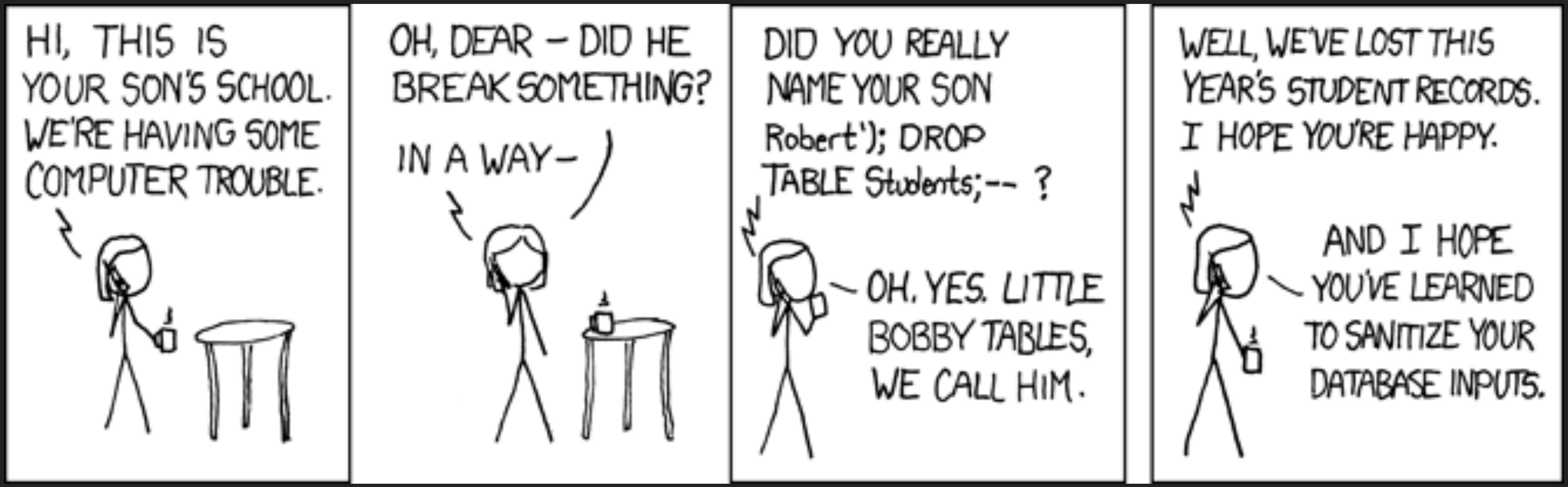
TOP 10 WEB APP VULNERABILITIES

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.

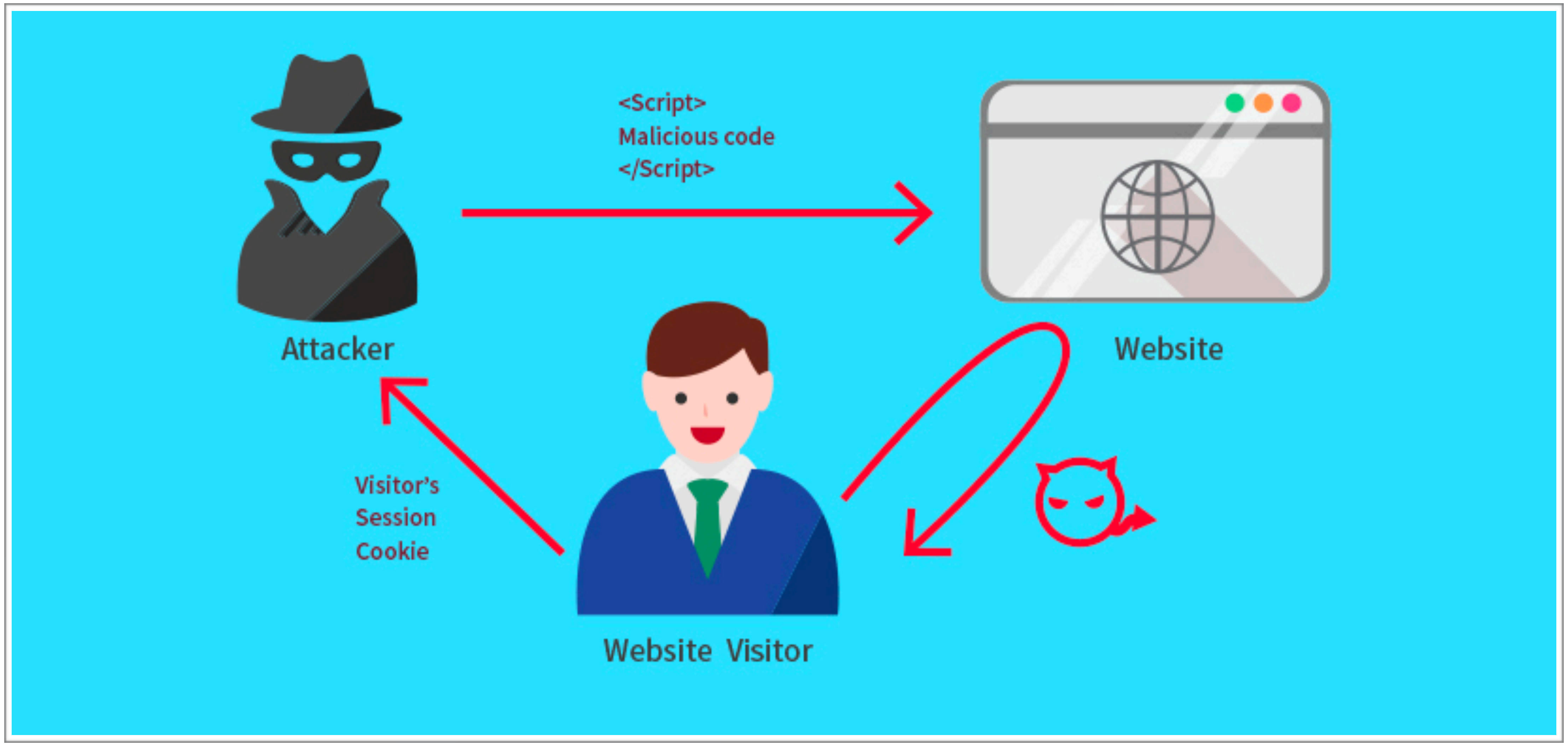
VULNERABLE NODE CODE

```
const user = await User.findOne({
  email: req.query.user.email,
});
```

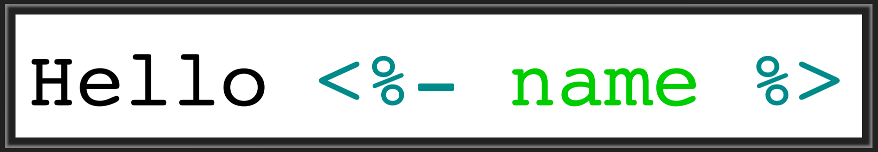
MITIGATION

- 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


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

VULNERABLE CODE



```
Hello <%- 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
- Node: https://www.npmjs.com/package/express-csp-header

3. BROKEN SESSION MANAGEMENT

EUROPÄISCHE UNION
BUNDESREPUBLIK DEUTSCHLAND
REISEPASS
ELAL אל על
הכי בבית בעולם
BOARDING PASS
ZONE C
MUC
LY351
28JUN
49J
C4
1720
WE DEPART ON TIME!
TLV
49J
SEQUENCE NO 0017

VULNERABLE CODE

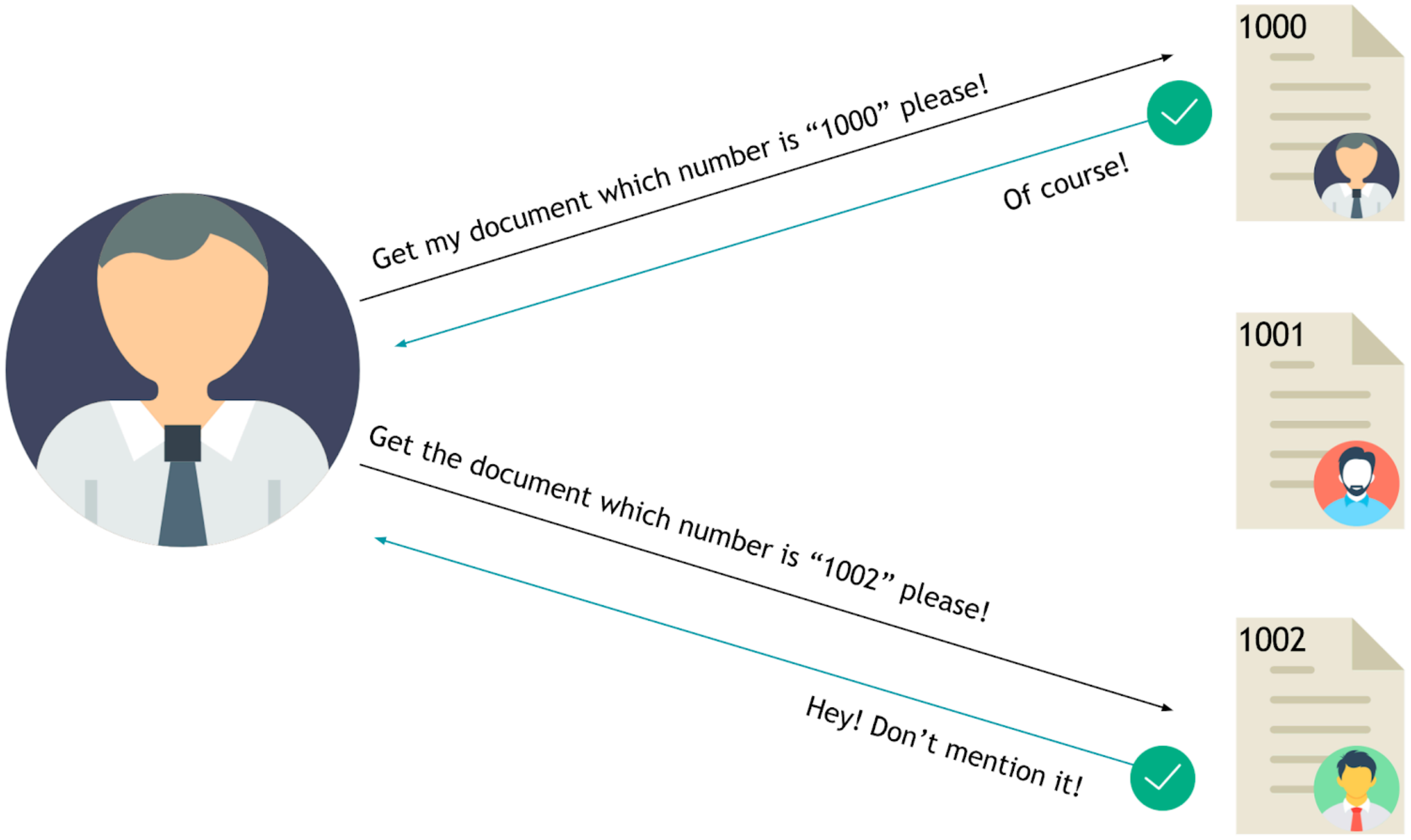
```
route.post('signin_with_barcode', async function(req, res, next) {
  const code = req.query.barcode;
  const user = await User.findOne({ barcode: String(code) });

  req.login(user, function(err){
    if(err) return next(err);
    res.redirect('/home');
  });
}
```

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!
1000
1001
Get the document which number is “1002” please!
Hey! Don’t mention it!
1002

VULNERABLE CODE

```
router.get('/:user_id', async function(req, res, next) {
  try {
    const id = new ObjectID(req.params.user_id);
    const user = await User.findOne(id);
    res.render('users/show', { user });
  } catch (err) {
    next(err);
  }
});
```

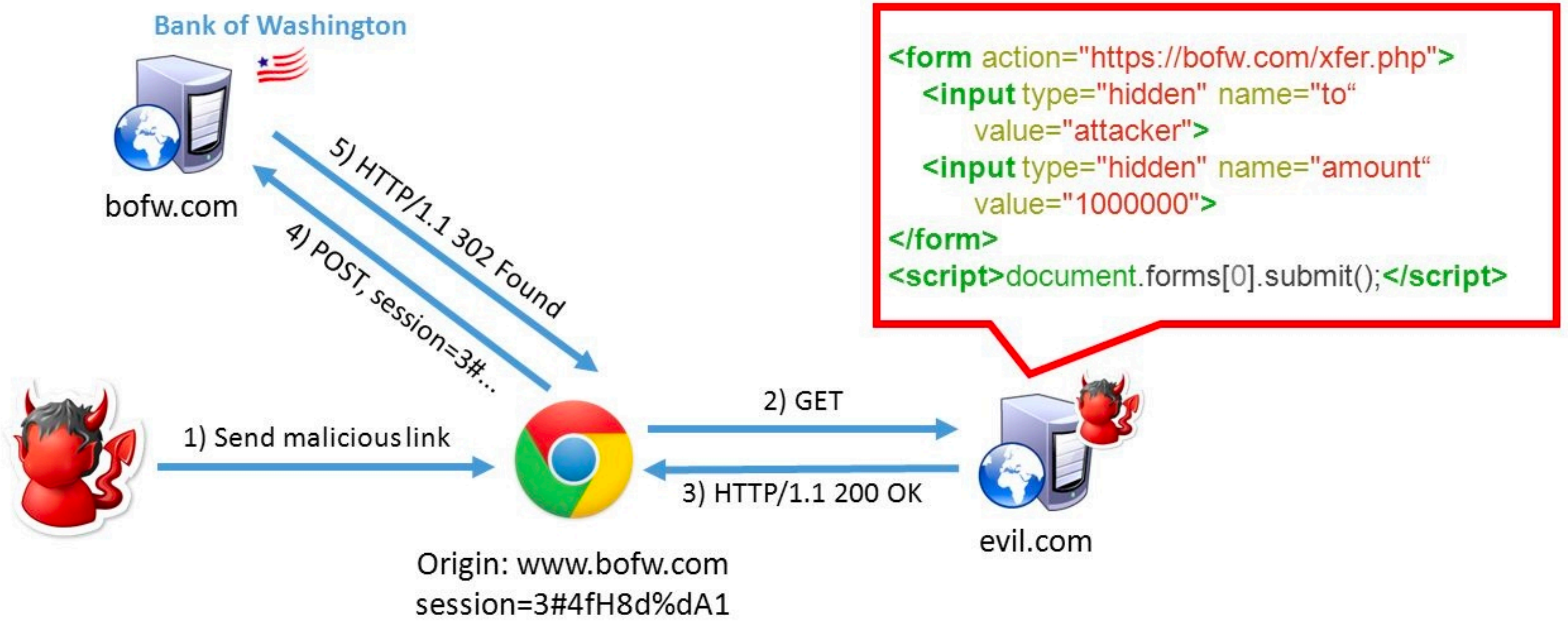
MITIGATION

- Use authorisation framework
- https://github.com/ForbesLindesay/connect-roles
- https://github.com/vadimdemedes/cancan

5. CSRF

CSRF Attack

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



```
<form action="https://bofw.com/xfer.php">
   <input type="hidden" name="to"
      value="attacker">
   <input type="hidden" name="amount"
      value="1000000">
</form>
<script>document.forms[0].submit();</script>
```

VULNERABLE CODE

```html
<form action="/users/delete_account" method="POST">
  <button>Delete Account</button>
</form>
```

MITIGATION

- Always use a CSRF token
- Verify origin header when using tokens
- Limit session duration

6. SECURITY MISCONFIGURATION

VULNERABLE CODE

```
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/users', { useNewUrlParser: true });
```

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 CODE

```
router.get('/admin', function(req, res, next) {
  res.render('admin/index');
});
```

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

https://example.com/login?url=http://example.com/bad/things

```
app.get('/login', function (req, res, next) {

  if(req.session.isAuthenticated()) {

    res.redirect(req.query.url);
  }
});
```

MITIGATIONS

- Always validate input before redirect
- Use a whitelist if redirect list is restricted

Q & A