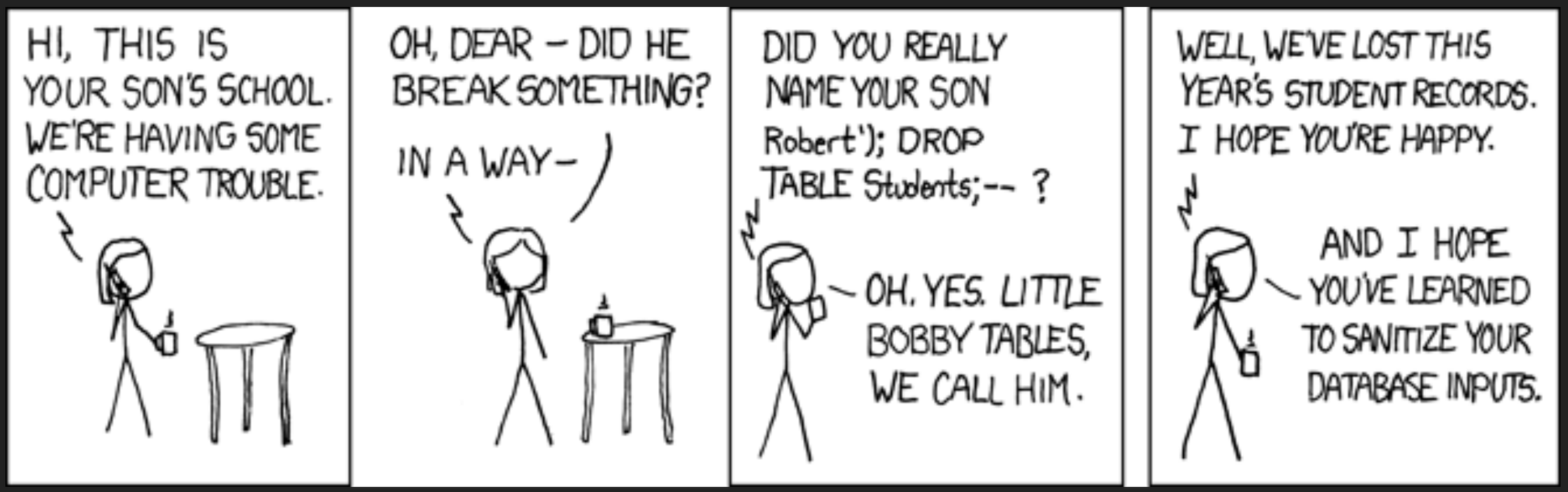
TOP 10 WEB APP VULNERABILITIES

YNON PEREK

1. INJECTIONS

THE PROBLEM


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

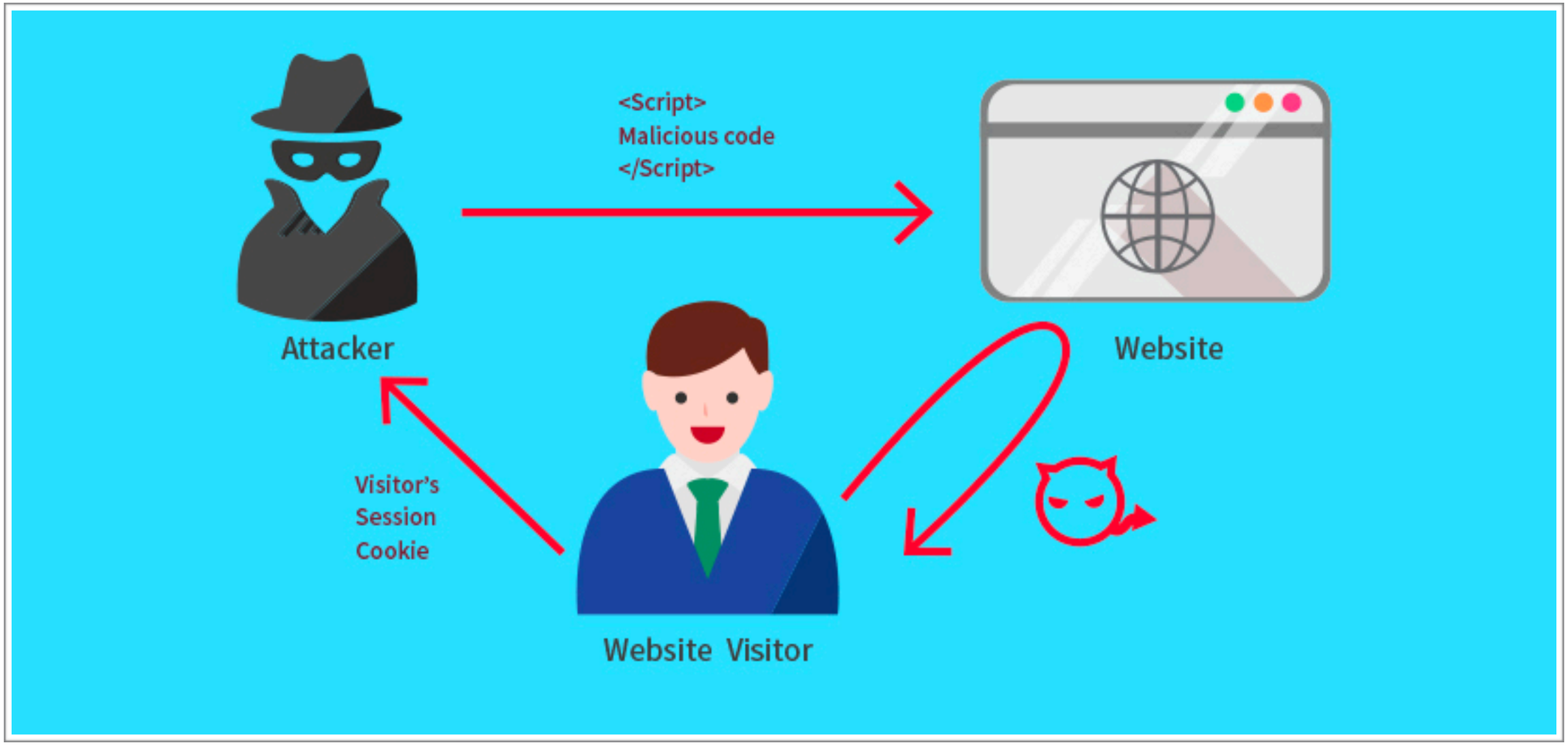
VULNERABLE RAILS CODE

```ruby
Project.where("name = '#{params[:name]}'")
```

MITIGATION

- Use automated tools to discover injections in your app
- Be careful using string concatenation in any context that creates “commands”
- Don’t forget:
  - MongoDB injection
  - Shell injection

2. XSS


<Script>
Malicious code
</Script>
Attacker
Website
Visitor's
Session
Cookie
Website Visitor

VULNERABLE RAILS CODE

```
Hello <%= raw @name %>
```

MITIGATIONS

- Rails automatically cleans your variables before making HTML
- Be careful with . html_safe / .raw
- Use automatic tools to find XSS in your site
- Use CSP

3. BROKEN SESSION MANAGEMENT


EUROPÄISCHE UNION
BUNDESREPUBLIK
DEUTSCHLAND
REISEPASS
ZONE C
WE DEPART ON TIME!
49J
TLV

VULNERABLE RAILS CODE

```ruby
def signin_with_barcode
  code = params[:barcode]
  user = User.find_by(code: code)
  sign_in(user)
end
```

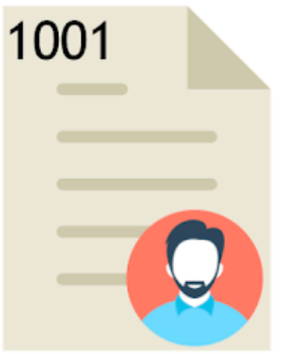
MITIGATION

- List all the ways users can
  - “Create a session”
  - “Continue a session”
- Make sure “logout” deletes the session
- Make it hard to continue somebody else’s session

4. INSECURE DIRECT OBJECT REFERENCES

Get my document which number is “1000” please!

Of course!



Get the document which number is “1002” please!

Hey! Don’t mention it!

VULNERABLE RAILS CODE

```ruby
def show
  @project = Project.find(params[:id])
end
```

MITIGATION

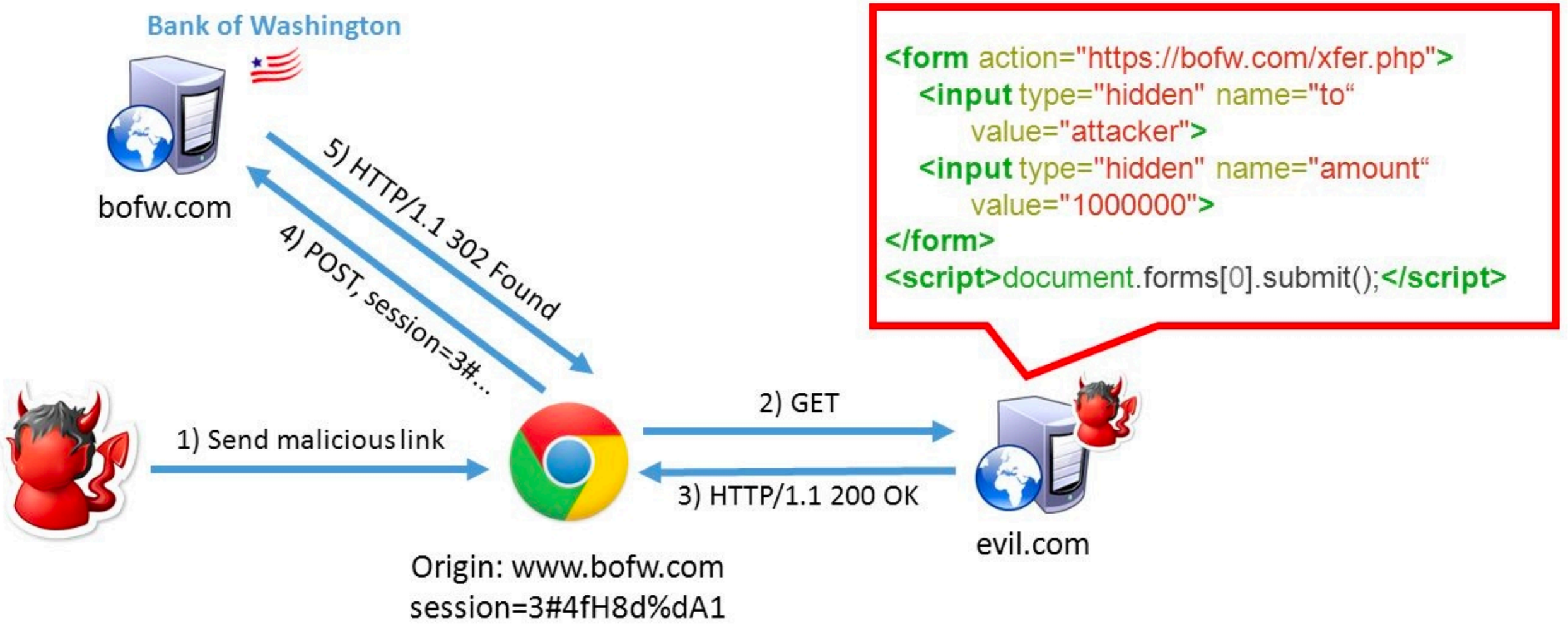
- cancancan

```
class ApplicationController < ActionController::Base
  check_authorization
end
```

5. CSRF

CSRF Attack

- Assume that the victim is logged-in to www.bofw.com



VULNERABLE RAILS CODE

```
class FooController < ApplicationController
  protect_from_forgery except: :index
  skip_forgery_protection
end
```

HOW IT WORKS IN RAILS

MITIGATION

- Always leave rails CSRF protection on
- Verify origin header when using tokens
- Limit session duration

6. SECURITY MISCONFIGURATION

VULNERABLE RAILS CODE

```
# file: config/database.yml
production:
  adapter: postgresql
  database: prod
  encoding: unicode
  pool: 5
  timeout: 5000
  username: ynon
```

SYMPTOMS

- DB / API connection without credentials
- Missing rate limit
- Information in HTTP headers

MITIGATIONS

- Use automatic tools to check your installation
- nmap
- https://securityheaders.com/

7. INSECURE CRYPTOGRAPHIC STORAGE

SYMPTOMS

- Passwords are saved plaintext or MD5 in the DB
- Backups are not encrypted
- Credentials stored in code

8. FAILURE TO RESTRICT URL ACCESS

VULNERABLE RAILS CODE

```ruby
def create
  @project = Project.new(project_params)
end
```

SYMPTOMS

- Router routes that are not accessible from UI
- Controller actions without authenticate

9. INSUFFICIENT TRANSPORT LAYER PROTECTION

RAILS SPECIFICS

- Use force_ssl = true
- Careful when your app is behind a proxy
- https://www.cdn77.com/tls-test

10. UNVALIDATED REDIRECTS AND FORWARDS

VULNERABLE CODE

```
class SessionsController < ApplicationController
  def create
    path = params[:url].presence || home_path
    user = User.authenticate!(users_params)
    session[:user] = user.id

    redirect_to path
  end
end
```

MITIGATIONS

- Always validate input before redirect
- use URI::parse(...) to validate the URL

Q & A