

# Check Your Thermal Exhaust Port: 10 Vulnerabilities Your Web App Probably Has Right Now

# Audience

- Anyone with a web app
- New to security
- Security Experts

# Agenda

- Series of lightning talks
- SQL Injection
- Directory Traversal/Backtracking
- Broken access control
- Package vulnerabilities
- Guids vs Ints
- Container Vulnerabilities
- JWT Stale Roles
- Data Leaking
- Clickjacking
- Lack of WAF

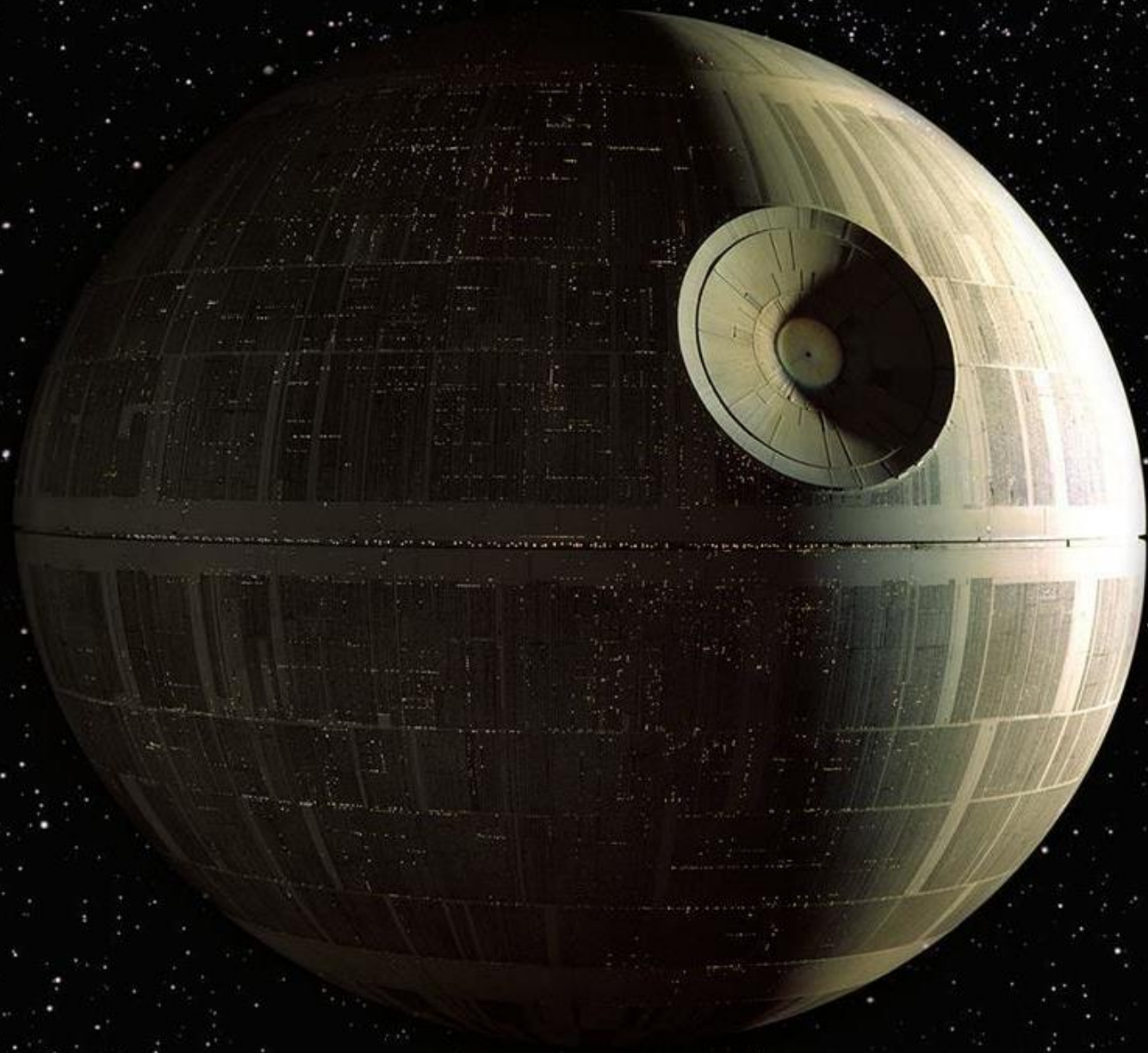
# Goals

- Exposure to security vulnerabilities
- Not just OWASP Top 10
- Check yourself before you wreck yourself

# Who am I?

- Director of Engineering at Lean TECHniques
- Co-organizer of [Iowa .NET User Group](#)
- Microsoft MVP
- [Friend of Redgate](#)
- Blog at [scottsauer.com](http://scottsauer.com)





One problem



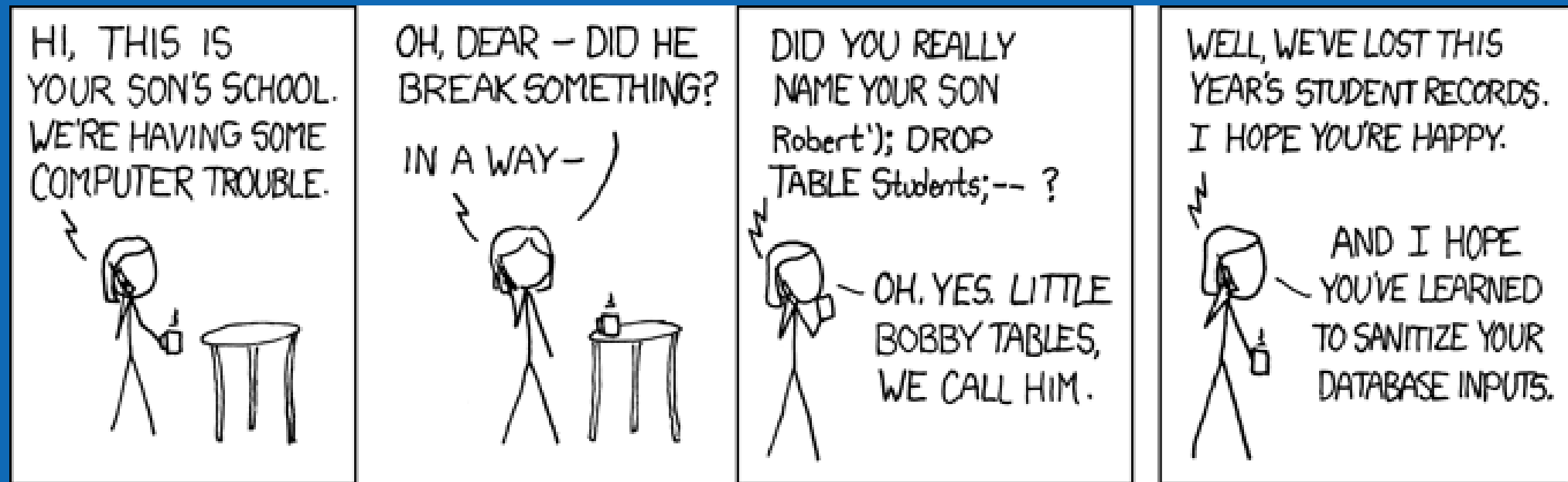
One problem





# It's the same with web apps

- One vulnerability can take down an otherwise rock solid web application
- There are dozens of attack types
- We're just going to cover 10



# SQL Injection

# SQL Injection

- What is it?
  - Allows attackers to execute queries against your database
- Why should I care?
  - They can get at some or all of your data
  - They can change or delete some or all of your data

# SQL Injection

- How do I fix it?
  - Parameterize Queries

- Vulnerable to SQL Injection

```
[HttpGet(template: "{id}")]
public Customer Get(string id)
{
    var query = $"SELECT id,
                    first_name as FirstName,
                    last_name as LastName
                    FROM customers
                    WHERE id='{id}'";

    return _dbConnection.QueryFirst<Customer>(query);
}
```

- Not Vulnerable to SQL Injection

```
[HttpGet(template: "{id}")]
public Customer Get(string id)
{
    var query = $"SELECT id,
                    first_name as FirstName,
                    last_name as LastName
                    FROM customers
                    WHERE id=@id";

    return _dbConnection.QueryFirst<Customer>(query, param: new {id});
}
```

# SQL Injection Demo



# Directory Traversal (aka Backtracking)

# Directory Traversal

- What is it?
  - Allows attackers to access folders and files they shouldn't have access to
- Why should I care?
  - Exposes data, potentially sensitive data



# Directory Traversal

- How do I fix it?
  - Filter user input
  - Do not intake a path from a user
  - One approach – file request based on GUID, path stored in database

- Vulnerable

```
[HttpGet]
public string GetFile(string fileName)
{
    return System.IO.File.ReadAllText(Path.Combine("Documents", fileName));
}
```

- Not Vulnerable

```
[HttpGet]
public string GetFile(Guid id)
{
    return System.IO.File.ReadAllText(GetFilePathById(id));
}
```

1 usage

```
private string GetFilePathById(Guid id) =>
    _dbConnection.QueryFirst<string>(sql: "SELECT file_path FROM documents WHERE id=@id", param: new { id });
```

# Directory Traversal Demo



# Directory Overwriting

# Directory Overwriting

- What is it?
  - Allows attackers to overwrite files or folders they shouldn't have access to
- Why should I care?
  - Loses valid files or folders

# Directory Overwriting

- How do I fix it?
  - Do not use the filename from the user to save the file to disk
  - On upload, rewrite the file path (i.e. use a GUID for a file name) and store path in the DB

- Vulnerable

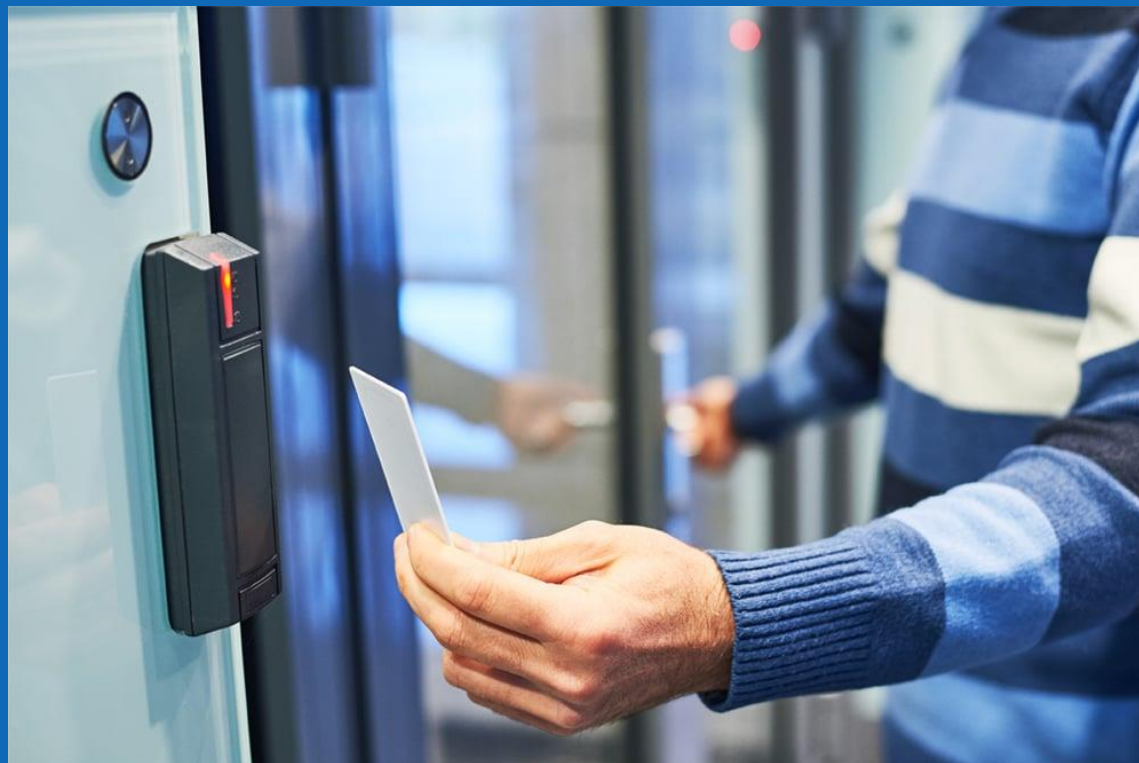
```
[HttpPost]
public IActionResult CreateFile([FromBody] CreateFileRequest request)
{
    System.IO.File.WriteAllText(Path.Combine("Documents", request.FilePath), request.TextContents);
    return StatusCode(StatusCode.Status201Created);
}
```

- Not Vulnerable

```
[HttpPost]
public IActionResult CreateFile([FromBody] CreateFileRequest request)
{
    System.IO.File.WriteAllText(Path.Combine("Documents", Guid.NewGuid().ToString()), request.TextContents);
    return StatusCode(StatusCode.Status201Created);
}
```

# Directory Overwriting Demo





# Broken Access Control

# Broken Access Control

- What is it?
  - Allows users to perform actions that should be outside of their control
- Why should I care?
  - Non-Admin users could perform Admin actions like Editing or Deleting data or viewing sensitive data

# Broken Access Control

- How do I fix it?
  - Enforce correct permissions SERVER SIDE (not just client side)
  - Client-side is just UX so they don't see things that aren't relevant



# Package Vulnerabilities

# Package vulnerabilities

- What is it?
  - Outdated and/or insecure packages
- Why should I care?
  - Depending on how the package is used, this could be a catastrophic vulnerability in your application


# Package Vulnerabilities

- How do I fix it?
  - Stay up to date on latest versions
  - Use maintained/supported packages

# Open SSF Scorecard

- <https://securityscorecards.dev/>
- Assesses open source packages for security risks
- Code Vulnerabilities
- Maintenance (Dependency Updating, Frequent History, License, etc)
- Continuous Testing (Tests, SAST, Fuzzing)
- Source Risk Assessment (Branch Protection, Code Review, etc.)
- Build Risk Assessment (Signed Releases, Pinned Deps, etc)

### Example GUID



{1f7531a6-2568-4ea7-b2a3-8cf80223e5bb}

**M**  
position or version  
(defines the GUID type)

**N**  
position or variant  
(defines the version)

© 2014 Microsoft Corporation. All rights reserved. See [http://go.microsoft.com/fwlink/?LinkId=254236](#) for more details.

# Using Ints instead of Guids

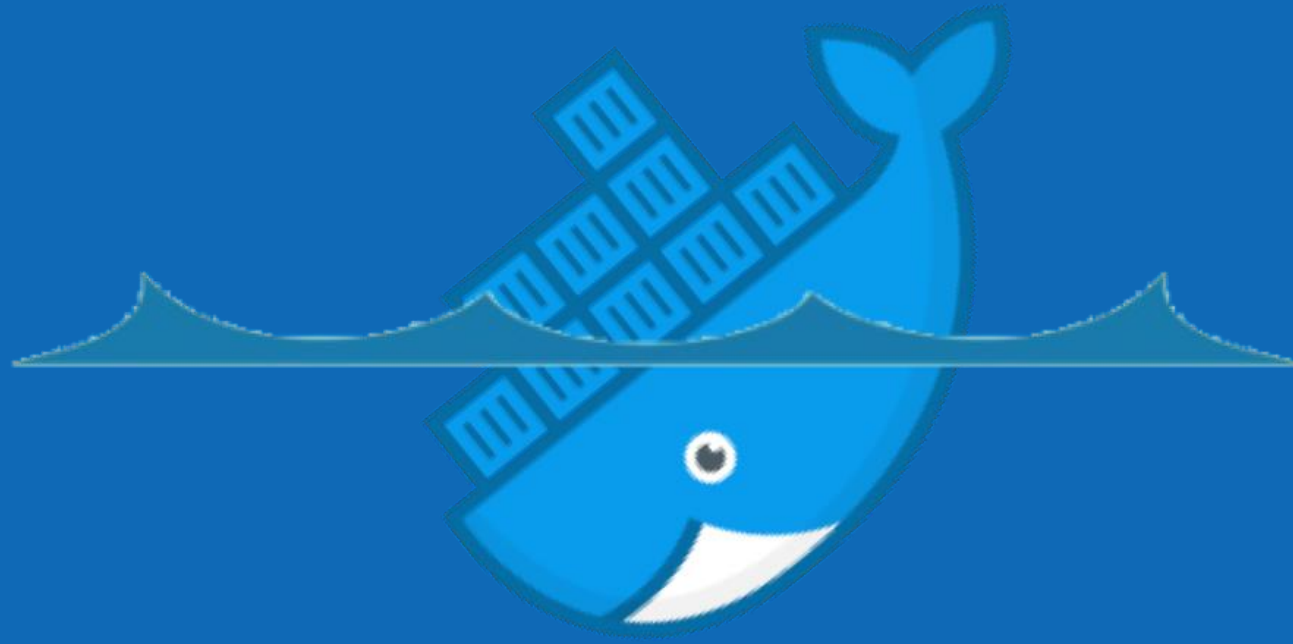


# Using Ints instead of Guids

- What is it?
  - Using auto-incrementing integers to uniquely identify database rows
- Why should I care?
  - Auto-incrementing ints can leak how much data is in the system (possible predictor of financials)
  - If Broken Access Control vulnerability is in place, they can easily enumerate your data

# Using Ints instead of Guids

- How do I fix it?
  - Use random Guids instead of Ints for primary keys
  - Yes it comes at a performance penalty... but will you notice?



# Container Image Vulnerabilities

# Container Image Vulnerabilities

- What is it?
  - Images provide an immutable artifact which is great for repeatability
  - The problem with immutable artifacts, is they don't get updates
  - These images are used for deployments (i.e. Azure App Service/Container Apps, AWS ECS, etc)
- Why should I care?
  - Depending on the vulnerability, your application may be partially or completely taken over

# Container Image Vulnerabilities

- How do I fix it?
  - Rebuild images regularly
  - Use a scanner
  - Use a PaaS service such as Azure App Service that auto-upgrades framework versions (i.e. .NET, Java, etc)



# JWT Stale Roles

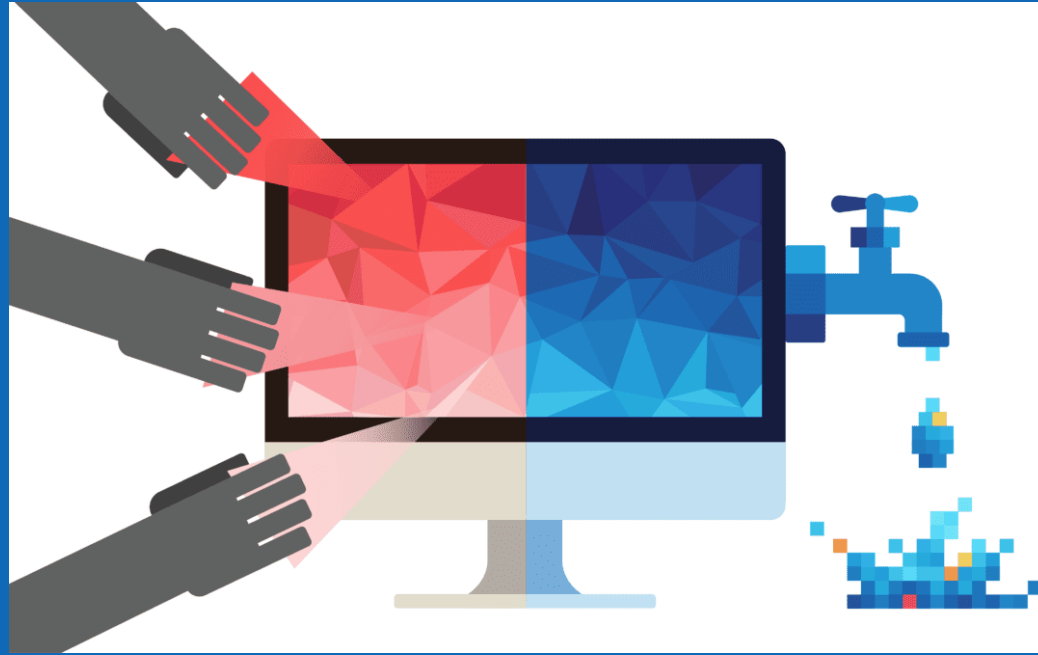
# JWT Stale Roles

- What is it?
  - Often times, web applications put role data in their JWT so they don't have to go fetch it every time.
- Why should I care?
  - The problem is, if a new permission is added or removed, the roles on the JWT will be stale
  - For example, if someone is terminated their JWT (and therefore access) might still be valid if you do local introspection

# JWT Stale Roles

- How do I fix it?
  - Round trip to get roles from the database every time
  - Performance or Security – pick one
  - Understand the tradeoffs





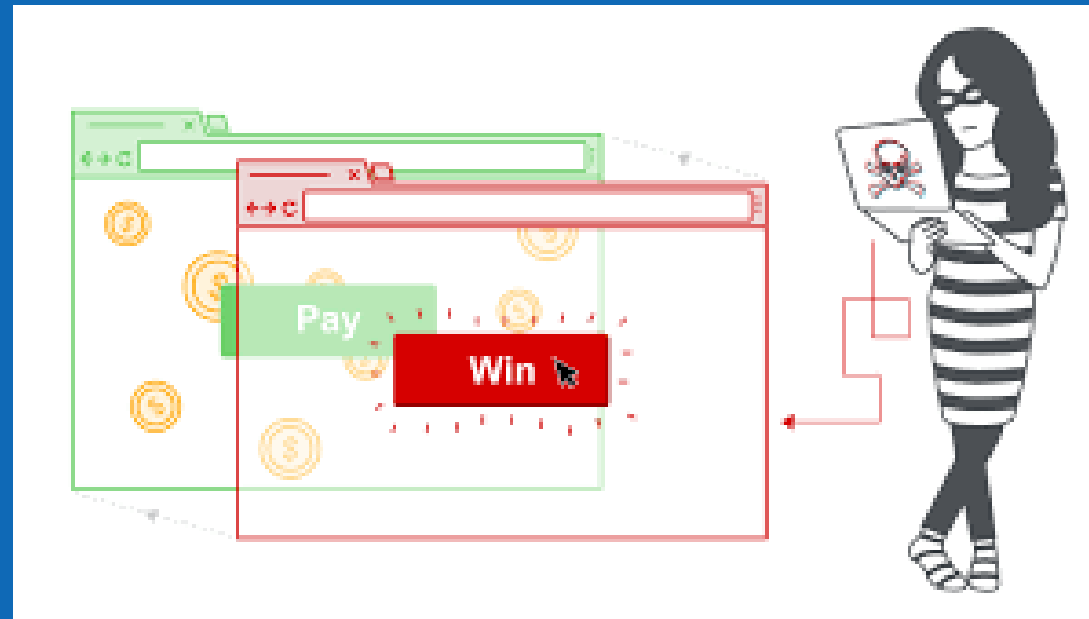
# Data Leaking

# Data Leaking

- What is it?
  - Exposing too much data
- Why should I care?
  - You could be exposing sensitive data to the wrong people

# Data Leaking

- How do I fix it?
  - Return back just the data you/your application/your customers need
  - Be wary of the spread operator in JS



# Clickjacking

# Clickjacking

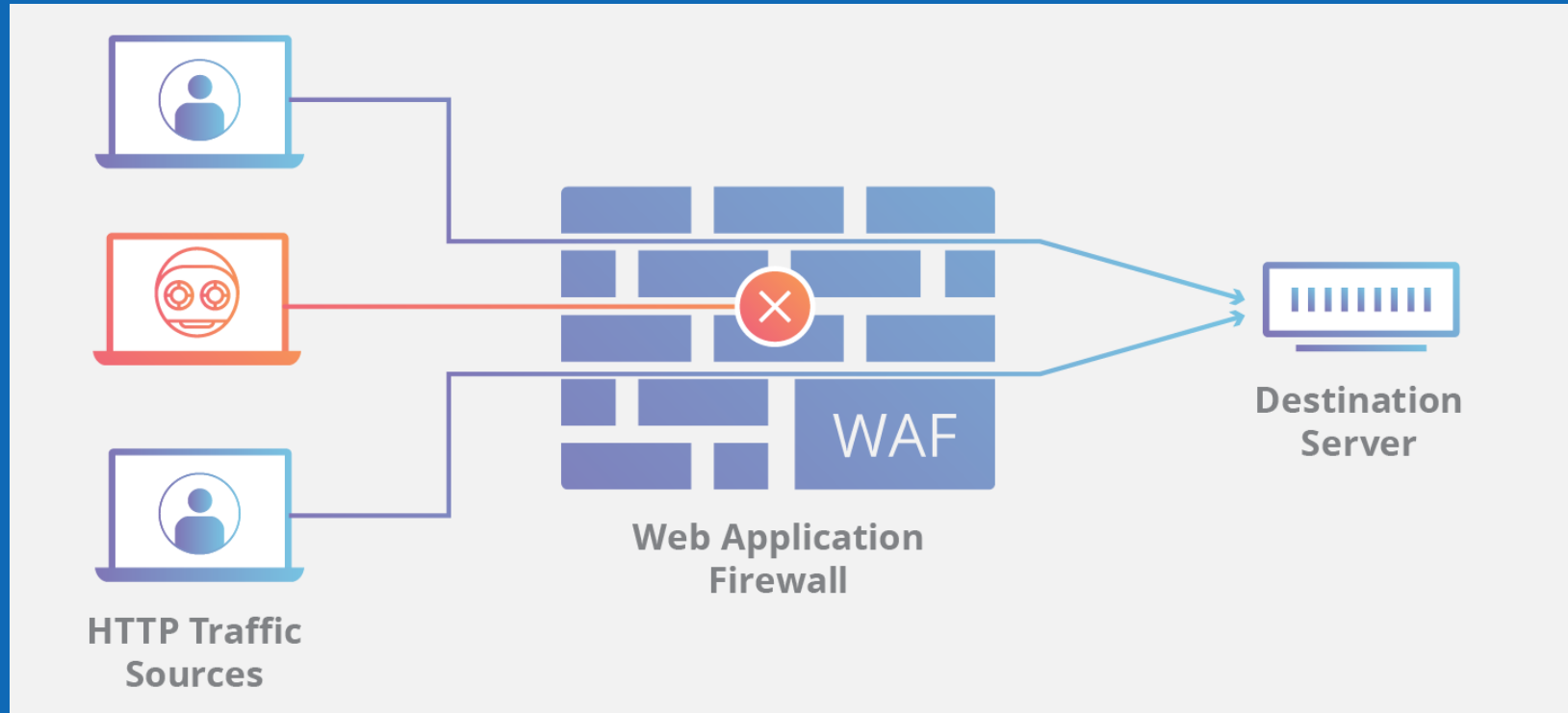
- What is it?
  - Tricking a user to perform an action they didn't mean to perform
  - Usually this is in the form of overlaying an iframe of a legitimate website, on top of a button on an illegitimate website
- Why should I care?
  - Could allow malicious users to trick privileged users to take actions they shouldn't be taking
  - Yes even internal apps are susceptible

# Clickjacking

- How do I fix it?
  - Restrict who can IFrame your web app
  - Apply the X-Frame-Options header with the value none or Content-Security-Policy with frame-ancestors setting to none

# Clickjacking Demo





# Lacking a WAF



# Lacking a WAF

- What is it?
  - Web Application Firewalls protect against many attacks (DDOS, scans, etc)
- Why should I care?
  - First line of defense against malicious attackers

# Lacking a WAF

- How do I fix it?
  - Purchase a WAF (i.e. Cloudflare, Imperva, Azure WAF, AWS WAF, etc)

# Takeaways

- Awareness different security vulnerabilities your web apps might have
- More than just OWASP Top 10

# Questions?

Contact: [ssauber@leantechniques.com](mailto:ssauber@leantechniques.com)



Slides at [scottsauber.com](http://scottsauber.com)

# Thanks!

