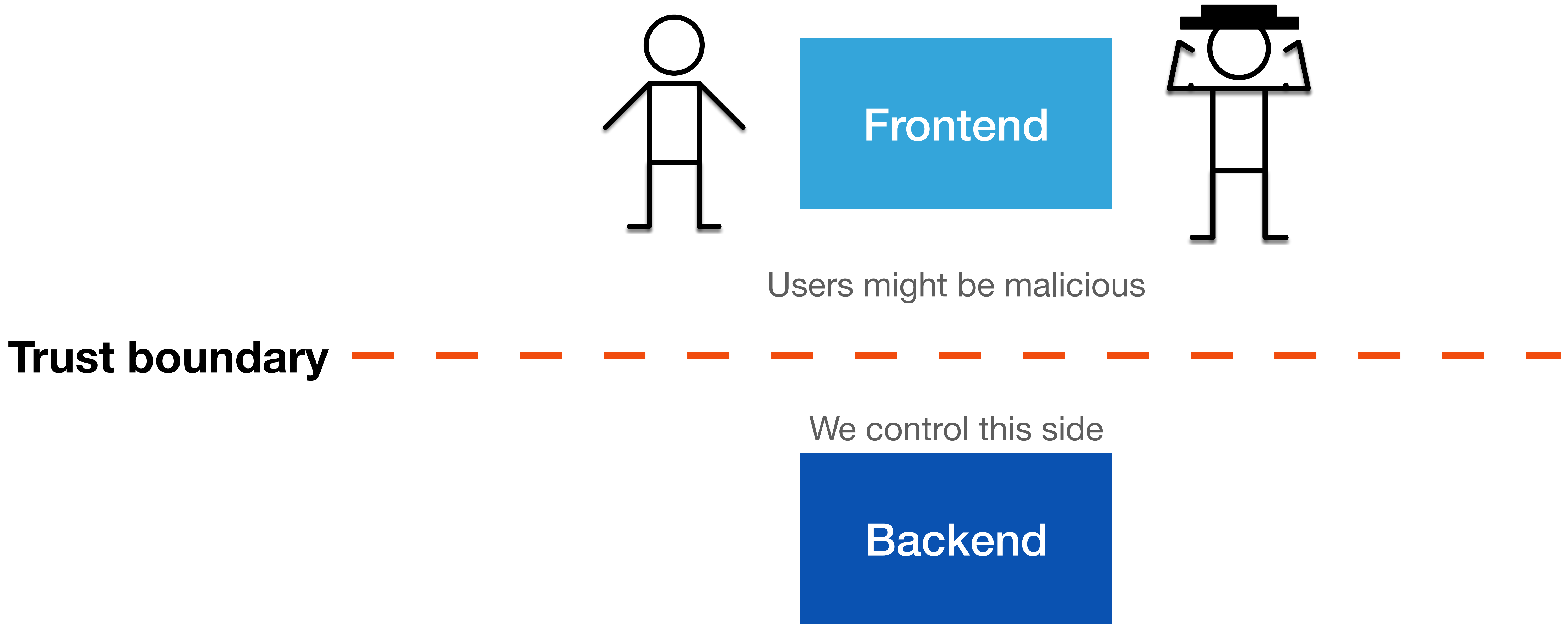
Authentication

```
function checkPassword(inputPassword: string){
  if(inputPassword === 'letmein'){
    return true;
  }
  return false;
}
```

Should this go in our frontend code?

Example: Client/server application

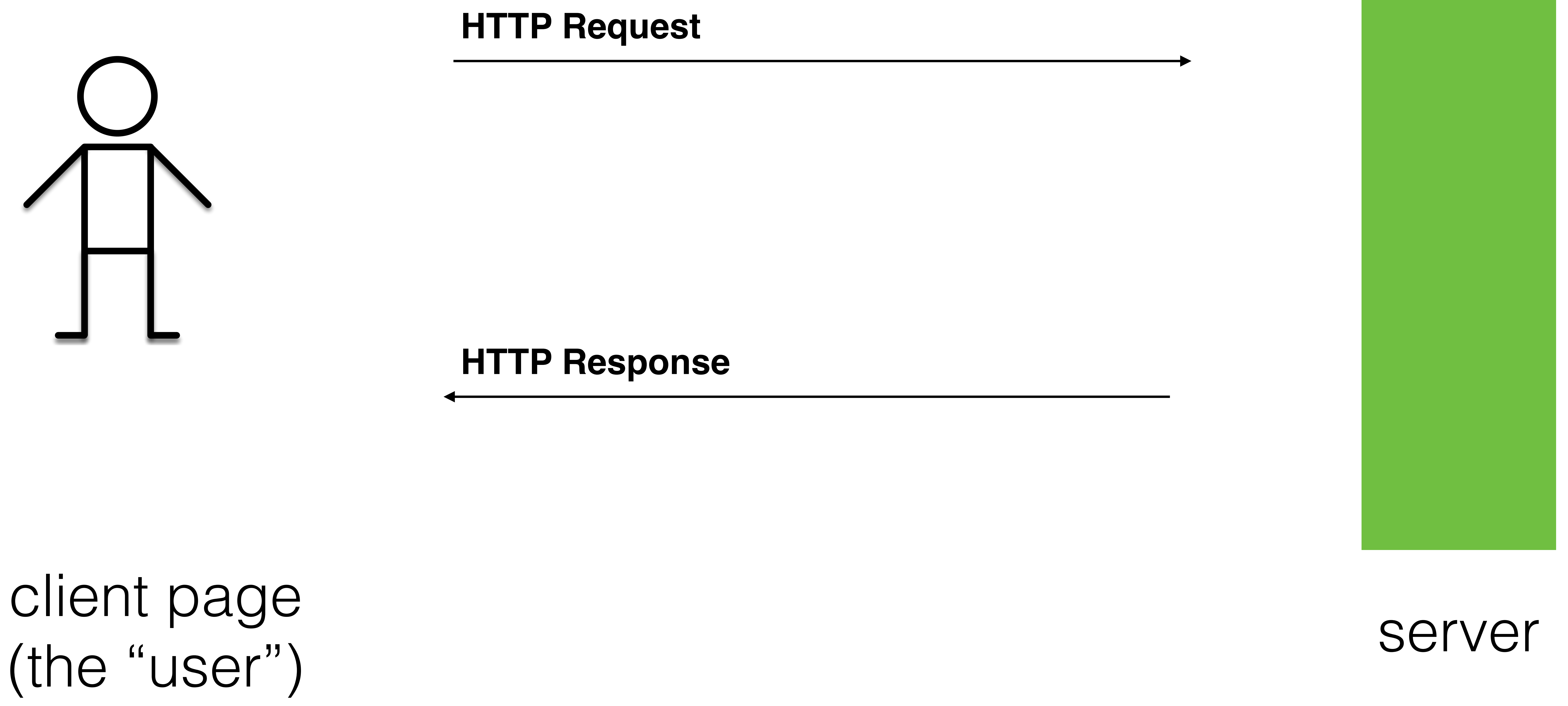
Authentication



```
function checkPassword(inputPassword: string){
  if(inputPassword === 'letmein'){
    return true;
  }
  return false;
}
```

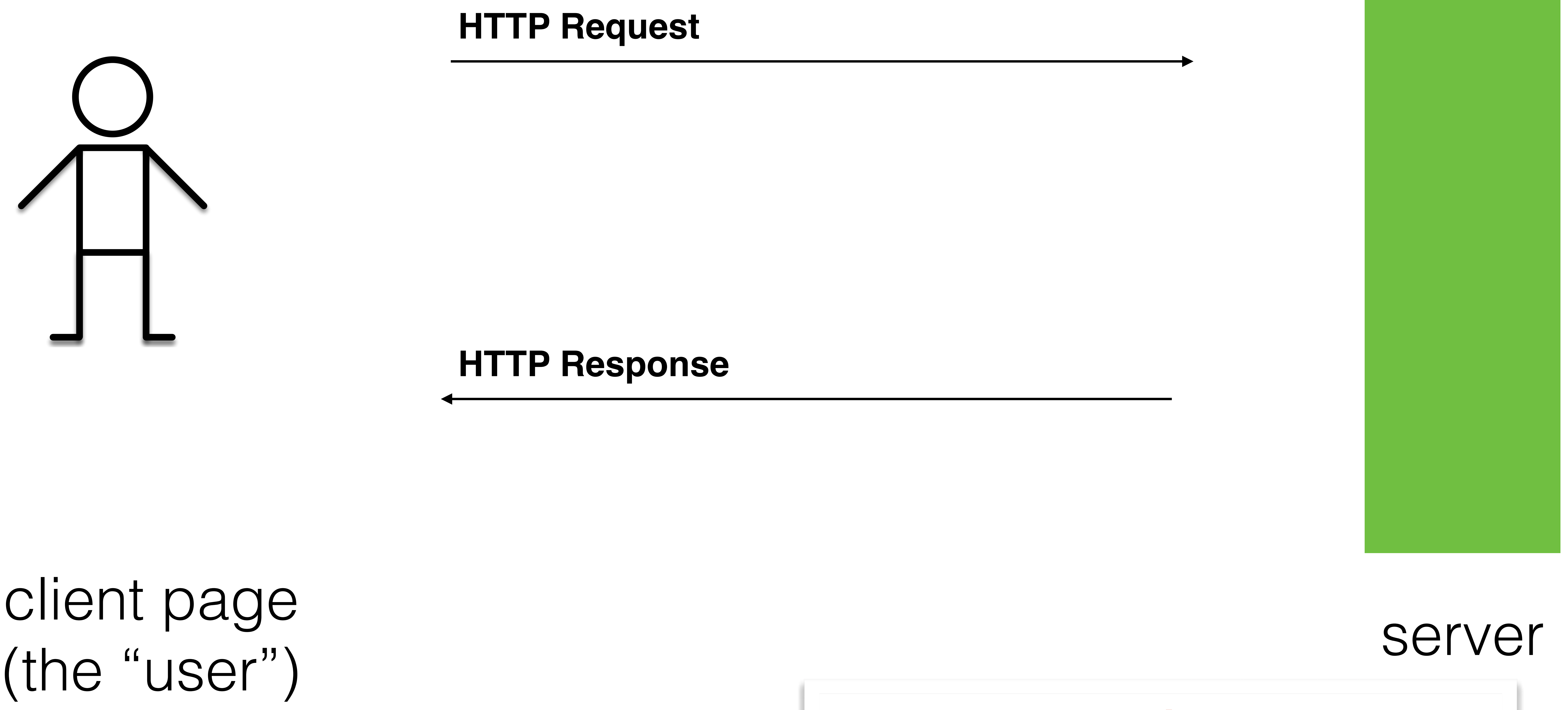
Example: Threat at the Boundary

Web Server



Example: Threat at the Boundary

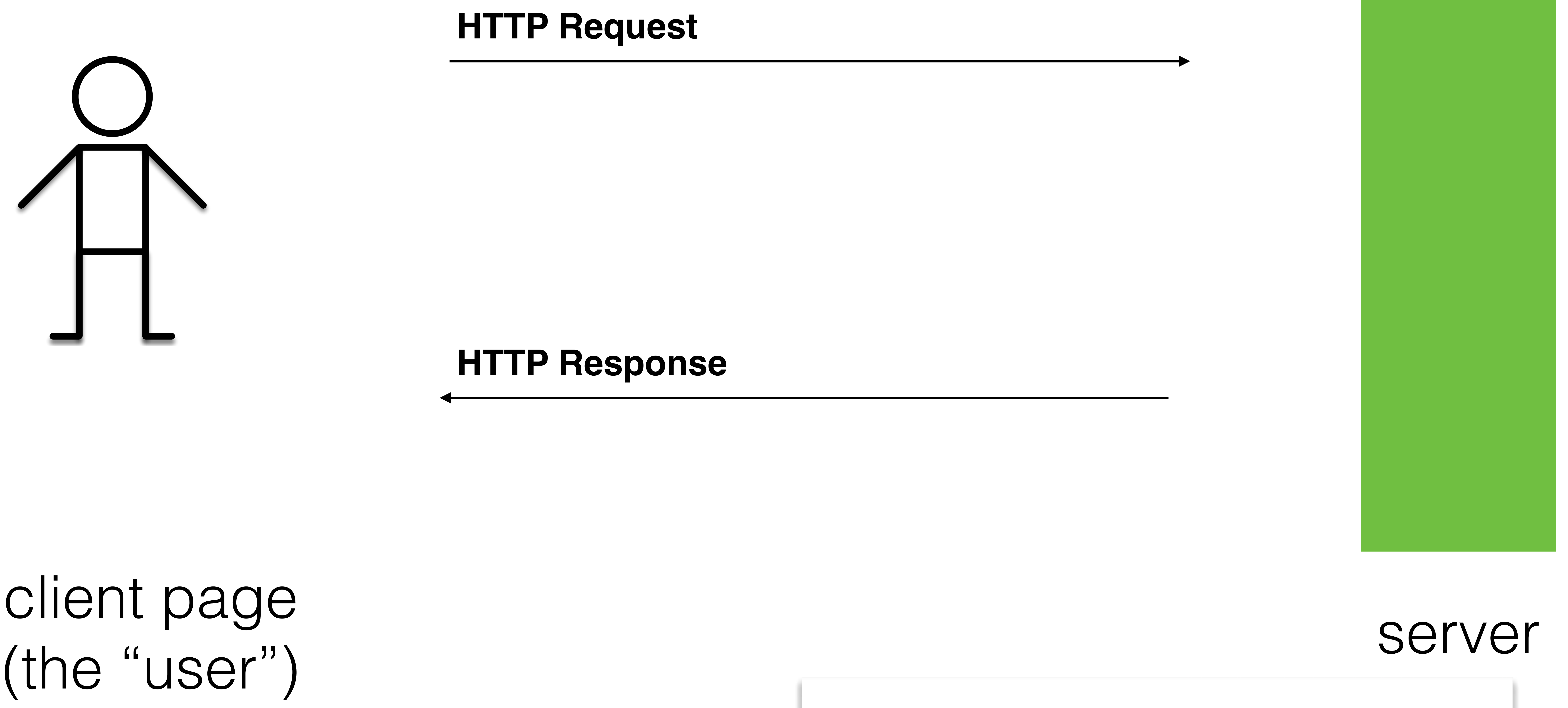
Web Server



Do I trust that this request *really* came from the user?

Example: Threat at the Boundary

Web Server

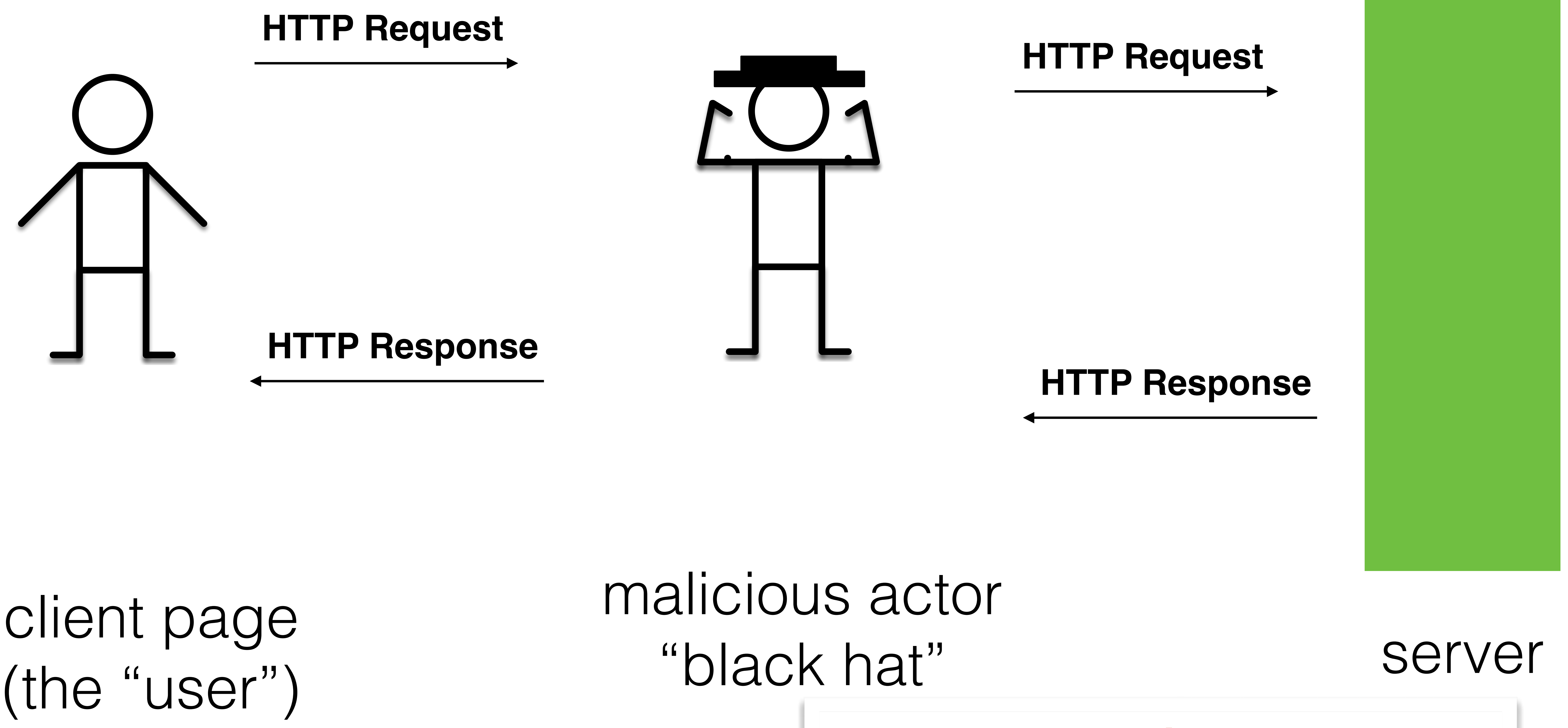


Do I trust that this response *really* came from the server?

Do I trust that this request *really* came from the user?

Example: Threat at the Boundary

Web Server



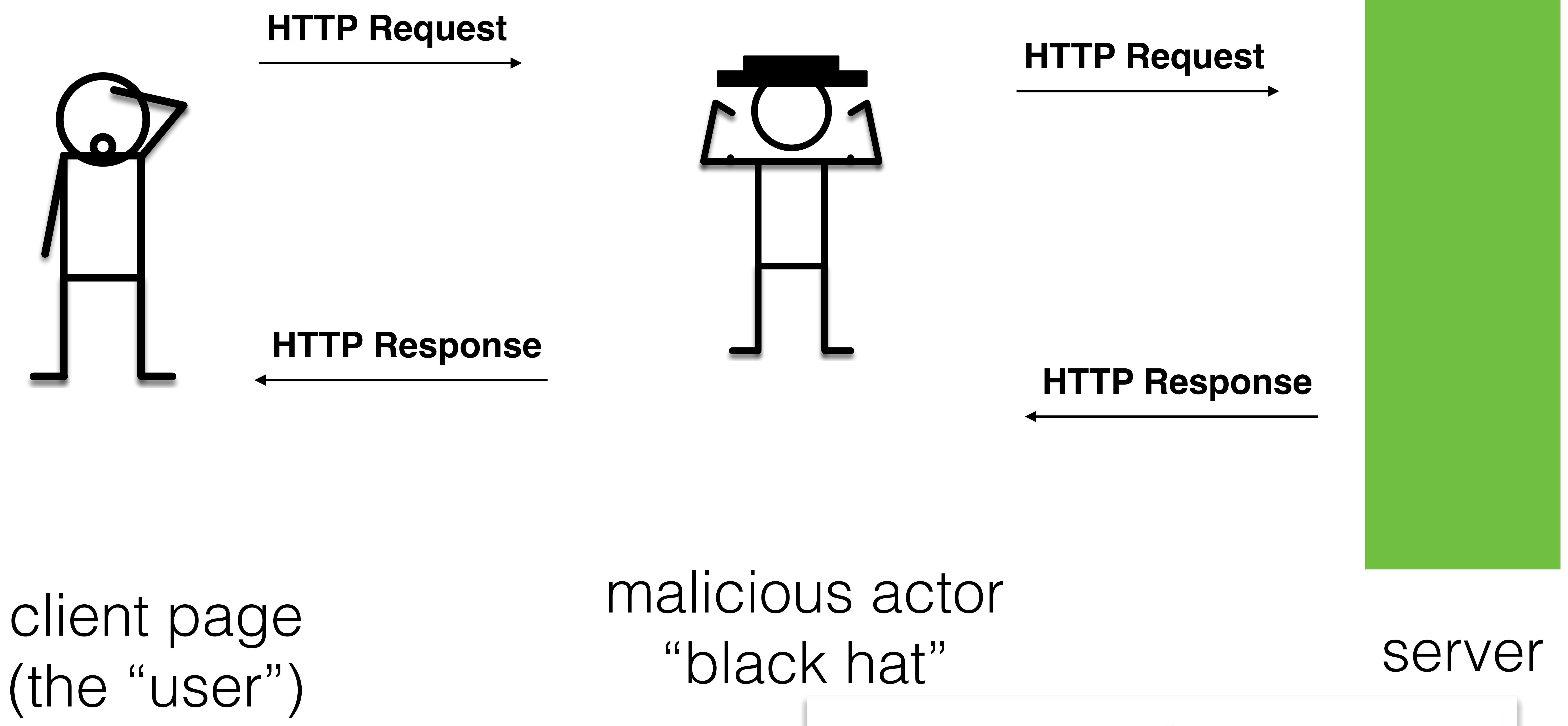
Do I trust that this response *really* came from the server?

Do I trust that this request *really* came from the user?

Example: Threa ary

Web Server

Might be “man in the middle” that intercepts requests and impersonates user or server.

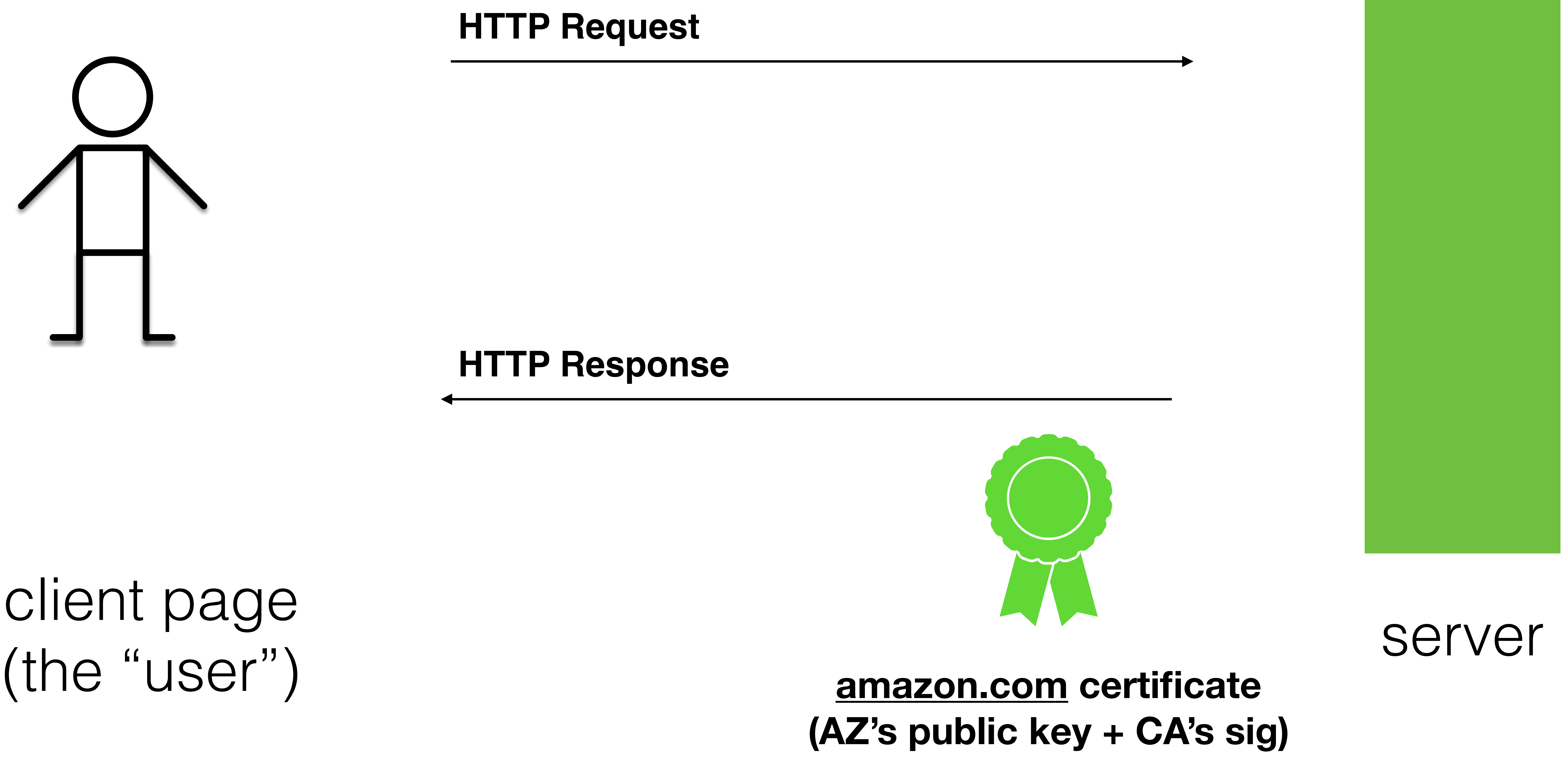


Do I trust that this response *really* came from the server?

Do I trust that this request *really* came from the user?

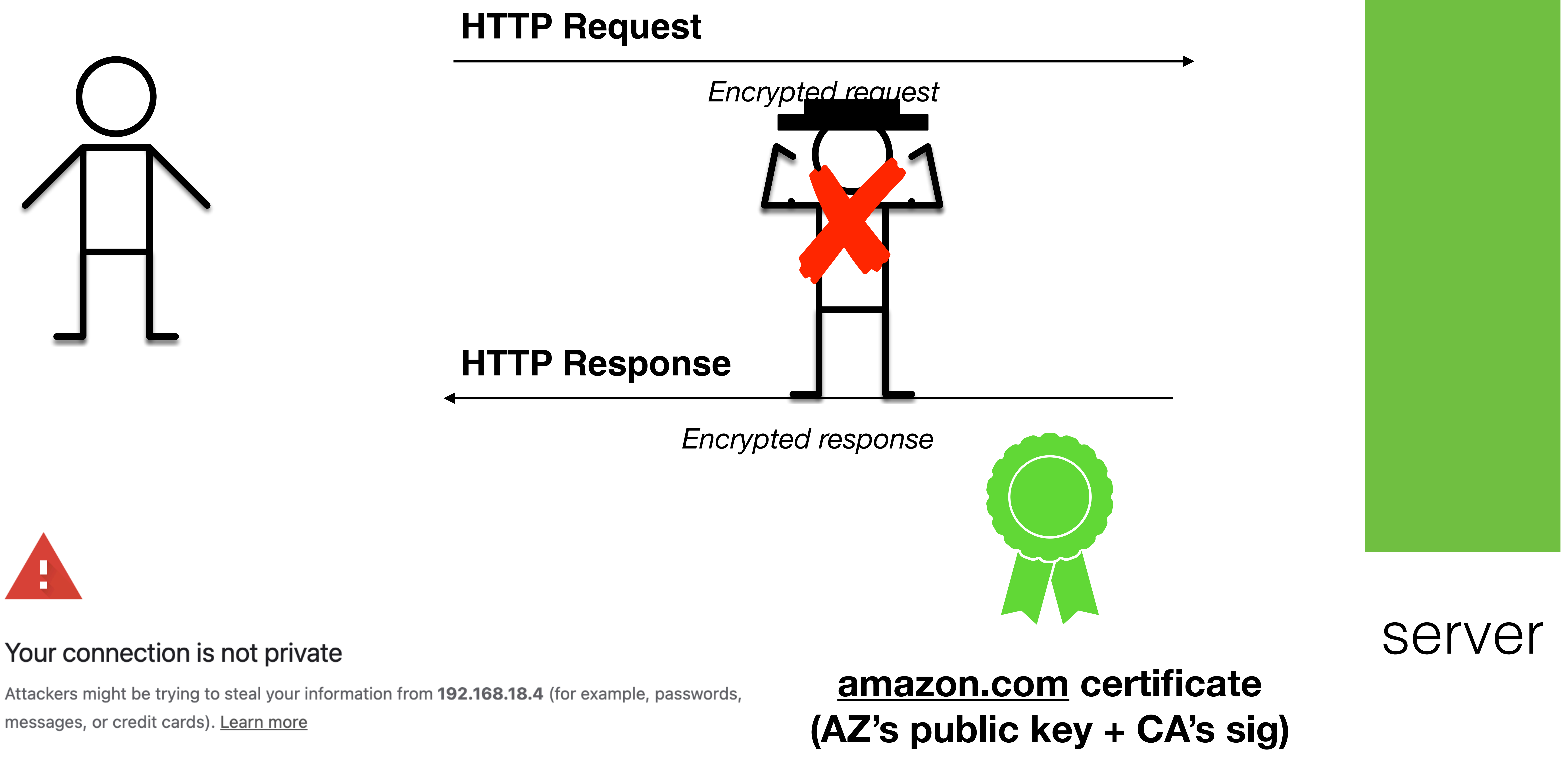
Threat Models: Web Server

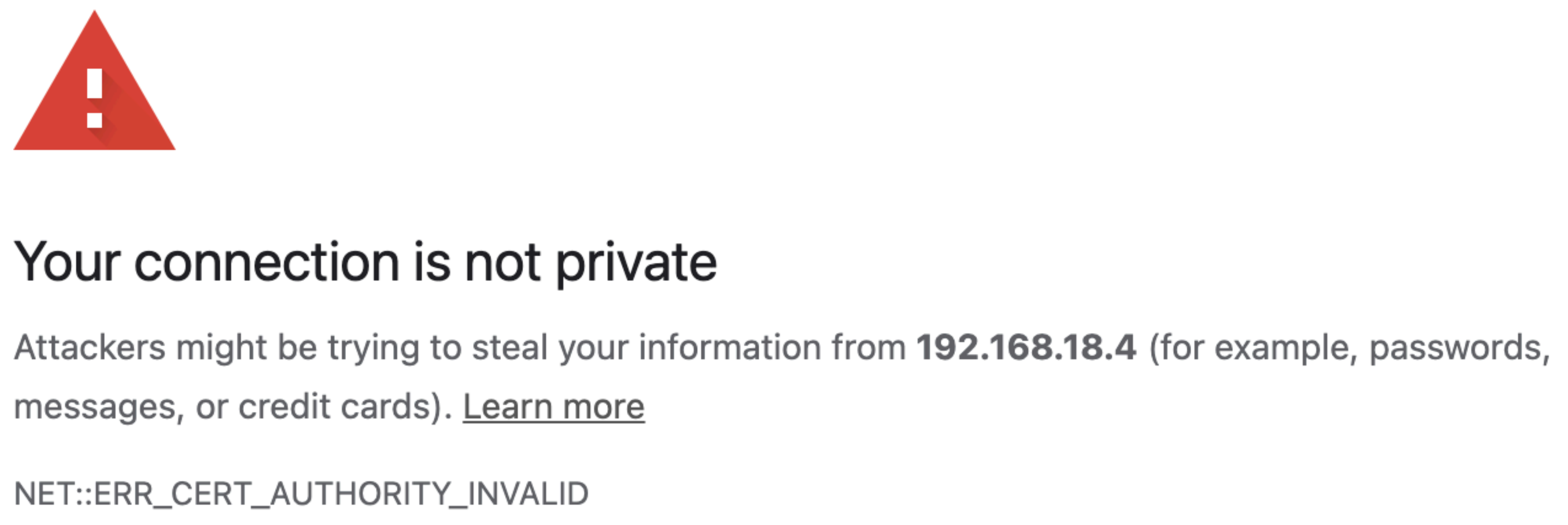
Preventing the man-in-the-middle with SSL



Threat Models: Web Server

Preventing the man-in-the-middle with SSL





Your connection is not private

Attackers might be trying to steal your information from **192.168.18.4** (for example, passwords, messages, or credit cards). Learn more

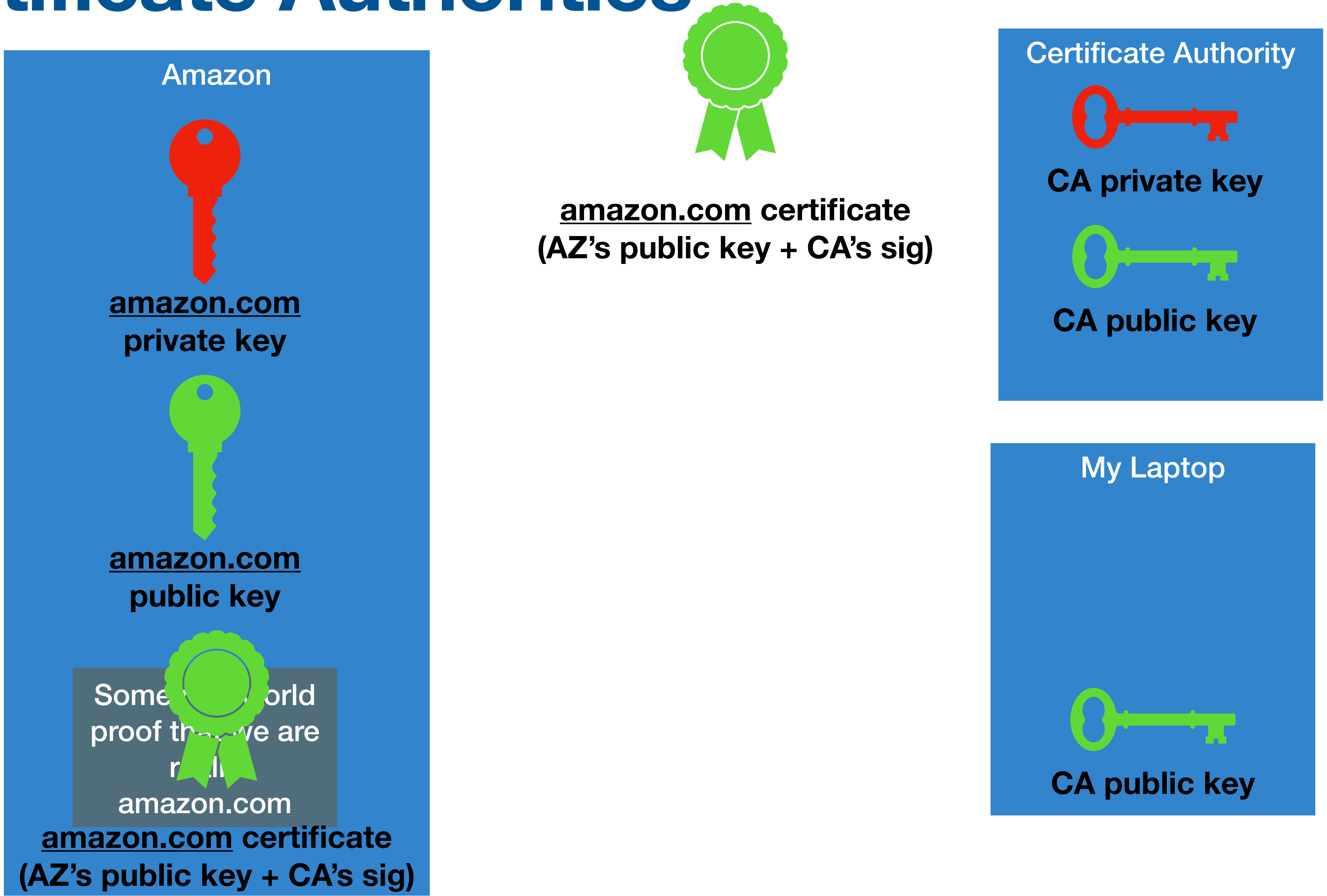
NET::ERR_CERT_AUTHORITY_INVALID

SSL: A perfect solution?

Certificate authorities

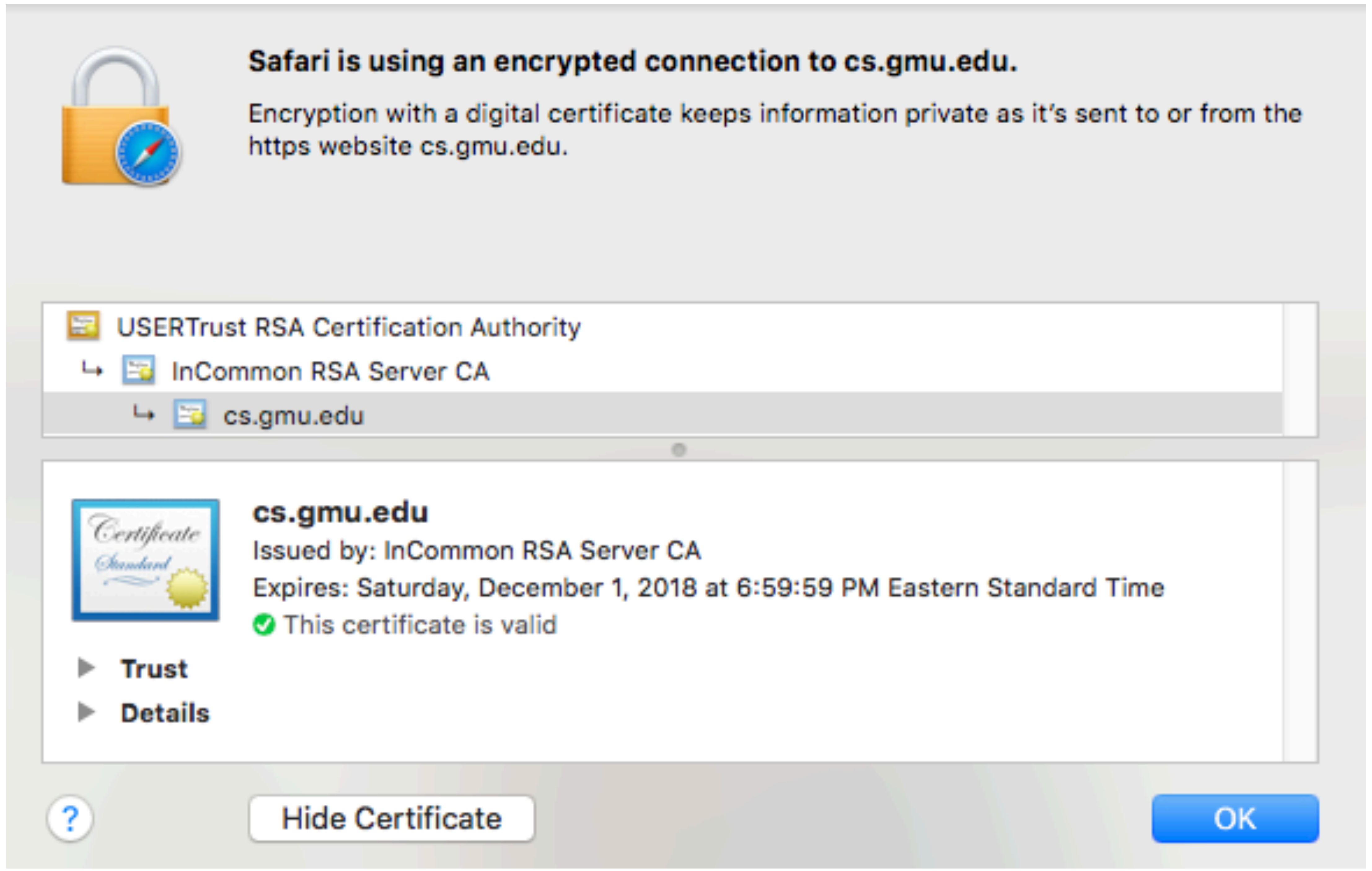
- A certificate authority (or CA) binds some public key to a real-world entity that we might be familiar with
- The CA is the clearinghouse that verifies that amazon.com is truly amazon.com
- CA creates a certificate that binds amazon.com's public key to the CA’s public key (signing it using the CA’s private key)

Certificate Authorities


Amazon
amazon.com private key
amazon.com public key
Some orld proof th we are r ll amazon.com
amazon.com certificate (AZ’s public key + CA’s sig)
amazon.com certificate (AZ’s public key + CA’s sig)
Certificate Authority
CA private key
CA public key
My Laptop
CA public key

Certificate Authorities

- Note: We had to already know the CA's public key
- There are a small set of “root” CA’s (think: root DNS servers)
- Every computer/browser is shipped with these root CA public keys



Safari is using an encrypted connection to cs.gmu.edu.

Encryption with a digital certificate keeps information private as it’s sent to or from the https website cs.gmu.edu.

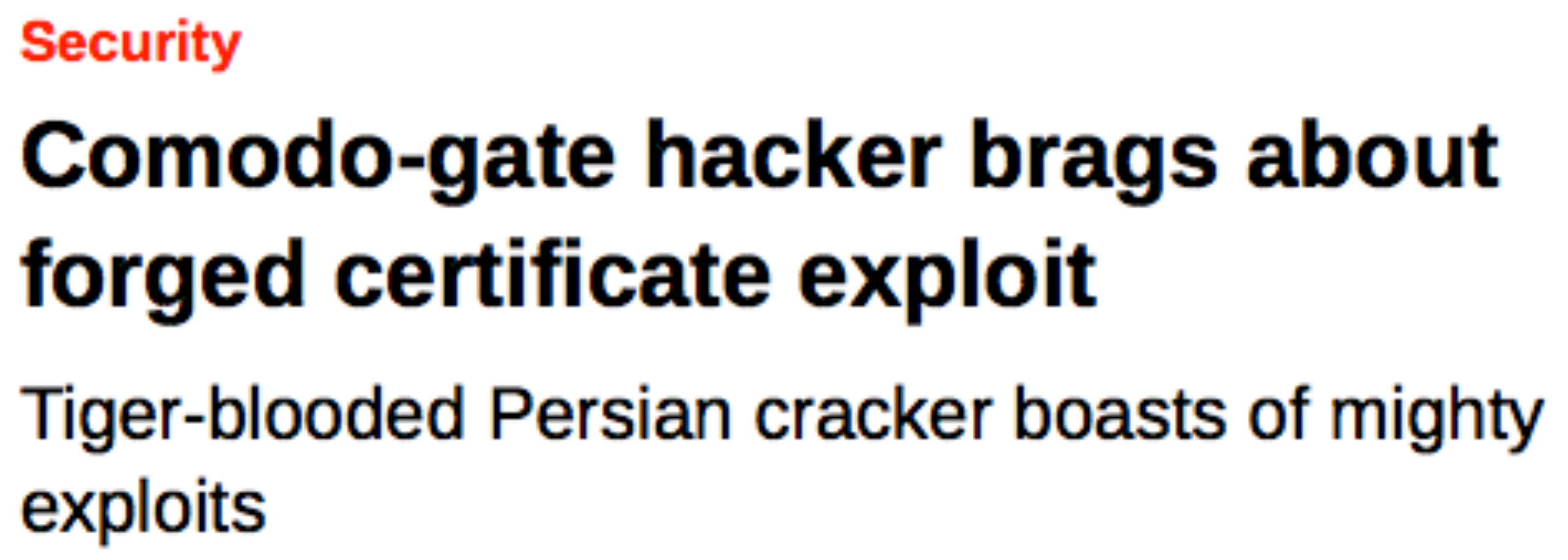
USERTrust RSA Certification Authority
↳ InCommon RSA Server CA
↳ cs.gmu.edu

cs.gmu.edu
Issued by: InCommon RSA Server CA
Expires: Saturday, December 1, 2018 at 6:59:59 PM Eastern Standard Time
✓ This certificate is valid

Certificate Authorities

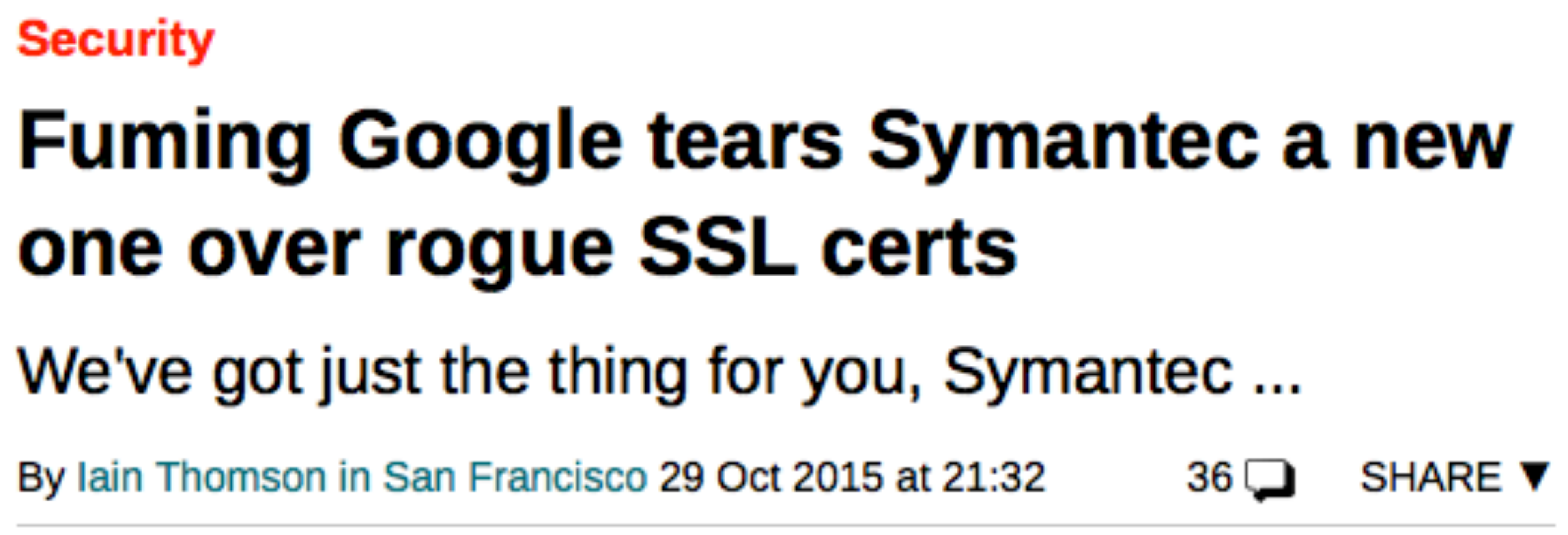
Nation-state-scale attackers

- What happens if a CA is compromised, and issues invalid certificates?
- Not good times.


Security

Comodo-gate hacker brags about forged certificate exploit

Tiger-blooded Persian cracker boasts of mighty exploits


Security

Fuming Google tears Symantec a new one over rogue SSL certs

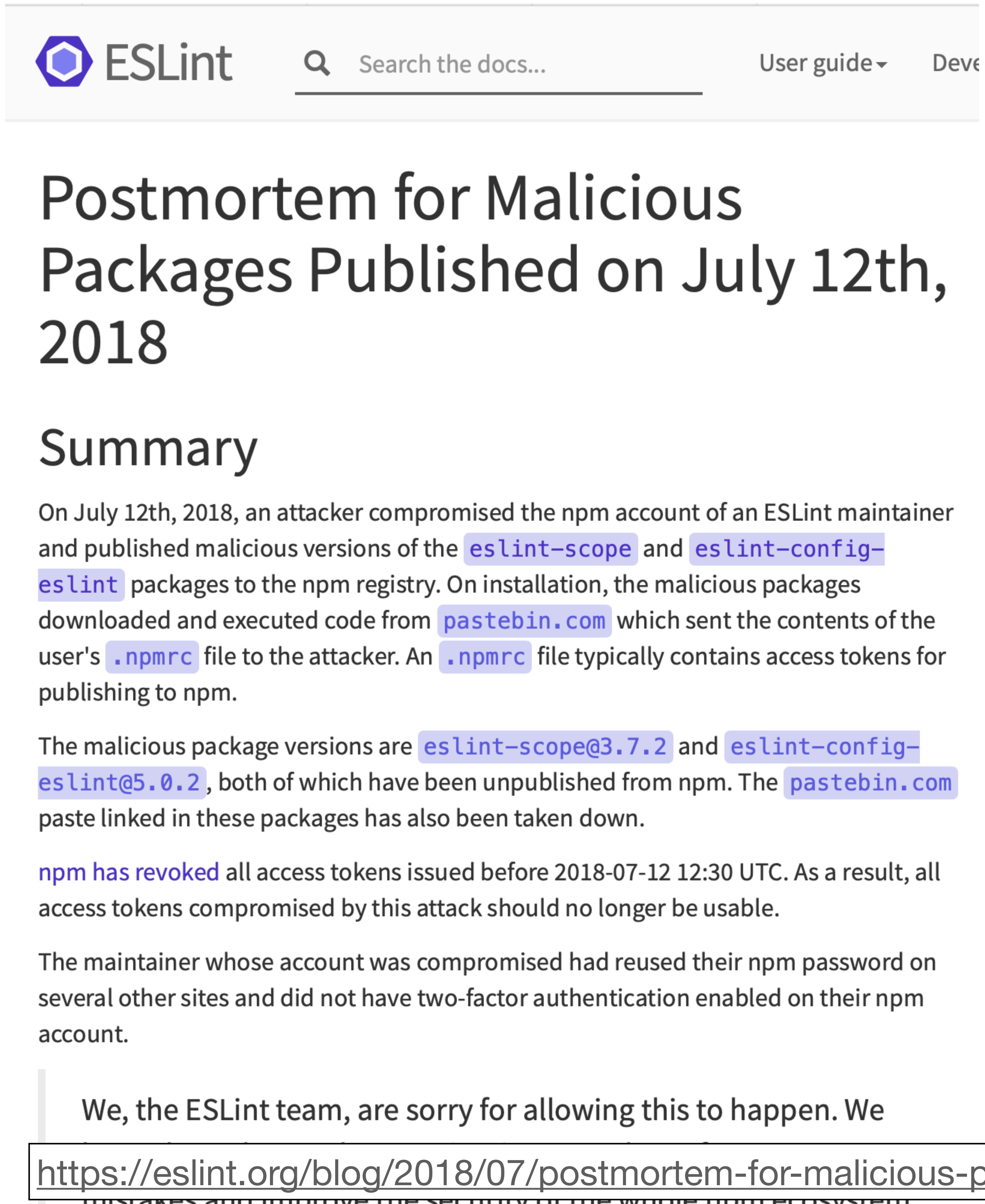
We've got just the thing for you, Symantec ...

By Iain Thomson in San Francisco 29 Oct 2015 at 21:32 36 SHARE ▼

Google has read the riot act to Symantec, scolding the security biz for its

Dependencies and Development Environment

Do we trust our own code? Third-party code provides an attack vector



Postmortem for Malicious Packages Published on July 12th, 2018

Summary

On July 12th, 2018, an attacker compromised the npm account of an ESLint maintainer and published malicious versions of the `eslint-scope` and `eslint-config-eslint` packages to the npm registry. On installation, the malicious packages downloaded and executed code from `pastebin.com` which sent the contents of the user's `.npmrc` file to the attacker. An `.npmrc` file typically contains access tokens for publishing to npm.

The malicious package versions are `eslint-scope@3.7.2` and `eslint-config-eslint@5.0.2`, both of which have been unpublished from npm. The `pastebin.com` paste linked in these packages has also been taken down.

npm has revoked all access tokens issued before 2018-07-12 12:30 UTC. As a result, all access tokens compromised by this attack should no longer be usable.

The maintainer whose account was compromised had reused their npm password on several other sites and did not have two-factor authentication enabled on their npm account.

> We, the ESLint team, are sorry for allowing this to happen. We

https://eslint.org/blog/2018/07/postmortem-for-malicious-package-publishes

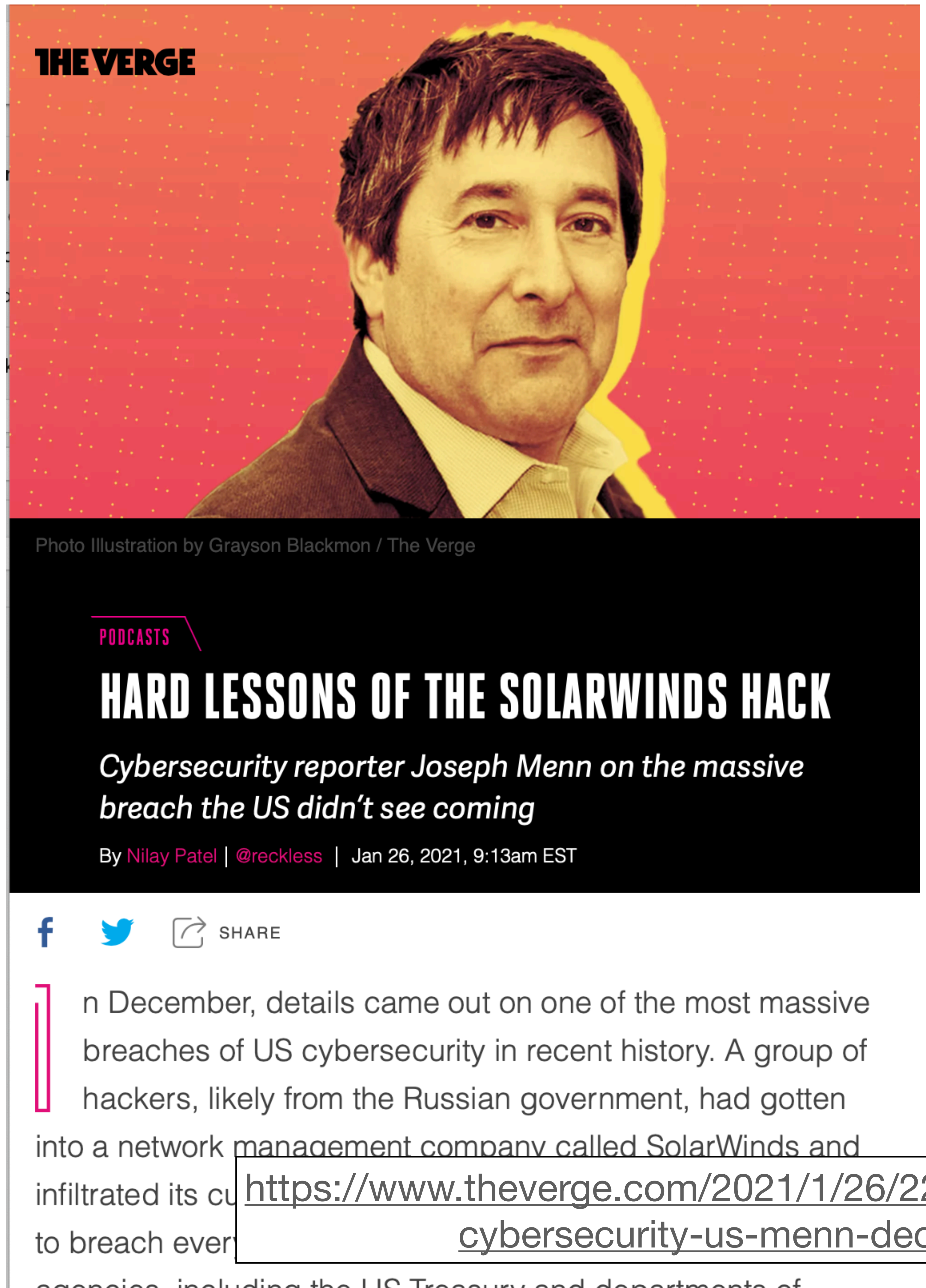


Photo Illustration by Grayson Blackmon / The Verge

PODCASTS

HARD LESSONS OF THE SOLARWINDS HACK

Cybersecurity reporter Joseph Menn on the massive breach the US didn't see coming

By Nilay Patel | @reckless | Jan 26, 2021, 9:13am EST

SHARE

In December, details came out on one of the most massive breaches of US cybersecurity in recent history. A group of hackers, likely from the Russian government, had gotten into a network management company called SolarWinds and infiltrated its cu
to breach ever

https://www.theverge.com/2021/1/26/22248631/solarwinds-hack-cybersecurity-us-menn-decoder-podcast

Costs & Benefits

We can fix everything at a cost…

- We can ensure our code is not tampered with by running all of it on our own machines (remove logic from frontend)
    - Increases latency
    - What if someone hacks into our server?
- We can fix the certificate authority issue by securely distributing our own certificate (out of band)
    - Cumbersome
    - What if someone hacks into the client and replaces certificate?
- Can we trust our own code? How?

Learning Objectives for this Lesson

By the end of this lesson, you should be able to…

- Describe that security is a spectrum, and be able to define a realistic threat model for a given system
- Evaluate the tradeoffs between security and performance in software engineering