CS 4530 & CS 5500 Software Engineering

Lecture 9.4: Engineering Secure Software

Jonathan Bell, John Boyland, Mitch Wand
Khoury College of Computer Sciences

Learning Objectives for this Lesson

By the end of this lesson, you should be able to…

- Recognize the causes of and common mitigations for common vulnerabilities in web applications
- Utilize static analysis tools to identify common weaknesses in code

OWASP Top Security Risks

All 10: https://owasp.org/www-project-top-ten/

- Code injection (various forms - SQL/command line/XSS/XML/deserialization)
- Broken authentication + access control
- Weakly protected sensitive data
- Using components with known vulnerabilities

Code Injection Example

OWASP A1:2017-Injection

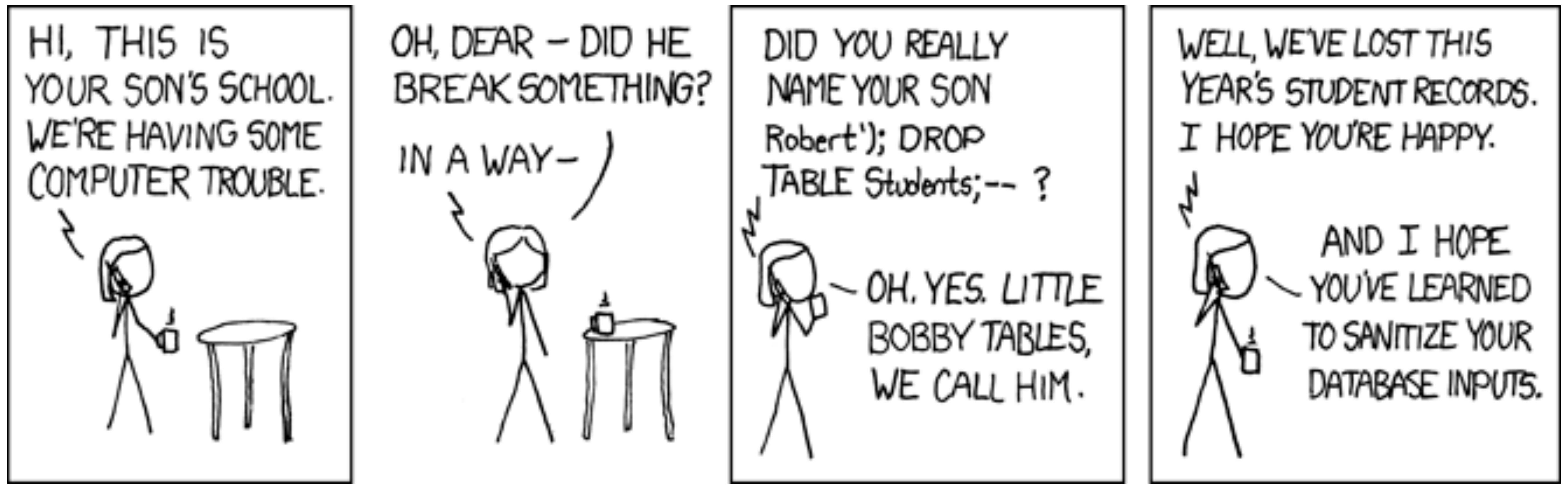
```
String query = "SELECT * FROM accounts WHERE
      name='" + request.getParameter("name") + "'";
```

Parameter `name`	Constructed Query	Effect
Alice	`SELECT * FROM accounts WHERE name='Alice';`	Select a single account
Alice O'Neal	`SELECT * FROM accounts WHERE name='Alice O'Neal';`	SQL Error
5' OR '1'='1	`SELECT * FROM accounts WHERE name='5' OR '1'='1';`	Select all accounts

THIS IS AN ATTACK

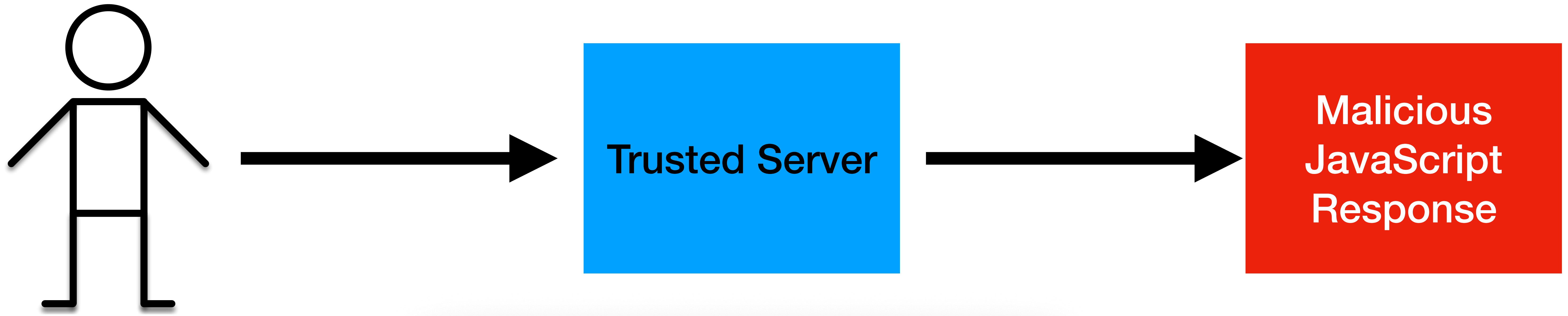
Code Injection Example

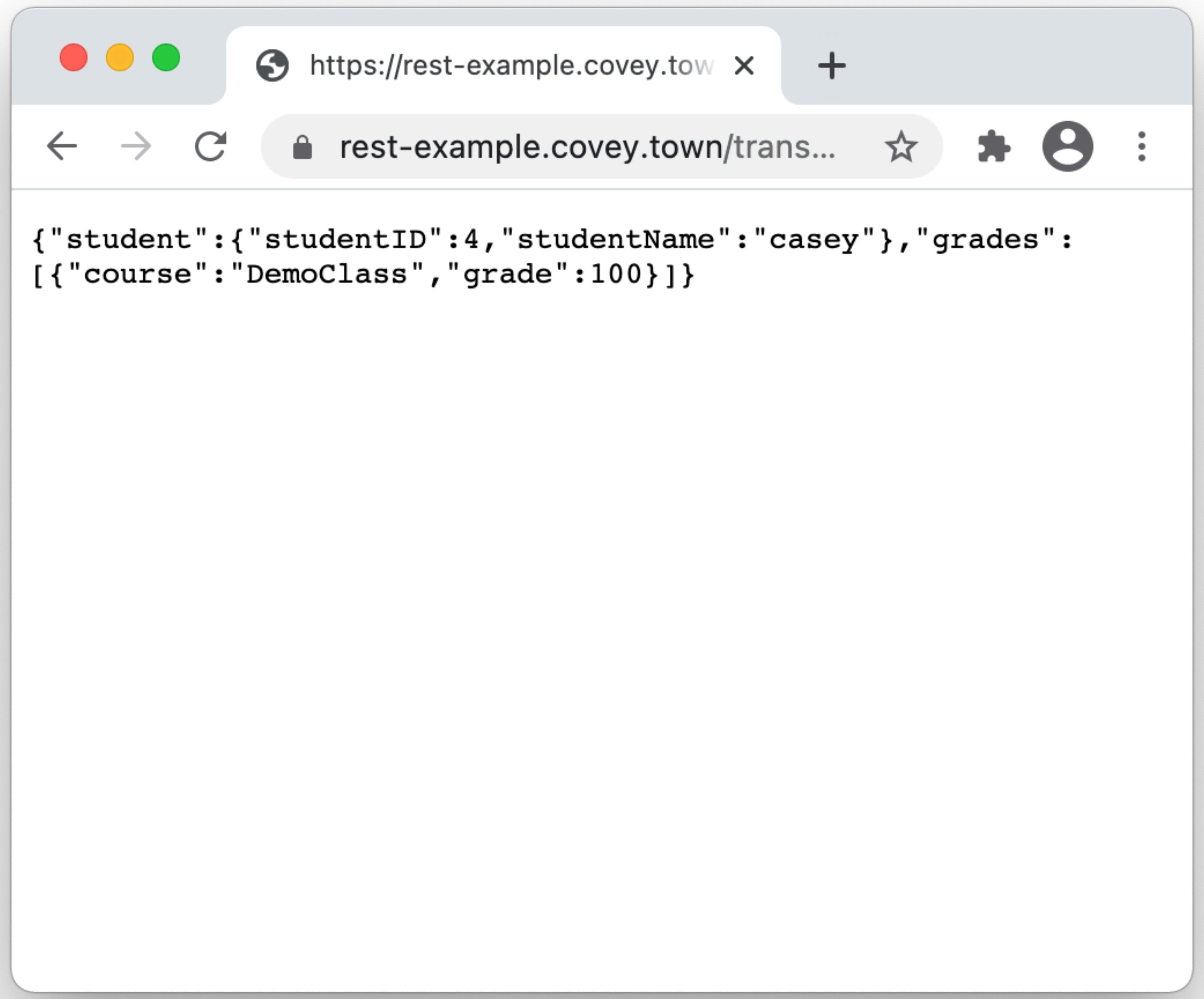
XKCD #327


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.

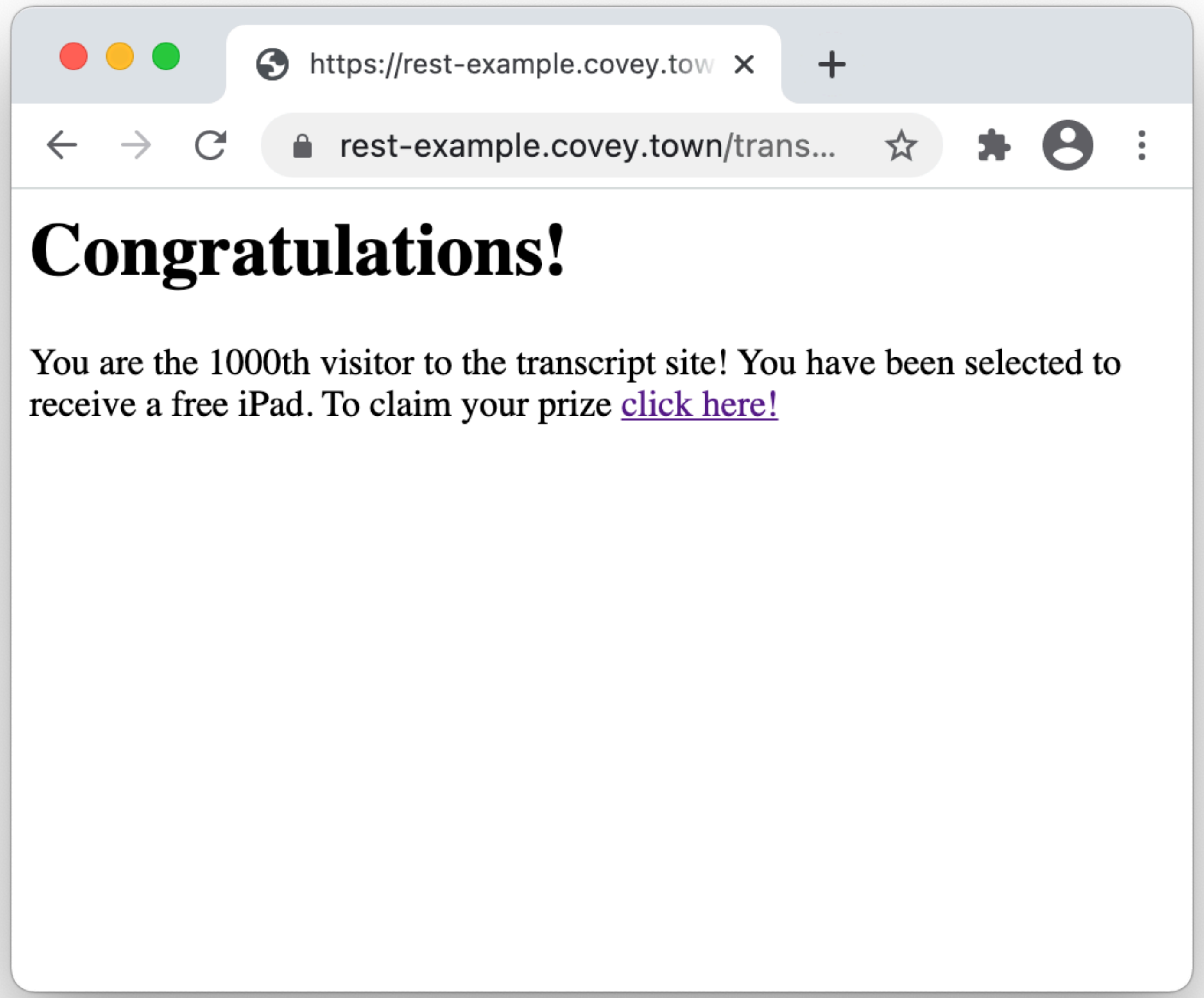
Code Injection Example

Cross-site scripting (XSS)




https://rest-example.covey.tow
rest-example.covey.town/trans...

```
{"student":{"studentID":4,"studentName":"casey"},"grades":
[{"course":"DemoClass","grade":100}]}
```


https://rest-example.covey.tow
rest-example.covey.town/trans...

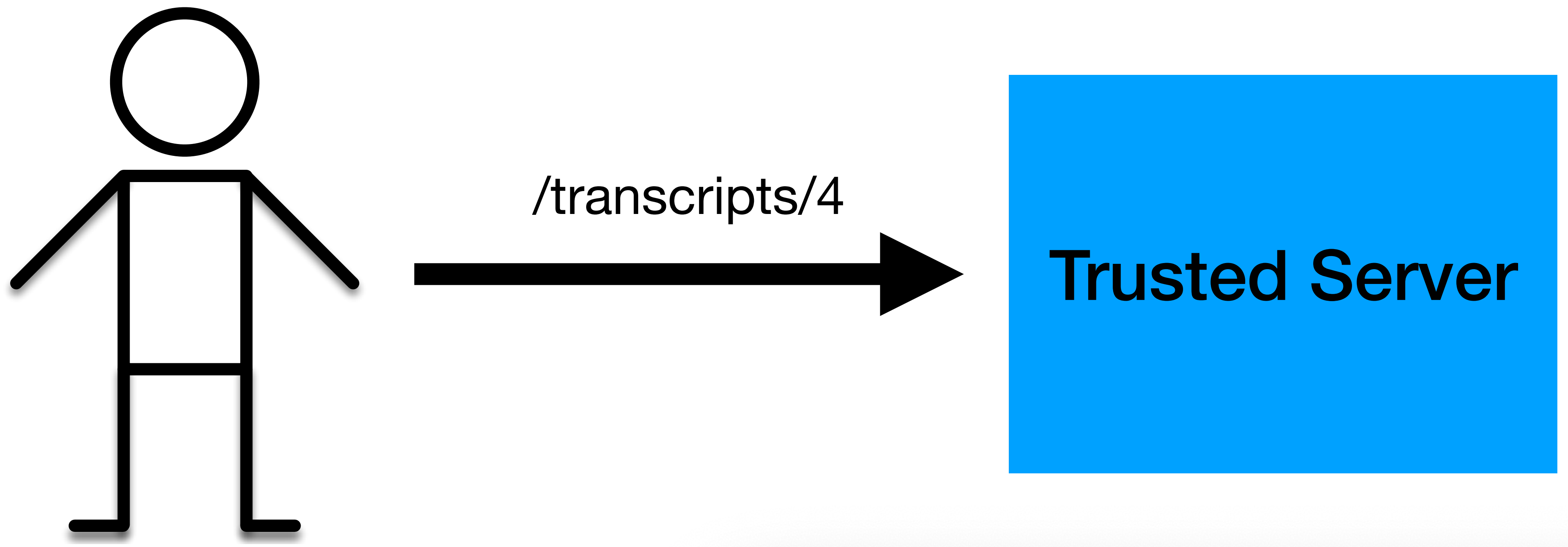
Congratulations!

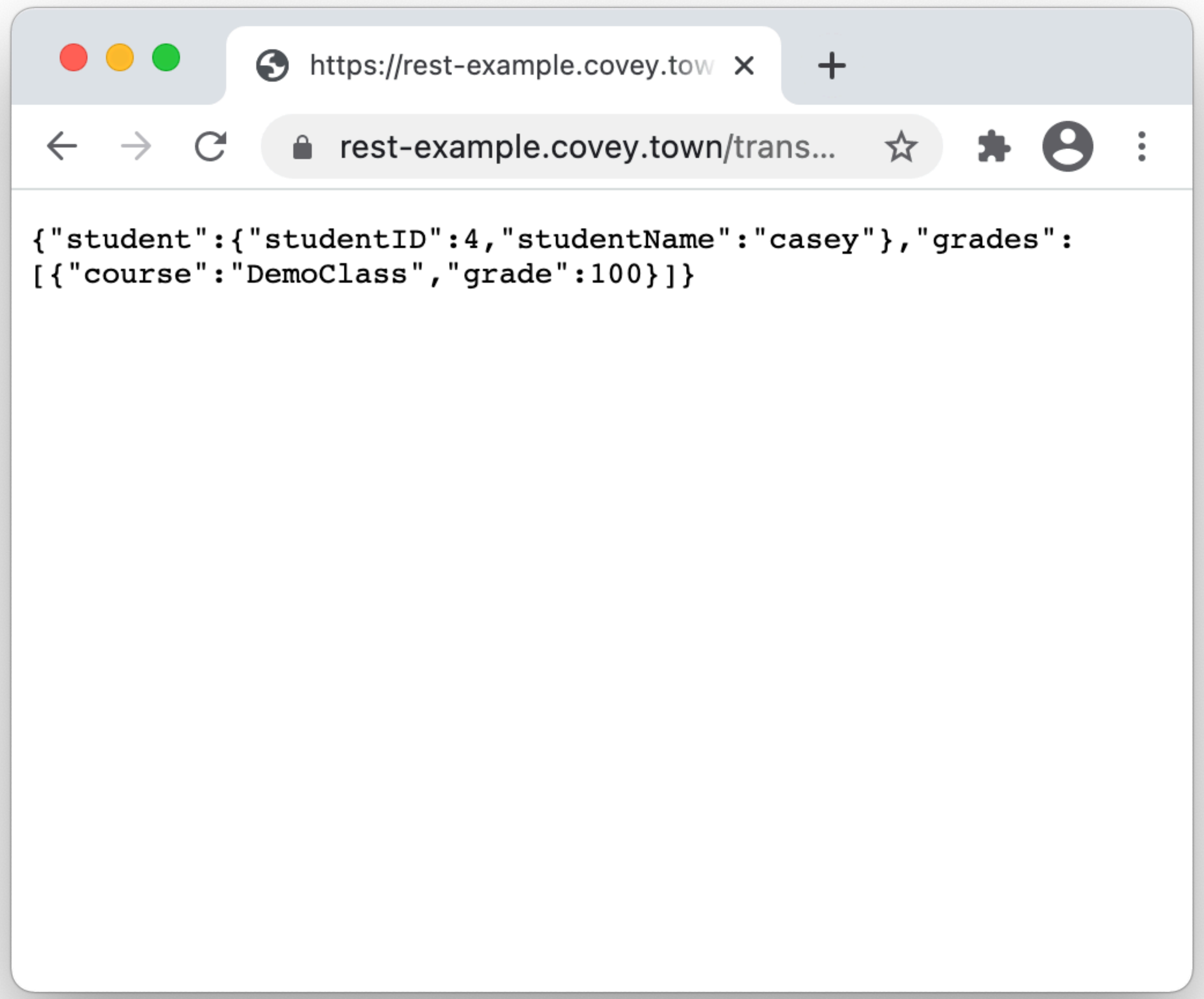
You are the 1000th visitor to the transcript site! You have been selected to receive a free iPad. To claim your prize click here!

Code Injection Example

Cross-site scripting (XSS)

```
app.get('/transcripts/:id', (req, res) => {
  // req.params to get components of the path
  const {id} = req.params;
  const theTranscript = db.getTranscript(parseInt(id));
  if (theTranscript === undefined) {
    res.status(404).send(`No student with id = ${id}`);
  }
  {
    res.status(200).send(theTranscript);
  }
});
```

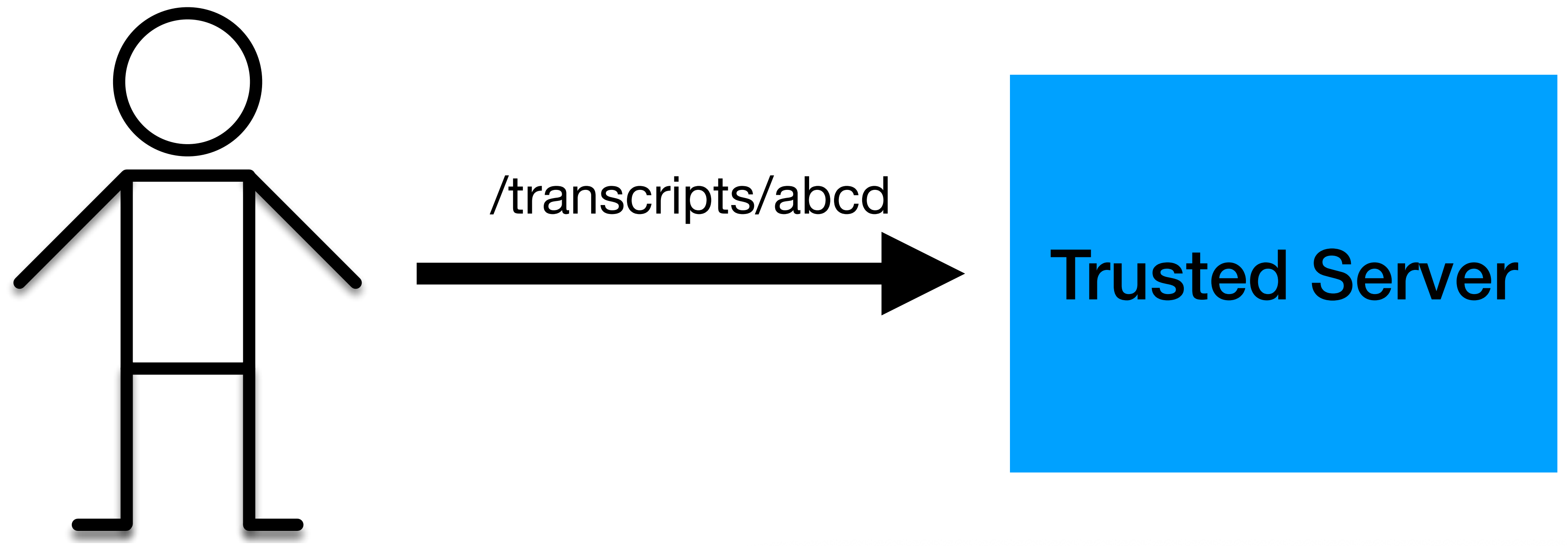



https://rest-example.covey.tow
rest-example.covey.town/trans...

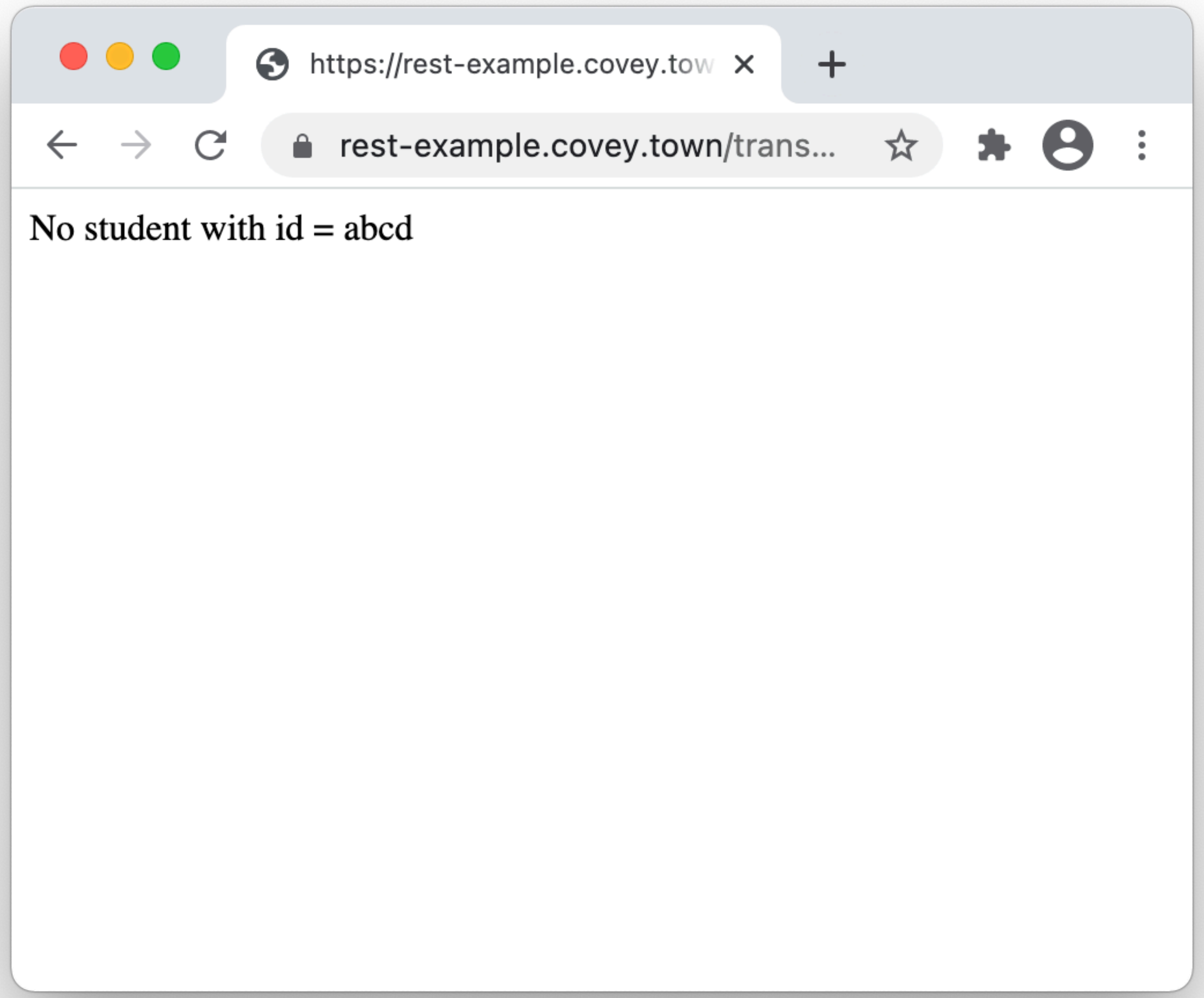
```
{"student":{"studentID":4,"studentName":"casey"},"grades":
[{"course":"DemoClass","grade":100}]}
```

Code Injection Example

Cross-site scripting (XSS)



```
app.get('/transcripts/:id', (req, res) => {
  // req.params to get components of the path
  const {id} = req.params;
  const theTranscript = db.getTranscript(parseInt(id));
  if (theTranscript === undefined) {
    res.status(404).send(`No student with id = ${id}`);
  }
  {
    res.status(200).send(theTranscript);
  }
});
```


https://rest-example.covey.tow
rest-example.covey.town/trans...

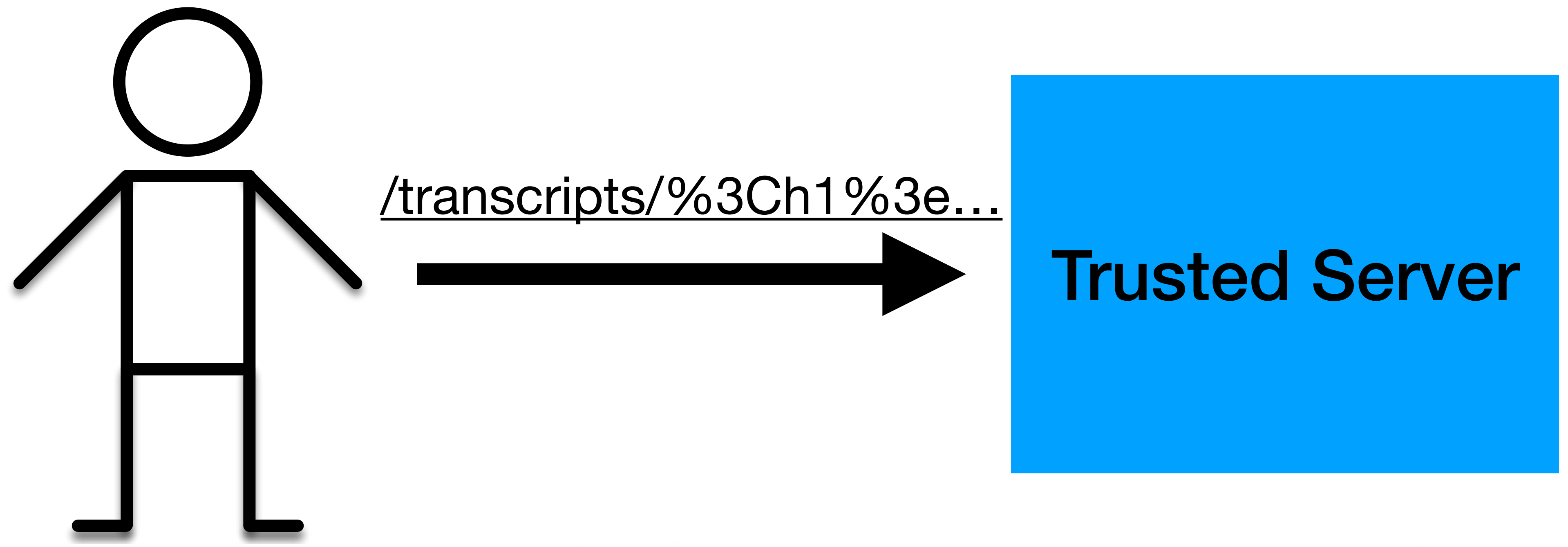
No student with id = abcd

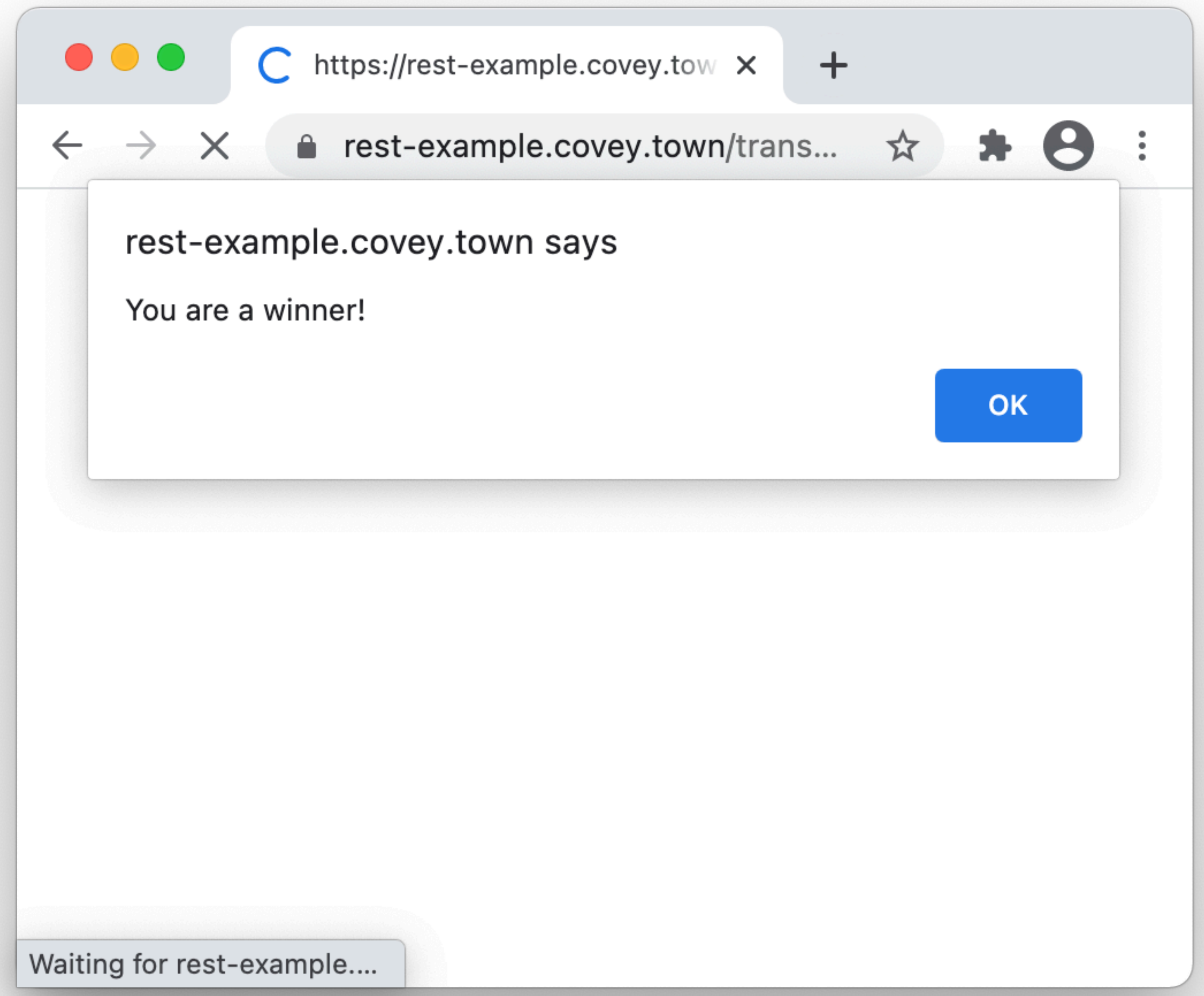
Code Injection Example

Cross-site scripting (XSS)

```
app.get('/transcripts/:id', (req, res) => {
  // req.params to get components of the path
  const {id} = req.params;
  const theTranscript = db.getTranscript(parseInt(id));
  if (theTranscript =  = undefined) {
    res.status(404).   d(`No student with id = ${id}`);
  }
  {
    res.status(200       theTranscript);
  }
});
```

```
<h1>Congratulations!</h1>
  You are the 1000th visitor to the
transcript site! You have been selected
to receive a free iPad. To claim your
prize <a href='https://www.youtube.com/
watch?v=DLzxrzFCyOs'>click here!</a>
  <script language="javascript">
document.getRootNode().body.innerHTML=
'<h1>Congratulations!</h1>You are the
1000th visitor to the transcript site!
You have been selected to receive a
free iPad. To claim your prize <a
href="https://www.youtube.com/watch?
v=DLzxrzFCyOs">click here!</a>';
alert('You are a winner!');
</script>
```

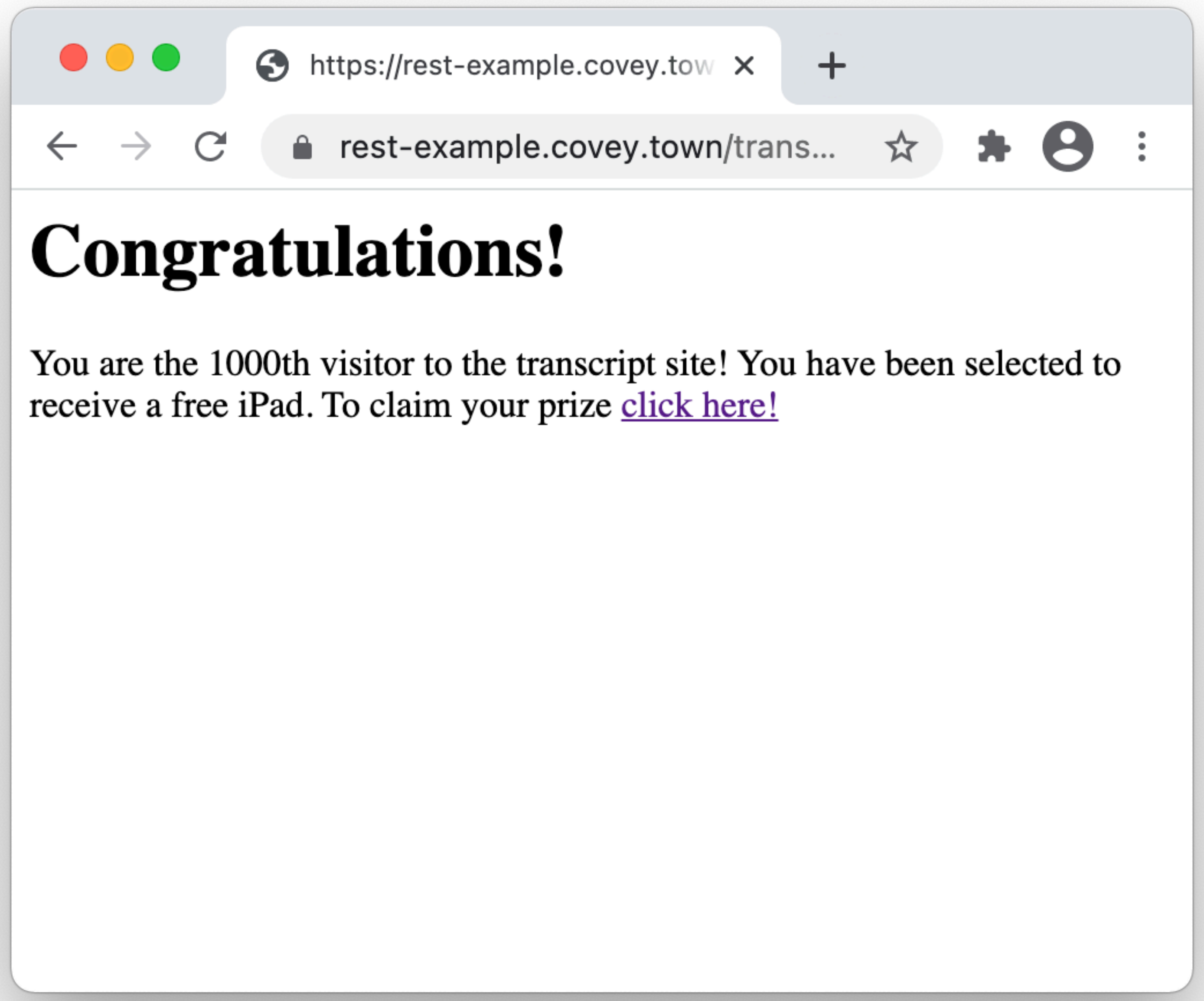



https://rest-example.covey.tow
rest-example.covey.town/trans...
OK

rest-example.covey.town says

You are a winner!

Waiting for rest-example....

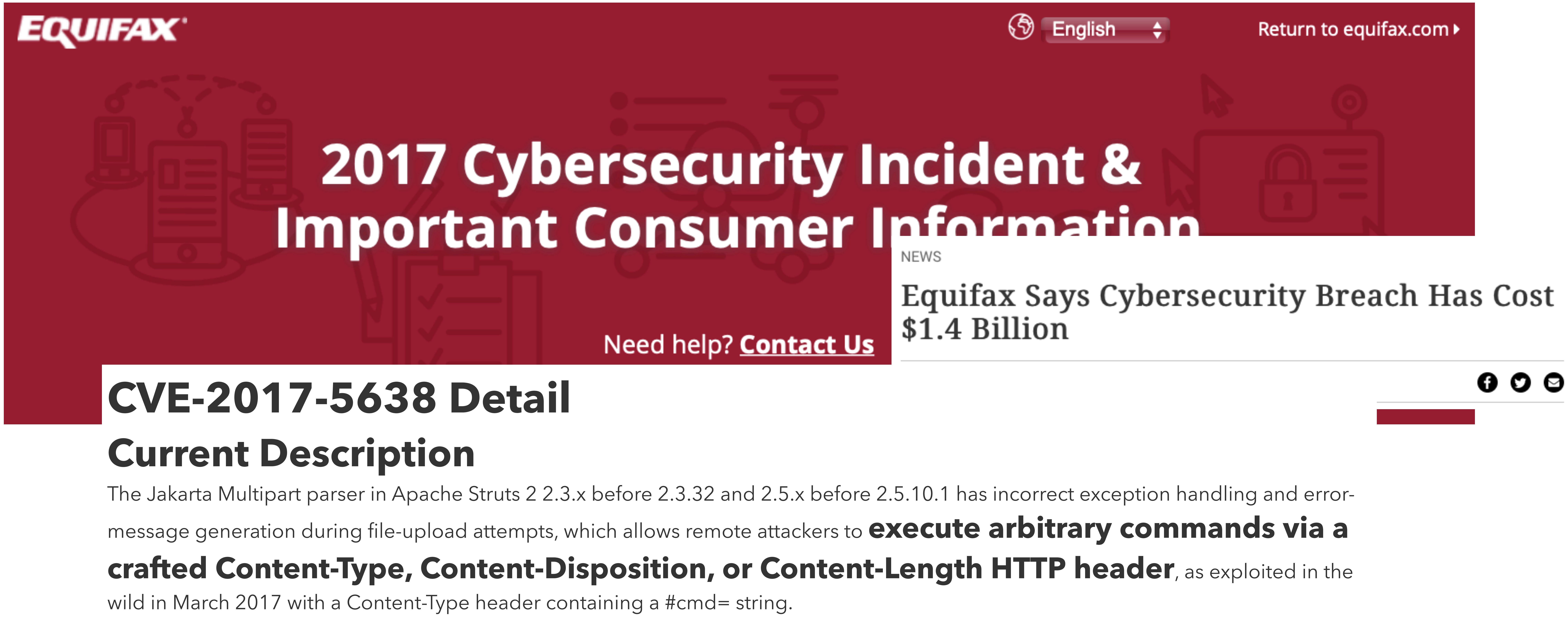

https://rest-example.covey.tow
rest-example.covey.town/trans...

Congratulations!

You are the 1000th visitor to the transcript site! You have been selected to receive a free iPad. To claim your prize click here!

Code Injection Example

Java code injection in Apache Struts (@Equifax)


EQUIFAX

English

Return to equifax.com

2017 Cybersecurity Incident & Important Consumer Information

Need help? Contact Us

NEWS

Equifax Says Cybersecurity Breach Has Cost $1.4 Billion

CVE-2017-5638 Detail

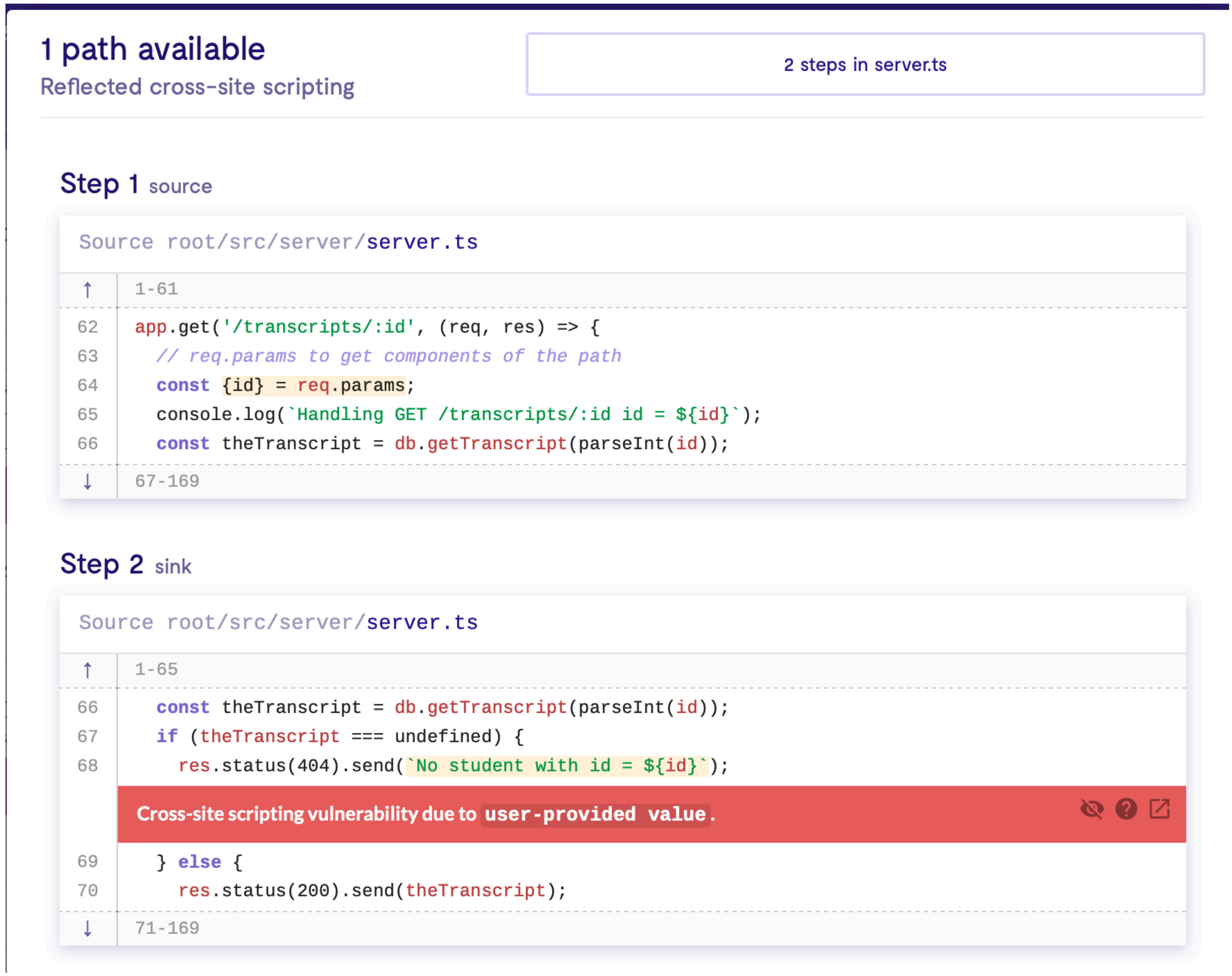
Current Description

The Jakarta Multipart parser in Apache Struts 2 2.3.x before 2.3.32 and 2.5.x before 2.5.10.1 has incorrect exception handling and error-message generation during file-upload attempts, which allows remote attackers to **execute arbitrary commands via a crafted Content-Type, Content-Disposition, or Content-Length HTTP header**, as exploited in the wild in March 2017 with a Content-Type header containing a #cmd= string.

Cross-site Scripting

How to fix it?

- **Sanitize** user-controlled inputs (remove HTML)
- Use tools like LGTM to detect vulnerable data flows
- Use middleware that side-steps the problem (e.g. return data as JSON, client puts that data into React component)



1 path available
Reflected cross-site scripting

2 steps in server.ts

Step 1 source

Source root/src/server/server.ts

```
1-61
app.get('/transcripts/:id', (req, res) => {
  // req.params to get components of the path
  const {id} = req.params;
  console.log(`Handling GET /transcripts/:id id = ${id}`);
  const theTranscript = db.getTranscript(parseInt(id));
67-169
```

Step 2 sink

Source root/src/server/server.ts

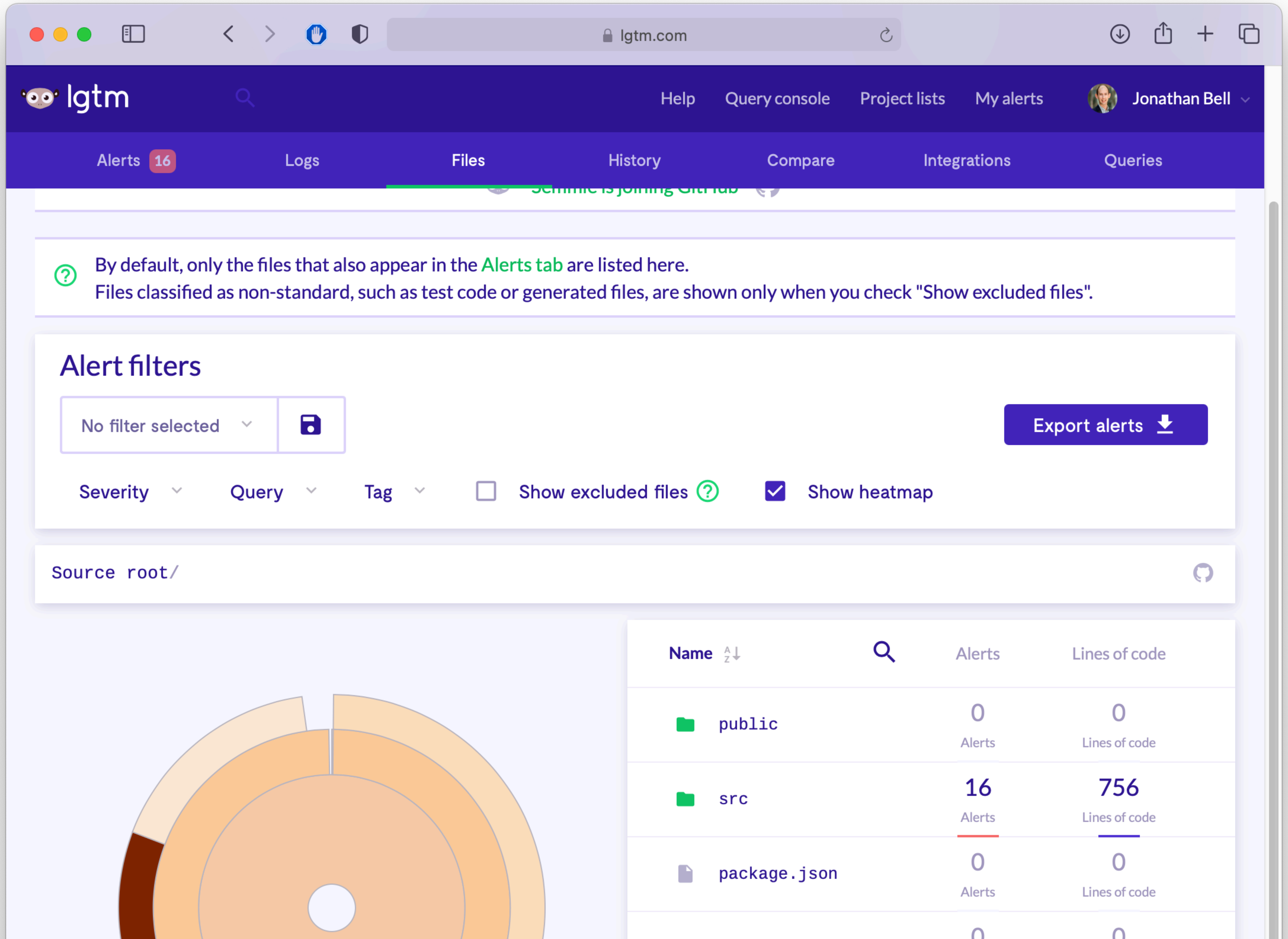
```
1-65
  const theTranscript = db.getTranscript(parseInt(id));
  if (theTranscript === undefined) {
    res.status(404).send(`No student with id = ${id}`);
```

Cross-site scripting vulnerability due to `user-provided value`.

```
  } else {
    res.status(200).send(theTranscript);
71-169
```

Detecting Weaknesses in Apps with Static Analysis

LGTM + CodeQL



By default, only the files that also appear in the Alerts tab are listed here.
Files classified as non-standard, such as test code or generated files, are shown only when you check "Show excluded files".

Alert filters

Source root/

Name	Alerts	Lines of code
public	0 Alerts	0 Lines of code
src	16 Alerts	756 Lines of code
package.json	0 Alerts	0 Lines of code

Clear text storage of sensitive information
Sensitive information stored without encryption or hashing can expose it to an attacker.

Clear-text logging of sensitive information
Logging sensitive information without encryption or hashing can expose it to an attacker.

Client-side cross-site scripting
Writing user input directly to the DOM allows for a cross-site scripting vulnerability.

Client-side URL redirect
Client-side URL redirection based on unvalidated user input may cause redirection to malicious web sites.

Code injection
Interpreting unsanitized user input as code allows a malicious user arbitrary code execution.

Download of sensitive file through insecure connection
Downloading executables and other sensitive files over an insecure connection opens up for potential man-in-the-middle attacks.

https://lgtm.com

OWASP Top Security Risks

All 10: https://owasp.org/www-project-top-ten/

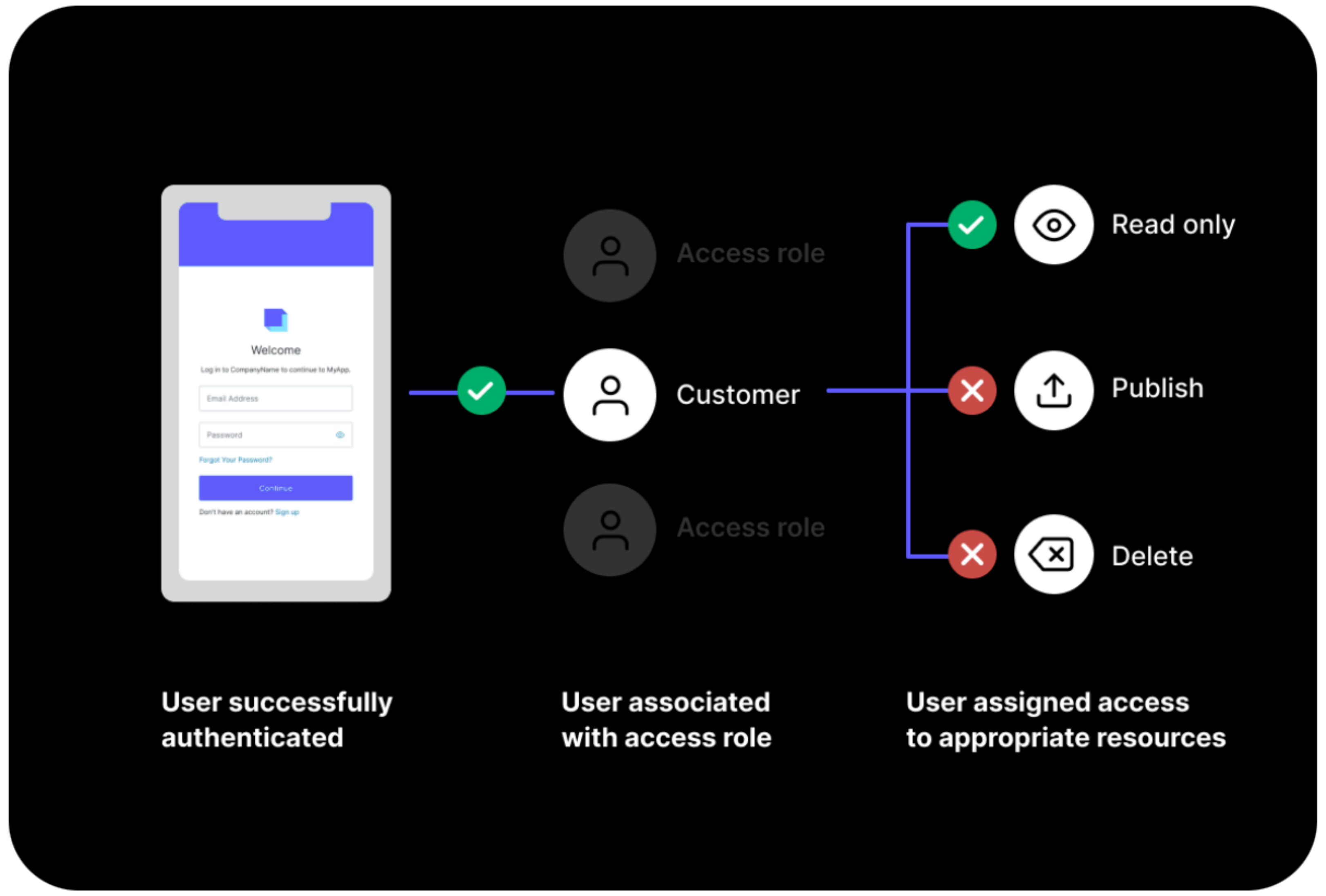
- Code injection (various forms - SQL/command line/XSS/XML/deserialization)
- Broken authentication + access control
- Weakly protected sensitive data
- Using components with known vulnerabilities

Broken Authentication + Access Control

How to fix it?

- Implement multi-factor authentication
- Implement weak-password checks
- Apply per-record access control
- Harden account creation, password reset pathways
- The software engineering approach: rely on a trusted component

Auth0

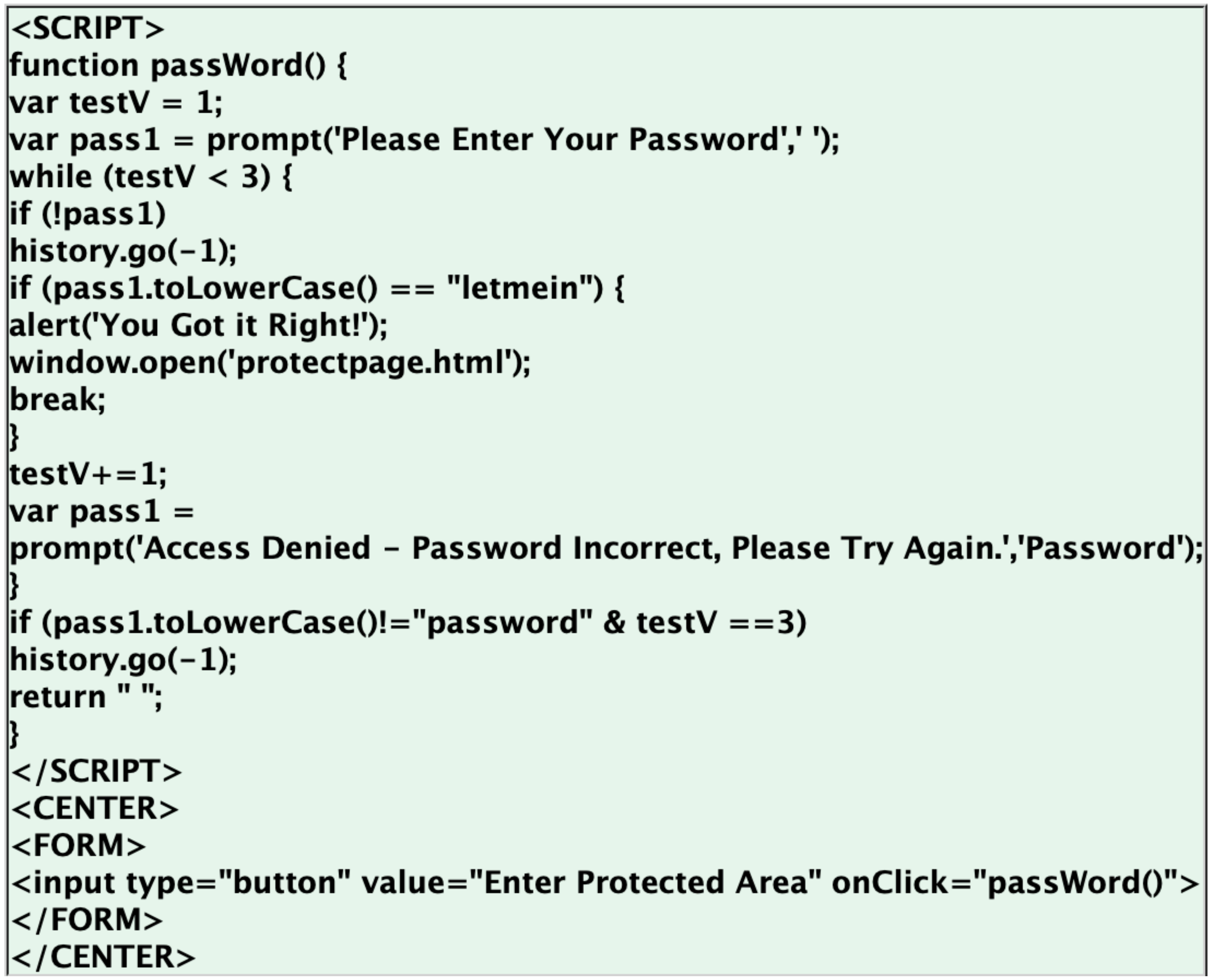


Welcome
Access role
Customer
Access role
Read only
Publish
Delete
User successfully authenticated
User associated with access role
User assigned access to appropriate resources

https://auth0.com

Broken Authentication + Access Control

Specifically: CWE-798: Use of Hard-coded Credentials



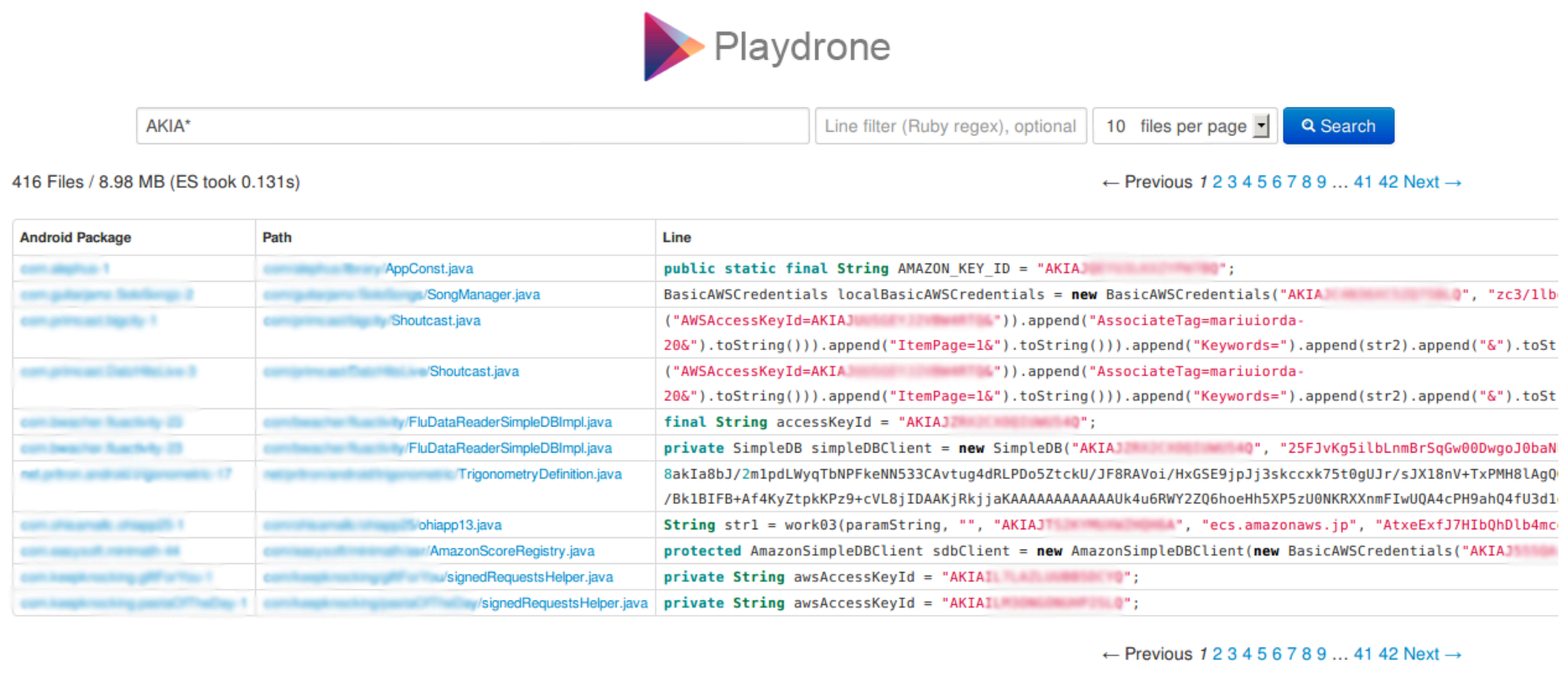
```
<SCRIPT>
function passWord() {
var testV = 1;
var pass1 = prompt('Please Enter Your Password',' ');
while (testV < 3) {
if (!pass1)
history.go(-1);
if (pass1.toLowerCase() == "letmein") {
alert('You Got it Right!');
window.open('protectpage.html');
break;
}
testV+=1;
var pass1 =
prompt('Access Denied - Password Incorrect, Please Try Again.','Password');
}
if (pass1.toLowerCase()!="password" & testV ==3)
history.go(-1);
return " ";
}
</SCRIPT>
<CENTER>
<FORM>
<input type="button" value="Enter Protected Area" onClick="passWord()">
</FORM>
</CENTER>
```

Broken Authentication + Access Control

CWE-798: Use of Hard-coded Credentials: Study of 1.1m Android Apps

	Amazon	Facebook	Twitter	Bitly	Flickr	Foursquare	Google	LinkedIn	Titanium
Total candidates	1,241	1,477	28,235	3,132	159	326	414	1,434	1,914
Unique candidates	308	460	6,228	616	89	177	225	181	1,783
Unique % valid	93.5%	71.7%	95.2%	88.8%	100%	97.7%	96.0%	97.2%	99.8%

Table 5: Credentials statistics from June 22, 2013 and validated on November 11, 2013. A credential may consist of an ID token and secret authentication token.



416 Files / 8.98 MB (ES took 0.131s)

← Previous 1 2 3 4 5 6 7 8 9 … 41 42 Next →

Android Package	Path	Line
[redacted]	[redacted]/AppConst.java	`public static final String AMAZON_KEY_ID = "AKIA[redacted]";`
[redacted]	[redacted]/SongManager.java	`BasicAWSCredentials localBasicAWSCredentials = new BasicAWSCredentials("AKIA[redacted]", "zc3/1lb`
[redacted]	[redacted]/Shoutcast.java	`("AWSAccessKeyId=AKIAJ[redacted]")).append("AssociateTag=mariuiorda-20&").toString())).append("ItemPage=1&").toString())).append("Keywords=").append(str2).append("&").toSt`
[redacted]	[redacted]/Shoutcast.java	`("AWSAccessKeyId=AKIA[redacted]")).append("AssociateTag=mariuiorda-20&").toString())).append("ItemPage=1&").toString())).append("Keywords=").append(str2).append("&").toSt`
[redacted]	[redacted]/FluDataReaderSimpleDBImpl.java	`final String accessKeyId = "AKIAJ[redacted]";`
[redacted]	[redacted]/FluDataReaderSimpleDBImpl.java	`private SimpleDB simpleDBClient = new SimpleDB("AKIAJ[redacted]", "25FJvKg5ilbLnmBrSqGw00DwgoJ0baN`
[redacted]	[redacted]/TrigonometryDefinition.java	`8akIa8bJ/2mlpdLWyqTbNPFkeNN533CAvtug4dRLPDo5ZtckU/JF8RAVoi/HxGSE9jpJj3skccxk75t0gUJr/sJX18nV+TxPMH8lAgQ/Bk1BIFB+Af4KyZtpkKPz9+cVL8jIDAAKjRkjjaKAAAAAAAAAAAAUk4u6RWY2ZQ6hoeHh5XP5zU0NKRXXnmFIwUQA4cPH9ahQ4fU3d1`
[redacted]	[redacted]/ohiapp13.java	`String str1 = work03(paramString, "", "AKIAJ[redacted]", "ecs.amazonaws.jp", "AtxeExfJ7HIbQhDlb4mc`
[redacted]	[redacted]/AmazonScoreRegistry.java	`protected AmazonSimpleDBClient sdbClient = new AmazonSimpleDBClient(new BasicAWSCredentials("AKIAJ[redacted]`
[redacted]	[redacted]/signedRequestsHelper.java	`private String awsAccessKeyId = "AKIAI[redacted]";`
[redacted]	[redacted]/signedRequestsHelper.java	`private String awsAccessKeyId = "AKIAI[redacted]";`

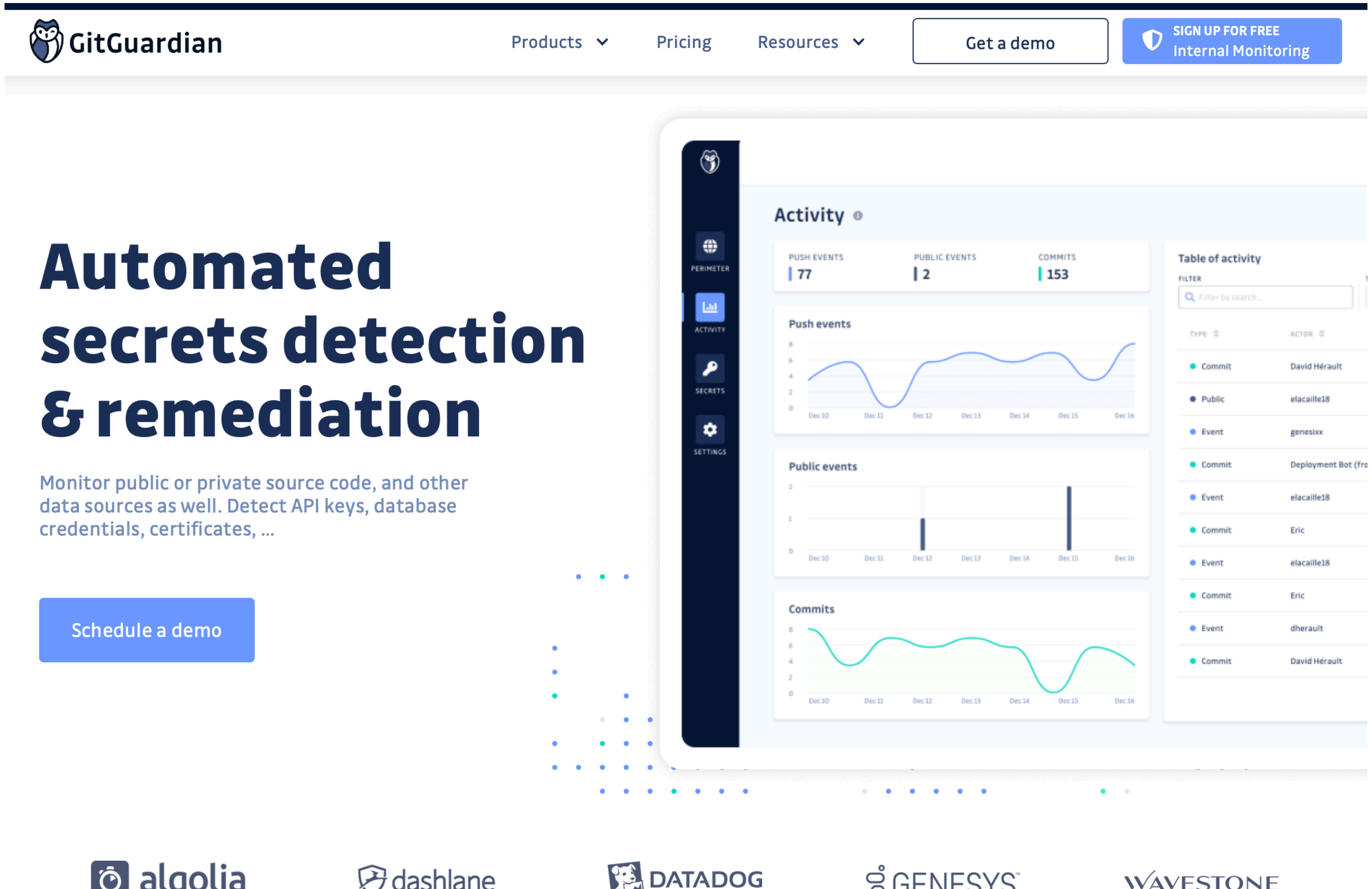
← Previous 1 2 3 4 5 6 7 8 9 … 41 42 Next →

Figure 9: PLAYDRONE's web interface to search decompiled sources showing Amazon Web Service tokens found in 130 ms.

"A Measurement Study of Google Play," Viennot et al, SIGMETRICS '14

Hardcoded Credentials: Automated Checker

GitGuardian (Launched in 2017)



Automated secrets detection & remediation

Monitor public or private source code, and other data sources as well. Detect API keys, database credentials, certificates, …

Activity

PUSH EVENTS	PUBLIC EVENTS	COMMITS
77	2	153

Push events

Public events

Commits

Table of activity

FILTER

Filter by search...

TYPE	ACTOR
Commit	David Hérault
Public	elacaille18
Event	genesixx
Commit	Deployment Bot (fro
Event	elacaille18
Commit	Eric
Event	elacaille18
Commit	Eric
Event	dherault
Commit	David Hérault

OWASP Top Security Risks

All 10: https://owasp.org/www-project-top-ten/

- Code injection (various forms - SQL/command line/XSS/XML/deserialization)
- Broken authentication + access control
- Weakly protected sensitive data
- Using components with known vulnerabilities

Weakly Protected Sensitive Data

How to fix it?

- Classify your data by sensitivity
- Encrypt sensitive data - in transit and at rest
- Make a plan for data controls, stick to it
- Software engineering fix: can we avoid storing sensitive data?
  - Payment processors: Stripe, Square, etc

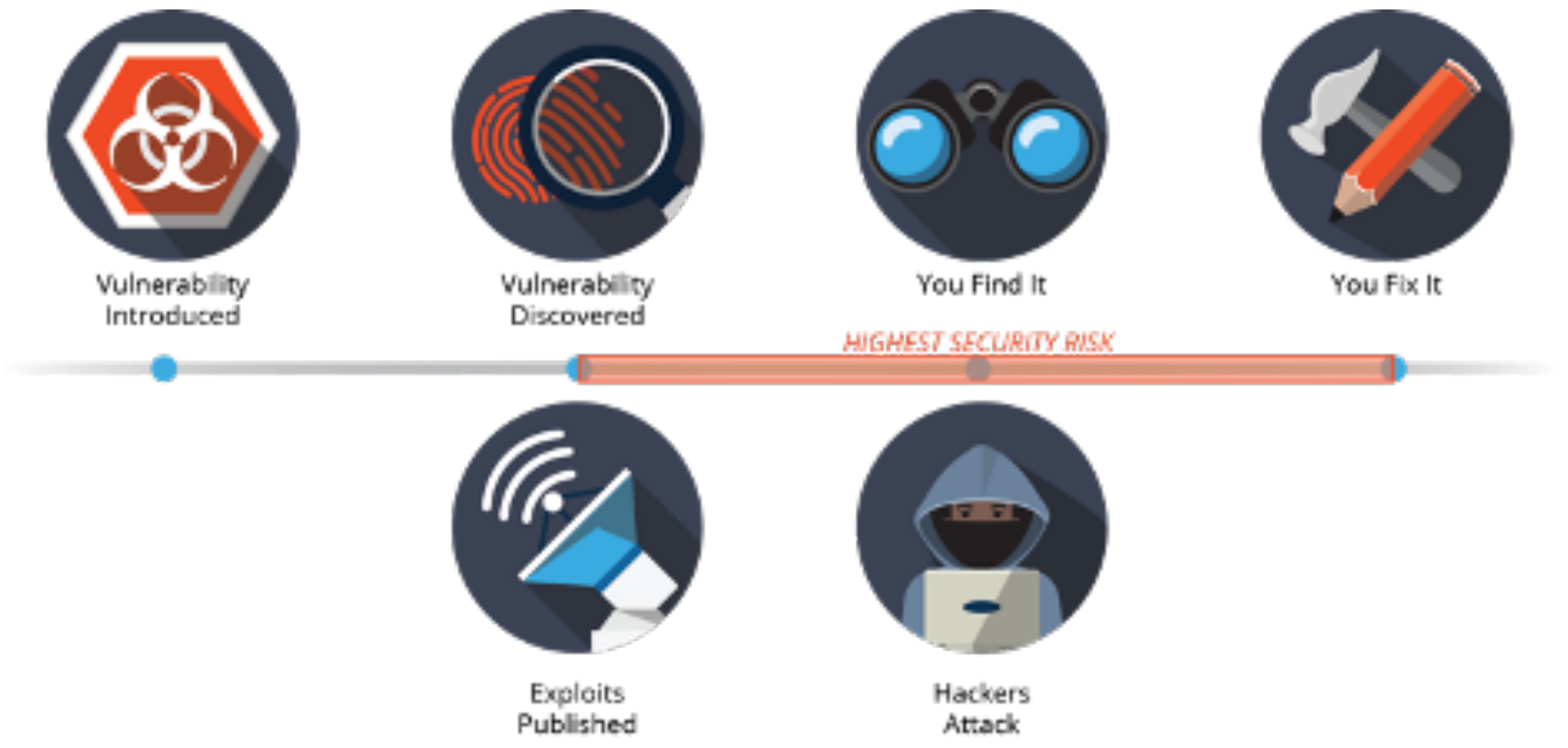
OWASP Top Security Risks

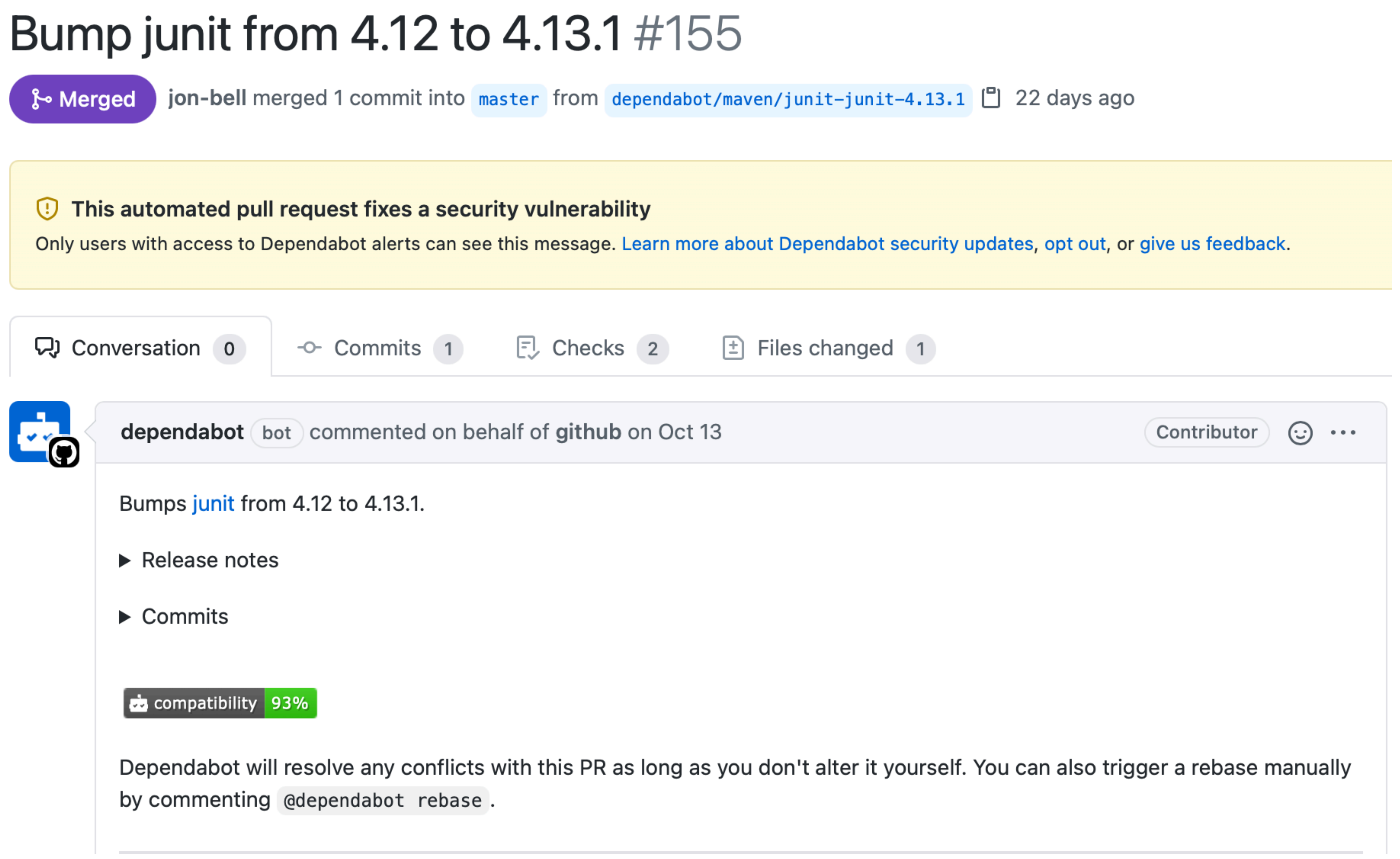
All 10: https://owasp.org/www-project-top-ten/

- Code injection (various forms - SQL/command line/XSS/XML/deserialization)
- Broken authentication + access control
- Weakly protected sensitive data
- Using components with known vulnerabilities

Using Components with Known Vulnerabilties

How to fix it?





Bump junit from 4.12 to 4.13.1 #155

Merged jon-bell merged 1 commit into `master` from `dependabot/maven/junit-junit-4.13.1` 22 days ago

This automated pull request fixes a security vulnerability

Only users with access to Dependabot alerts can see this message. Learn more about Dependabot security updates, opt out, or give us feedback.

dependabot bot commented on behalf of **github** on Oct 13 Contributor

Bumps junit from 4.12 to 4.13.1.

▶ Release notes

▶ Commits

compatibility 93%

Dependabot will resolve any conflicts with this PR as long as you don't alter it yourself. You can also trigger a rebase manually by commenting `@dependabot rebase`.

Learning Objectives for this Lesson

By the end of this lesson, you should be able to…

- Recognize the causes of and common mitigations for common vulnerabilities in web applications
- Utilize static analysis tools to identify common weaknesses in code

This work is licensed under a Creative Commons Attribution-ShareAlike license