



SonarQube SAST Report

generated by ReflectSonar

Date: 2025-10-06 13:29:29

SonarQube Project Name: juice-shop

Security 1 Open Issues	E	Reliability 46 Open Issues	C	Maintainability 452 Open Issues	A
Accepted Issues 0 <small>Valid issues that were not fixed</small>	C	Coverage 0.0% <small>on 15106 lines to cover</small>	B	Duplications 11.4% <small>on 68620 lines</small>	D
Security Hotspots 263 Open Issues					

This report is generated by ReflectSonar, an open-source tool to add the report generation mechanism to SonarQube Community and Developer Edition. It is not affiliated with Sonar. The report is generated based on SonarQube instance that its information is provided. All data is fetched from SonarQube API. ReflectSonar just provides a way to generate the report.




Security Issues

Total: 1 issues (Blocker: 1)

Severity	File Path	Rule & Message
	lib/insecurity.ts (Line 23)	secrets:S6706 Make sure this private key gets revoked, changed, and removed from the code.
<div><div>■ Problematic Code:</div><pre>20: import * as z85 from 'z85' 21: 22: export const publicKey = fs ? fs.readFileSync('encryptionkeys/jwt.pub', 'utf8') : 'placeholder-public-key' >>> 23: const privateKey = '-----BEGIN RSA PRIVATE KEY-----\r\nMIICXAIBAAKBgQDNWqLEe9wgTXCbC7+ RPdDbBbegjdbs4kOPOIGzqLpXvJXlxxW8iMz0EaM4BKUqYsIa+ndv3NAn2RxCd5ubVdJJcX43zO6Ko0TFEZx/65gY3BE0O6 syCEmUP4qbSd6exou/F+WTISzbQ5FBVPVmhYhG/kpwt/cIxK5iUn5hm+4tQIDAQABAoGBAI+8xiPoOrA+KMnG/T4jJsG6T sHQcDHvJi7o1IKC/hnIXha0atTX5AUkRRce95qSfvKFweXdJXSQ0JMGJyfuXgU6dI0TcseFRfewXAa/ssxAC+iUVR6KUMh1 PE2wXLitfeI6JLvVtrBYswm2I7CtY0q8n5AGimHWVXJPLfGV7m0BAkEA+fqFt2LXbLtyg6wZyxMA/cnmt5Nt3U2dAu77MzF JvibANUNHE4HPLZxjGNXN+a6m0K6TD4kDdh5HfUYLWWRBYQJBANK3carmulBwqzcDBjsJ0YrIONBpCAsXxk8idXb8jL9aNI gl5Wumm2enqqObahDHB5jnGOLmbasizvSVqypfM9UCQCQ18xIqy+YgURXzXCn+kwUgHinrutZms87Jyi+D8Br8NY0+Nlf+z HvXAomD2W5CsEK7C+8SLBr3k/TsnRWHJuECQHF9RA2OP8WoaLPuGCyFXaxzICThSRZYluVnWkZtxsBhW2W8z1b8PvWUE7k My7TnkzeJS2LSnaNHoyxi7IaPQUCQCwWU4U+v41D7uYBw00Ga/xt+7+UqFP1PVdz1yyr4q24Zxaw0LgmuEvgU5dycq8N7Jx jTubX0MIRR+G9fmDBB18=\r\n-----END RSA PRIVATE KEY-----' 24: 25: interface ResponseWithUser { 26: status?: string</pre></div>		

Reliability Issues

Total: 46 issues (High: 2, Medium: 14, Low: 30)

Severity	File Path	Rule & Message
	data/datacache.ts (Line 33)	typescript:S6861 Exporting mutable 'let' binding, use 'const' instead.
■ Problematic Code: <pre> 30: } 31: export const notifications: Notification[] = [] 32: >>> 33: export let retrieveBlueprintChallengeFile: string null = null 34: export function setRetrieveBlueprintChallengeFile (retrieveBlueprintChallengeFileArg: string) { 35: retrieveBlueprintChallengeFile = retrieveBlueprintChallengeFileArg 36: }</pre>		
	routes/chatbot.ts (Line 25)	typescript:S6861 Exporting mutable 'let' binding, use 'const' instead.
■ Problematic Code: <pre> 22: 23: let trainingFile = config.get<string>('application.chatBot.trainingData') 24: let testCommand: string >>> 25: export let bot: Bot null = null 26: 27: export async function initializeChatbot () { 28: if (utils.isUrl(trainingFile)) {</pre>		
	frontend/src/app/score-board/components challenge-card challenge-card.component.scss (Line 143)	css:S4656 Unexpected duplicate "color"
■ Problematic Code: <pre> 140: 141: border-radius: calc(\$badge-size / 2); 142: color: var(--theme-text) !important; >>> 143: color: var(--theme-text); 144: cursor: pointer; 145: 146: display: flex;</pre>		
	test/server/blueprintSpec.ts (Line 20)	typescript:S1848 Either remove this useless object instantiation of "ExifImage" or use it.

<p>■ Problematic Code:</p> <pre> 17: async function parseExifData (path: string): Promise<any> { 18: return await new Promise((resolve, reject) => { 19: // eslint-disable-next-line no-new >>> 20: new ExifImage({ image: path }, (error: Error null, exifData: any) => { 21: if (error != null) { 22: expect.fail(`Could not read EXIF data from \${path}`) 23: reject(error) </pre>		
M	frontend/src/app/payment payment.component.html (Line 84)	Web:S6853 A form label must be associated with a control.
<p>■ Problematic Code:</p> <pre> 81: <div class="row responsive-row"> 82: <div class="col col-34 mat-elevation-z0"> 83: <div class="payment-label"> >>> 84: <label translate>LABEL_DONATIONS</label> 85: </div> 86: <small> 87: (@if (applicationName === 'OWASP Juice Shop') { </pre>		
M	frontend/src/app/photo-wall photo-wall.component.html (Line 14)	Web:S6851 Remove redundant word "image" from the "alt" attribute of your "img" tag.
<p>■ Problematic Code:</p> <pre> 11: <div class="grid"> 12: @for (image of slideshowDataSource; track image) { 13: >>> 14: 15: <div class="overlay"> 16: <div>{{ image.caption }}</div> 17: @if (twitterHandle) { </pre>		
M	frontend/src/app/order-completion order-completion.component.html (Line 76)	Web:S5256 Add "" headers to this "".
<p>■ Problematic Code:</p> <pre> 73: <mat-header-cell *matHeaderCellDef translate>LABEL_TOTAL_PRICE</mat-header-cell> 74: <mat-cell *matCellDef="let element" class="price-align">{{ (element.total).toFixed(2) }}&current;</mat-cell> 75: <mat-footer-cell *matFooterCellDef> >>> 76: <table class="price-align"> 77: <tr class="mat-row"> 78: <td>{{ orderDetails.itemTotal?.toFixed(2) }}&current;</td> 79: </tr> </pre>		
M	frontend/src/app/order-summary order-summary.component.html (Line 50)	Web:S5256 Add "" headers to this "".

<p>■ Problematic Code:</p> <pre> 47: <mat-card appearance="outlined" class="mat-elevation-z0 fill"> 48: <div class="mdc-card"> 49: <div class="order-summary">{{"ORDER_SUMMARY" translate}}</div> >>> 50: <table class="mat-table"> 51: <tr class="mat-row"> 52: <td class="mat-cell label">{{"ITEMS" translate}}</td> 53: <td class="mat-cell price">{{ itemTotal?.toFixed(2) }}&current;</td> </pre>		
M	frontend/src/app/order-completion/order-completion.component.html (Line 56)	Web:S5256 Add "" headers to this "".
<p>■ Problematic Code:</p> <pre> 53: {{element.quantity}} 54: </mat-cell> 55: <mat-footer-cell *matFooterCellDef> >>> 56: <table class="mat-table"> 57: <tr class="mat-row"> 58: <td translate>ITEMS</td> 59: </tr> </pre>		
M	frontend/src/app/contact/contact.component.html (Line 38)	Web:S6853 A form label must be associated with a control.
<p>■ Problematic Code:</p> <pre> 35: </mat-form-field> 36: 37: <div class="rating-container"> >>> 38: <label style="font-weight:500; margin-right: 8px; float:left;" translate>LABEL_RATING</label> 39: <mat-slider id="rating" min="1" max="5" step="1" showTickMarks discrete [displayWith]="formatRating" aria-label="Slider for selecting the star rating"> 40: <input matSliderThumb [(ngModel)]="rating" /> 41: </mat-slider> </pre>		
M	frontend/src/app/contact/contact.component.html (Line 45)	Web:S6853 A form label must be associated with a control.
<p>■ Problematic Code:</p> <pre> 42: </div> 43: 44: <div style="margin-bottom: 10px; margin-top: 10px; "> >>> 45: <label style="font-weight:500;">CAPTCHA:</label>&nbsp;&nbsp;&nbsp;LABEL_WHAT_IS&nbsp;&nbsp;&nbsp; 47: <code id="captcha" aria-label="CAPTCHA code which must be solved">{{captcha}}</code>&nbsp;&nbsp;&nbsp;<label 48: style="font-size:small;">?</label> </pre>		
M	frontend/src/app/contact/contact.component.html (Line 47)	Web:S6853 A form label must be associated with a control.

<p>■ Problematic Code:</p> <pre> 44: <div style="margin-bottom: 10px; margin-top: 10px; "> 45: <label style="font-weight:500;">CAPTCHA:</label>&nbsp;&nbsp;LABEL_WHAT_IS&nbsp;&nbsp; >>> 47: <code id="captcha" aria-label="CAPTCHA code which must be solved">{{captcha}}</code>&nbsp;<label 48: style="font-size:small;">?</label> 49: </div> 50: <mat-form-field appearance="outline" color="accent"> </pre>		
M	frontend/src/app/payment payment.component.html (Line 118)	Web:S6853 A form label must be associated with a control.
<p>■ Problematic Code:</p> <pre> 115: 116: <div class="col col-65 mat-elevation-z0"> 117: <div class="payment-label"> >>> 118: <label translate>LABEL_MERCHANDISE</label> 119: </div> 120: <small> 121: (@if (applicationName === 'OWASP Juice Shop') { </pre>		
M	routes/languages.ts (Line 69)	typescript:S6671 Expected the Promise rejection reason to be an Error.
<p>■ Problematic Code:</p> <pre> 66: } 67: resolve((differentStrings / totalStrings) * 100) 68: } catch (err) { >>> 69: reject(err) 70: } 71: }) 72: } </pre>		
M	lib/startup/validatePreconditions.ts (Line 103)	typescript:S6671 Expected the Promise rejection reason to be an Error.
<p>■ Problematic Code:</p> <pre> 100: return await new Promise((resolve, reject) => { 101: portscanner.checkPortStatus(portNumber, function (error: unknown, status: string) { 102: if (error) { >>> 103: reject(error) 104: } else { 105: if (status === 'open') { 106: logger.warn(`Port \${colors.bold(port.toString())} is in use (\${colors.red('NOT OK')})`) </pre>		
M	frontend/src/assets/private threejs-demo.html (Line 2)	Web:S5254 Add "lang" and/or "xml:lang" attributes to this "" element

■ Problematic Code: <pre> 1: <!DOCTYPE html> >>> 2: <html> 3: <head> 4: <title>Welcome to Planet Orangeuze</title> 5: <script src="/assets/private/three.js"></script> </pre>		
L	frontend/src/app/chatbot chatbot.component.html (Line 18)	Web:ImgWithoutAltCheck Add an "alt" attribute to this image.
■ Problematic Code: <pre> 15: @for (message of messages; track message; let index = \$index) { 16: <div class="message-container"> 17: @if (message.author == 'bot') { >>> 18: 19: } 20: @if (message.author == 'user') { 21: </pre>		
L	frontend/src/app/chatbot chatbot.component.html (Line 21)	Web:ImgWithoutAltCheck Add an "alt" attribute to this image.
■ Problematic Code: <pre> 18: 19: } 20: @if (message.author == 'user') { >>> 21: 22: } 23: <div class="{{message.author == 'user' ? 'speech-bubble-right' : 'speech-bubble-left'}}"> 24: {{message.body}} </pre>		
L	frontend/src/app/code-area code-area.component.html (Line 2)	Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.
■ Problematic Code: <pre> 1: <div id="code-area"> >>> 2: <pre id="code"><div id="emphasize">@for (marker of lineMarkers; track marker) {<div class="lineMarker" [id]='line' + marker.lineNumber" (click)="selectLines(marker.lineNumber)">{{ marker.marked ? '■' : '■'}}</div>} 3: </div><code 4: [highlight]="code" [lineNumbers]="true" 5: [languages]="langs" </pre>		
L	frontend/src/app/sidenav sidenav.component.html (Line 54)	Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.

<p>■ Problematic Code:</p> <pre> 51: } 52: 53: @if (isLoggedIn()) { >>> 54: <mat-list-item (click)="showOrdersSubmenu = !showOrdersSubmenu" class="parent" 55: aria-label="Show Orders and Payment Menu"> 56: <mat-icon> 57: check_circle_outline </pre>	
<p>L</p> <p>frontend/src/app/sidenav sidenav.component.html (Line 129)</p>	<p>Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.</p>
<p>■ Problematic Code:</p> <pre> 126: } 127: 128: @if (isLoggedIn()) { >>> 129: <mat-list-item (click)="showPrivacySubmenu = !showPrivacySubmenu" class="parent" 130: aria-label="Show Privacy and Security Menu"> 131: <mat-icon> 132: security </pre>	
<p>L</p> <p>frontend/src/app/address address.component.html (Line 19)</p>	<p>Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.</p>
<p>■ Problematic Code:</p> <pre> 16: <ng-container matColumnDef="Selection"> 17: <mat-header-cell *matHeaderCellDef style="display: none;"></mat-header-cell> 18: <mat-cell *matCellDef="let element; let row"> >>> 19: <mat-radio-button 20: (click)="emitSelectionToParent(element.id)" 21: [checked]="selection.isSelected(row)" 22: (change)=" \$event ? selection.toggle(row) : null"> </pre>	
<p>L</p> <p>frontend/src/app/address address.component.html (Line 61)</p>	<p>Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.</p>
<p>■ Problematic Code:</p> <pre> 58: </mat-cell> 59: </ng-container> 60: <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row> >>> 61: <mat-row 62: *matRowDef="let row; columns: displayedColumns; let element" 63: (click)="selection.toggle(row); emitSelectionToParent(element.id)"> 64: </mat-row> </pre>	
<p>L</p> <p>frontend/src/app/search-result search-result.component.html (Line 38)</p>	<p>Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.</p>

<p>■ Problematic Code:</p> <pre> 35: {{ "LABEL_SOLD_OUT" translate }} 36: </div> 37: } >>> 38: <div class="product" (click)="showDetail(item)" aria-label="Click for more information about the product" 39: matTooltip="Click for more information" matTooltipPosition="above"> 40: </pre>		
<p>L</p>	<p>frontend/src/app/navbar navbar.component.html (Line 134)</p>	<p>Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.</p>
<p>■ Problematic Code:</p> <pre> 131: </mat-form-field> 132: </div> 133: @for (language of filteredLanguages; track language.key) { >>> 134: <mat-radio-button 135: (click)="changeLanguage(language.key)" 136: [value]="language" 137: class="mat-menu-item" </pre>		
<p>L</p>	<p>frontend/src/app/navbar navbar.component.html (Line 126)</p>	<p>Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.</p>
<p>■ Problematic Code:</p> <pre> 123: </button> 124: 125: <mat-menu #menu="matMenu" [overlapTrigger]="true"> >>> 126: <div class="language-search-container" (click)="\$event.stopPropagation()"> 127: <mat-form-field class="language-search-field"> 128: <mat-icon matPrefix class="search-icon">search</mat-icon> 129: <mat-label>{{ 'SEARCH' translate }}</mat-label> </pre>		
<p>L</p>	<p>frontend/src/app/administration administration.component.html (Line 59)</p>	<p>Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.</p>
<p>■ Problematic Code:</p> <pre> 56: 57: <ng-container matColumnDef="comment"> 58: <mat-header-cell *matHeaderCellDef translate></mat-header-cell> >>> 59: <mat-cell *matCellDef="let feedback" (click)="showFeedbackDetails(feedback.comment, feedback.UserId)"> 60: <p [innerHTML]="feedback.comment" matTooltip="Click for more information" matTooltipPosition="above"></p> 61: </mat-cell> 62: </ng-container> </pre>		
<p>L</p>	<p>frontend/src/app/administration administration.component.html (Line 119)</p>	<p>Web:ImgWithoutAltCheck Add an "alt" attribute to this image.</p>

■ Problematic Code: <pre> 116: <mat-row *matRowDef="let row; columns: userColumns; "></mat-row> 117: </mat-table> 118: >>> 119: </pre>		
L	frontend/src/app/mat-search-bar mat-search-bar.component.html (Line 9)	Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.
■ Problematic Code: <pre> 6: 7: 8: >>> 9: <mat-icon class="mat-search_icon-close" (click)="close()" matRipple> 10: close 11: </mat-icon> 12: <mat-icon class="mat-search_icon-search" (click)="open()" matRipple> </pre>		
L	frontend/src/app/mat-search-bar mat-search-bar.component.html (Line 12)	Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.
■ Problematic Code: <pre> 9: <mat-icon class="mat-search_icon-close" (click)="close()" matRipple> 10: close 11: </mat-icon> >>> 12: <mat-icon class="mat-search_icon-search" (click)="open()" matRipple> 13: search 14: </mat-icon> 15: </pre>		
L	frontend/src/app/web3-sandbox web3-sandbox.component.html (Line 169)	Web:ImgWithoutAltCheck Add an "alt" attribute to this image.
■ Problematic Code: <pre> 166: } 167: </div> 168: </div> >>> 169: 170: </div> 171: </pre>		
L	frontend/src/app/faucet faucet.component.html (Line 6)	Web:ImgWithoutAltCheck Add an "alt" attribute to this image.

<p>■ Problematic Code:</p> <pre> 3: <div class="mdc-card"> 4: <h1>{{"TITLE_BEE_HAVEN" translate}}</h1> 5: <div class="faucet-image"> >>> 6: 7: </div> 8: <p>{{"BEE_HAVEN_INTRO" translate}}</p> 9: <div class="faucet-balance"> </pre>		
<p>L</p>	<p>frontend/src/app/faucet faucet.component.html (Line 67)</p>	<p>Web:ImgWithoutAltCheck Add an "alt" attribute to this image.</p>
<p>■ Problematic Code:</p> <pre> 64: <mat-card appearance="outlined" class="mat-elevation-z6 honeypot-container"> 65: <div class="mdc-card"> 66: <div class="honeypot-image mat-elevation-z6"> >>> 67: 68: </div> 69: <h2>{{"BEE_HONEYPOT_TITLE" translate}}</h2> 70: <p>{{"BEE_HONEYPOT_DESCRIPTION" translate}}</p> </pre>		
<p>L</p>	<p>frontend/src/app/nft-unlock nft-unlock.component.html (Line 13)</p>	<p>Web:ImgWithoutAltCheck Add an "alt" attribute to this image.</p>
<p>■ Problematic Code:</p> <pre> 10: 11: <div class="detail-container offer-container align-self-center"> 12: <div> >>> 13: 14: </div> 15: 16: @if (successResponse) { </pre>		
<p>L</p>	<p>frontend/src/app/delivery-method delivery-method.component.html (Line 28)</p>	<p>Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.</p>
<p>■ Problematic Code:</p> <pre> 25: <ng-container matColumnDef="Selection"> 26: <mat-header-cell *matHeaderCellDef></mat-header-cell> 27: <mat-cell *matCellDef="let element; let row"> >>> 28: <mat-radio-button (click)=selectMethod(element.id) [checked]="selection.isSelected(row)" 29: (change)=" \$event ? selection.toggle(row) : null"></mat-radio-button> 30: </mat-cell> 31: </ng-container> </pre>		
<p>L</p>	<p>frontend/src/app/delivery-method delivery-method.component.html (Line 45)</p>	<p>Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.</p>

<p>■ Problematic Code:</p> <pre> 42: <mat-cell *matCellDef="let element">{{ element.eta }} {{ "LABEL_DAYS" translate}}</mat-cell> 43: </ng-container> 44: <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row> >>> 45: <mat-row *matRowDef="let row; columns: displayedColumns; let element" 46: (click)="selection.toggle(row); selectMethod(element.id)"></mat-row> 47: </mat-table> 48: </div> </pre>		
L	frontend/src/app/token-sale token-sale.component.html (Line 101)	Web:ImgWithoutAltCheck Add an "alt" attribute to this image.
<p>■ Problematic Code:</p> <pre> 98: </div> 99: </mat-card> 100: </div> >>> 101: 102: </div> 103: </mat-card> 104: </pre>		
L	frontend/src/app/privacy-policy privacy-policy.component.html (Line 172)	Web:ImgWithoutAltCheck Add an "alt" attribute to this image.
<p>■ Problematic Code:</p> <pre> 169: 170: </section> 171: >>> 172: 173: </div> 174: </mat-card> 175: </pre>		
L	frontend/src/app/search-result search-result.component.html (Line 40)	Web:S6842 Non-interactive elements should not be assigned interactive roles.
<p>■ Problematic Code:</p> <pre> 37: } 38: <div class="product" (click)="showDetail(item)" aria-label="Click for more information about the product" 39: matTooltip="Click for more information" matTooltipPosition="above"> >>> 40: 42: <div class="info-box"> 43: <div class="item-name">{{item.name}}</div> </pre>		
L	frontend/src/app/search-result search-result.component.html (Line 40)	Web:MouseEventWithoutKeyboardEquivalentCheck Add a 'onKeyPress onKeyDown onKeyUp' attribute to this tag.

■ Problematic Code:

```

37:         }
38:         <div class="product" (click)="showDetail(item)" aria-label="Click for
more information about the product"
39:             matTooltip="Click for more information" matTooltipPosition="above">
>>> 40:         <img mat-card-image
[src]='assets/public/images/products/' + item.image" alt={{item.name}}
41:             class="img-responsive img-thumbnail" role="button">
42:         <div class="info-box">
43:             <div class="item-name">{{item.name}}</div>

```

L

frontend/src/app/payment-method
payment-method.component.html
(Line 14)

Web:MouseEventWithoutKeyboardEquivalentCheck
Add a 'onKeyPress|onKeyDown|onKeyUp' attribute to this tag.

■ Problematic Code:

```

11:         <ng-container matColumnDef="Selection">
12:             <mat-header-cell *matHeaderCellDef></mat-header-cell>
13:             <mat-cell *matCellDef="let element">
>>> 14:             <mat-radio-button
(click)="emitSelectionToParent(element.id)"></mat-radio-button>
15:             </mat-cell>
16:         </ng-container>
17:         <ng-container matColumnDef="Number">

```

L

frontend/src/app/product-details
product-details.component.html
(Line 51)

Web:MouseEventWithoutKeyboardEquivalentCheck
Add a 'onKeyPress|onKeyDown|onKeyUp' attribute to this tag.

■ Problematic Code:

```

48:         @for (review of reviews$|async; track review) {
49:             <div class="comment">
50:                 <div class="review-row">
>>> 51:                     <div class="review-text"
52:                         (click)="review.author !== 'Anonymous' && review.author === author &&
editReview(review)"
53:                         matTooltipDisabled="{{review.author !== author}}"
matTooltip="{{ 'LABEL_EDIT_REVIEW' | translate }}"
54:                         matTooltipPosition="right">

```

L

frontend/src/app/score-board
score-board.component.html
(Line 53)

Web:ImgWithoutAltCheck
Add an "alt" attribute to this image.

■ Problematic Code:

```

50:         </div>
51:     }
52: }
>>> 53: 
54: }
55:
56:

```

<div>L</div>	frontend/src/app/token-sale token-sale.component.html (Line 96)	Web:S6850 Headings must have content and the content must be accessible by a screen reader.
	<p>■ Problematic Code:</p> <pre> 93: 94: <mat-card appearance="outlined" class="mat-elevation-z6"> 95: <div class="mdc-card"> >>> 96: <h5><i class='fas fa-comments fa-2x'></i> </h5> 97: <small class="text-justify">{{"ICO_FAQ_ANSWER" translate}}</small> 98: </div> 99: </mat-card> </pre>	
<div>L</div>	frontend/src/app/recycle recycle.component.html (Line 64)	Web:ImgWithoutAltCheck Add an "alt" attribute to this image.
	<p>■ Problematic Code:</p> <pre> 61: <h3 class="responsibility-header">{{"SECTION_PRESS_JUICE_RESPONSIBLY" translate}}</h3> 62: 63: <mat-card class="mat-elevation-z0 card-row"> >>> 64: 65: <mat-card-content> 66: <div> 67: <small>Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut </pre>	
<div>L</div>	frontend/src/app/recycle recycle.component.html (Line 76)	Web:ImgWithoutAltCheck Add an "alt" attribute to this image.
	<p>■ Problematic Code:</p> <pre> 73: </mat-card> 74: 75: <mat-card class="mat-elevation-z0 card-row"> >>> 76: 77: <mat-card-content> 78: <div> 79: <small>Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor </pre>	

Maintainability Issues

Total: 452 issues (Blocker: 25, High: 105, Medium: 206, Low: 92, Info: 24)

Severity	File Path	Rule & Message
	frontend/src/app/change-password/change-password.component.ts (Line 80)	typescript:S7655 Lifecycle interface 'OnInit' should be implemented for method 'ngOnInit'. (https://angular.dev/style-guide#style-09-01)
■ Problematic Code: <pre> 77: private readonly translate: TranslateService 78:) {} 79: >>> 80: ngOnInit (): void { 81: this.formSubmitService.attachEnterKeyHandler(82: 'password-form', 83: 'changeButton', </pre>		
	frontend/src/app/last-login-ip/last-login-ip.component.ts (Line 23)	typescript:S7655 Lifecycle interface 'OnInit' should be implemented for method 'ngOnInit'. (https://angular.dev/style-guide#style-09-01)
■ Problematic Code: <pre> 20: lastLoginIp: any = '?' 21: constructor (private readonly sanitizer: DomSanitizer) {} 22: >>> 23: ngOnInit (): void { 24: try { 25: this.parseAuthToken() 26: } catch (err) { </pre>		
	frontend/src/app/nft-unlock/nft-unlock.component.ts (Line 32)	typescript:S7655 Lifecycle interface 'OnInit' should be implemented for method 'ngOnInit'. (https://angular.dev/style-guide#style-09-01)
■ Problematic Code: <pre> 29: 30: constructor (private readonly keysService: KeysService) {} 31: >>> 32: ngOnInit (): void { 33: this.checkChallengeStatus() 34: } 35: </pre>		
	frontend/src/app/two-factor-auth/two-factor-auth.component.ts (Line 59)	typescript:S7655 Lifecycle interface 'OnInit' should be implemented for method 'ngOnInit'. (https://angular.dev/style-guide#style-09-01)

■ Problematic Code:

```

56:
57:   constructor (private readonly twoFactorAuthService: TwoFactorAuthService, private
readonly configurationService: ConfigurationService, private readonly snackBar: MatSnackBar,
private readonly translateService: TranslateService, private readonly snackBarHelperService:
SnackBarHelperService) {}
58:
>>> 59:   ngOnInit (): void {
60:       this.updateStatus()
61:   }
62:

```

B

frontend/src/app/mat-search-bar
mat-search-bar.component.ts
(Line 55)

typescript:S7652

Output bindings, including aliases, should not be named "on", nor prefixed with it (<https://angular.dev/guide/components/outputs#choosing-event-names>)

■ Problematic Code:

```

52:   @Input() matAutocomplete: MatAutocomplete
53:   @Input() placeholder = ''
54:   @Input() alwaysOpen: boolean = false
>>> 55:   @Output() onBlur = new EventEmitter<string>()
56:   @Output() onClose = new EventEmitter<void>()
57:   @Output() onEnter = new EventEmitter<string>()
58:   @Output() onFocus = new EventEmitter<string>()

```

B

frontend/src/app/mat-search-bar
mat-search-bar.component.ts
(Line 56)

typescript:S7652

Output bindings, including aliases, should not be named "on", nor prefixed with it (<https://angular.dev/guide/components/outputs#choosing-event-names>)

■ Problematic Code:

```

53:   @Input() placeholder = ''
54:   @Input() alwaysOpen: boolean = false
55:   @Output() onBlur = new EventEmitter<string>()
>>> 56:   @Output() onClose = new EventEmitter<void>()
57:   @Output() onEnter = new EventEmitter<string>()
58:   @Output() onFocus = new EventEmitter<string>()
59:   @Output() onOpen = new EventEmitter<void>()

```

B

frontend/src/app/mat-search-bar
mat-search-bar.component.ts
(Line 57)

typescript:S7652

Output bindings, including aliases, should not be named "on", nor prefixed with it (<https://angular.dev/guide/components/outputs#choosing-event-names>)

■ Problematic Code:

```





54:   @Input() alwaysOpen: boolean = false
55:   @Output() onBlur = new EventEmitter<string>()
56:   @Output() onClose = new EventEmitter<void>()
>>> 57:   @Output() onEnter = new EventEmitter<string>()
58:   @Output() onFocus = new EventEmitter<string>()
59:   @Output() onOpen = new EventEmitter<void>()
60:

```


<div>B</div>	frontend/src/app/mat-search-bar mat-search-bar.component.ts (Line 58)	typescript:S7652 Output bindings, including aliases, should not be named "on", nor prefixed with it (https://angular.dev/guide/components/outputs#choosing-event-names)
<div> <div>■ Problematic Code:</div> <pre> 55: @Output() onBlur = new EventEmitter<string>() 56: @Output() onClose = new EventEmitter<void>() 57: @Output() onEnter = new EventEmitter<string>() >>> 58: @Output() onFocus = new EventEmitter<string>() 59: @Output() onOpen = new EventEmitter<void>() 60: 61: searchVisible = false </pre> </div>		
<div>B</div>	frontend/src/app/mat-search-bar mat-search-bar.component.ts (Line 59)	typescript:S7652 Output bindings, including aliases, should not be named "on", nor prefixed with it (https://angular.dev/guide/components/outputs#choosing-event-names)
<div> <div>■ Problematic Code:</div> <pre> 56: @Output() onClose = new EventEmitter<void>() 57: @Output() onEnter = new EventEmitter<string>() 58: @Output() onFocus = new EventEmitter<string>() >>> 59: @Output() onOpen = new EventEmitter<void>() 60: 61: searchVisible = false 62: </pre> </div>		
<div>B</div>	frontend/src/app/code-fixes code-fixes.component.ts (Line 28)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)
<div> <div>■ Problematic Code:</div> <pre> 25: @Input('fixes') 26: public fixes: string[] = [] 27: >>> 28: @Input('selectedFix') 29: public selectedFix: number = 0 30: 31: @Input('randomFixes') </pre> </div>		
<div>B</div>	frontend/src/app/code-fixes code-fixes.component.ts (Line 31)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)
<div> <div>■ Problematic Code:</div> <pre> 28: @Input('selectedFix') 29: public selectedFix: number = 0 30: >>> 31: @Input('randomFixes') 32: public randomFixes: RandomFixes[] = [] 33: 34: @Input('format') </pre> </div>		





<div>B</div>	frontend/src/app/address address.component.ts (Line 31)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)
<div> <div>■ Problematic Code:</div> <pre> 28: }) 29: export class AddressComponent implements OnInit { 30: @Output() emitSelection = new EventEmitter() >>> 31: @Input('allowEdit') public allowEdit: boolean = false 32: @Input('addNewAddressDiv') public addNewAddressDiv: boolean = true 33: @Input('showNextButton') public showNextButton: boolean = false 34: public addressId: any = undefined </pre> </div>		
<div>B</div>	frontend/src/app/address address.component.ts (Line 32)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)
<div> <div>■ Problematic Code:</div> <pre> 29: export class AddressComponent implements OnInit { 30: @Output() emitSelection = new EventEmitter() 31: @Input('allowEdit') public allowEdit: boolean = false >>> 32: @Input('addNewAddressDiv') public addNewAddressDiv: boolean = true 33: @Input('showNextButton') public showNextButton: boolean = false 34: public addressId: any = undefined 35: public displayedColumns = ['Name', 'Address', 'Country'] </pre> </div>		
<div>B</div>	frontend/src/app/web3-sandbox web3-sandbox.component.ts (Line 54)	typescript:S7655 Lifecycle interface 'OnInit' should be implemented for method 'ngOnInit'. (https://angular.dev/style-guide#style-09-01)
<div> <div>■ Problematic Code:</div> <pre> 51: private readonly changeDetectorRef: ChangeDetectorRef 52:) {} 53: >>> 54: ngOnInit (): void { 55: this.handleAuth() 56: window.ethereum.on('chainChanged', this.handleChainChanged.bind(this)) 57: } </pre> </div>		
<div>B</div>	frontend/src/app/wallet-web3 wallet-web3.component.ts (Line 52)	typescript:S7655 Lifecycle interface 'OnInit' should be implemented for method 'ngOnInit'. (https://angular.dev/style-guide#style-09-01)
<div> <div>■ Problematic Code:</div> <pre> 49: challengeSolved = false 50: errorMessage = '' 51: metamaskAddress = '' >>> 52: ngOnInit (): void { 53: this.handleAuth() 54: window.ethereum.on('chainChanged', this.handleChainChanged.bind(this)) 55: } </pre> </div>		
<div>B</div>	frontend/src/app/faucet faucet.component.ts (Line 64)	typescript:S7655 Lifecycle interface 'OnInit' should be implemented for method 'ngOnInit'. (https://angular.dev/style-guide#style-09-01)

■ Problematic Code: <pre> 61: errorMessage = '' 62: metamaskAddress = '' 63: >>> 64: ngOnInit (): void { 65: this.translateService.get('NFT_MINT_TEXT_INTRO').subscribe((translatedString: string) => { 66: this.nftMintText = translatedString 67: }) </pre>		
B	frontend/src/app/code-fixes code-fixes.component.ts (Line 34)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)
■ Problematic Code: <pre> 31: @Input('randomFixes') 32: public randomFixes: RandomFixes[] = [] 33: >>> 34: @Input('format') 35: public format: string = 'SideBySide' 36: 37: @ViewChild('codeComponent', { static: false }) codeComponent: NgxTextDiffComponent </pre>		
B	frontend/src/app/code-fixes code-fixes.component.ts (Line 25)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)
■ Problematic Code: <pre> 22: @Input('snippet') 23: public snippet: string = '' 24: >>> 25: @Input('fixes') 26: public fixes: string[] = [] 27: 28: @Input('selectedFix') </pre>		
B	frontend/src/app/code-fixes code-fixes.component.ts (Line 22)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)
■ Problematic Code: <pre> 19: this.differ = this.differs.find({}).create() 20: } 21: >>> 22: @Input('snippet') 23: public snippet: string = '' 24: 25: @Input('fixes') </pre>		
B	frontend/src/app/code-area code-area.component.ts (Line 32)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)

<p>■ Problematic Code:</p> <pre> 29: this._code = value '' 30: } 31: >>> 32: @Input('vulnLines') 33: public vulnLines: number[] 34: 35: public lineMarkers: LineMarker[] </pre>		
	frontend/src/app/purchase-basket purchase-basket.component.ts (Line 30)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)
<p>■ Problematic Code:</p> <pre> 27: export class PurchaseBasketComponent implements OnInit { 28: @Input('allowEdit') public allowEdit: boolean = false 29: @Input('displayTotal') public displayTotal: boolean = false >>> 30: @Input('totalPrice') public totalPrice: boolean = true 31: @Output() emitTotal = new EventEmitter() 32: @Output() emitProductCount = new EventEmitter() 33: public tableColumns = ['image', 'product', 'quantity', 'price'] </pre>		
	frontend/src/app/address address.component.ts (Line 33)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)
<p>■ Problematic Code:</p> <pre> 30: @Output() emitSelection = new EventEmitter() 31: @Input('allowEdit') public allowEdit: boolean = false 32: @Input('addNewAddressDiv') public addNewAddressDiv: boolean = true >>> 33: @Input('showNextButton') public showNextButton: boolean = false 34: public addressId: any = undefined 35: public displayedColumns = ['Name', 'Address', 'Country'] 36: selection = new SelectionModel<Element>(false, []) </pre>		
	frontend/src/app/purchase-basket purchase-basket.component.ts (Line 28)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)
<p>■ Problematic Code:</p> <pre> 25: imports: [MatTable, MatColumnDef, MatHeaderCellDef, MatHeaderCell, MatCellDef, MatCell, MatFooterCellDef, MatFooterCell, MatIconButton, MatHeaderRowDef, MatHeaderRow, MatRowDef, MatRow, MatFooterRowDef, MatFooterRow, TranslateModule] 26: }) 27: export class PurchaseBasketComponent implements OnInit { >>> 28: @Input('allowEdit') public allowEdit: boolean = false 29: @Input('displayTotal') public displayTotal: boolean = false 30: @Input('totalPrice') public totalPrice: boolean = true 31: @Output() emitTotal = new EventEmitter() </pre>		
	frontend/src/app/purchase-basket purchase-basket.component.ts (Line 29)	typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)

<p>■ Problematic Code:</p> <pre> 26: }) 27: export class PurchaseBasketComponent implements OnInit { 28: @Input('allowEdit') public allowEdit: boolean = false >>> 29: @Input('displayTotal') public displayTotal: boolean = false 30: @Input('totalPrice') public totalPrice: boolean = true 31: @Output() emitTotal = new EventEmitter() 32: @Output() emitProductCount = new EventEmitter() </pre>		
<p>B</p>	<p>frontend/src/app/payment-method payment-method.component.ts (Line 34)</p>	<p>typescript:S7649 Input bindings should not be aliased (https://angular.dev/style-guide#style-05-13)</p>
<p>■ Problematic Code:</p> <pre> 31: 32: export class PaymentMethodComponent implements OnInit { 33: @Output() emitSelection = new EventEmitter() >>> 34: @Input('allowDelete') public allowDelete: boolean = false 35: public displayedColumns = ['Number', 'Name', 'Expiry'] 36: public nameControl: UntypedFormControl = new UntypedFormControl('', [Validators.required]) 37: // eslint-disable-next-line @typescript-eslint/no-loss-of-precision </pre>		
<p>H</p>	<p>frontend/src/app/search-result search-result.component.ts (Line 74)</p>	<p>typescript:S3776 Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.</p>
<p>■ Problematic Code:</p> <pre> 71: const products = this.productService.search('') 72: const quantities = this.quantityService.getAll() 73: forkJoin([quantities, products]).subscribe({ >>> 74: next: ([quantities, products]) => { 75: const dataTable: TableEntry[] = [] 76: this.tableData = products 77: this.trustProductDescription(products) // vuln-code-snippet neutral-line restfulXssChallenge </pre>		
<p>H</p>	<p>frontend/src/app/search-result search-result.component.ts (Line 209)</p>	<p>typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.</p>
<p>■ Problematic Code:</p> <pre> 206: this.basketService.put(existingBasketItem.id, { quantity: newQuantity }).subscribe({ 207: next: (updatedBasketItem) => { 208: this.productService.get(updatedBasketItem.ProductId).subscribe({ >>> 209: next: (product) => { 210: this.translateService.get('BASKET_ADD_SAME_PRODUCT', { product: product.name }).subscribe({ 211: next: (basketAddSameProduct) => { 212: this.snackBarHelperService.open(basketAddSameProduct, 'confirmBar') </pre>		

<div>H</div>	frontend/src/app/search-result search-result.component.ts (Line 221)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div> <div>■ Problematic Code:</div> <pre> 218: } 219: }) 220: }, >>> 221: error: (err) => { console.log(err) } 222: }) 223: }, 224: error: (err) => { </pre> </div>		
<div>H</div>	frontend/src/app/search-result search-result.component.ts (Line 241)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div> <div>■ Problematic Code:</div> <pre> 238: this.productService.get(newBasketItem.ProductId).subscribe({ 239: next: (product) => { 240: this.translateService.get('BASKET_ADD_PRODUCT', { product: product.name }).subscribe({ >>> 241: next: (basketAddProduct) => { 242: this.snackBarHelperService.open(basketAddProduct, 'confirmBar') 243: this.basketService.updateNumberOfCartItems() 244: }, </pre> </div>		
<div>H</div>	frontend/src/app/search-result search-result.component.ts (Line 245)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div> <div>■ Problematic Code:</div> <pre> 242: this.snackBarHelperService.open(basketAddProduct, 'confirmBar') 243: this.basketService.updateNumberOfCartItems() 244: }, >>> 245: error: (translationId) => { 246: this.snackBarHelperService.open(translationId, 'confirmBar') 247: this.basketService.updateNumberOfCartItems() 248: } </pre> </div>		
<div>H</div>	frontend/src/app server-started-notification server-started-notification.compone nt.ts (Line 42)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div> <div>■ Problematic Code:</div> <pre> 39: this.challengeService.restoreProgress(encodeURIComponent(continueCode)).subs cribe({ 40: next: () => { 41: this.translate.get('AUTO_RESTORED_PROGRESS').subscribe({ >>> 42: next: (notificationServerStarted) => { 43: this.hackingProgress.autoRestoreMessage = notificationServerStarted 44: }, 45: error: (translationId) => { </pre> </div>		

	frontend/src/app server-started-notification server-started-notification.component.ts (Line 45)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div> <div>■ Problematic Code:</div> <pre> 42: next: (notificationServerStarted) => { 43: this.hackingProgress.autoRestoreMessage = notificationServerStarted 44: }, >>> 45: error: (translationId) => { 46: this.hackingProgress.autoRestoreMessage = translationId 47: } 48: }) </pre> </div>		
	frontend/src/app server-started-notification server-started-notification.component.ts (Line 53)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div> <div>■ Problematic Code:</div> <pre> 50: error: (error) => { 51: console.log(error) 52: this.translate.get('AUTO_RESTORE_PROGRESS_FAILED', { error }).subscribe({ >>> 53: next: (notificationServerStarted) => { 54: this.hackingProgress.autoRestoreMessage = notificationServerStarted 55: }, 56: error: (translationId) => { </pre> </div>		
	frontend/src/app server-started-notification server-started-notification.component.ts (Line 56)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div> <div>■ Problematic Code:</div> <pre> 53: next: (notificationServerStarted) => { 54: this.hackingProgress.autoRestoreMessage = notificationServerStarted 55: }, >>> 56: error: (translationId) => { 57: this.hackingProgress.autoRestoreMessage = translationId 58: } 59: }) </pre> </div>		
	test/server/fileUploadSpec.ts (Line 33)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code:

```

30:         challenges.uploadSizeChallenge = { solved: false, save } as unknown as
Challenge
31:         req.file.size = size
32:
>>> 33:         checkUploadSize(req, res, () => {})
34:
35:         expect(challenges.uploadSizeChallenge.solved).to.equal(false)
36:     })

```



routes/dataExport.ts
(Line 88)

typescript:S2004

Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code:

```

85:         likedBy: string
86:     }>) => {
87:         if (reviews.length > 0) {
>>> 88:             reviews.forEach(review => {
89:                 userData.reviews.push({
90:                     message: review.message,
91:                     author: review.author,

```



routes/dataExport.ts
(Line 100)

typescript:S2004

Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code:

```

97:         }
98:         const emailHash = security.hash(email).slice(0, 4)
99:         for (const order of userData.orders) {
>>> 100:             challengeUtils.solveIf(challenges.dataExportChallenge, () => { return
order.orderId.split('-')[0] !== emailHash })
101:         }
102:         res.status(200).send({ userData: JSON.stringify(userData, null, 2),
confirmation: 'Your data export will open in a new Browser window.' })
103:     },

```



routes/languages.ts
(Line 25)

typescript:S2004

Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code:

```

22:         if (err != null) {
23:             next(new Error(`Unable to read i18n directory: ${err.message}`))
24:         }
>>> 25:         languageFiles.forEach((fileName) => {
26:             // eslint-disable-next-line @typescript-eslint/no-misused-promises
27:             fs.readFile('frontend/dist/frontend/assets/i18n/' + fileName, 'utf-8', async
(err, content) => {
28:                 if (err != null) {

```



routes/metrics.ts
(Line 170)

typescript:S2004

Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code:

```

167:         }
168:
169:         void retrieveChallengesWithCodeSnippet().then(challenges => {
>>> 170:             ChallengeModel.count({ where: { codingChallengeStatus: { [Op.eq]: 1 } }
    }).then((count: number) => {
171:                 codingChallengesProgressMetrics.set({ phase: 'find it' }, count)
172:             }).catch(() => {
173:                 throw new Error('Unable to retrieve and count such challenges. Please try
again')

```



routes/metrics.ts
(Line 172)

typescript:S2004

Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code:

```

169:         void retrieveChallengesWithCodeSnippet().then(challenges => {
170:             ChallengeModel.count({ where: { codingChallengeStatus: { [Op.eq]: 1 } }
    }).then((count: number) => {
171:                 codingChallengesProgressMetrics.set({ phase: 'find it' }, count)
>>> 172:             }).catch(() => {
173:                 throw new Error('Unable to retrieve and count such challenges. Please try
again')
174:             })
175:

```



routes/metrics.ts
(Line 176)

typescript:S2004

Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code:

```

173:             throw new Error('Unable to retrieve and count such challenges. Please try
again')
174:         })
175:
>>> 176:         ChallengeModel.count({ where: { codingChallengeStatus: { [Op.eq]: 2 } }
    }).then((count: number) => {
177:                 codingChallengesProgressMetrics.set({ phase: 'fix it' }, count)
178:             }).catch(_: unknown) => {
179:                 throw new Error('Unable to retrieve and count such challenges. Please try
again')

```



routes/metrics.ts
(Line 178)

typescript:S2004

Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code:

```

175:
176:         ChallengeModel.count({ where: { codingChallengeStatus: { [Op.eq]: 2 } }
    }).then((count: number) => {
177:                 codingChallengesProgressMetrics.set({ phase: 'fix it' }, count)
>>> 178:             }).catch(_: unknown) => {
179:                 throw new Error('Unable to retrieve and count such challenges. Please try
again')
180:             })
181:





```

	routes/metrics.ts (Line 182)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div data-bbox="177 362 458 387">■ Problematic Code:</div> <pre data-bbox="177 398 1404 741"> 179: throw new Error('Unable to retrieve and count such challenges. Please try again') 180: }) 181: >>> 182: ChallengeModel.count({ where: { codingChallengeStatus: { [Op.ne]: 0 } } }).then((count: number) => { 183: codingChallengesProgressMetrics.set({ phase: 'unsolved' }, challenges.length - count) 184: }).catch(_: unknown) => { 185: throw new Error('Unable to retrieve and count such challenges. Please try again')</pre>		
	routes/metrics.ts (Line 184)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div data-bbox="177 884 458 909">■ Problematic Code:</div> <pre data-bbox="177 920 1404 1234"> 181: 182: ChallengeModel.count({ where: { codingChallengeStatus: { [Op.ne]: 0 } } }).then((count: number) => { 183: codingChallengesProgressMetrics.set({ phase: 'unsolved' }, challenges.length - count) >>> 184: }).catch(_: unknown) => { 185: throw new Error('Unable to retrieve and count such challenges. Please try again') 186: }) 187: })</pre>		
	routes/order.ts (Line 70)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div data-bbox="177 1373 458 1397">■ Problematic Code:</div> <pre data-bbox="177 1408 1404 1722"> 67: let totalPoints = 0 68: basket.Products?.forEach(({ BasketItem, price, deluxePrice, name, id }) => { 69: if (BasketItem !== null) { >>> 70: challengeUtils.solveIf(challenges.christmasSpecialChallenge, () => { return BasketItem.ProductId === products.christmasSpecial.id }) 71: QuantityModel.findOne({ where: { ProductId: BasketItem.ProductId } }).then((product: any) => { 72: const newQuantity = product.quantity - BasketItem.quantity 73: QuantityModel.update({ quantity: newQuantity }, { where: { ProductId: BasketItem?.ProductId } }).catch((error: unknown) => {</pre>		
	routes/order.ts (Line 71)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

	■ Problematic Code: <pre> 68: basket.Products?.forEach(({ BasketItem, price, deluxePrice, name, id }) => { 69: if (BasketItem != null) { 70: challengeUtils.solveIf(challenges.christmasSpecialChallenge, () => { return BasketItem.ProductId === products.christmasSpecial.id }) >>> 71: QuantityModel.findOne({ where: { ProductId: BasketItem.ProductId } }).then((product: any) => { 72: const newQuantity = product.quantity - BasketItem.quantity 73: QuantityModel.update({ quantity: newQuantity }, { where: { ProductId: BasketItem?.ProductId } }).catch((error: unknown) => { 74: next(error) </pre>	
	routes/order.ts (Line 76)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
	■ Problematic Code: <pre> 73: QuantityModel.update({ quantity: newQuantity }, { where: { ProductId: BasketItem?.ProductId } }).catch((error: unknown) => { 74: next(error) 75: }) >>> 76: }).catch((error: unknown) => { 77: next(error) 78: }) 79: let itemPrice: number </pre>	
	routes/resetPassword.ts (Line 37)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
	■ Problematic Code: <pre> 34: }).then((data: SecurityAnswerModel null) => { 35: if ((data != null) && security.hmac(answer) === data.answer) { 36: UserModel.findByPk(data.UserId).then((user: UserModel null) => { >>> 37: user?.update({ password: newPassword }).then((user: UserModel) => { 38: verifySecurityAnswerChallenges(user, answer) 39: res.json({ user }) 40: }).catch((error: unknown) => { </pre>	
	routes/resetPassword.ts (Line 40)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
	■ Problematic Code: <pre> 37: user?.update({ password: newPassword }).then((user: UserModel) => { 38: verifySecurityAnswerChallenges(user, answer) 39: res.json({ user }) >>> 40: }).catch((error: unknown) => { 41: next(error) 42: }) 43: }).catch((error: unknown) => { </pre>	
	routes/vulnCodeSnippet.ts (Line 70)	typescript:S3776 Refactor this function to reduce its Cognitive Complexity from 20 to the 15 allowed.


<div> <div>■ Problematic Code:</div> <pre> 67: return notOkLines.length === 0 68: } 69: >>> 70: export const checkVulnLines = () => async (req: Request<Record<string, unknown>, Record<string, unknown>, VerdictRequestBody>, res: Response, next: NextFunction) => { 71: const key = req.body.key 72: let snippetData 73: try { </pre> </div>		
<div>H</div>	test/server/verifySpec.ts (Line 216)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div> <div>■ Problematic Code:</div> <pre> 213: describe('"changeProductChallenge"', () => { 214: beforeEach(() => { 215: challenges.changeProductChallenge = { solved: false, save } as unknown as Challenge >>> 216: products.osaft = { reload () { return { then (cb: any) { cb() } } } } as unknown as Product 217: }) 218: 219: it(`is solved when the link in the O-Saft product goes to \${config.get<string>('challenges.overrideUrlForProductTamperingChallenge')}`, () => { </pre> </div>		
<div>H</div>	routes/nftMint.ts (Line 19)	typescript:S3735 Remove this use of the "void" operator.
<div> <div>■ Problematic Code:</div> <pre> 16: const provider = new WebSocketProvider('wss://eth-sepolia.g.alchemy.com/v2/FZDapFZSs1l6yhHW4VnQqsi18qSd-3GJ') 17: const contract = new Contract(nftAddress, nftABI, provider) 18: if (!isEventListenerCreated) { >>> 19: void contract.on('NFTMinted', (minter: string) => { 20: if (!addressesMinted.has(minter)) { 21: addressesMinted.add(minter) 22: } </pre> </div>		
<div>H</div>	routes/web3Wallet.ts (Line 21)	typescript:S3735 Remove this use of the "void" operator.
<div> <div>■ Problematic Code:</div> <pre> 18: const provider = new WebSocketProvider('wss://eth-sepolia.g.alchemy.com/v2/FZDapFZSs1l6yhHW4VnQqsi18qSd-3GJ') 19: const contract = new Contract(web3WalletAddress, web3WalletABI, provider) 20: if (!isEventListenerCreated) { >>> 21: void contract.on('ContractExploited', (exploiter: string) => { 22: if (walletsConnected.has(exploiter)) { 23: walletsConnected.delete(exploiter) 24: challengeUtils.solveIf(challenges.web3WalletChallenge, () => true) </pre> </div>		
<div>H</div>	routes/profileImageUrlUpload.ts (Line 17)	typescript:S3776 Refactor this function to reduce its Cognitive Complexity from 19 to the 15 allowed.





■ Problematic Code: <pre> 14: import logger from '../lib/logger' 15: 16: export function profileImageUrlUpload () { >>> 17: return async (req: Request, res: Response, next: NextFunction) => { 18: if (req.body.imageUrl !== undefined) { 19: const url = req.body.imageUrl 20: if (url.match(/(.)*solve\/challenges\/server-side(.)*\/) !== null) req.app.locals.abused_ssrf_bug = true </pre>		
H	test/server/blueprintSpec.ts (Line 35)	typescript:S3776 Refactor this function to reduce its Cognitive Complexity from 23 to the 15 allowed.
■ Problematic Code: <pre> 32: let pathToImage: string = 'assets/public/images/products/' 33: 34: describe('checkExifData', () => { >>> 35: it('should contain properties from exifForBlueprintChallenge', async () => { 36: for (const product of products) { 37: if (product.fileForRetrieveBlueprintChallenge && product.image) { 38: if (utils.isUrl(product.image)) { </pre>		
H	rsn/rsnUtil.ts (Line 47)	typescript:S3776 Refactor this function to reduce its Cognitive Complexity from 31 to the 15 allowed.
■ Problematic Code: <pre> 44: }, {}) 45: for (const val of keys) { 46: await retrieveCodeSnippet(val.split('_')[0]) >>> 47: .then(snippet => { 48: if (snippet == null) return 49: process.stdout.write(val + ': ') 50: const fileData = fs.readFileSync(fixesPath + '/' + val).toString() </pre>		
H	routes/fileUpload.ts (Line 108)	typescript:S3776 Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.
■ Problematic Code: <pre> 105: next() 106: } 107: >>> 108: function handleYamlUpload ({ file }: Request, res: Response, next: NextFunction) { 109: if (utils.endsWith(file?.originalname.toLowerCase(), '.yaml') utils.endsWith(file?.originalname.toLowerCase(), '.yml')) { 110: challengeUtils.solveIf(challenges.deprecatedInterfaceChallenge, () => { return true }) 111: if (((file?.buffer) != null) && utils.isChallengeEnabled(challenges.deprecatedInterfaceChallenge)) { </pre>		
H	test/cypress/e2e/chatbot.spec.ts (Line 45)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code: <pre> 42: cy.get('#message-input').type('hi').type('{enter}') 43: cy.get('#message-input').type('...').type('{enter}') 44: >>> 45: const genArr = Array.from({ length: 100 }, (v, k) => k + 1) 46: cy.wrap(genArr).eachSeries(() => { 47: cy.get('#message-input') 48: .type(couponIntent.utterances[0]) </pre>		
	frontend/src/app/about about.component.ts (Line 76)	typescript:S3776 Refactor this function to reduce its Cognitive Complexity from 17 to the 15 allowed.
■ Problematic Code: <pre> 73: console.error(err) 74: return EMPTY 75: }) >>> 76:).subscribe((config) => { 77: if (config?.application?.social) { 78: if (config.application.social.blueSkyUrl) { 79: this.blueSkyUrl = config.application.social.blueSkyUrl </pre>		
	test/cypress/e2e/geoStalking.spec.ts (Line 11)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 8: it('Should be possible to find the answer to a security question in the meta-data of a photo on the photo wall', () => { 9: cy.task<string>('GetFromMemories', 'geoStalkingMetaSecurityAnswer').then(10: (answer: string) => { >>> 11: cy.task<string>('GetFromConfig', 'application.domain').then((appDomain: string) => { 12: cy.get('#email').type(`john@\${appDomain}`) 13: cy.wait('@securityQuestion') 14: cy.get('#securityAnswer').should('not.be.disabled').focus().type(answer) </pre>		
	test/cypress/e2e/geoStalking.spec.ts (Line 30)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 27: it('Should be possible to determine the answer to a security question by looking closely at an image on the photo wall', () => { 28: cy.task<string>('GetFromMemories', 'geoStalkingVisualSecurityAnswer').then(29: (answer: string) => { >>> 30: cy.task<string>('GetFromConfig', 'application.domain').then((appDomain: string) => { 31: cy.get('#email').type(`emma@\${appDomain}`) 32: cy.wait('@securityQuestion') 33: cy.get('#securityAnswer').should('not.be.disabled').focus().type(answer) </pre>		
	test/cypress/e2e/restApi.spec.ts (Line 51)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.


	<p>■ Problematic Code:</p> <pre> 48: describe('challenge "changeProduct"', () => { 49: it('should be possible to change product via PUT request without being logged in', () => { 50: cy.task<number>('GetTamperingProductId').then((tamperingProductId: number) => { >>> 51: cy.task<string>('GetOverwriteUrl').then((overwriteUrl: string) => { 52: cy.window().then(async () => { 53: const response = await fetch(54: `\${Cypress.config('baseUrl')}/api/Products/\${tamperingProductId}`, </pre>	
	test/cypress/e2e/b2bOrder.spec.ts (Line 8)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
	<p>■ Problematic Code:</p> <pre> 5: if (!isDocker) { 6: cy.login({ email: 'admin', password: 'admin123' }) 7: >>> 8: cy.window().then(async () => { 9: const response = await fetch(10: `\${Cypress.config('baseUrl')}/b2b/v2/orders/`, 11: { </pre>	
	test/cypress/e2e/b2bOrder.spec.ts (Line 38)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
	<p>■ Problematic Code:</p> <pre> 35: cy.task('isDocker').then((isDocker) => { 36: if (!isDocker) { 37: cy.login({ email: 'admin', password: 'admin123' }) >>> 38: cy.window().then(async () => { 39: const response = await fetch(40: `\${Cypress.config('baseUrl')}/b2b/v2/orders/`, 41: { </pre>	
	test/cypress/e2e/complain.spec.ts (Line 158)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
	<p>■ Problematic Code:</p> <pre> 155: cy.get('#submitButton').click() 156: cy.visit('/promotion') 157: >>> 158: cy.on('window:alert', (t) => { 159: expect(t).to.equal('xss') 160: }) 161: cy.visit('/') </pre>	
	test/cypress/e2e/contact.spec.ts (Line 64)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

<div> <div>■ Problematic Code:</div> <pre> 61: cy.get('#submitButton').click({ force: true }) // FIXME Analyze Cypress recordings to properly fix behavior during test 62: 63: cy.visit('/#/about') >>> 64: cy.on('window:alert', (t) => { 65: expect(t).to.equal('xss') 66: }) 67: </pre> </div>		
<div>H</div>	test/cypress/e2e/contact.spec.ts (Line 69)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div> <div>■ Problematic Code:</div> <pre> 66: }) 67: 68: cy.visit('/#/administration') >>> 69: cy.on('window:alert', (t) => { 70: expect(t).to.equal('xss') 71: }) 72: cy.expectChallengeSolved({ challenge: 'Server-side XSS Protection' }) </pre> </div>		
<div>H</div>	test/cypress/e2e/forgedJwt.spec.ts (Line 19)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div> <div>■ Problematic Code:</div> <pre> 16: it('should accept a token HMAC-signed with public RSA key with email rsa_lord@juice-sh.op in the payload ', () => { 17: cy.task('isWindows').then((isWindows) => { 18: if (!isWindows) { >>> 19: cy.window().then(() => { 20: localStorage.setItem(21: 'token', 22: 'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRhIjp7ImVtYWlsIjoicnNhX2xvcm RAanVpY2Utc2gub3AifSwiaWF0IjoxNTgzMDM3NzExfQ.gShXDT5TrE5736mpIbfVDEcQbLfteJaQUg7Z0PH8Xc8' </pre> </div>		
<div>H</div>	test/cypress/e2e/noSql.spec.ts (Line 13)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div> <div>■ Problematic Code:</div> <pre> 10: it('should be possible to inject a command into the get route', () => { 11: cy.task('isDocker').then((isDocker) => { 12: if (!isDocker) { >>> 13: cy.window().then(() => { 14: void fetch(15: `\${Cypress.config('baseUrl')}/rest/products/sleep(1000)/reviews`, 16: { </pre> </div>		
<div>H</div>	test/cypress/e2e/noSql.spec.ts (Line 34)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.




■ Problematic Code: <pre> 31: it('should be possible to inject and get all the orders', () => { 32: cy.task('isDocker').then((isDocker) => { 33: if (!isDocker) { >>> 34: cy.window().then(async () => { 35: await fetch(36: `\${Cypress.config('baseUrl')}/rest/track-order/%27%20%7C%7C%20true%20%7C%7C%20%27`, 37: { </pre>		
	test/cypress/e2e/profile.spec.ts (Line 30)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 27: cy.get('#username').type('<a ascript>alert(`xss`)</script>') 28: cy.get('#submit').click() 29: >>> 30: cy.on('window:alert', (t) => { 31: expect(t).to.equal('xss') 32: }) 33: </pre>		
	test/cypress/e2e/register.spec.ts (Line 17)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 14: it('should be possible to bypass validation by directly using Rest API', async () => { 15: cy.task('isDocker').then((isDocker) => { 16: if (!isDocker) { >>> 17: cy.window().then(async () => { 18: const response = await fetch(19: `\${Cypress.config('baseUrl')}/api/Users/`, 20: { </pre>		
	test/cypress/e2e/register.spec.ts (Line 40)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 37: }) 38: 39: cy.visit('/#/administration') >>> 40: cy.on('window:alert', (t) => { 41: expect(t).to.equal('xss') 42: }) 43: cy.expectChallengeSolved({ challenge: 'Client-side XSS Protection' }) </pre>		
	test/cypress/e2e/restApi.spec.ts (Line 12)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.




■ Problematic Code: <pre> 9: xit('should be possible to create a new product when logged in', () => { 10: cy.task('isDocker').then((isDocker) => { 11: if (!isDocker) { >>> 12: cy.window().then(async () => { 13: const response = await fetch(14: `\${Cypress.config('baseUrl')}/api/Products`, 15: { </pre>		
	test/cypress/e2e/restApi.spec.ts (Line 38)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 35: cy.reload() 36: cy.get('img[alt="RestXSS"]').click() 37: >>> 38: cy.on('window:alert', (t) => { 39: expect(t).to.equal('xss') 40: }) 41: </pre>		
	test/cypress/e2e/restApi.spec.ts (Line 89)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 86: it('should be possible to save log-in IP when logged in', () => { 87: cy.task('isDocker').then((isDocker) => { 88: if (!isDocker) { >>> 89: cy.window().then(async () => { 90: const response = await fetch(91: `\${Cypress.config('baseUrl')}/rest/saveLoginIp`, 92: { </pre>		
	test/cypress/e2e/trackOrder.spec.ts (Line 7)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 4: xit('Order Id should be susceptible to reflected XSS attacks', () => { 5: cy.task('isDocker').then((isDocker) => { 6: if (!isDocker) { >>> 7: cy.on('uncaught:exception', (_err, _runnable) => { 8: return false 9: }) 10: </pre>		
	test/cypress/e2e/trackOrder.spec.ts (Line 15)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.





■ Problematic Code: <pre> 12: cy.visit('/#/track-result?id=<iframe src="javascript:alert(`xss`)">') 13: cy.reload() 14: >>> 15: cy.on('window:alert', (t) => { 16: expect(t).to.equal('xss') 17: }) 18: </pre>		
H	routes/fileUpload.ts (Line 75)	typescript:S3776 Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.
■ Problematic Code: <pre> 72: next() 73: } 74: >>> 75: function handleXmlUpload ({ file }: Request, res: Response, next: NextFunction) { 76: if (utils.endsWith(file?.originalname.toLowerCase(), '.xml')) { 77: challengeUtils.solveIf(challenges.deprecatedInterfaceChallenge, () => { return true 78: 79: if (((file?.buffer) != null) && 80: utils.isChallengeEnabled(challenges.deprecatedInterfaceChallenge)) { // XXE attacks in 81: Docker/Heroku containers regularly cause "segfault" crashes 82: } 83: } </pre>		
H	routes/search.ts (Line 47)	typescript:S3735 Remove this use of the "void" operator.
■ Problematic Code: <pre> 44: } 45: if (challengeUtils.notSolved(challenges.dbSchemaChallenge)) { 46: let solved = true >>> 47: void models.sequelize.query('SELECT sql FROM sqlite_master').then(([data]: 48: any) => { 49: const tableDefinitions = utils.queryResultToJson(data) 50: if (tableDefinitions.data?.length) { 51: for (let i = 0; i < tableDefinitions.data.length; i++) { </pre>		
H	data/datacache.ts (Line 33)	typescript:S6861 Exporting mutable 'let' binding, use 'const' instead.
■ Problematic Code: <pre> 30: } 31: export const notifications: Notification[] = [] 32: >>> 33: export let retrieveBlueprintChallengeFile: string null = null 34: export function setRetrieveBlueprintChallengeFile (retrieveBlueprintChallengeFileArg: 35: string) { 36: retrieveBlueprintChallengeFile = retrieveBlueprintChallengeFileArg 37: } </pre>		
H	test/cypress/e2e/scoreBoard.spec.ts (Line 45)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code: <pre> 42:) 43: 44: cy.get('@showSolvedNotifications').then((showSolvedNotifications) => { >>> 45: cy.get('@showFlagsInNotifications').then((showFlagsInNotifications) => { 46: if (showSolvedNotifications && showFlagsInNotifications) { 47: cy.get('.challenge-solved-toast').then((arrayOfSolvedToasts) => { 48: const alertsBefore = Cypress.\$(arrayOfSolvedToasts).length </pre>		
	test/cypress/e2e/basket.spec.ts (Line 80)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 77: }) 78: describe('challenge "manipulateClock"', () => { 79: it('should be possible to enter WMNSDY2019 coupon & place order with this expired coupon', () => { >>> 80: cy.window().then(() => { 81: window.localStorage.couponPanelExpanded = false 82: }) 83: cy.visit('/#/payment/shop') </pre>		
	test/cypress/e2e/basket.spec.ts (Line 85)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 82: }) 83: cy.visit('/#/payment/shop') 84: >>> 85: cy.window().then((win) => { 86: cy.on('uncaught:exception', (_err, _runnable) => { 87: // Introduced to disable the uncaught:exception we get after the eval under this as TypeError: Date.now is not a function 88: return false </pre>		
	test/cypress/e2e/basket.spec.ts (Line 141)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 138: 139: cy.visit('/#/payment/shop') 140: cy.get('#collapseCouponElement').click() >>> 141: cy.task<string>('GenerateCoupon', 90).then((coupon: string) => { 142: cy.get('#coupon').type(coupon) 143: cy.get('#applyCouponButton').click() 144: }) </pre>		
	test/cypress/e2e/search.spec.ts (Line 64)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code: <pre> 61: cy.request("/rest/products/search?q=''))--") 62: .its('body') 63: .then((sourceContent) => { >>> 64: cy.task<Product>('GetPastebinLeakProduct').then((pastebinLeakProduct: Product) => { 65: let foundProduct = false 66: 67: sourceContent.data.forEach((product: Product) => { </pre>		
	test/cypress/e2e/search.spec.ts (Line 91)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 88: cy.request("/rest/products/search?q=''))--") 89: .its('body') 90: .then((sourceContent) => { >>> 91: cy.task<Product>('GetChristmasProduct').then((christmasProduct: Product) => { 92: let foundProduct = false 93: 94: sourceContent.data.forEach((product: Product) => { </pre>		
	test/cypress/e2e/search.spec.ts (Line 109)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 106: cy.request('/api/products') 107: .its('body') 108: .then((sourceContent) => { >>> 109: cy.task<Product>('GetChristmasProduct').then((christmasProduct: Product) => { 110: sourceContent.data.forEach((product: Product) => { 111: if (product.name === christmasProduct.name) { 112: cy.window().then(async () => { </pre>		
	test/cypress/e2e/totpSetup.spec.ts (Line 33)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 30: // console.log(\$val); 31: cy.get('#currentPasswordSetup').type('K1f.....') 32: >>> 33: cy.task<string>('GenerateAuthenticator', \$val).then((secret: string) => { 34: cy.get('#initialToken').type(secret) 35: cy.get('#setupTwoFactorAuth').click() 36: </pre>		
	routes/metrics.ts (Line 209)	typescript:S3735 Remove this use of the "void" operator.

	<p>■ Problematic Code:</p> <pre> 206: if (count) userMetrics.set({ type: 'deluxe' }, count) 207: }) 208: >>> 209: void UserModel.count().then((count: number) => { 210: if (count) userTotalMetrics.set(count) 211: }) 212: </pre>	
	routes/metrics.ts (Line 213)	typescript:S3735 Remove this use of the "void" operator.
	<p>■ Problematic Code:</p> <pre> 210: if (count) userTotalMetrics.set(count) 211: }) 212: >>> 213: void WalletModel.sum('balance').then((totalBalance: number) => { 214: if (totalBalance) walletMetrics.set(totalBalance) 215: }) 216: </pre>	
	routes/chatbot.ts (Line 50)	typescript:S3776 Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.
	<p>■ Problematic Code:</p> <pre> 47: 48: void initializeChatbot() 49: >>> 50: async function processQuery (user: User, req: Request, res: Response, next: NextFunction) { 51: if (bot == null) { 52: res.status(503).send() 53: return </pre>	
	routes/order.ts (Line 37)	typescript:S3776 Refactor this function to reduce its Cognitive Complexity from 34 to the 15 allowed.
	<p>■ Problematic Code:</p> <pre> 34: return (req: Request, res: Response, next: NextFunction) => { 35: const id = req.params.id 36: BasketModel.findOne({ where: { id }, include: [{ model: ProductModel, paranoid: false, as: 'Products' }] }) >>> 37: .then(async (basket: BasketModel null) => { 38: if (basket != null) { 39: const customer = security.authenticatedUsers.from(req) 40: const email = customer ? customer.data ? customer.data.email : '' : '' </pre>	
	frontend/src/app/web3-sandbox web3-sandbox.component.ts (Line 190)	typescript:S3776 Refactor this function to reduce its Cognitive Complexity from 16 to the 15 allowed.

■ Problematic Code: <pre> 187: } 188: } 189: >>> 190: async invokeFunction (func) { 191: if (!this.session) { 192: this.snackBarHelperService.open('PLEASE_CONNECT_WEB3_WALLET', 'errorBar') 193: return </pre>		
	routes/chatbot.ts (Line 25)	typescript:S6861 Exporting mutable 'let' binding, use 'const' instead.
■ Problematic Code: <pre> 22: 23: let trainingFile = config.get<string>('application.chatBot.trainingData') 24: let testCommand: string >>> 25: export let bot: Bot null = null 26: 27: export async function initializeChatbot () { 28: if (utils.isUrl(trainingFile)) { </pre>		
	test/cypress/e2e/chatbot.spec.ts (Line 46)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 43: cy.get('#message-input').type('...').type('{enter}') 44: 45: const genArr = Array.from({ length: 100 }, (v, k) => k + 1) >>> 46: cy.wrap(genArr).eachSeries(() => { 47: cy.get('#message-input') 48: .type(couponIntent.utterances[0]) 49: .type('{enter}') </pre>		
	test/cypress/e2e/basket.spec.ts (Line 9)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 6: 7: describe('challenge "negativeOrder"', () => { 8: it('should be possible to update a basket to a negative quantity via the Rest API', () => { >>> 9: cy.window().then(async () => { 10: const response = await fetch(11: `\${Cypress.config('baseUrl')}/api/BasketItems/1`, 12: { </pre>		
	test/cypress/e2e/basket.spec.ts (Line 30)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code: <pre> 27: 28: cy.get('mat-cell.mat-column-quantity > span') 29: .first() >>> 30: .then((\$ele) => { 31: const quantity = \$ele.text() 32: expect(quantity).toMatch(/-100000/) 33: }) </pre>		
	test/cypress/e2e/basket.spec.ts (Line 45)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 42: 43: describe('challenge "basketAccessChallenge"', () => { 44: it('should access basket with id from session storage instead of the one associated to logged-in user', () => { >>> 45: cy.window().then(() => { 46: window.sessionStorage.bid = 3 47: }) 48: }) </pre>		
	test/cypress/e2e/basket.spec.ts (Line 58)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 55: 56: describe('challenge "basketManipulateChallenge"', () => { 57: it('should manipulate basket of other user instead of the one associated to logged-in user', () => { >>> 58: cy.window().then(async () => { 59: await fetch(`\${Cypress.config('baseUrl')}/api/BasketItems/`, { 60: method: 'POST', 61: cache: 'no-cache', </pre>		
	test/cypress/e2e/basket.spec.ts (Line 111)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 108: }) 109: 110: it('should be possible to add a product in the basket', () => { >>> 111: cy.window().then(async () => { 112: const response = await fetch(113: `\${Cypress.config('baseUrl')}/api/BasketItems/`, 114: { </pre>		
	test/cypress/e2e/basket.spec.ts (Line 135)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code: <pre> 132: }) 133: 134: it('should be possible to enter a coupon that gives an 80% discount', () => { >>> 135: cy.window().then(() => { 136: window.localStorage.couponPanelExpanded = false 137: }) 138: }) </pre>		
	test/cypress/e2e/contact.spec.ts (Line 154)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 151: await sendPostRequest(responseJson) 152: } 153: >>> 154: async function sendPostRequest (captcha: { 155: captchaId: number 156: answer: string 157: }) { </pre>		
	test/cypress/e2e/contact.spec.ts (Line 202)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 199: await sendPostRequest(responseJson) 200: } 201: >>> 202: async function sendPostRequest (captcha: { 203: captchaId: number 204: answer: string 205: }) { </pre>		
	test/cypress/e2e/deluxe.spec.ts (Line 29)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 26: url: '/rest/deluxe-membership', 27: method: 'POST', 28: headers: { Authorization: `Bearer \${token?.value}` } >>> 29: }).then((response) => { 30: expect(response.body.status).contains('success') 31: }) 32: }) </pre>		
	test/cypress/e2e/forgotPassword.spec.ts (Line 54)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code: <pre> 51: describe('for his internal account', () => { 52: it('should be able to reset password with his security answer', () => { 53: cy.task<string>('GetFromConfig', 'application.domain').then(>>> 54: (appDomain: string) => { 55: cy.get('#email').type(`bjoern@\${appDomain}`) 56: } 57:) </pre>		
	test/cypress/e2e/login.spec.ts (Line 128)	typescript:S3735 Remove this use of the "void" operator.
■ Problematic Code: <pre> 125: 126: cy.task<string>('GenerateAuthenticator', 'IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH').then(127: (totpToken: string) => { >>> 128: void cy.get('#totpToken').type(totpToken) 129: void cy.get('#totpSubmitButton').click() 130: } 131:) </pre>		
	test/cypress/e2e/login.spec.ts (Line 129)	typescript:S3735 Remove this use of the "void" operator.
■ Problematic Code: <pre> 126: cy.task<string>('GenerateAuthenticator', 'IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH').then(127: (totpToken: string) => { 128: void cy.get('#totpToken').type(totpToken) >>> 129: void cy.get('#totpSubmitButton').click() 130: } 131:) 132: cy.expectChallengeSolved({ challenge: 'Two Factor Authentication' }) </pre>		
	test/cypress/e2e/noSql.spec.ts (Line 14)	typescript:S3735 Remove this use of the "void" operator.
■ Problematic Code: <pre> 11: cy.task('isDocker').then((isDocker) => { 12: if (!isDocker) { 13: cy.window().then(() => { >>> 14: void fetch(15: `\${Cypress.config('baseUrl')}/rest/products/sleep(1000)/reviews`, 16: { 17: method: 'GET', </pre>		
	test/cypress/e2e/noSql.spec.ts (Line 97)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code: <pre> 94: await editReview(reviewId) 95: } 96: >>> 97: async function editReview (reviewId: string) { 98: const response = await fetch(99: `\${Cypress.config('baseUrl')}/rest/products/reviews`, 100: { </pre>		
H	test/cypress/e2e/noSql.spec.ts (Line 126)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 123: it('should be possible to like reviews multiple times', () => { 124: cy.visit('/') 125: cy.window().then(async () => { >>> 126: async function sendPostRequest (reviewId: string) { 127: const anotherResponse = await fetch(128: `\${Cypress.config('baseUrl')}/rest/products/reviews`, 129: { </pre>		
H	data/datacreator.ts (Line 406)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 403: product: id, 404: likesCount: 0, 405: likedBy: [] >>> 406: }).catch((err: unknown) => { 407: logger.error(`Could not insert Product Review \${text}: \${utils.getErrorMessage(err)}`) 408: }) 409:) </pre>		
H	routes/metrics.ts (Line 201)	typescript:S3735 Remove this use of the "void" operator.
■ Problematic Code: <pre> 198: if (reviewCount) interactionsMetrics.set({ type: 'review' }, reviewCount) 199: }) 200: >>> 201: void UserModel.count({ where: { role: { [Op.eq]: 'customer' } } }).then((count: number) => { 202: if (count) userMetrics.set({ type: 'standard' }, count) 203: }) 204: </pre>		
H	routes/metrics.ts (Line 205)	typescript:S3735 Remove this use of the "void" operator.

■ Problematic Code:

```

202:         if (count) userMetrics.set({ type: 'standard' }, count)
203:     })
204:
>>> 205:     void UserModel.count({ where: { role: { [Op.eq]: 'deluxe' } } }).then((count:
number) => {
206:         if (count) userMetrics.set({ type: 'deluxe' }, count)
207:     })
208:

```



routes/metrics.ts
(Line 217)

typescript:S3735

Remove this use of the "void" operator.

■ Problematic Code:

```

214:         if (totalBalance) walletMetrics.set(totalBalance)
215:     })
216:
>>> 217:     void FeedbackModel.count().then((count: number) => {
218:         if (count) interactionsMetrics.set({ type: 'feedback' }, count)
219:     })
220:

```



routes/metrics.ts
(Line 221)

typescript:S3735

Remove this use of the "void" operator.

■ Problematic Code:

```

218:         if (count) interactionsMetrics.set({ type: 'feedback' }, count)
219:     })
220:
>>> 221:     void ComplaintModel.count().then((count: number) => {
222:         if (count) interactionsMetrics.set({ type: 'complaint' }, count)
223:     })
224:     } catch (e: unknown) {

```



routes/fileUpload.ts
(Line 49)

typescript:S2004

Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code:

```

46:         } else {
47:             entry.autodrain()
48:         }
>>> 49:     }).on('error', function (err: unknown) { next(err) })
50:     })
51:     })
52:     })





```



routes/order.ts
(Line 48)

typescript:S3735

Remove this use of the "void" operator.

■ Problematic Code: <pre> 45: const fileWriter = doc.pipe(fs.createWriteStream(path.join('ftp/', pdfFile))) 46: 47: fileWriter.on('finish', async () => { >>> 48: void basket.update({ coupon: null }) 49: await BasketItemModel.destroy({ where: { BasketId: id } }) 50: res.json({ orderConfirmation: orderId }) 51: }) </pre>		
	routes/verify.ts (Line 175)	typescript:S3735 Remove this use of the "void" operator.
■ Problematic Code: <pre> 172: 173: function changeProductChallenge (osoft: Product) { 174: let urlForProductTamperingChallenge: string null = null >>> 175: void osoft.reload().then(() => { 176: for (const product of config.get<ProductConfig[]>('products')) { 177: if (product.urlForProductTamperingChallenge !== undefined) { 178: urlForProductTamperingChallenge = product.urlForProductTamperingChallenge </pre>		
	routes/fileUpload.ts (Line 40)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 37: fs.close(fd, function () { 38: fs.createReadStream(tempFile) 39: .pipe(unzipper.Parse()) >>> 40: .on('entry', function (entry: any) { 41: const fileName = entry.path 42: const absolutePath = path.resolve('uploads/complaints/' + fileName) 43: challengeUtils.solveIf(challenges.fileWriteChallenge, () => { return absolutePath === path.resolve('ftp/legal.md') }) }) </pre>		
	test/server/verifySpec.ts (Line 168)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 165: describe('is solved when an error occurs on a response with error', () => { 166: const httpStatus = [402, 403, 404, 500] 167: httpStatus.forEach(statusCode => { >>> 168: it(`\${statusCode} status code`, () => { 169: res.statusCode = statusCode 170: err = new Error() 171: </pre>		
	test/server/verifySpec.ts (Line 191)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code: <pre> 188: describe('is not solved when no error occurs on a response with error', () => { 189: const httpStatus = [401, 402, 404, 500] 190: httpStatus.forEach(statusCode => { >>> 191: it(`\${statusCode} status code`, () => { 192: res.statusCode = statusCode 193: err = undefined 194: </pre>		
	data/static/codefixes adminSectionChallenge_3.ts (Line 3)	typescript:S3504 Unexpected var, use let or const instead.
■ Problematic Code: <pre> 1: const routes: Routes = [2: { >>> 3: path: (function(){var t=Array.prototype.slice.call(arguments),G=t.shift();return t.reverse().map(function(e,w){return String.fromCharCode(e-G-2-w)}).join('')})(55,167,171,165,1 68,158,154)+(62749278960).toString(36).toLowerCase()+((function(){var b=Array.prototype.slice.call(arguments),V=b.shift();return b.reverse().map(function(l,S){return String.fromCharCode(l-V-43-S)}).join('')})(58,211), 4: component: AdministrationComponent, 5: canActivate: [AuthGuard] 6: }, </pre>		
	data/static/codefixes adminSectionChallenge_3.ts (Line 3)	typescript:S3504 Unexpected var, use let or const instead.
■ Problematic Code: <pre> 1: const routes: Routes = [2: { >>> 3: path: (function(){var t=Array.prototype.slice.call(arguments),G=t.shift();return t.reverse().map(function(e,w){return String.fromCharCode(e-G-2-w)}).join('')})(55,167,171,165,1 68,158,154)+(62749278960).toString(36).toLowerCase()+((function(){var b=Array.prototype.slice.call(arguments),V=b.shift();return b.reverse().map(function(l,S){return String.fromCharCode(l-V-43-S)}).join('')})(58,211), 4: component: AdministrationComponent, 5: canActivate: [AuthGuard] 6: }, </pre>		
	test/api/dataExportApiSpec.ts (Line 218)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
■ Problematic Code: <pre> 215: .expect('status', 200) 216: .expect('header', 'content-type', /application\/json/) 217: .expect('json', 'confirmation', 'Your data export will open in a new Browser window.') >>> 218: .then(({ json }) => { 219: const parsedData = JSON.parse(json.userData) 220: expect(parsedData.username).toBe('') 221: expect(parsedData.email).toBe('jim@' + config.get<string>('application.domain')) </pre>		

H	test/api/dataExportApiSpec.ts (Line 352)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div data-bbox="159 324 1476 660"> <div>■ Problematic Code:</div> <pre> 349: }) 350: .expect('status', 200) 351: .expect('header', 'content-type', /application\/json/) >>> 352: .then(({ json: captchaAnswer }) => { 353: return frisby.post(REST_URL + '/user/data-export', { 354: headers: { Authorization: 'Bearer ' + jsonLogin.authentication.token, 'content-type': 'application/json' }, 355: body: { </pre> </div>		
H	test/api/dataExportApiSpec.ts (Line 98)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div data-bbox="159 750 1476 1086"> <div>■ Problematic Code:</div> <pre> 95: .expect('status', 200) 96: .expect('header', 'content-type', /application\/json/) 97: .expect('json', 'confirmation', 'Your data export will open in a new Browser window.') >>> 98: .then(({ json }) => { 99: const parsedData = JSON.parse(json.userData) 100: expect(parsedData.username).toBe('bkimminich') 101: expect(parsedData.email).toBe('bjoern.kimminich@gmail.com') </pre> </div>		
H	test/api/dataExportApiSpec.ts (Line 131)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div data-bbox="159 1176 1476 1545"> <div>■ Problematic Code:</div> <pre> 128: .expect('status', 200) 129: .expect('header', 'content-type', /application\/json/) 130: .expect('json', 'confirmation', 'Your data export will open in a new Browser window.') >>> 131: .then(({ json }) => { 132: const parsedData = JSON.parse(json.userData) 133: expect(parsedData.username).toBe('') 134: expect(parsedData.email).toBe('amy@' + config.get<string>('application.domain')) </pre> </div>		
H	test/api/dataExportApiSpec.ts (Line 249)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.
<div data-bbox="159 1635 1476 1971"> <div>■ Problematic Code:</div> <pre> 246: }) 247: .expect('status', 200) 248: .expect('header', 'content-type', /application\/json/) >>> 249: .then(({ json: captchaAnswer }) => { 250: return frisby.post(REST_URL + '/user/data-export', { 251: headers: { Authorization: 'Bearer ' + jsonLogin.authentication.token, 'content-type': 'application/json' }, 252: body: { </pre> </div>		
H	test/api/dataExportApiSpec.ts (Line 303)	typescript:S2004 Refactor this code to not nest functions more than 4 levels deep.

■ Problematic Code: <pre> 300: .expect('status', 200) 301: .expect('header', 'content-type', /application\/json/) 302: .expect('json', 'confirmation', 'Your data export will open in a new Browser window.') >>> 303: .then(({ json }) => { 304: const parsedData = JSON.parse(json.userData) 305: expect(parsedData.username).toBe('') 306: expect(parsedData.email).toBe('jim@' + config.get<string>('application.domain')) </pre>		
M	frontend/src/app/address address.component.scss (Line 72)	css:S4666 Unexpected duplicate selector ".mat-column-Address", first used at line 63
■ Problematic Code: <pre> 69: width: 48px; 70: } 71: >>> 72: .mat-column-Address { 73: word-break: break-word; 74: } 75: </pre>		
M	frontend/src/app/token-sale token-sale.component.scss (Line 117)	css:S4666 Unexpected duplicate selector ".mdc-card", first used at line 79
■ Problematic Code: <pre> 114: width: 10px; 115: } 116: >>> 117: .mdc-card { border: 0; } 118: 119: .offer-container { display: flex; 120: flex-direction: column; </pre>		
M	frontend/src/app/token-sale token-sale.component.scss (Line 119)	css:S4666 Unexpected duplicate selector ".offer-container", first used at line 94
■ Problematic Code: <pre> 116: 117: .mdc-card { border: 0; } 118: >>> 119: .offer-container { display: flex; 120: flex-direction: column; 121: gap: 16px; } 122: </pre>		
M	frontend/src/app/token-sale token-sale.component.scss (Line 123)	css:S4666 Unexpected duplicate selector ".faq-container", first used at line 46

<p>■ Problematic Code:</p> <pre> 120: flex-direction: column; 121: gap: 16px; } 122: >>> 123: .faq-container { display: flex; 124: flex-direction: column; 125: gap: 4px; 126: margin-top: -7px; }</pre>		
M	Dockerfile (Line 22)	docker:S6596 Use a specific version tag for the image.
<p>■ Problematic Code:</p> <pre> 19: RUN npm install -g @cyclonedx/cyclonedx-npm@\$CYCLONEDX_NPM_VERSION 20: RUN npm run sbom 21: >>> 22: FROM gcr.io/distroless/nodejs22-debian12 23: ARG BUILD_DATE 24: ARG VCS_REF 25: LABEL maintainer="Bjoern Kimminich <bjoern.kimminich@owasp.org>" \</pre>		
M	server.ts (Line 19)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 16: import colors from 'colors/safe' 17: import serveIndex from 'serve-index' 18: import bodyParser from 'body-parser' >>> 19: // @ts-expect-error FIXME due to non-existing type definitions for finale-rest 20: import * as finale from 'finale-rest' 21: import compression from 'compression' 22: // @ts-expect-error FIXME due to non-existing type definitions for express-robots-txt</pre>		
M	server.ts (Line 22)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 19: // @ts-expect-error FIXME due to non-existing type definitions for finale-rest 20: import * as finale from 'finale-rest' 21: import compression from 'compression' >>> 22: // @ts-expect-error FIXME due to non-existing type definitions for express-robots-txt 23: import robots from 'express-robots-txt' 24: import cookieParser from 'cookie-parser' 25: import * as Prometheus from 'prom-client'</pre>		
M	server.ts (Line 29)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<p>■ Problematic Code:</p> <pre> 26: import swaggerUi from 'swagger-ui-express' 27: import featurePolicy from 'feature-policy' 28: import { IpFilter } from 'express-ipfilter' >>> 29: // @ts-expect-error FIXME due to non-existing type definitions for express-security.txt 30: import securityTxt from 'express-security.txt' 31: import { rateLimit } from 'express-rate-limit' 32: import { getStream } from 'file-stream-rotator' </pre>		
M	routes/userProfile.ts (Line 90)	typescript:S6397 Replace this character class by the character itself.
<p>■ Problematic Code:</p> <pre> 87: const CSP = `img-src 'self' \${user?.profileImage}; script-src 'self' 'unsafe-eval' https://code.getmdl.io http://ajax.googleapis.com` 88: 89: challengeUtils.solveIf(challenges.usernameXssChallenge, () => { >>> 90: return username && user?.profileImage.match(/;[]*script-src(.*'unsafe-inline'/g) !== null && utils.contains(username, '<script>alert(`xss`)</script>') 91: }) 92: 93: res.set({ </pre>		
M	routes/b2bOrder.ts (Line 8)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 5: 6: import vm from 'node:vm' 7: import { type Request, type Response, type NextFunction } from 'express' >>> 8: // @ts-expect-error FIXME due to non-existing type definitions for notevil 9: import { eval as safeEval } from 'notevil' 10: 11: import * as challengeUtils from '../lib/challengeUtils' </pre>		
M	data/datacreator.ts (Line 36)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 33: import { AllHtmlEntities as Entities } from 'html-entities' 34: import * as datacache from './datacache' 35: import * as security from '../lib/insecurity' >>> 36: // @ts-expect-error FIXME due to non-existing type definitions for replace 37: import replace from 'replace' 38: 39: const entities = new Entities() </pre>		
M	test/cypress/e2e/complain.spec.ts (Line 113)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

	<p>■ Problematic Code:</p> <pre> 110: }) 111: }) 112: >>> 113: xit('should be solved either through dev/random or Quadratic Blowup attack', () => { // FIXME Unreliable during CI/CD as sometimes the Quadratic Blowup is blocked for entity loops 114: cy.task('isDocker').then((isDocker) => { 115: if (!isDocker) { 116: cy.expectChallengeSolved({ challenge: 'XXE DoS' }) </pre>	
M	test/api/fileUploadSpec.ts (Line 182)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
	<p>■ Problematic Code:</p> <pre> 179: .expect('status', 204) 180: }) 181: >>> 182: xit('POST valid file with tampered content length', () => { // FIXME Fails on CI/CD pipeline 183: const file = path.resolve(__dirname, '../files/validSizeAndTypeForClient.pdf') 184: const form = frisby.formData() 185: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type </pre>	
M	test/api/profileImageUploadSpec.ts (Line 155)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
	<p>■ Problematic Code:</p> <pre> 152: .expect('bodyContains', 'Error: Blocked illegal activity') 153: }) 154: >>> 155: xit('POST valid image with tampered content length', () => { // FIXME Fails on CI/CD pipeline 156: const file = path.resolve(__dirname, '../files/validProfileImage.jpg') 157: const form = frisby.formData() 158: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type </pre>	
M	test/api/fileUploadSpec.ts (Line 187)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
	<p>■ Problematic Code:</p> <pre> 184: const form = frisby.formData() 185: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 186: >>> 187: // @ts-expect-error FIXME form.getHeaders() is not found 188: return frisby.post(URL + '/file-upload', { headers: { 'Content-Type': form.getHeaders()['content-type'], 'Content-Length': 42 }, body: form }) 189: .expect('status', 500) 190: .expect('bodyContains', 'Unexpected end of form') </pre>	
M	test/api/profileImageUploadSpec.ts (Line 172)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

■	Problematic Code:	<pre> 169: return frisby.post(`\${URL}/profile/image/file`, { 170: headers: { 171: Cookie: `token=\${jsonLogin.authentication.token}`, >>> 172: // @ts-expect-error FIXME form.getHeaders() is not found 173: 'Content-Type': form.getHeaders()['content-type'], 174: 'Content-Length': 42 175: }, </pre>	
M	test/api/fileUploadSpec.ts (Line 143)	typescript:S1134 Take the required action to fix the issue indicated by this comment.	
■	Problematic Code:	<pre> 140: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 141: 142: return frisby.post(URL + '/file-upload', { >>> 143: // @ts-expect-error FIXME form.getHeaders() is not found 144: headers: { 'Content-Type': form.getHeaders()['content-type'] }, 145: body: form 146: }).then((res) => { </pre>	
M	frontend/src/app/search-result search-result.component.scss (Line 25)	css:S4666 Unexpected duplicate selector ".heading", first used at line 13	
■	Problematic Code:	<pre> 22: width: 60% !important; /* image ~60%, info-box ~40% */ 23: } 24: >>> 25: .heading { 26: margin-bottom: 10px; 27: } 28: </pre>	
M	frontend/src/app/search-result search-result.component.scss (Line 151)	css:S4666 Unexpected duplicate selector ".product", first used at line 29	
■	Problematic Code:	<pre> 148: margin-top: 10px; 149: } 150: >>> 151: .product { 152: margin-left: -17px; 153: margin-right: 6px; 154: margin-top: -17px; </pre>	
M	frontend/src/app/faucet faucet.component.scss (Line 139)	css:S4666 Unexpected duplicate selector ".withdraw-container button", first used at line 85	

■ Problematic Code: <pre> 136: width: 100%; 137: } 138: >>> 139: .withdraw-container button { 140: height: 68px; 141: } 142: @media (max-width: 1100px) { </pre>		
M	frontend/src/app/token-sale token-sale.component.scss (Line 83)	css:S4666 Unexpected duplicate selector ".faq-container", first used at line 46
■ Problematic Code: <pre> 80: border: 0; 81: } 82: >>> 83: .faq-container { 84: display: flex; 85: flex-direction: column; 86: gap: 4px; </pre>		
M	frontend/src/app/navbar navbar.component.scss (Line 43)	css:S4666 Unexpected duplicate selector "mat-toolbar", first used at line 6
■ Problematic Code: <pre> 40: width: 26px; 41: } 42: >>> 43: mat-toolbar { 44: mat-icon { 45: font-size: 23px; 46: height: 24px; </pre>		
M	frontend/src/app/forgot-password forgot-password.component.scss (Line 30)	css:S4666 Unexpected duplicate selector "mat-form-field", first used at line 12
■ Problematic Code: <pre> 27: width: 60%; 28: } 29: >>> 30: mat-form-field { 31: input { 32: color: #fff !important; 33: } </pre>		
M	test/cypress/e2e/changePassword.s pec.ts (Line 14)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<p>■ Problematic Code:</p> <pre> 11: it('should be able to change password', () => { 12: cy.get('#currentPassword').type('focusOnScienceMorty!focusOnScience') 13: cy.get('#newPassword').type('GonorrheaCantSeeUs!') >>> 14: cy.get('#newPasswordRepeat').type('GonorrheaCantSeeUs!', { force: true }) // FIXME Analyze Cypress recordings to properly fix behavior during test 15: cy.get('#changeButton').click() 16: 17: cy.get('.confirmation').should('not.be.hidden') </pre>		
M	test/cypress/e2e/contact.spec.ts (Line 25)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 22: cy.get('#userId').clear().type('2') 23: cy.get('#rating').type('{rightarrow}{rightarrow}{rightarrow}') 24: cy.get('#comment').type('Picard stinks!') >>> 25: cy.get('#submitButton').click({ force: true }) // FIXME Analyze Cypress recordings to properly fix behavior during test 26: 27: cy.visit('/#/administration') 28: </pre>		
M	test/cypress/e2e/contact.spec.ts (Line 61)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 58: cy.get('#comment').type(59: '<<script>Foo</script>iframe src="javascript:alert(`xss`)">' 60:) >>> 61: cy.get('#submitButton').click({ force: true }) // FIXME Analyze Cypress recordings to properly fix behavior during test 62: 63: cy.visit('/#/about') 64: cy.on('window:alert', (t) => { </pre>		
M	test/cypress/e2e/contact.spec.ts (Line 84)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 81: cy.get('#comment').type('sanitize-html 1.4.2 is non-recursive.') 82: cy.get('#comment').type('express-jwt 0.1.3 has broken crypto.') 83: >>> 84: cy.get('#submitButton').click({ force: true }) // FIXME Analyze Cypress recordings to properly fix behavior during test 85: cy.expectChallengeSolved({ challenge: 'Vulnerable Library' }) 86: }) 87: }) </pre>		
M	test/cypress/e2e/contact.spec.ts (Line 95)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<div> <div>■ Problematic Code:</div> <pre> 92: cy.get('#comment').type(93: 'The following libraries are bad for crypto: z85, base85, md5 and hashids' 94:) >>> 95: cy.get('#submitButton').click({ force: true }) // FIXME Analyze Cypress recordings to properly fix behavior during test 96: cy.expectChallengeSolved({ challenge: 'Weird Crypto' }) 97: }) 98: }) </pre> </div>		
<div>M</div>	test/cypress/e2e/contact.spec.ts (Line 106)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 103: cy.get('#comment').type(104: 'You are a typosquatting victim of this NPM package: epilogue-js' 105:) >>> 106: cy.get('#submitButton').click({ force: true }) // FIXME Analyze Cypress recordings to properly fix behavior during test 107: cy.expectChallengeSolved({ challenge: 'Legacy Typosquatting' }) 108: }) 109: }) </pre> </div>		
<div>M</div>	test/cypress/e2e/contact.spec.ts (Line 117)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 114: cy.get('#comment').type(115: 'You are a typosquatting victim of this NPM package: ngy-cookie' 116:) >>> 117: cy.get('#submitButton').click({ force: true }) // FIXME Analyze Cypress recordings to properly fix behavior during test 118: cy.expectChallengeSolved({ challenge: 'Frontend Typosquatting' }) 119: }) 120: }) </pre> </div>		
<div>M</div>	test/cypress/e2e/contact.spec.ts (Line 128)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 125: cy.get('#comment').type(126: 'Pickle Rick is hiding behind one of the support team ladies' 127:) >>> 128: cy.get('#submitButton').click({ force: true }) // FIXME Analyze Cypress recordings to properly fix behavior during test 129: cy.expectChallengeSolved({ challenge: 'Steganography' }) 130: }) 131: }) </pre> </div>		
<div>M</div>	test/cypress/e2e/contact.spec.ts (Line 232)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<p>■ Problematic Code:</p> <pre> 12: import logger from './logger' 13: import { type NextFunction, type Request, type Response } from 'express' 14: import * as utils from './utils' >>> 15: // @ts-expect-error FIXME due to non-existing type definitions for median 16: import median from 'median' 17: import { type ChallengeKey } from 'models/challenge' 18: </pre>		
M	lib/startup/customizeApplication.ts (Line 9)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 6: import fs from 'node:fs' 7: import config from 'config' 8: import * as utils from './utils' >>> 9: // @ts-expect-error FIXME due to non-existing type definitions for replace 10: import replace from 'replace' 11: 12: const customizeApplication = async () => { </pre>		
M	lib/startup/customizeEasterEgg.ts (Line 8)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 5: 6: import config from 'config' 7: import * as utils from './utils' >>> 8: // @ts-expect-error FIXME due to non-existing type definitions for replace 9: import replace from 'replace' 10: 11: const customizeEasterEgg = async () => { </pre>		
M	lib/startup/validateConfig.ts (Line 10)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 7: import config from 'config' 8: import process from 'node:process' 9: import colors from 'colors/safe' >>> 10: // @ts-expect-error FIXME due to non-existing type definitions for yaml-schema-validator 11: import validateSchema from 'yaml-schema-validator/src' 12: 13: import type { AppConfig, Memory as MemoryConfig, Product as ProductConfig } from '../config.types' </pre>		
M	lib/startup/validateDependencies.ts (Line 9)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

	<p>■ Problematic Code:</p> <pre> 6: import colors from 'colors/safe' 7: import * as utils from '../utils' 8: import logger from '../logger' >>> 9: // @ts-expect-error FIXME due to non-existing type definitions for check-dependencies 10: import dependencyChecker from 'check-dependencies' 11: 12: const validateDependencies = async ({ packageDir = '.', exitOnFailure = true } = {}) => { </pre>	
<div>M</div>	lib/startup/validateDependenciesBas ic.ts (Line 10)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
	<p>■ Problematic Code:</p> <pre> 7: // otherwise this check would be useless as the app would fail on a random import before even reaching this point 8: const validateIfDependencyCheckerIsInstalled = async () => { 9: try { >>> 10: // @ts-expect-error FIXME due to non-existing type definitions for check-dependencies 11: await import('check-dependencies') 12: } catch (err) { 13: console.error('Please run "npm install" before starting the application!') </pre>	
<div>M</div>	lib/startup/validatePreconditions.ts (Line 15)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
	<p>■ Problematic Code:</p> <pre> 12: import process from 'node:process' 13: import semver from 'semver' 14: import portscanner from 'portscanner' >>> 15: // @ts-expect-error FIXME due to non-existing type definitions for check-internet-connected 16: import checkInternetConnected from 'check-internet-connected' 17: 18: const domainDependencies = { </pre>	
<div>M</div>	lib/antiCheat.ts (Line 77)	typescript:S3358 Extract this nested ternary operation into an independent statement.

	<p>■ Problematic Code:</p> <pre> 74: cheatScore = Math.min(1, cheatScore) 75: } 76: >>> 77: logger.info(`Cheat score for \${areCoupled(challenge, previous().challenge) ? 'coupled' : (isTrivial(challenge) ? 'trivial' : '')}\${challenge.tutorialOrder ? 'tutorial' : ''}\${colors.cyan(challenge.key)} solved in \${Math.round(minutesSincePreviousSolve)}min (expected ~\${minutesExpectedToSolve}min) with\${config.get('challenges.showHints') ? '' : 'out'} hints allowed\${percentPrecedingInteraction > -1 ? (' and ' + percentPrecedingInteraction * 100 + '% expected preceding URL interaction') : ''}: \${cheatScore < 0.33 ? colors.green(cheatScore.toString()) : (cheatScore < 0.66 ? colors.yellow(cheatScore.toString()) : colors.red(cheatScore.toString()))}`) 78: solves.push({ challenge, phase: 'hack it', timestamp, cheatScore }) 79: return cheatScore 80: }</pre>	
<div>M</div>	lib/antiCheat.ts (Line 77)	typescript:S3358 Extract this nested ternary operation into an independent statement.
	<p>■ Problematic Code:</p> <pre> 74: cheatScore = Math.min(1, cheatScore) 75: } 76: >>> 77: logger.info(`Cheat score for \${areCoupled(challenge, previous().challenge) ? 'coupled' : (isTrivial(challenge) ? 'trivial' : '')}\${challenge.tutorialOrder ? 'tutorial' : ''}\${colors.cyan(challenge.key)} solved in \${Math.round(minutesSincePreviousSolve)}min (expected ~\${minutesExpectedToSolve}min) with\${config.get('challenges.showHints') ? '' : 'out'} hints allowed\${percentPrecedingInteraction > -1 ? (' and ' + percentPrecedingInteraction * 100 + '% expected preceding URL interaction') : ''}: \${cheatScore < 0.33 ? colors.green(cheatScore.toString()) : (cheatScore < 0.66 ? colors.yellow(cheatScore.toString()) : colors.red(cheatScore.toString()))}`) 78: solves.push({ challenge, phase: 'hack it', timestamp, cheatScore }) 79: return cheatScore 80: }</pre>	
<div>M</div>	lib/startup/validatePreconditions.ts (Line 90)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
	<p>■ Problematic Code:</p> <pre> 87: }) 88: .catch(() => { 89: logger.warn(`Domain \${colors.bold(domain)} is not reachable (\${colors.yellow('NOT OK')}) in a future major release`) >>> 90: // @ts-expect-error FIXME Type problem by accessing key via variable 91: domainDependencies[domain].forEach((dependency: string) => { 92: logger.warn(`\${colors.italic(dependency)} will not work as intended without access to \${colors.bold(domain)}`) 93: }) </pre>	
<div>M</div>	lib/startup/registerWebsocketEvents.ts (Line 21)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

■ Problematic Code: <pre> 18: 19: const registerWebsocketEvents = (server: any) => { 20: const io = new Server(server, { cors: { origin: 'http://localhost:4200' } }) >>> 21: // @ts-expect-error FIXME Type safety issue when setting global socket-io object 22: globalWithSocketIO.io = io 23: 24: io.on('connection', (socket: any) => { </pre>		
M	routes/login.ts (Line 48)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 45: } 46: }) 47: } else if (user.data?.id) { >>> 48: // @ts-expect-error FIXME some properties missing in user - vuln-code-snippet hide-line 49: afterLogin(user, res, next) 50: } else { 51: res.status(401).send(res.___('Invalid email or password.')) </pre>		
M	test/server/b2bOrderSpec.ts (Line 39)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 36: expect(challenges.rceChallenge.solved).to.equal(true) 37: }) 38: >>> 39: // FIXME Disabled as test started failing on Linux regularly 40: xit('timeout after 2 seconds solves "rceOccupyChallenge"', () => { 41: req.body.orderLinesData = '/((a+)+b)/.test("aaaaaaaaaaaaaaaaaaaaaaaaaaaaa")' 42: </pre>		
M	routes/2fa.ts (Line 49)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 46: const [basket] = await BasketModel.findOrCreate({ where: { UserId: userId } }) 47: 48: const token = security.authorize(plainUser) >>> 49: // @ts-expect-error FIXME set new property for original basket 50: plainUser.bid = basket.id // keep track of original basket for challenge solution check 51: security.authenticatedUsers.put(token, plainUser) 52: </pre>		
M	routes/chatbot.ts (Line 98)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<div> <div>■ Problematic Code:</div> <pre> 95: try { 96: const response = await bot.respond(req.body.query, `\${user.id}`) 97: if (response.action === 'function') { >>> 98: // @ts-expect-error FIXME unclear usage of any type as index 99: if (response.handler && botUtils[response.handler]) { 100: // @ts-expect-error FIXME unclear usage of any type as index 101: res.status(200).json(await botUtils[response.handler](req.body.query, user)) </pre> </div>		
<div>M</div>	routes/chatbot.ts (Line 100)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 97: if (response.action === 'function') { 98: // @ts-expect-error FIXME unclear usage of any type as index 99: if (response.handler && botUtils[response.handler]) { >>> 100: // @ts-expect-error FIXME unclear usage of any type as index 101: res.status(200).json(await botUtils[response.handler](req.body.query, user)) 102: } else { 103: res.status(200).json({ </pre> </div>		
<div>M</div>	data/mongodb.ts (Line 6)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 3: * SPDX-License-Identifier: MIT 4: */ 5: >>> 6: // @ts-expect-error FIXME due to non-existing type definitions for MarsDB 7: import * as MarsDB from 'marsdb' 8: 9: export const reviewsCollection = new MarsDB.Collection('posts') </pre> </div>		
<div>M</div>	lib/antiCheat.ts (Line 127)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 124: } 125: 126: function areCoupled (challenge: Challenge, previousChallenge: Challenge) { >>> 127: // @ts-expect-error FIXME any type issues 128: return coupledChallenges[challenge.key]?.indexOf(previousChallenge.key) > -1 coupledChallenges[previousChallenge.key]?.indexOf(challenge.key) > -1 129: } 130: </pre> </div>		
<div>M</div>	lib/insecurity.ts (Line 19)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

■ Problematic Code: <pre> 16: 17: /* tslint node: true */ 18: // eslint-disable-next-line @typescript-eslint/prefer-ts-expect-error >>> 19: // @ts-expect-error FIXME no typescript definitions for z85 :(20: import * as z85 from 'z85' 21: 22: export const publicKey = fs ? fs.readFileSync('encryptionkeys/jwt.pub', 'utf8') : 'placeholder-public-key' </pre>		
M	lib/startup/validateConfig.ts (Line 81)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 78: export const checkUnambiguousMandatorySpecialProducts = (products: ProductConfig[]) => { 79: let success = true 80: specialProducts.forEach(({ name, key }) => { >>> 81: // @ts-expect-error FIXME Ignoring any type issue on purpose 82: const matchingProducts = products.filter((product) => product[key]) 83: if (matchingProducts.length === 0) { 84: logger.warn(`No product is configured as \${colors.italic(name)} but one is required (\${colors.red('NOT OK')})`) </pre>		
M	lib/startup/validateConfig.ts (Line 97)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 94: export const checkNecessaryExtraKeysOnSpecialProducts = (products: ProductConfig[]) => { 95: let success = true 96: specialProducts.forEach(({ name, key, extra = {} }) => { >>> 97: // @ts-expect-error FIXME implicit any type issue 98: const matchingProducts = products.filter((product) => product[key]) 99: // @ts-expect-error FIXME implicit any type issue 100: if (extra.key && matchingProducts.length === 1 && !matchingProducts[0][extra.key]) { </pre>		
M	lib/startup/validateConfig.ts (Line 99)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 96: specialProducts.forEach(({ name, key, extra = {} }) => { 97: // @ts-expect-error FIXME implicit any type issue 98: const matchingProducts = products.filter((product) => product[key]) >>> 99: // @ts-expect-error FIXME implicit any type issue 100: if (extra.key && matchingProducts.length === 1 && !matchingProducts[0][extra.key]) { 101: logger.warn(`Product \${colors.italic(matchingProducts[0].name)} configured as \${colors.italic(name)} doesn't contain necessary \${colors.italic(`\${extra.name}`)} (\${colors.red('NOT OK')})`) 102: success = false </pre>		
M	lib/startup/validateConfig.ts (Line 111)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<div> <div>■ Problematic Code:</div> <pre> 108: export const checkUniqueSpecialOnProducts = (products: ProductConfig[]) => { 109: let success = true 110: products.forEach((product) => { >>> 111: // @ts-expect-error FIXME any type issue 112: const appliedSpecials = specialProducts.filter(({ key }) => product[key]) 113: if (appliedSpecials.length > 1) { 114: logger.warn(`Product \${colors.italic(product.name)} is used as \${appliedSpecials.map(({ name }) => `\${colors.italic(name)}`).join(' and ')} but can only be used for one challenge (\${colors.red('NOT OK')})`) </pre> </div>		
<div>M</div>	lib/utils.ts (Line 70)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 67: 68: export const version = (module?: string) => { 69: if (module) { >>> 70: // @ts-expect-error FIXME Ignoring any type issue on purpose 71: return packageJson.dependencies[module] 72: } else { 73: return packageJson.version </pre> </div>		
<div>M</div>	models/relations.ts (Line 44)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 41: name: 'BasketId' 42: } 43: }) >>> 44: // @ts-expect-error FIXME type mismatch 45: makeKeyNonUpdatable(BasketItemModel, 'BasketId') 46: 47: CardModel.belongsTo(UserModel, { </pre> </div>		
<div>M</div>	models/relations.ts (Line 97)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 94: name: 'ProductId' 95: } 96: }) >>> 97: // @ts-expect-error FIXME type mismatch 98: makeKeyNonUpdatable(BasketItemModel, 'ProductId') 99: 100: QuantityModel.belongsTo(ProductModel, { </pre> </div>		
<div>M</div>	routes/showProductReviews.ts (Line 16)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

■ Problematic Code: <pre> 13: import * as utils from '../lib/utils' 14: 15: // Blocking sleep function as in native MongoDB >>> 16: // @ts-expect-error FIXME Type safety broken for global object 17: global.sleep = (time: number) => { 18: // Ensure that users don't accidentally dos their servers for too long 19: if (time > 2000) { </pre>		
M	server.ts (Line 243)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 240: const serveIndexMiddleware = (req: Request, res: Response, next: NextFunction) => { 241: // eslint-disable-next-line @typescript-eslint/unbound-method 242: const origEnd = res.end >>> 243: // @ts-expect-error FIXME assignment broken due to seemingly void return value 244: res.end = function () { 245: if (arguments.length) { 246: const reqPath = req.originalUrl.replace(/\?.*\$/, '') </pre>		
M	server.ts (Line 261)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 258: return 'a href="' + relativePath + '"' 259: }) 260: } >>> 261: // @ts-expect-error FIXME passed argument has wrong type 262: origEnd.apply(this, arguments) 263: } 264: next() </pre>		
M	server.ts (Line 311)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 308: 309: app.use(bodyParser.text({ type: '*' })) 310: app.use(function jsonParser (req: Request, res: Response, next: NextFunction) { >>> 311: // @ts-expect-error FIXME intentionally saving original request in this property 312: req.rawBody = req.body 313: if (req.headers['content-type']?.includes('application/json')) { 314: if (!req.body) { </pre>		
M	test/api/2faSpec.ts (Line 21)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<div> <div>■ Problematic Code:</div> <pre> 18: const jsonHeader = { 'content-type': 'application/json' } 19: 20: async function login ({ email, password, totpSecret }: { email: string, password: string, totpSecret?: string }) { >>> 21: // @ts-expect-error FIXME promise return handling broken 22: const loginRes = await frisby 23: .post(REST_URL + '/user/login', { 24: email,</pre> </div>		
M	test/api/2faSpec.ts (Line 38)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 35: throw new Error('login with totp required but no totp secret provided to login function') 36: } 37: >>> 38: // @ts-expect-error FIXME promise return handling broken 39: const totpRes = await frisby 40: .post(REST_URL + '/2fa/verify', { 41: tmpToken: loginRes.json.data.tmpToken,</pre> </div>		
M	test/api/2faSpec.ts (Line 52)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 49: } 50: 51: async function register ({ email, password, totpSecret }: { email: string, password: string, totpSecret?: string }) { >>> 52: // @ts-expect-error FIXME promise return handling broken 53: const res = await frisby 54: .post(API_URL + '/Users/', { 55: email,</pre> </div>		
M	test/api/2faSpec.ts (Line 67)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 64: if (totpSecret) { 65: const { token } = await login({ email, password }) 66: >>> 67: // @ts-expect-error FIXME promise return handling broken 68: await frisby.post(69: REST_URL + '/2fa/setup', 70: {</pre> </div>		
M	test/api/2faSpec.ts (Line 111)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<div> <div>■ Problematic Code:</div> <pre> 108: 109: const totpToken = otplib.authenticator.generate('IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH') 110: >>> 111: // @ts-expect-error FIXME promise return handling broken 112: await frisby.post(REST_URL + '/2fa/verify', { 113: headers: jsonHeader, 114: body: { </pre> </div>		
<div>M</div>	test/api/2faSpec.ts (Line 139)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 136: 137: const totpToken = otplib.authenticator.generate('THIS9ISNT8THE8RIGHT8SECRET') 138: >>> 139: // @ts-expect-error FIXME promise return handling broken 140: await frisby.post(REST_URL + '/2fa/verify', { 141: headers: jsonHeader, 142: body: { </pre> </div>		
<div>M</div>	test/api/2faSpec.ts (Line 158)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 155: 156: const totpToken = otplib.authenticator.generate('IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH') 157: >>> 158: // @ts-expect-error FIXME promise return handling broken 159: await frisby.post(REST_URL + '/2fa/verify', { 160: headers: jsonHeader, 161: body: { </pre> </div>		
<div>M</div>	test/api/2faSpec.ts (Line 178)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 175: totpSecret: 'IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH' 176: }) 177: >>> 178: // @ts-expect-error FIXME promise return handling broken 179: await frisby.get(180: REST_URL + '/2fa/status', 181: { </pre> </div>		
<div>M</div>	test/api/2faSpec.ts (Line 203)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

■ Problematic Code: <pre> 200: password: '0Y8rMnww\$*9VFYE\$59-!Fg1L6t&6lB' 201: }) 202: >>> 203: // @ts-expect-error FIXME promise return handling broken 204: await frisby.get(205: REST_URL + '/2fa/status', 206: { </pre>		
M	test/api/2faSpec.ts (Line 227)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 224: }) 225: 226: it('GET should return 401 when not logged in', async () => { >>> 227: // @ts-expect-error FIXME promise return handling broken 228: await frisby.get(REST_URL + '/2fa/status') 229: .expect('status', 401) 230: }) </pre>		
M	test/api/2faSpec.ts (Line 243)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 240: await register({ email, password }) 241: const { token } = await login({ email, password }) 242: >>> 243: // @ts-expect-error FIXME promise return handling broken 244: await frisby.post(245: REST_URL + '/2fa/setup', 246: { </pre>		
M	test/api/2faSpec.ts (Line 262)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 259: }) 260: .expect('status', 200) 261: >>> 262: // @ts-expect-error FIXME promise return handling broken 263: await frisby.get(264: REST_URL + '/2fa/status', 265: { </pre>		
M	test/api/2faSpec.ts (Line 289)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

■ Problematic Code: <pre> 286: await register({ email, password }) 287: const { token } = await login({ email, password }) 288: >>> 289: // @ts-expect-error FIXME promise return handling broken 290: await frisby.post(291: REST_URL + '/2fa/setup', 292: { </pre>		
M	test/api/2faSpec.ts (Line 318)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 315: await register({ email, password }) 316: const { token } = await login({ email, password }) 317: >>> 318: // @ts-expect-error FIXME promise return handling broken 319: await frisby.post(320: REST_URL + '/2fa/setup', 321: { </pre>		
M	test/api/2faSpec.ts (Line 347)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 344: await register({ email, password }) 345: const { token } = await login({ email, password }) 346: >>> 347: // @ts-expect-error FIXME promise return handling broken 348: await frisby.post(349: REST_URL + '/2fa/setup', 350: { </pre>		
M	test/api/2faSpec.ts (Line 374)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 371: 372: const { token } = await login({ email, password, totpSecret }) 373: >>> 374: // @ts-expect-error FIXME promise return handling broken 375: await frisby.post(376: REST_URL + '/2fa/setup', 377: { </pre>		
M	test/api/2faSpec.ts (Line 404)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<div> <div>■ Problematic Code:</div> <pre> 401: await register({ email, password, totpSecret }) 402: const { token } = await login({ email, password, totpSecret }) 403: >>> 404: // @ts-expect-error FIXME promise return handling broken 405: await getStatus(token) 406: .expect('status', 200) 407: .expect('json', { </pre> </div>		
<div>M</div>	test/api/2faSpec.ts (Line 411)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 408: setup: true 409: }) 410: >>> 411: // @ts-expect-error FIXME promise return handling broken 412: await frisby.post(413: REST_URL + '/2fa/disable', 414: { </pre> </div>		
<div>M</div>	test/api/2faSpec.ts (Line 425)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 422: } 423:).expect('status', 200) 424: >>> 425: // @ts-expect-error FIXME promise return handling broken 426: await getStatus(token) 427: .expect('status', 200) 428: .expect('json', { </pre> </div>		
<div>M</div>	test/api/2faSpec.ts (Line 441)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 438: await register({ email, password, totpSecret }) 439: const { token } = await login({ email, password, totpSecret }) 440: >>> 441: // @ts-expect-error FIXME promise return handling broken 442: await getStatus(token) 443: .expect('status', 200) 444: .expect('json', { </pre> </div>		
<div>M</div>	test/api/2faSpec.ts (Line 448)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<p>■ Problematic Code:</p> <pre> 445: setup: true 446: }) 447: >>> 448: // @ts-expect-error FIXME promise return handling broken 449: await frisby.post(450: REST_URL + '/2fa/disable', 451: { </pre>		
M	test/api/2faSpec.ts (Line 462)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 459: } 460:).expect('status', 401) 461: >>> 462: // @ts-expect-error FIXME promise return handling broken 463: await getStatus(token) 464: .expect('status', 200) 465: .expect('json', { </pre>		
M	test/api/chatBotSpec.ts (Line 19)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 16: let trainingData: { data: any[] } 17: 18: async function login ({ email, password }: { email: string, password: string }) { >>> 19: // @ts-expect-error FIXME promise return handling broken 20: const loginRes = await frisby 21: .post(REST_URL + '/user/login', { 22: email, </pre>		
M	test/api/dataExportApiSpec.ts (Line 202)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 199: return frisby.post(REST_URL + '/memories', { 200: headers: { 201: Authorization: 'Bearer ' + jsonLogin.authentication.token, >>> 202: // @ts-expect-error FIXME form.getHeaders() is not found 203: 'Content-Type': form.getHeaders()['content-type'] 204: }, 205: body: form </pre>		
M	test/api/dataExportApiSpec.ts (Line 340)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<p>■ Problematic Code:</p> <pre> 337: return frisby.post(REST_URL + '/memories', { 338: headers: { 339: Authorization: 'Bearer ' + jsonLogin.authentication.token, >>> 340: // @ts-expect-error FIXME form.getHeaders() is not found 341: 'Content-Type': form.getHeaders()['content-type'] 342: }, 343: body: form </pre>		
M	test/api/deluxeApiSpec.ts (Line 14)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 11: const API_URL = 'http://localhost:3000/api' 12: 13: async function login ({ email, password }: { email: string, password: string }) { >>> 14: // @ts-expect-error FIXME promise return handling broken 15: const loginRes = await frisby 16: .post(`\${REST_URL}/user/login`, { 17: email, </pre>		
M	test/api/fileUploadSpec.ts (Line 21)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 18: const form = frisby.formData() 19: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 20: >>> 21: // @ts-expect-error FIXME form.getHeaders() is not found 22: return frisby.post(URL + '/file-upload', { headers: { 'Content-Type': form.getHeaders()['content-type'] }, body: form }) 23: .expect('status', 204) 24: }) </pre>		
M	test/api/fileUploadSpec.ts (Line 31)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 28: const form = frisby.formData() 29: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 30: >>> 31: // @ts-expect-error FIXME form.getHeaders() is not found 32: return frisby.post(URL + '/file-upload', { headers: { 'Content-Type': form.getHeaders()['content-type'] }, body: form }) 33: .expect('status', 204) 34: }) </pre>		
M	test/api/fileUploadSpec.ts (Line 41)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<p>■ Problematic Code:</p> <pre> 38: const form = frisby.formData() 39: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 40: >>> 41: // @ts-expect-error FIXME form.getHeaders() is not found 42: return frisby.post(URL + '/file-upload', { headers: { 'Content-Type': form.getHeaders()['content-type'] }, body: form }) 43: .expect('status', 204) 44: }) </pre>		
M	test/api/fileUploadSpec.ts (Line 51)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 48: const form = frisby.formData() 49: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 50: >>> 51: // @ts-expect-error FIXME form.getHeaders() is not found 52: return frisby.post(URL + '/file-upload', { headers: { 'Content-Type': form.getHeaders()['content-type'] }, body: form }) 53: .expect('status', 410) 54: }) </pre>		
M	test/api/fileUploadSpec.ts (Line 61)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 58: const form = frisby.formData() 59: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 60: >>> 61: // @ts-expect-error FIXME form.getHeaders() is not found 62: return frisby.post(URL + '/file-upload', { headers: { 'Content-Type': form.getHeaders()['content-type'] }, body: form }) 63: .expect('status', 410) 64: }) </pre>		
M	test/api/fileUploadSpec.ts (Line 73)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 70: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 71: 72: return frisby.post(URL + '/file-upload', { >>> 73: // @ts-expect-error FIXME form.getHeaders() is not found 74: headers: { 'Content-Type': form.getHeaders()['content-type'] }, 75: body: form 76: }) </pre>		
M	test/api/fileUploadSpec.ts (Line 86)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<p>■ Problematic Code:</p> <pre> 83: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 84: 85: return frisby.post(URL + '/file-upload', { >>> 86: // @ts-expect-error FIXME form.getHeaders() is not found 87: headers: { 'Content-Type': form.getHeaders()['content-type'] }, 88: body: form 89: }) </pre>	
<p>M</p> <p>test/api/fileUploadSpec.ts (Line 99)</p>	<p>typescript:S1134 Take the required action to fix the issue indicated by this comment.</p>
<p>■ Problematic Code:</p> <pre> 96: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 97: 98: return frisby.post(URL + '/file-upload', { >>> 99: // @ts-expect-error FIXME form.getHeaders() is not found 100: headers: { 'Content-Type': form.getHeaders()['content-type'] }, 101: body: form 102: }) </pre>	
<p>M</p> <p>test/api/fileUploadSpec.ts (Line 113)</p>	<p>typescript:S1134 Take the required action to fix the issue indicated by this comment.</p>
<p>■ Problematic Code:</p> <pre> 110: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 111: 112: return frisby.post(URL + '/file-upload', { >>> 113: // @ts-expect-error FIXME form.getHeaders() is not found 114: headers: { 'Content-Type': form.getHeaders()['content-type'] }, 115: body: form 116: }).then((res) => { </pre>	
<p>M</p> <p>test/api/fileUploadSpec.ts (Line 127)</p>	<p>typescript:S1134 Take the required action to fix the issue indicated by this comment.</p>
<p>■ Problematic Code:</p> <pre> 124: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 125: 126: return frisby.post(URL + '/file-upload', { >>> 127: // @ts-expect-error FIXME form.getHeaders() is not found 128: headers: { 'Content-Type': form.getHeaders()['content-type'] }, 129: body: form 130: }).then((res) => { </pre>	
<p>M</p> <p>test/api/fileUploadSpec.ts (Line 157)</p>	<p>typescript:S1134 Take the required action to fix the issue indicated by this comment.</p>

<p>■ Problematic Code:</p> <pre> 154: const form = frisby.formData() 155: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 156: >>> 157: // @ts-expect-error FIXME form.getHeaders() is not found 158: return frisby.post(URL + '/file-upload', { headers: { 'Content-Type': form.getHeaders()['content-type'] }, body: form }) 159: .expect('status', 500) 160: }) </pre>		
M	test/api/fileUploadSpec.ts (Line 167)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 164: const form = frisby.formData() 165: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 166: >>> 167: // @ts-expect-error FIXME form.getHeaders() is not found 168: return frisby.post(URL + '/file-upload', { headers: { 'Content-Type': form.getHeaders()['content-type'] }, body: form }) 169: .expect('status', 204) 170: }) </pre>		
M	test/api/fileUploadSpec.ts (Line 177)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 174: const form = frisby.formData() 175: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 176: >>> 177: // @ts-expect-error FIXME form.getHeaders() is not found 178: return frisby.post(URL + '/file-upload', { headers: { 'Content-Type': form.getHeaders()['content-type'] }, body: form }) 179: .expect('status', 204) 180: }) </pre>		
M	test/api/memoryApiSpec.ts (Line 46)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<p>■ Problematic Code:</p> <pre> 43: 44: return frisby.post(REST_URL + '/memories', { 45: headers: { >>> 46: // @ts-expect-error FIXME form.getHeaders() is not found 47: 'Content-Type': form.getHeaders()['content-type'] 48: }, 49: body: form </pre>		
M	test/api/memoryApiSpec.ts (Line 72)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

■ Problematic Code:	<pre> 69: return frisby.post(REST_URL + '/memories', { 70: headers: { 71: Authorization: 'Bearer ' + jsonLogin.authentication.token, >>> 72: // @ts-expect-error FIXME form.getHeaders() is not found 73: 'Content-Type': form.getHeaders()['content-type'] 74: }, 75: body: form </pre>	
M	test/api/memoryApiSpec.ts (Line 150)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code:	<pre> 147: return frisby.post(REST_URL + '/memories', { 148: headers: { 149: Authorization: 'Bearer ' + jsonLogin.authentication.token, >>> 150: // @ts-expect-error FIXME form.getHeaders() is not found 151: 'Content-Type': form.getHeaders()['content-type'] 152: }, 153: body: form </pre>	
M	test/api/metricsApiSpec.ts (Line 38)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code:	<pre> 35: const form = frisby.formData() 36: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 37: >>> 38: // @ts-expect-error FIXME form.getHeaders() is not found 39: return frisby.post(URL + '/file-upload', { headers: { 'Content-Type': form.getHeaders()['content-type'] }, body: form }) 40: .expect('status', 204) 41: .then(() => { </pre>	
M	test/api/metricsApiSpec.ts (Line 54)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code:	<pre> 51: const form = frisby.formData() 52: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the fileStream type 53: >>> 54: // @ts-expect-error FIXME form.getHeaders() is not found 55: return frisby.post(URL + '/file-upload', { headers: { 'Content-Type': form.getHeaders()['content-type'] }, body: form }) 56: .expect('status', 500) 57: .then(() => { </pre>	
M	test/api/profileImageUploadSpec.ts (Line 33)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

■ Problematic Code: <pre> 30: return frisby.post(`\${URL}/profile/image/file`, { 31: headers: { 32: Cookie: `token=\${jsonLogin.authentication.token}`, >>> 33: // @ts-expect-error FIXME form.getHeaders() is not found 34: 'Content-Type': form.getHeaders()['content-type'] 35: }, 36: body: form, </pre>		
M	test/api/profileImageUploadSpec.ts (Line 60)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 57: return frisby.post(`\${URL}/profile/image/file`, { 58: headers: { 59: Cookie: `token=\${jsonLogin.authentication.token}`, >>> 60: // @ts-expect-error FIXME form.getHeaders() is not found 61: 'Content-Type': form.getHeaders()['content-type'] 62: }, 63: body: form </pre>		
M	test/api/profileImageUploadSpec.ts (Line 78)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 75: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the FileStream type 76: 77: return frisby.post(`\${URL}/profile/image/file`, { >>> 78: // @ts-expect-error FIXME form.getHeaders() is not found 79: headers: { 'Content-Type': form.getHeaders()['content-type'] }, 80: body: form 81: }) </pre>		
M	test/api/profileImageUploadSpec.ts (Line 105)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 102: return frisby.post(`\${URL}/profile/image/url`, { 103: headers: { 104: Cookie: `token=\${jsonLogin.authentication.token}`, >>> 105: // @ts-expect-error FIXME form.getHeaders() is not found 106: 'Content-Type': form.getHeaders()['content-type'] 107: }, 108: body: form, </pre>		
M	test/api/profileImageUploadSpec.ts (Line 131)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

<div> <div>■ Problematic Code:</div> <pre> 128: return frisby.post(`\${URL}/profile/image/url`, { 129: headers: { 130: Cookie: `token=\${jsonLogin.authentication.token}`, >>> 131: // @ts-expect-error FIXME form.getHeaders() is not found 132: 'Content-Type': form.getHeaders()['content-type'] 133: }, 134: body: form, </pre> </div>		
<div> <div>M</div> </div>	test/api/profileImageUploadSpec.ts (Line 146)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 143: form.append('imageUrl', 'cataas.com/cat') 144: 145: return frisby.post(`\${URL}/profile/image/url`, { >>> 146: // @ts-expect-error FIXME form.getHeaders() is not found 147: headers: { 'Content-Type': form.getHeaders()['content-type'] }, 148: body: form 149: }) </pre> </div>		
<div> <div>M</div> </div>	test/api/userProfileSpec.ts (Line 51)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 48: form.append('username', 'Localhorst') 49: 50: return frisby.post(`\${URL}/profile`, { >>> 51: // @ts-expect-error FIXME form.getHeaders() is not found 52: headers: { 'Content-Type': form.getHeaders()['content-type'], Cookie: authHeader.Cookie }, 53: body: form, 54: redirect: 'manual' </pre> </div>		
<div> <div>M</div> </div>	test/api/userProfileSpec.ts (Line 64)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 61: form.append('username', 'Localhorst') 62: 63: return frisby.post(`\${URL}/profile`, { >>> 64: // @ts-expect-error FIXME form.getHeaders() is not found 65: headers: { 'Content-Type': form.getHeaders()['content-type'] }, 66: body: form 67: }) </pre> </div>		
<div> <div>M</div> </div>	test/server/insecuritySpec.ts (Line 6)	typescript:S1134 Take the required action to fix the issue indicated by this comment.

■ Problematic Code: <pre> 3: * SPDX-License-Identifier: MIT 4: */ 5: >>> 6: // @ts-expect-error FIXME no typescript definitions for z85 :(7: import z85 from 'z85' 8: import chai from 'chai' 9: import * as security from '../../lib/insecurity' </pre>		
M	frontend/src/app/score-board/components filter-settings/pipes difficulty-selection-summary.pipe.ts (Line 34)	typescript:S6660 'If' statement should not be the only statement in 'else' block
■ Problematic Code: <pre> 31: if (currentGroup === null) { 32: currentGroup = { start: difficulty, end: difficulty } 33: } else { >>> 34: if (difficulty === currentGroup.end + 1) { 35: currentGroup.end = difficulty 36: } else { 37: difficultyGroups.push(currentGroup) </pre>		
M	frontend/src/app/wallet-web3 wallet-web3.component.ts (Line 72)	typescript:S1854 Remove this useless assignment to variable "txConfirmation".
■ Problematic Code: <pre> 69: value: ethers.utils.parseEther(depositAmount) 70: }) 71: // eslint-disable-next-line @typescript-eslint/no-unused-vars >>> 72: const txConfirmation = await transaction.wait() 73: this.getUserEthBalance() 74: } catch (error) { 75: this.errorMessage = error.message </pre>		
M	frontend/src/app/wallet-web3 wallet-web3.component.ts (Line 90)	typescript:S1854 Remove this useless assignment to variable "txConfirmation".
■ Problematic Code: <pre> 87: ethers.utils.parseEther(withdrawalAmount) 88:) 89: // eslint-disable-next-line @typescript-eslint/no-unused-vars >>> 90: const txConfirmation = await transaction.wait() 91: this.getUserEthBalance() 92: } catch (error) { 93: this.errorMessage = error.message </pre>		
M	routes/checkKeys.ts (Line 21)	typescript:S6660 'If' statement should not be the only statement in 'else' block

<p>■ Problematic Code:</p> <pre> 18: if (req.body.privateKey === privateKey) { 19: res.status(200).json({ success: true, message: 'Challenge successfully solved', status: challenges.nftUnlockChallenge }) 20: } else { >>> 21: if (req.body.privateKey === address) { 22: res.status(401).json({ success: false, message: 'Looks like you entered the public address of my ethereum wallet!', status: challenges.nftUnlockChallenge }) 23: } else if (req.body.privateKey === publicKey) { 24: res.status(401).json({ success: false, message: 'Looks like you entered the public key of my ethereum wallet!', status: challenges.nftUnlockChallenge }) </pre>		
	Dockerfile (Line 19)	docker:S6570 Surround this variable with double quotes; otherwise, it can lead to unexpected behavior.
<p>■ Problematic Code:</p> <pre> 16: RUN rm il8n/*.json true 17: 18: ARG CYCLONEDX_NPM_VERSION=latest >>> 19: RUN npm install -g @cyclonedx/cyclonedx-npm@\$CYCLONEDX_NPM_VERSION 20: RUN npm run sbom 21: 22: FROM gcr.io/distroless/nodejs22-debian12 </pre>		
	data/static/codefixes exposedMetricsChallenge_1.ts (Line 18)	typescript:S4624 Refactor this code to not use nested template literals.
<p>■ Problematic Code:</p> <pre> 15: metricsUpdateLoop = Metrics.updateLoop() 16: 17: server.listen(port, () => { >>> 18: logger.info(colors.cyan(`Server listening on port \${colors.bold(`\${port}`)}`)) 19: startupGauge.set({ task: 'ready' }, (Date.now() - startTime) / 1000) 20: if (process.env.BASE_PATH !== '') { 21: logger.info(colors.cyan(`Server using proxy base path \${colors.bold(`\${process.env.BASE_PATH}`)} for redirects`)) </pre>		
	data/static/codefixes exposedMetricsChallenge_1.ts (Line 21)	typescript:S4624 Refactor this code to not use nested template literals.
<p>■ Problematic Code:</p> <pre> 18: logger.info(colors.cyan(`Server listening on port \${colors.bold(`\${port}`)}`)) 19: startupGauge.set({ task: 'ready' }, (Date.now() - startTime) / 1000) 20: if (process.env.BASE_PATH !== '') { >>> 21: logger.info(colors.cyan(`Server using proxy base path \${colors.bold(`\${process.env.BASE_PATH}`)} for redirects`)) 22: } 23: registerWebsocketEvents(server) 24: if (readyCallback) { </pre>		

<div>M</div>	data/static/codefixes exposedMetricsChallenge_2.ts (Line 15)	typescript:S4624 Refactor this code to not use nested template literals.
<div> <div>■ Problematic Code:</div> <pre> 12: process.env.BASE_PATH = process.env.BASE_PATH ?? config.get('server.basePath') 13: 14: server.listen(port, () => { >>> 15: logger.info(colors.cyan(`Server listening on port \${colors.bold(`\${port}`)}`)) 16: startupGauge.set({ task: 'ready' }, (Date.now() - startTime) / 1000) 17: if (process.env.BASE_PATH !== '') { 18: logger.info(colors.cyan(`Server using proxy base path \${colors.bold(`\${process.env.BASE_PATH}`)} for redirects`)) </pre> </div>		
<div>M</div>	data/static/codefixes exposedMetricsChallenge_2.ts (Line 18)	typescript:S4624 Refactor this code to not use nested template literals.
<div> <div>■ Problematic Code:</div> <pre> 15: logger.info(colors.cyan(`Server listening on port \${colors.bold(`\${port}`)}`)) 16: startupGauge.set({ task: 'ready' }, (Date.now() - startTime) / 1000) 17: if (process.env.BASE_PATH !== '') { >>> 18: logger.info(colors.cyan(`Server using proxy base path \${colors.bold(`\${process.env.BASE_PATH}`)} for redirects`)) 19: } 20: registerWebsocketEvents(server) 21: if (readyCallback) { </pre> </div>		
<div>M</div>	data/static/codefixes exposedMetricsChallenge_3_correc t.ts (Line 18)	typescript:S4624 Refactor this code to not use nested template literals.
<div> <div>■ Problematic Code:</div> <pre> 15: metricsUpdateLoop = Metrics.updateLoop() 16: 17: server.listen(port, () => { >>> 18: logger.info(colors.cyan(`Server listening on port \${colors.bold(`\${port}`)}`)) 19: startupGauge.set({ task: 'ready' }, (Date.now() - startTime) / 1000) 20: if (process.env.BASE_PATH !== '') { 21: logger.info(colors.cyan(`Server using proxy base path \${colors.bold(`\${process.env.BASE_PATH}`)} for redirects`)) </pre> </div>		
<div>M</div>	data/static/codefixes exposedMetricsChallenge_3_correc t.ts (Line 21)	typescript:S4624 Refactor this code to not use nested template literals.

■ Problematic Code:

```

18:     logger.info(colors.cyan(`Server listening on port ${colors.bold(`${port}`)})`)
19:     startupGauge.set({ task: 'ready' }, (Date.now() - startTime) / 1000)
20:     if (process.env.BASE_PATH !== '') {
>>> 21:         logger.info(colors.cyan(`Server using proxy base path
${colors.bold(`${process.env.BASE_PATH}`)} for redirects`))
22:     }
23:     registerWebsocketEvents(server)
24:     if (readyCallback) {

```

M

lib/accuracy.ts
(Line 58)

typescript:S3358

Extract this nested ternary operation into an independent statement.

■ Problematic Code:

```

55:     if (solves[challengeKey][phase]) {
56:         accuracy = 1 / solves[challengeKey].attempts[phase]
57:     }
>>> 58:     logger.info(`Accuracy for '${phase} == 'fix it' ? 'Fix It' : 'Find It'` phase of
coding challenge ${colors.cyan(challengeKey)}: ${accuracy > 0.5 ?
colors.green(accuracy.toString()) : (accuracy > 0.25 ? colors.yellow(accuracy.toString()) :
colors.red(accuracy.toString()))}`)
59:     return accuracy
60: }
61:

```

M

lib/antiCheat.ts
(Line 102)

typescript:S3358

Extract this nested ternary operation into an independent statement.

■ Problematic Code:

```

99:     const minutesSincePreviousSolve = (timestamp.getTime() -
previous().timestamp.getTime()) / 60000
100:     cheatScore += Math.max(0, 1 - (minutesSincePreviousSolve / minutesExpectedToSolve))
101:
>>> 102:     logger.info(`Cheat score for "Find it" phase of ${challenge.key} ==
'scoreBoardChallenge' && config.get('hackingInstructor.isEnabled') ? 'tutorial ' :
''}${colors.cyan(challenge.key)} solved in ${Math.round(minutesSincePreviousSolve)}min (expected
~${minutesExpectedToSolve}min): ${cheatScore < 0.33 ? colors.green(cheatScore.toString()) :
(cheatScore < 0.66 ? colors.yellow(cheatScore.toString()) :
colors.red(cheatScore.toString()))}`)
103:     solves.push({ challenge, phase: 'find it', timestamp, cheatScore })
104:
105:     return cheatScore

```

M

lib/antiCheat.ts
(Line 117)

typescript:S3358

Extract this nested ternary operation into an independent statement.

■ Problematic Code:

```

114:   const minutesSincePreviousSolve = (timestamp.getTime() -
previous().timestamp.getTime()) / 60000
115:   cheatScore += Math.max(0, 1 - (minutesSincePreviousSolve / minutesExpectedToSolve))
116:
>>> 117:   logger.info(`Cheat score for "Fix it" phase of ${colors.cyan(challenge.key)} solved
in ${Math.round(minutesSincePreviousSolve)}min (expected ~${minutesExpectedToSolve}min):
${cheatScore < 0.33 ? colors.green(cheatScore.toString()) : (cheatScore < 0.66 ?
colors.yellow(cheatScore.toString()) : colors.red(cheatScore.toString()))}`)
118:   solves.push({ challenge, phase: 'fix it', timestamp, cheatScore })
119:   return cheatScore
120: }

```

M

lib/insecurity.ts
(Line 61)

typescript:S6397

Replace this character class by the character itself.

■ Problematic Code:

```

58: export const decode = (token: string) => { return jws.decode(token)?.payload }
59:
60: export const sanitizeHtml = (html: string) => sanitizeHtmlLib(html)
>>> 61: export const sanitizeLegacy = (input = '') => input.replace(/<(?:\w+)\W+?[\w]/gi, '')
62: export const sanitizeFilename = (filename: string) => sanitizeFilenameLib(filename)
63: export const sanitizeSecure = (html: string): string => {
64:   const sanitized = sanitizeHtml(html)

```

M

lib/startup/validateConfig.ts
(Line 101)

typescript:S4624

Refactor this code to not use nested template literals.

■ Problematic Code:

```

98:   const matchingProducts = products.filter((product) => product[key])
99:   // @ts-expect-error FIXME implicit any type issue
100:   if (extra.key && matchingProducts.length === 1 && !matchingProducts[0][extra.key])
{
>>> 101:     logger.warn(`Product ${colors.italic(matchingProducts[0].name)} configured as
${colors.italic(name)} doesn't contain necessary ${colors.italic(`${extra.name}`)}
(${colors.red('NOT OK')})`)
102:     success = false
103:   }
104: })

```

M

lib/utlis.ts
(Line 22)

typescript:S1134

Take the required action to fix the issue indicated by this comment.

■ Problematic Code:

```

19: import isWindows from './is-windows'
20: export { default as isDocker } from './is-docker'
21: export { default as isWindows } from './is-windows'
>>> 22: // import isGitpod from 'is-gitpod' // FIXME Roll back to this when
https://github.com/dword-design/is-gitpod/issues/94 is resolve
23: const isGitpod = () => false
24:
25: const months = ['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT',
'NOV', 'DEC']

```

<div>M</div>	server.ts (Line 727)	typescript:S4624 Refactor this code to not use nested template literals.
<div> <div>■ Problematic Code:</div> <pre> 724: metricsUpdateLoop = Metrics.updateLoop() // vuln-code-snippet neutral-line exposedMetricsChallenge 725: 726: server.listen(port, () => { >>> 727: logger.info(colors.cyan(`Server listening on port \${colors.bold(`\${port}`)}`)) 728: startupGauge.set({ task: 'ready' }, (Date.now() - startTime) / 1000) 729: if (process.env.BASE_PATH !== '') { 730: logger.info(colors.cyan(`Server using proxy base path \${colors.bold(`\${process.env.BASE_PATH}`)} for redirects`)) </pre> </div>		
<div>M</div>	server.ts (Line 730)	typescript:S4624 Refactor this code to not use nested template literals.
<div> <div>■ Problematic Code:</div> <pre> 727: logger.info(colors.cyan(`Server listening on port \${colors.bold(`\${port}`)}`)) 728: startupGauge.set({ task: 'ready' }, (Date.now() - startTime) / 1000) 729: if (process.env.BASE_PATH !== '') { >>> 730: logger.info(colors.cyan(`Server using proxy base path \${colors.bold(`\${process.env.BASE_PATH}`)} for redirects`)) 731: } 732: registerWebsocketEvents(server) 733: if (readyCallback) { </pre> </div>		
<div>M</div>	test/api/productReviewApiSpec.ts (Line 42)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 39: .expect('jsonTypes', reviewResponseSchema) 40: }) 41: >>> 42: xit('GET product reviews by alphanumeric non-mongoDB-command product id', () => { // FIXME Turn on when #1960 is resolved 43: return frisby.get(`\${REST_URL}/products/kaboom/reviews`) 44: .expect('status', 400) 45: }) </pre> </div>		
<div>M</div>	test/cypress/e2e/profile.spec.ts (Line 107)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 104: headers: { 105: 'Content-type': 'application/x-www-form-urlencoded', 106: Authorization: `Bearer \${localStorage.getItem('token')}`, >>> 107: Origin: 'http://html5edit.squarefree.com', // FIXME Not allowed by browser due to "unsafe header not permitted" 108: Cookie: `token=\${localStorage.getItem('token')}` // FIXME Not allowed by browser due to "unsafe header not permitted" 109: }, 110: body: formData </pre> </div>		

<div>M</div>	test/cypress/e2e/profile.spec.ts (Line 108)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 105: 'Content-type': 'application/x-www-form-urlencoded', 106: Authorization: `Bearer \${localStorage.getItem('token')}`, 107: Origin: 'http://htmledit.squarefree.com', // FIXME Not allowed by browser due to "unsafe header not permitted" >>> 108: Cookie: `token=\${localStorage.getItem('token')}` // FIXME Not allowed by browser due to "unsafe header not permitted" 109: }, 110: body: formData 111: }) </pre> </div>		
<div>M</div>	test/cypress/e2e/complain.spec.ts (Line 14)	typescript:S6666 Use the spread operator instead of '.apply()'.
<div> <div>■ Problematic Code:</div> <pre> 11: describe('challenge "uploadSize"', () => { 12: it('should be possible to upload files greater 100 KB directly through backend', () => { 13: cy.window().then(async () => { >>> 14: const over100KB = Array.apply(null, new Array(11000)).map(15: // eslint-disable-next-line @typescript-eslint/unbound-method 16: String.prototype.valueOf, 17: '1234567890' </pre> </div>		
<div>M</div>	test/api/metricsApiSpec.ts (Line 14)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 11: const API_URL = 'http://localhost:3000/metrics' 12: 13: describe('/metrics', () => { >>> 14: xit('GET metrics via public API that are available instantaneously', () => { // FIXME Flaky on CI/CD on at least Windows 15: return frisby.get(API_URL) 16: .expect('status', 200) 17: .expect('header', 'content-type', /text\/plain/) </pre> </div>		
<div>M</div>	test/api/metricsApiSpec.ts (Line 33)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 30: .expect('bodyContains', /^http_requests_count{status_code="[0-9]XX",app=".*"} [0-9]*\$/gm) 31: }) 32: >>> 33: xit('GET file upload metrics via public API', () => { // FIXME Flaky on CI/CD on at least Windows 34: const file = path.resolve(__dirname, '../files/validSizeAndTypeForClient.pdf') 35: const form = frisby.formData() 36: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the FileStream type </pre> </div>		

<div>M</div>	test/api/metricsApiSpec.ts (Line 49)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 46: }) 47: }) 48: >>> 49: xit('GET file upload error metrics via public API', () => { // FIXME Flaky on CI/CD on at least Windows 50: const file = path.resolve(__dirname, '../files/invalidSizeForServer.pdf') 51: const form = frisby.formData() 52: form.append('file', fs.createReadStream(file) as unknown as Blob) // casting to blob as the frisby types are wrong and wont accept the FileStream type </pre> </div>		
<div>M</div>	frontend/src/app/last-login-ip last-login-ip.component.spec.ts (Line 60)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 57: expect(console.log).toHaveBeenCalled() 58: }) 59: >>> 60: xit('should set Last-Login IP from JWT as trusted HTML', () => { // FIXME Expected state seems to leak over from previous test case occasionally 61: localStorage.setItem('token', 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjpb7Im xhc3Rmb2dpbkIWIjoimS4yLjMuNCJ9fQ.RAKmdqWNypuOxv3SDjP04xMKvd1CddKvDFYDBfUt3bg') 62: component.ngOnInit() 63: expect(sanitizer.bypassSecurityTrustHtml).toHaveBeenCalled() all>') </pre> </div>		
<div>M</div>	frontend/src/app/last-login-ip last-login-ip.component.spec.ts (Line 66)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 63: expect(sanitizer.bypassSecurityTrustHtml).toHaveBeenCalled() 64: }) 65: >>> 66: xit('should not set Last-Login IP if none is present in JWT', () => { // FIXME Expected state seems to leak over from previous test case occasionally 67: localStorage.setItem('token', 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjpb7fX 0.bVBhvl16Iaer3aUdoOeyR8YZe2S2DfhGAXTGfd9enLw') 68: component.ngOnInit() 69: expect(sanitizer.bypassSecurityTrustHtml).not.toHaveBeenCalled() </pre> </div>		
<div>M</div>	test/cypress/e2e/totpSetup.spec.ts (Line 30)	typescript:S125 Remove this commented out code.

	<p>■ Problematic Code:</p> <pre> 27: cy.get('#initialToken') 28: .should('have.attr', 'data-test-totp-secret') 29: .then((\$val) => { >>> 30: // console.log(\$val); 31: cy.get('#currentPasswordSetup').type('K1f.....') 32: 33: cy.task<string>('GenerateAuthenticator', \$val).then((secret: string) => { </pre>	
<div>M</div>	test/cypress/e2e/profile.spec.ts (Line 63)	typescript:S125 Remove this commented out code.
	<p>■ Problematic Code:</p> <pre> 60: '/solve/challenges/server-side?key=tRy_H4rd3r_n0thIng_iS_Imp0ssibl3' 61:) 62: cy.visit('/') >>> 63: // void browser.driver.sleep(10000); 64: // void browser.waitForAngularEnabled(true); 65: cy.expectChallengeSolved({ challenge: 'SSTi' }) 66: } </pre>	
<div>M</div>	test/cypress/e2e/profile.spec.ts (Line 72)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
	<p>■ Problematic Code:</p> <pre> 69: }) 70: 71: describe('challenge "csrf"', () => { >>> 72: // FIXME Only works on Chrome <80 but Protractor uses latest Chrome version. Test can probably never be turned on again. 73: xit('should be possible to perform a CSRF attack against the user profile page', () => { 74: cy.visit('http://html5edit.squarefree.com') 75: /* The script executed below is equivalent to pasting this string into http://html5edit.squarefree.com: */ </pre>	
<div>M</div>	lib/startup/validateChatBot.ts (Line 36)	typescript:S6660 'If' statement should not be the only statement in 'else' block
	<p>■ Problematic Code:</p> <pre> 33: logger.warn(`Intent \${colors.italic(intent)} is missing in chatbot training data (\${colors.red('NOT OK')})`) 34: success = false 35: } else { >>> 36: if (intentData[0].answers.filter((answer: { action: string, handler: string }) => answer.action === 'function' && answer.handler === handler).length === 0) { 37: logger.warn(`Answer with \${colors.italic('function')} action and handler \${colors.italic(handler)} is missing for intent \${colors.italic(intent)} (\${colors.red('NOT OK')})`) 38: success = false 39: } </pre>	
<div>M</div>	rsn/rsnUtil.ts (Line 129)	typescript:S4043 Move this array "sort" operation to a separate statement or replace it with "toSorted".

■ Problematic Code:

```
126: function checkData (data: CacheData, fileData: CacheData) {
127:   const filesWithDiff = []
128:   for (const key in data) {
>>> 129:     const fileDataValueAdded = fileData[key].added.sort((a, b) => a - b)
130:     const dataValueAdded = data[key].added.sort((a, b) => a - b)
131:     const fileDataValueRemoved = fileData[key].added.sort((a, b) => a - b)
132:     const dataValueAddedRemoved = data[key].added.sort((a, b) => a - b)
```

M

rsn/rsnUtil.ts
(Line 130)

typescript:S4043

Move this array "sort" operation to a separate statement or replace it with "toSorted".

■ Problematic Code:

```
127:   const filesWithDiff = []
128:   for (const key in data) {
129:     const fileDataValueAdded = fileData[key].added.sort((a, b) => a - b)
>>> 130:     const dataValueAdded = data[key].added.sort((a, b) => a - b)
131:     const fileDataValueRemoved = fileData[key].added.sort((a, b) => a - b)
132:     const dataValueAddedRemoved = data[key].added.sort((a, b) => a - b)
133:     if (fileDataValueAdded.length === dataValueAdded.length &&
fileDataValueRemoved.length === dataValueAddedRemoved.length) {
```

M

rsn/rsnUtil.ts
(Line 131)

typescript:S4043

Move this array "sort" operation to a separate statement or replace it with "toSorted".

■ Problematic Code:

```
128:   for (const key in data) {
129:     const fileDataValueAdded = fileData[key].added.sort((a, b) => a - b)
130:     const dataValueAdded = data[key].added.sort((a, b) => a - b)
>>> 131:     const fileDataValueRemoved = fileData[key].added.sort((a, b) => a - b)
132:     const dataValueAddedRemoved = data[key].added.sort((a, b) => a - b)
133:     if (fileDataValueAdded.length === dataValueAdded.length &&
fileDataValueRemoved.length === dataValueAddedRemoved.length) {
134:       if (!dataValueAdded.every((val: number, ind: number) => fileDataValueAdded[ind]
=== val)) {
```

M

rsn/rsnUtil.ts
(Line 132)

typescript:S4043

Move this array "sort" operation to a separate statement or replace it with "toSorted".

■ Problematic Code:

```
129:     const fileDataValueAdded = fileData[key].added.sort((a, b) => a - b)
130:     const dataValueAdded = data[key].added.sort((a, b) => a - b)
131:     const fileDataValueRemoved = fileData[key].added.sort((a, b) => a - b)
>>> 132:     const dataValueAddedRemoved = data[key].added.sort((a, b) => a - b)
133:     if (fileDataValueAdded.length === dataValueAdded.length &&
fileDataValueRemoved.length === dataValueAddedRemoved.length) {
134:       if (!dataValueAdded.every((val: number, ind: number) => fileDataValueAdded[ind]
=== val)) {
135:         console.log(colors.red(key))
```

M

lib/utlis.ts
(Line 41)

typescript:S6557

Use 'String#startsWith' method instead.

	<p>■ Problematic Code:</p> <pre> 38: return startsWith(url, 'http') 39: } 40: >>> 41: export const startsWith = (str: string, prefix: string) => str ? str.indexOf(prefix) === 0 : false 42: 43: export const endsWith = (str?: string, suffix?: string) => (str && suffix) ? str.includes(suffix, str.length - suffix.length) : false 44: </pre>	
<div>M</div>	frontend/src/environments/environm ent.ts (Line 21)	typescript:S125 Remove this commented out code.
	<p>■ Problematic Code:</p> <pre> 18: * import the following file, but please comment it out in production mode 19: * because it will have performance impact when throw error 20: */ >>> 21: // import 'zone.js/plugins/zone-error'; // Included with Angular CLI. 22: </pre>	
<div>M</div>	frontend/src/hacking-instructor challenges/codingChallenges.ts (Line 20)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
	<p>■ Problematic Code:</p> <pre> 17: fixture: 'app-navbar', 18: fixtureAfter: true, 19: unskippable: true, >>> 20: resolved: waitForAngularRouteToBeVisited('score-board') // FIXME The tutorial does not progress automatically. Workaround ^^^^^^^^^^^^^^^^^ instruction above should be removed when fixed. 21: }, 22: { 23: text: </pre>	
<div>M</div>	data/static/codefixes dbSchemaChallenge_1.ts (Line 7)	typescript:S1854 Remove this useless assignment to variable "dataString".
	<p>■ Problematic Code:</p> <pre> 4: criteria = (criteria.length <= 200) ? criteria : criteria.substring(0, 200) 5: models.sequelize.query("SELECT * FROM Products WHERE ((name LIKE '%"+criteria+"%' OR description LIKE '%"+criteria+"%') AND deletedAt IS NULL) ORDER BY name") 6: .then(([products]: any) => { >>> 7: const dataString = JSON.stringify(products) 8: for (let i = 0; i < products.length; i++) { 9: products[i].name = req.__(products[i].name) 10: products[i].description = req.__(products[i].description) </pre>	
<div>M</div>	data/static/codefixes dbSchemaChallenge_2_correct.ts (Line 9)	typescript:S1854 Remove this useless assignment to variable "dataString".

■ Problematic Code:

```

6:         `SELECT * FROM Products WHERE ((name LIKE '%:criteria%' OR description LIKE
':criteria%') AND deletedAt IS NULL) ORDER BY name`,
7:         { replacements: { criteria } }
8:     ).then(([products]: any) => {
>>> 9:         const dataString = JSON.stringify(products)
10:         for (let i = 0; i < products.length; i++) {
11:             products[i].name = req.__(products[i].name)
12:             products[i].description = req.__(products[i].description)

```

M

data/static/codefixes
dbSchemaChallenge_3.ts
(Line 13)

typescript:S1854

Remove this useless assignment to variable "dataString".

■ Problematic Code:

```

10:     }
11:     models.sequelize.query(`SELECT * FROM Products WHERE ((name LIKE '%${criteria}%' OR
description LIKE '%${criteria}%') AND deletedAt IS NULL) ORDER BY name`)
12:     .then(([products]: any) => {
>>> 13:         const dataString = JSON.stringify(products)
14:         for (let i = 0; i < products.length; i++) {
15:             products[i].name = req.__(products[i].name)
16:             products[i].description = req.__(products[i].description)

```

M

data/static/codefixes
unionSqlInjectionChallenge_1.ts
(Line 8)

typescript:S1854

Remove this useless assignment to variable "dataString".

■ Problematic Code:

```

5:     criteria.replace(/"'|;|and|or/i, "")
6:     models.sequelize.query(`SELECT * FROM Products WHERE ((name LIKE '%${criteria}%' OR
description LIKE '%${criteria}%') AND deletedAt IS NULL) ORDER BY name`)
7:     .then(([products]: any) => {
>>> 8:         const dataString = JSON.stringify(products)
9:         for (let i = 0; i < products.length; i++) {
10:             products[i].name = req.__(products[i].name)
11:             products[i].description = req.__(products[i].description)

```

M

data/static/codefixes
unionSqlInjectionChallenge_2_corre
ct.ts
(Line 9)

typescript:S1854

Remove this useless assignment to variable "dataString".

■ Problematic Code:

```

6:         `SELECT * FROM Products WHERE ((name LIKE '%:criteria%' OR description LIKE
':criteria%') AND deletedAt IS NULL) ORDER BY name`,
7:         { replacements: { criteria } }
8:     ).then(([products]: any) => {
>>> 9:         const dataString = JSON.stringify(products)
10:         for (let i = 0; i < products.length; i++) {
11:             products[i].name = req.__(products[i].name)
12:             products[i].description = req.__(products[i].description)

```

<div>M</div>	data/static/codefixes unionSqlInjectionChallenge_3.ts (Line 12)	typescript:S1854 Remove this useless assignment to variable "dataString".
<div> <div>■ Problematic Code:</div> <pre> 9: } 10: models.sequelize.query(`SELECT * FROM Products WHERE ((name LIKE '%\${criteria}%' OR description LIKE '%\${criteria}%') AND deletedAt IS NULL) ORDER BY name`) 11: .then(([products]: any) => { >>> 12: const dataString = JSON.stringify(products) 13: for (let i = 0; i < products.length; i++) { 14: products[i].name = req.__(products[i].name) 15: products[i].description = req.__(products[i].description) </pre> </div>		
<div>M</div>	data/static/codefixes noSqlReviewsChallenge_1.ts (Line 3)	typescript:S1854 Remove this useless assignment to variable "user".
<div> <div>■ Problematic Code:</div> <pre> 1: export function updateProductReviews () { 2: return (req: Request, res: Response, next: NextFunction) => { >>> 3: const user = security.authenticatedUsers.from(req) 4: 5: if (req.body.id['\$ne'] !== undefined) { 6: res.status(400).send() </pre> </div>		
<div>M</div>	data/static/codefixes noSqlReviewsChallenge_2.ts (Line 3)	typescript:S1854 Remove this useless assignment to variable "user".
<div> <div>■ Problematic Code:</div> <pre> 1: export function updateProductReviews () { 2: return (req: Request, res: Response, next: NextFunction) => { >>> 3: const user = security.authenticatedUsers.from(req) 4: db.reviewsCollection.update(5: { _id: req.body.id }, 6: { \$set: { message: req.body.message } } </pre> </div>		
<div>M</div>	data/static/codefixes noSqlReviewsChallenge_3_correct.ts (Line 3)	typescript:S1854 Remove this useless assignment to variable "user".
<div> <div>■ Problematic Code:</div> <pre> 1: export function updateProductReviews () { 2: return (req: Request, res: Response, next: NextFunction) => { >>> 3: const user = security.authenticatedUsers.from(req) 4: 5: if (typeof req.body.id !== 'string') { 6: res.status(400).send() </pre> </div>		
<div>M</div>	data/static/codefixes forgedReviewChallenge_3.ts (Line 3)	typescript:S1854 Remove this useless assignment to variable "user".

■ Problematic Code: <pre> 1: export function updateProductReviews () { 2: return (req: Request, res: Response, next: NextFunction) => { >>> 3: const user = security.authenticatedUsers.from(req) 4: db.reviewsCollection.update(5: { _id: req.body.id }, 6: { \$set: { message: req.body.message } } </pre>		
M	test/api/loginApiSpec.ts (Line 261)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 258: }) 259: }) 260: >>> 261: xit('GET last login IP will be saved as remote IP when True-Client-IP is not present', () => { // FIXME Started to fail regularly on CI under Linux 262: return frisby.post(REST_URL + '/user/login', { 263: headers: jsonHeader, 264: body: { </pre>		
M	test/api/productApiSpec.ts (Line 103)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 100: .expect('json', 'data', { description: 'More...' }) 101: }) 102: >>> 103: xit('PUT update existing product does not filter XSS attacks', () => { // FIXME Started to fail regularly on CI under Linux 104: return frisby.put(API_URL + '/Products/1', { 105: header: jsonHeader, 106: body: { </pre>		
M	test/server/b2bOrderSpec.ts (Line 31)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 28: challenges.rceChallenge = { solved: false, save } as unknown as Challenge 29: }) 30: >>> 31: xit('infinite loop payload does not succeed but solves "rceChallenge"', () => { // FIXME Started failing on Linux regularly 32: req.body.orderLinesData = '(function dos() { while(true); })()' 33: 34: b2bOrder()(req, res, next) </pre>		
M	frontend/src/assets/public/css dataErasure.css (Line 104)	css:S4666 Unexpected duplicate selector "img", first used at line 86

■ Problematic Code: <pre> 101: .home-page { 102: margin-top: 30px; 103: } >>> 104: img { 105: padding-top: 5px; 106: padding-left: 50px; 107: } </pre>		
M	test/server/insecuritySpec.ts (Line 58)	typescript:S4623 Remove this redundant "undefined".
■ Problematic Code: <pre> 55: 56: describe('discountFromCoupon', () => { 57: it('returns undefined when not passing in a coupon code', () => { >>> 58: expect(security.discountFromCoupon(undefined)).toEqual(undefined) 59: }) 60: 61: it('returns undefined for malformed coupon code', () => { </pre>		
M	test/server/insecuritySpec.ts (Line 145)	typescript:S4623 Remove this redundant "undefined".
■ Problematic Code: <pre> 142: describe('sanitizeLegacy', () => { 143: it('returns empty string for undefined input', () => { 144: expect(security.sanitizeLegacy()).toEqual('') >>> 145: expect(security.sanitizeLegacy(undefined)).toEqual('') 146: }) 147: 148: it('returns input unchanged for plain text input', () => { </pre>		
M	frontend/src/app/two-factor-auth-enter two-factor-auth-enter.component.spec.ts (Line 121)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
■ Problematic Code: <pre> 118: expect(sessionStorage.getItem('bid')).toBe('42') 119: }) 120: >>> 121: xit('should notify about user login after 2FA verification', () => { // FIXME Spy call is not registered at all 122: component.verify() 123: 124: expect(userService.isLoggedIn.next).toHaveBeenCalled() </pre>		
M	frontend/src/hacking-instructor/helpers helpers.ts (Line 63)	typescript:S1871 This branch's code block is the same as the block for the branch on line 61.

	<p>■ Problematic Code:</p> <pre> 60: while (true) { 61: if (options.ignoreCase && inputElement.value.toLowerCase() !== value.toLowerCase()) { 62: break >>> 63: } else if (!options.ignoreCase && inputElement.value !== value) { 64: break 65: } 66: await sleep(100) </pre>	
<div>M</div>	frontend/src/hacking-instructor/helpers.ts (Line 46)	typescript:S1871 This branch's code block is the same as the block for the branch on line 44.
	<p>■ Problematic Code:</p> <pre> 43: while (true) { 44: if (options.ignoreCase && inputElement.value.toLowerCase() === value.toLowerCase()) { 45: break >>> 46: } else if (!options.ignoreCase && inputElement.value === value) { 47: break 48: } 49: await sleep(100) </pre>	
<div>M</div>	frontend/src/hacking-instructor/helpers.ts (Line 81)	typescript:S1871 This branch's code block is the same as the block for the branch on line 79.
	<p>■ Problematic Code:</p> <pre> 78: if (inputElement.value !== '') { 79: if (options.ignoreCase && inputElement.value.toLowerCase() !== value.toLowerCase()) { 80: break >>> 81: } else if (!options.ignoreCase && inputElement.value !== value) { 82: break 83: } 84: } </pre>	
<div>M</div>	test/api/internetResourcesSpec.ts (Line 17)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
	<p>■ Problematic Code:</p> <pre> 14: .expect('bodyContains', 'this coupled with Eurogium Edule was sometimes found fatal') 15: }) 16: >>> 17: xit('for 7MS configuration (https://pastebin.com/8SMbWPxc)', () => { // FIXME Test would need to confirm/bypass PasteBin SMART filter to retrieve content 18: return frisby.get('https://pastebin.com/8SMbWPxc') 19: .expect('status', 200) 20: .expect('bodyContains', 'TAYLOR SWIFT') </pre>	

<div>M</div>	test/api/internetResourcesSpec.ts (Line 34)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 31: .expect('bodyContains', 'Note to self: Option (1) and (3) of the above should not be combined.') 32: }) 33: >>> 34: xit('for Mozilla configuration (https://pastebin.com/t8jqEly7)', () => { // FIXME Test would need to confirm/bypass PasteBin SMART filter to retrieve content 35: return frisby.get('https://pastebin.com/t8jqEly7') 36: .expect('status', 200) 37: .expect('bodyContains', 'Fixed a bug that, when this plugin was installed together with both the') </pre> </div>		
<div>M</div>	test/api/userProfileSpec.ts (Line 59)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 56: .expect('status', 302) 57: }) 58: >>> 59: xit('POST update username is forbidden for unauthenticated user', () => { // FIXME runs into "socket hang up" 60: const form = frisby.formData() 61: form.append('username', 'Localhorst') 62: </pre> </div>		
<div>M</div>	lib/startup/validateConfig.ts (Line 114)	typescript:S4624 Refactor this code to not use nested template literals.
<div> <div>■ Problematic Code:</div> <pre> 111: // @ts-expect-error FIXME any type issue 112: const appliedSpecials = specialProducts.filter(({ key }) => product[key]) 113: if (appliedSpecials.length > 1) { >>> 114: logger.warn(`Product \${colors.italic(product.name)} is used as \${appliedSpecials.map(({ name }) => `\${colors.italic(name)}`).join(' and ')} but can only be used for one challenge (\${colors.red('NOT OK')}`) 115: success = false 116: } 117: }) </pre> </div>		
<div>M</div>	lib/startup/validateConfig.ts (Line 162)	typescript:S4624 Refactor this code to not use nested template literals.

<div> <div>■ Problematic Code:</div> <pre> 159: memories.forEach((memory) => { 160: const appliedSpecials = specialMemories.filter(({ keys }) => memory[keys[0]] && memory[keys[1]]) 161: if (appliedSpecials.length > 1) { >>> 162: logger.warn(`Memory \${colors.italic(memory.caption)} is used as \${appliedSpecials.map(({ name }) => `\${colors.italic(name)}`).join(' and ')} but can only be used for one challenge (\${colors.red('NOT OK')})`) 163: success = false 164: } 165: }) </pre> </div>		
<div>M</div>	test/api/profileImageUploadSpec.ts (Line 141)	typescript:S1134 Take the required action to fix the issue indicated by this comment.
<div> <div>■ Problematic Code:</div> <pre> 138: }) 139: }) 140: >>> 141: xit('POST profile image URL forbidden for anonymous user', () => { // FIXME runs into "socket hang up" 142: const form = frisby.formData() 143: form.append('imageUrl', 'cataas.com/cat') 144: </pre> </div>		
<div>M</div>	test/files/decrypt.py (Line 9)	python:PrintStatementUsage Replace print statement by built-in function.
<div> <div>■ Problematic Code:</div> <pre> 6: d = 8948942500927444436822854592177309391966958606588425744549785445648767483962981839 09349419732628796167979706089172836798754993315741611138540888132754881105882471930775825272784 37906504015680623423550067240042466665654232383502922215493623289472138866445818789127946123407 807725702626644091036502372545139713 7: 8: with open('announcement_encrypted.md', 'r') as fl: >>> 9: print "".join([chr(pow(int(f[:-1]), d, N)) for f in fl.readlines()]) 10: </pre> </div>		
<div>M</div>	test/files/decrypt_bruteforce.py (Line 15)	python:PrintStatementUsage Replace print statement by built-in function.
<div> <div>■ Problematic Code:</div> <pre> 12: encrypted_chars[str(c)] = char 13: 14: with open('announcement_encrypted.md', 'r') as fl: >>> 15: print "".join([encrypted_chars[f[:-1]] for f in fl.readlines()]) 16: </pre> </div>		
<div>M</div>	frontend/src/app/deluxe-user deluxe-user.component.ts (Line 61)	typescript:S6557 Use 'String#startsWith' method instead.

■ Problematic Code: <pre> 58: if (config.application.logo) { 59: let logo: string = config.application.logo 60: >>> 61: if (logo.substring(0, 4) === 'http') { 62: logo = decodeURIComponent(logo.substring(logo.lastIndexOf('/') + 1)) 63: } 64: this.logoSrc = `assets/public/images/\${decalParam logo}` </pre>		
M	test/smoke/Dockerfile (Line 1)	docker:S6596 Use a specific version tag for the image.
■ Problematic Code: <pre> >>> 1: FROM alpine 2: 3: RUN apk add curl 4: </pre>		
M	test/smoke/Dockerfile (Line 3)	docker:S6587 Remove cache after installing packages or store it in a cache mount.
■ Problematic Code: <pre> 1: FROM alpine 2: >>> 3: RUN apk add curl 4: 5: COPY smoke-test.sh smoke-test.sh 6: </pre>		
M	frontend/src/app/photo-wall mime-type.validator.ts (Line 38)	typescript:S4165 Review this redundant assignment: "isValid" already holds the assigned value along all execution paths.
■ Problematic Code: <pre> 35: isValid = true 36: break 37: default: >>> 38: isValid = false 39: break 40: } 41: if (isValid) { </pre>		
M	frontend/src/app/photo-wall photo-wall.component.html (Line 14)	Web:S6851 Remove redundant word "image" from the "alt" attribute of your "img" tag.

■ Problematic Code: <pre> 11: <div class="grid"> 12: @for (image of slideshowDataSource; track image) { 13: >>> 14: 15: <div class="overlay"> 16: <div>{{ image.caption }}</div> 17: @if (twitterHandle) { </pre>		
M	frontend/src/app/search-result search-result.component.spec.ts (Line 264)	typescript:S4623 Remove this redundant "undefined".
■ Problematic Code: <pre> 261: it('should not add anything to basket on error retrieving basket', fakeAsync(() => { 262: basketService.find.and.returnValue(throwError('Error')) 263: sessionStorage.setItem('bid', '815') >>> 264: component.addToBasket(undefined) 265: expect(snackBar.open).not.toHaveBeenCalled() 266: })) 267: </pre>		
M	frontend/src/app/search-result search-result.component.html (Line 40)	Web:S6819 Use or instead of the button role to ensure accessibility across all devices.
■ Problematic Code: <pre> 37: } 38: <div class="product" (click)="showDetail(item)" aria-label="Click for more information about the product" 39: matTooltip="Click for more information" matTooltipPosition="above"> >>> 40: 42: <div class="info-box"> 43: <div class="item-name">{{item.name}}</div> </pre>		
M	routes/imageCaptcha.ts (Line 39)	typescript:S3358 Extract this nested ternary operation into an independent statement.
■ Problematic Code: <pre> 36: 37: export const verifyImageCaptcha = () => (req: Request, res: Response, next: NextFunction) => { 38: const user = security.authenticatedUsers.from(req) >>> 39: const UserId = user ? user.data ? user.data.id : undefined : undefined 40: ImageCaptchaModel.findAll({ 41: limit: 1, 42: where: { </pre>		
M	routes/languages.ts (Line 41)	typescript:S3358 Extract this nested ternary operation into an independent statement.





■ Problematic Code: <pre> 38: icons: locale?.icons, 39: shortKey: locale?.shortKey, 40: percentage, >>> 41: gauge: (percentage > 90 ? 'full' : (percentage > 70 ? 'three-quarters' : (percentage > 50 ? 'half' : (percentage > 30 ? 'quarter' : 'empty')))) 42: } 43: if (!(fileName === 'en.json' fileName === 'tlh_AA.json')) { 44: languages.push(lang) </pre>		
M	routes/languages.ts (Line 41)	typescript:S3358 Extract this nested ternary operation into an independent statement.
■ Problematic Code: <pre> 38: icons: locale?.icons, 39: shortKey: locale?.shortKey, 40: percentage, >>> 41: gauge: (percentage > 90 ? 'full' : (percentage > 70 ? 'three-quarters' : (percentage > 50 ? 'half' : (percentage > 30 ? 'quarter' : 'empty')))) 42: } 43: if (!(fileName === 'en.json' fileName === 'tlh_AA.json')) { 44: languages.push(lang) </pre>		
M	routes/languages.ts (Line 41)	typescript:S3358 Extract this nested ternary operation into an independent statement.
■ Problematic Code: <pre> 38: icons: locale?.icons, 39: shortKey: locale?.shortKey, 40: percentage, >>> 41: gauge: (percentage > 90 ? 'full' : (percentage > 70 ? 'three-quarters' : (percentage > 50 ? 'half' : (percentage > 30 ? 'quarter' : 'empty')))) 42: } 43: if (!(fileName === 'en.json' fileName === 'tlh_AA.json')) { 44: languages.push(lang) </pre>		
M	lib/startup/validatePreconditions.ts (Line 105)	typescript:S6660 'If' statement should not be the only statement in 'else' block
■ Problematic Code: <pre> 102: if (error) { 103: reject(error) 104: } else { >>> 105: if (status === 'open') { 106: logger.warn(`Port \${colors.bold(port.toString())} is in use (\${colors.red('NOT OK')})`) 107: resolve(false) 108: } else { </pre>		
M	frontend/src/app/two-factor-auth-enter two-factor-auth-enter.component.ts (Line 61)	typescript:S125 Remove this commented out code.

	<p>■ Problematic Code:</p> <pre> 58: this.cookieService.put('token', authentication.token, { expires }) 59: sessionStorage.setItem('bid', authentication.bid?.toString()) 60: /* Use userService to notify if user has logged in */ >>> 61: /* this.userService.isLoggedIn = true; */ 62: this.userService.isLoggedIn.next(true) 63: this.ngZone.run(async () => await this.router.navigate(['/search'])) 64: }, </pre>	
<div>M</div>	frontend/src/app/user-details user-details.component.html (Line 18)	Web:AvoidCommentedOutCodeCheck Remove this commented out code.
	<p>■ Problematic Code:</p> <pre> 15: <strong translate="LABEL_EMAIL"> 16: <p>{{user?.email}}</p> 17: </div> >>> 18: <!--<div> 19: <strong translate="LABEL_PASSWORD"> 20: <p>{{user?.password}}</p> 21: </div>--> </pre>	
<div>M</div>	routes/order.ts (Line 40)	typescript:S3358 Extract this nested ternary operation into an independent statement.
	<p>■ Problematic Code:</p> <pre> 37: .then(async (basket: BasketModel null) => { 38: if (basket != null) { 39: const customer = security.authenticatedUsers.from(req) >>> 40: const email = customer ? customer.data ? customer.data.email : '' : '' 41: const orderId = security.hash(email).slice(0, 4) + '-' + utils.randomHexString(16) 42: const pdfFile = `order_\${orderId}.pdf` 43: const doc = new PDFDocument() </pre>	
<div>M</div>	frontend/src/app/navbar navbar.component.ts (Line 132)	typescript:S6557 Use 'String#startsWith' method instead.
	<p>■ Problematic Code:</p> <pre> 129: if (config?.application?.logo) { 130: let logo: string = config.application.logo 131: >>> 132: if (logo.substring(0, 4) === 'http') { 133: logo = decodeURIComponent(logo.substring(logo.lastIndexOf('/') + 1)) 134: } 135: this.logoSrc = 'assets/public/images/' + logo </pre>	
<div>M</div>	server.ts (Line 203)	typescript:S6397 Replace this character class by the character itself.

■ Problematic Code: <pre> 200: 201: /* Remove duplicate slashes from URL which allowed bypassing subsequent filters */ 202: app.use((req: Request, res: Response, next: NextFunction) => { >>> 203: req.url = req.url.replace(/[/]+/g, '/') 204: next() 205: }) 206: </pre>		
L	server.ts (Line 551)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
■ Problematic Code: <pre> 548: // translate hints on-the-fly 549: if (name === 'Hint') { 550: resource.list.fetch.after((req: Request, res: Response, context: { instance: string any[], continue: any }) => { >>> 551: for (let i = 0; i < context.instance.length; i++) { 552: context.instance[i].text = req.__(context.instance[i].text) 553: } 554: return context.continue </pre>		
L	frontend/src/app/change-password change-password.component.ts (Line 92)	typescript:S6594 Use the "RegExp.exec()" method instead.
■ Problematic Code: <pre> 89: 90: changePassword () { 91: if (>>> 92: localStorage.getItem('email')?.match(/support@.*/) && 93: !this.newPasswordControl.value.match(94: /(?!.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@\$!%*?&])[A-Za-z\d@\$!%*?&]{12,30}/ 95:) </pre>		
L	frontend/src/app/search-result search-result.component.ts (Line 199)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
■ Problematic Code: <pre> 196: next: (basket) => { 197: const productsInBasket: any = basket.Products 198: let found = false >>> 199: for (let i = 0; i < productsInBasket.length; i++) { 200: if (productsInBasket[i].id === id) { 201: found = true 202: this.basketService.get(productsInBasket[i].BasketItem.id).subscribe({ </pre>		
L	routes/deluxe.ts (Line 51)	typescript:S2486 Handle this exception or don't catch it at all.



■ Problematic Code: <pre> 48: const updatedToken = security.authorize(userWithStatus) 49: security.authenticatedUsers.put(updatedToken, userWithStatus) 50: res.status(200).json({ status: 'success', data: { confirmation: 'Congratulations! You are now a deluxe member!', token: updatedToken } }) >>> 51: } catch (error) { 52: res.status(400).json({ status: 'error', error: 'Something went wrong. Please try again!' }) 53: } 54: } catch (err: unknown) { </pre>		
L	test/server/webhookSpec.ts (Line 23)	typescript:S2486 Handle this exception or don't catch it at all.
■ Problematic Code: <pre> 20: it('ignores errors where no webhook URL is provided via environment variable', async () => { 21: try { 22: await webhook.notify(challenge) >>> 23: } catch (error) { 24: chai.assert.fail('webhook.notify should not throw an error when no webhook URL is provided') 25: } 26: }) </pre>		
L	routes/likeProductReviews.ts (Line 58)	typescript:S2486 Handle this exception or don't catch it at all.
■ Problematic Code: <pre> 55: } catch (err) { 56: res.status(500).json(err) 57: } >>> 58: } catch (err) { 59: res.status(400).json({ error: 'Wrong Params' }) 60: } 61: } </pre>		
L	frontend/src/app/web3-sandbox web3-sandbox.component.ts (Line 312)	typescript:S2486 Handle this exception or don't catch it at all.
■ Problematic Code: <pre> 309: } 310: console.log('session', this.session) 311: this.changeDetectorRef.detectChanges() >>> 312: } catch (err) { 313: console.log('An error occurred') 314: } 315: } </pre>		
L	frontend/src/app/password-strength password-strength.component.ts (Line 59)	typescript:S6353 Use concise character class syntax '\d' instead of '[0-9]'.

■ Problematic Code: <pre> 56: } 57: 58: get containAtLeastOneDigit (): boolean { >>> 59: return /^(?=.*?[0-9])/\.test(this.password) 60: } 61: 62: get containAtLeastOneSpecialChar (): boolean { </pre>		
	frontend/src/main.ts (Line 9)	typescript:S1128 Remove this unused import of 'AppModule'.
■ Problematic Code: <pre> 6: import { enableProdMode, importProvidersFrom } from '@angular/core' 7: import { platformBrowserDynamic } from '@angular/platform-browser-dynamic' 8: >>> 9: import { AppModule, HttpLoaderFactory } from './app/app.module' 10: import { environment } from './environments/environment' 11: import { AppComponent } from './app/app.component'; 12: import { MatAutocompleteModule } from '@angular/material/autocomplete'; </pre>		
	frontend/src/main.ts (Line 51)	typescript:S1128 Remove this unused import of 'TranslateHttpLoader'.
■ Problematic Code: <pre> 48: import { provideAnimations } from '@angular/platform-browser/animations'; 49: import { ReactiveFormsModule, FormsModule } from '@angular/forms'; 50: >>> 51: import { TranslateHttpLoader } from '@ngx-translate/http-loader'; 52: import { TranslateModule, TranslateLoader } from '@ngx-translate/core'; 53: import { Routing } from './app/app.routing'; 54: import { BrowserModule, bootstrapApplication } from '@angular/platform-browser'; </pre>		
	lib/startup/validateDependenciesBas ic.ts (Line 12)	typescript:S2486 Handle this exception or don't catch it at all.
■ Problematic Code: <pre> 9: try { 10: // @ts-expect-error FIXME due to non-existing type definitions for check-dependencies 11: await import('check-dependencies') >>> 12: } catch (err) { 13: console.error('Please run "npm install" before starting the application!') 14: process.exit(1) 15: } </pre>		
	frontend/src/hacking-instructor/index .ts (Line 201)	typescript:S6571 'unknown' overrides all other types in this union type.

■ Problematic Code: <pre> 198: } 199: 200: // eslint-disable-next-line @typescript-eslint/no-invalid-void-type >>> 201: const continueConditions: Array<Promise<void unknown>> = [202: hint.resolved() 203:] 204: </pre>		
	routes/b2bOrder.ts (Line 26)	typescript:S6594 Use the "RegExp.exec()" method instead.
■ Problematic Code: <pre> 23: vm.runInContext('safeEval(orderLinesData)', sandbox, { timeout: 2000 }) 24: res.json({ cid: body.cid, orderNo: uniqueOrderNumber(), paymentDue: dateTwoWeeksFromNow() }) 25: } catch (err) { >>> 26: if (utils.getErrorMessage(err).match(/Script execution timed out.*/) != null) { 27: challengeUtils.solveIf(challenges.rceOccupyChallenge, () => { return true }) 28: res.status(503) 29: next(new Error('Sorry, we are temporarily not available! Please try again later.')) </pre>		
	test/cypress/support/commands.ts (Line 36)	typescript:S6594 Use the "RegExp.exec()" method instead.
■ Problematic Code: <pre> 33: 'login', 34: (context: { email: string, password: string, totpSecret?: string }) => { 35: cy.visit('/#/login') >>> 36: if (context.email.match(/\S+@\S+\.\S+/) != null) { 37: cy.get('#email').type(context.email) 38: } else { 39: cy.task<string>('GetFromConfig', 'application.domain').then(</pre>		
	lib/is-docker.ts (Line 27)	typescript:S6606 Prefer using nullish coalescing operator (`??=`) instead of an assignment expression, as it is simpler to read.
■ Problematic Code: <pre> 24: 25: export default function isDocker () { 26: // TODO: Use `??=` when targeting Node.js 16. >>> 27: if (isDockerCached === undefined) { 28: isDockerCached = hasDockerEnv() hasDockerCGroup() 29: } 30: return isDockerCached </pre>		
	routes/chatbot.ts (Line 111)	typescript:S2486 Handle this exception or don't catch it at all.

<p>■ Problematic Code:</p> <pre> 108: } else { 109: res.status(200).json(response) 110: } >>> 111: } catch (err) { 112: try { 113: await bot.respond(testCommand, `\${user.id}`) 114: res.status(200).json({ </pre>		
L	test/server/blueprintSpec.ts (Line 58)	typescript:S2486 Handle this exception or don't catch it at all.
<p>■ Problematic Code:</p> <pre> 55: for (const property of product.exifForBlueprintChallenge) { 56: expect(properties).to.include(property) 57: } >>> 58: } catch (error) { 59: expect.fail(`Could not read EXIF data from \${pathToImage}`) 60: } 61: } </pre>		
L	lib/codingChallenges.ts (Line 37)	typescript:S2486 Handle this exception or don't catch it at all.
<p>■ Problematic Code:</p> <pre> 34:) { 35: matches.push({ path: currPath, content: code }) 36: } >>> 37: } catch (e) { 38: logger.warn(`File \${currPath} could not be read. it might have been moved or deleted. If coding challenges are contained in the file, they will not be available.`) 39: } 40: } </pre>		
L	lib/codingChallenges.ts (Line 59)	typescript:S6594 Use the "RegExp.exec()" method instead.
<p>■ Problematic Code:</p> <pre> 56: } 57: 58: function getCodingChallengeFromFileContent (source: string, challengeKey: string) { >>> 59: const snippets = source.match(`[/#]{0,2} vuln-code-snippet start.*\${challengeKey}([^\s]*)vuln-code-snippet end.*\${challengeKey}`) 60: if (snippets == null) { 61: throw new BrokenBoundary('Broken code snippet boundaries for: ' + challengeKey) 62: } </pre>		
L	routes/chatbot.ts (Line 71)	typescript:S2486 Handle this exception or don't catch it at all.

	<p>■ Problematic Code:</p> <pre> 68: action: 'response', 69: body: bot.greet(`\${user.id}`) 70: }) >>> 71: } catch (err) { 72: next(new Error('Blocked illegal activity by ' + req.socket.remoteAddress)) 73: } 74: return </pre>	
<div>L</div>	<p>routes/chatbot.ts (Line 81)</p>	<p>typescript:S2486 Handle this exception or don't catch it at all.</p>
	<p>■ Problematic Code:</p> <pre> 78: bot.addUser(`\${user.id}`, username) 79: try { 80: bot.addUser(`\${user.id}`, username) >>> 81: } catch (err) { 82: next(new Error('Blocked illegal activity by ' + req.socket.remoteAddress)) 83: return 84: } </pre>	
<div>L</div>	<p>routes/chatbot.ts (Line 199)</p>	<p>typescript:S2486 Handle this exception or don't catch it at all.</p>
	<p>■ Problematic Code:</p> <pre> 196: status: bot.training.state, 197: body: bot.training.state ? bot.greet(`\${user.id}`) : `\${config.get<string>('application.chatBot.name')}` isn't ready at the moment, please wait while I set things up` 198: }) >>> 199: } catch (err) { 200: next(new Error('Blocked illegal activity by ' + req.socket.remoteAddress)) 201: } 202: } </pre>	
<div>L</div>	<p>routes/search.ts (Line 50)</p>	<p>typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.</p>
	<p>■ Problematic Code:</p> <pre> 47: void models.sequelize.query('SELECT sql FROM sqlite_master').then(([data]: any) => { 48: const tableDefinitions = utils.queryResultToJson(data) 49: if (tableDefinitions.data?.length) { >>> 50: for (let i = 0; i < tableDefinitions.data.length; i++) { 51: if (tableDefinitions.data[i].sql) { 52: solved = solved && utils.containsOrEscaped(dataString, tableDefinitions.data[i].sql) 53: if (!solved) { </pre>	
<div>L</div>	<p>lib/insecurity.ts (Line 120)</p>	<p>typescript:S6353 Use concise character class syntax `d` instead of `[0-9]`.</p>

■ Problematic Code: <pre> 117: } 118: 119: function hasValidFormat (coupon: string) { >>> 120: return coupon.match(/(JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC)[0-9]{2}-[0-9]{2}/) 121: } 122: 123: // vuln-code-snippet start redirectCryptoCurrencyChallenge redirectChallenge </pre>		
	lib/insecurity.ts (Line 120)	typescript:S6353 Use concise character class syntax '\d' instead of '[0-9]'.
■ Problematic Code: <pre> 117: } 118: 119: function hasValidFormat (coupon: string) { >>> 120: return coupon.match(/(JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC)[0-9]{2}-[0-9]{2}/) 121: } 122: 123: // vuln-code-snippet start redirectCryptoCurrencyChallenge redirectChallenge </pre>		
	lib/insecurity.ts (Line 120)	typescript:S6594 Use the "RegExp.exec()" method instead.
■ Problematic Code: <pre> 117: } 118: 119: function hasValidFormat (coupon: string) { >>> 120: return coupon.match(/(JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC)[0-9]{2}-[0-9]{2}/) 121: } 122: 123: // vuln-code-snippet start redirectCryptoCurrencyChallenge redirectChallenge </pre>		
	routes/basket.ts (Line 26)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
■ Problematic Code: <pre> 23: return user && id && id !== 'undefined' && id !== 'null' && id !== 'NaN' && user.bid && user?.bid !== parseInt(id, 10) // eslint-disable-line eqeqeq 24: }) 25: if ((basket?.Products) !== null) && basket.Products.length > 0) { >>> 26: for (let i = 0; i < basket.Products.length; i++) { 27: basket.Products[i].name = req.__(basket.Products[i].name) 28: } 29: } </pre>		
	routes/payment.ts (Line 9)	typescript:S101 Rename interface "displayCard" to match the regular expression ^[A-Z][a-zA-Z0-9]*\$.

■ Problematic Code: <pre> 6: import { type Request, type Response, type NextFunction } from 'express' 7: import { CardModel } from '../models/card' 8: >>> 9: interface displayCard { 10: UserId: number 11: id: number 12: fullName: string </pre>		
L	server.ts (Line 218)	typescript:S1874 The signature '(from: number, length?: number undefined): string' of 'locale.key.substr' is deprecated.
■ Problematic Code: <pre> 215: contact: config.get('application.securityTxt.contact'), 216: encryption: config.get('application.securityTxt.encryption'), 217: acknowledgements: config.get('application.securityTxt.acknowledgements'), >>> 218: 'Preferred-Languages': [...new Set(locales.map((locale: { key: string }) => locale.key.substr(0, 2)))]].join(', '), 219: hiring: config.get('application.securityTxt.hiring'), 220: csaf: config.get<string>('server.baseUrl') + config.get<string>('application.securityTxt.csaf'), 221: expires: securityTxtExpiration.toUTCString() </pre>		
L	frontend/src/app/search-result search-result.component.ts (Line 134)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
■ Problematic Code: <pre> 131: } 132: 133: trustProductDescription (tableData: any[]) { // vuln-code-snippet neutral-line restfulXssChallenge >>> 134: for (let i = 0; i < tableData.length; i++) { // vuln-code-snippet neutral-line restfulXssChallenge 135: tableData[i].description = this.sanitizer.bypassSecurityTrustHtml(tableData[i].description) // vuln-code-snippet vuln-line restfulXssChallenge 136: } // vuln-code-snippet neutral-line restfulXssChallenge 137: } // vuln-code-snippet neutral-line restfulXssChallenge </pre>		
L	data/static/codefixes dbSchemaChallenge_1.ts (Line 8)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
■ Problematic Code: <pre> 5: models.sequelize.query("SELECT * FROM Products WHERE ((name LIKE '%"+criteria+"%' OR description LIKE '%"+criteria+"%') AND deletedAt IS NULL) ORDER BY name") 6: .then(([products]: any) => { 7: const dataString = JSON.stringify(products) >>> 8: for (let i = 0; i < products.length; i++) { 9: products[i].name = req.__(products[i].name) 10: products[i].description = req.__(products[i].description) 11: } </pre>		

<div>L</div>	data/static/codefixes dbSchemaChallenge_2_correct.ts (Line 10)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 7: { replacements: { criteria } } 8:).then(([products]: any) => { 9: const dataString = JSON.stringify(products) >>> 10: for (let i = 0; i < products.length; i++) { 11: products[i].name = req.__(products[i].name) 12: products[i].description = req.__(products[i].description) 13: } </pre> </div>		
<div>L</div>	data/static/codefixes dbSchemaChallenge_3.ts (Line 14)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 11: models.sequelize.query(`SELECT * FROM Products WHERE ((name LIKE '%\${criteria}%' OR description LIKE '%\${criteria}%') AND deletedAt IS NULL) ORDER BY name`) 12: .then(([products]: any) => { 13: const dataString = JSON.stringify(products) >>> 14: for (let i = 0; i < products.length; i++) { 15: products[i].name = req.__(products[i].name) 16: products[i].description = req.__(products[i].description) 17: } </pre> </div>		
<div>L</div>	data/static/codefixes unionSqlInjectionChallenge_1.ts (Line 9)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 6: models.sequelize.query(`SELECT * FROM Products WHERE ((name LIKE '%\${criteria}%' OR description LIKE '%\${criteria}%') AND deletedAt IS NULL) ORDER BY name`) 7: .then(([products]: any) => { 8: const dataString = JSON.stringify(products) >>> 9: for (let i = 0; i < products.length; i++) { 10: products[i].name = req.__(products[i].name) 11: products[i].description = req.__(products[i].description) 12: } </pre> </div>		
<div>L</div>	data/static/codefixes unionSqlInjectionChallenge_2_correct.ts (Line 10)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 7: { replacements: { criteria } } 8:).then(([products]: any) => { 9: const dataString = JSON.stringify(products) >>> 10: for (let i = 0; i < products.length; i++) { 11: products[i].name = req.__(products[i].name) 12: products[i].description = req.__(products[i].description) 13: } </pre> </div>		

<div>L</div>	data/static/codefixes unionSqlInjectionChallenge_3.ts (Line 13)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 10: models.sequelize.query(`SELECT * FROM Products WHERE ((name LIKE '%\${criteria}%' OR description LIKE '%\${criteria}%') AND deletedAt IS NULL) ORDER BY name`) 11: .then(([products]: any) => { 12: const dataString = JSON.stringify(products) >>> 13: for (let i = 0; i < products.length; i++) { 14: products[i].name = req.__(products[i].name) 15: products[i].description = req.__(products[i].description) 16: } </pre> </div>		
<div>L</div>	routes/vulnCodeFixes.ts (Line 11)	typescript:S101 Rename interface "codeFix" to match the regular expression <code>^[A-Z][a-zA-Z0-9]*\$</code> .
<div> <div>■ Problematic Code:</div> <pre> 8: 9: const FixesDir = 'data/static/codefixes' 10: >>> 11: interface codeFix { 12: fixes: string[] 13: correct: number 14: } </pre> </div>		
<div>L</div>	frontend/src/app/Services code-fixes.service.ts (Line 7)	typescript:S101 Rename interface "result" to match the regular expression <code>^[A-Z][a-zA-Z0-9]*\$</code> .
<div> <div>■ Problematic Code:</div> <pre> 4: import { catchError, map } from 'rxjs/operators' 5: import { type Observable } from 'rxjs' 6: >>> 7: export interface result { 8: verdict: boolean 9: } 10: </pre> </div>		
<div>L</div>	frontend/src/app/Services vuln-lines.service.ts (Line 6)	typescript:S101 Rename interface "result" to match the regular expression <code>^[A-Z][a-zA-Z0-9]*\$</code> .
<div> <div>■ Problematic Code:</div> <pre> 3: import { HttpClient } from '@angular/common/http' 4: import { catchError, map } from 'rxjs/operators' 5: >>> 6: export interface result { 7: verdict: boolean 8: hint: string 9: } </pre> </div>		

<div>L</div>	routes/vulnCodeSnippet.ts (Line 26)	typescript:S1301 Replace this "switch" statement by "if" statements to increase readability.
<div> <div>■ Problematic Code:</div> <pre> 23: } 24: 25: const setStatusCode = (error: any) => { >>> 26: switch (error.name) { 27: case 'BrokenBoundary': 28: return 422 29: default: </pre> </div>		
<div>L</div>	routes/verify.ts (Line 133)	typescript:S6594 Use the "RegExp.exec()" method instead.
<div> <div>■ Problematic Code:</div> <pre> 130: } 131: 132: function hasEmail (token: { data: { email: string } }, email: string RegExp) { >>> 133: return token?.data?.email?.match(email) 134: } 135: 136: export const databaseRelatedChallenges = () => (req: Request, res: Response, next: NextFunction) => { </pre> </div>		
<div>L</div>	lib/startup/validatePreconditions.ts (Line 118)	typescript:S1874 The signature '(from: number, length?: number undefined): string' of 'pathRelativeToProjectRoot.substr' is deprecated.
<div> <div>■ Problematic Code:</div> <pre> 115: } 116: 117: export const checkIfRequiredFileExists = async (pathRelativeToProjectRoot: string) => { >>> 118: const fileName = pathRelativeToProjectRoot.substr(pathRelativeToProjectRoot.lastIndexOf('/') + 1) 119: 120: return await access(path.resolve(pathRelativeToProjectRoot)).then(() => { 121: logger.info(`Required file \${colors.bold(fileName)} is present (\${colors.green('OK')})`) </pre> </div>		
<div>L</div>	frontend/src/app/data-export data-export.component.ts (Line 69)	typescript:S1874 '(...text: string[]): void' is deprecated.
<div> <div>■ Problematic Code:</div> <pre> 66: this.error = null 67: this.confirmation = data.confirmation 68: this.userData = data.userData >>> 69: window.open('', '_blank', 'width=500').document.write(this.userData) 70: this.lastSuccessfulTry = new Date() 71: localStorage.setItem('lstdtxpirt', JSON.stringify(this.lastSuccessfulTry)) 72: this.ngOnInit() </pre> </div>		

<div>L</div>	routes/basketItems.ts (Line 26)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 23: const basketIds = [] 24: const quantities = [] 25: >>> 26: for (let i = 0; i < result.length; i++) { 27: if (result[i].key === 'ProductId') { 28: productIds.push(result[i].value) 29: } else if (result[i].key === 'BasketId') { </pre> </div>		
<div>L</div>	routes/showProductReviews.ts (Line 40)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 37: const t1 = new Date().getTime() 38: challengeUtils.solveIf(challenges.noSqlCommandChallenge, () => { return (t1 - t0) > 2000 }) 39: const user = security.authenticatedUsers.from(req) >>> 40: for (let i = 0; i < reviews.length; i++) { 41: if (user === undefined reviews[i].likedBy.includes(user.data.email)) { 42: reviews[i].liked = true 43: } </pre> </div>		
<div>L</div>	frontend/src/app/oauth oauth.component.ts (Line 69)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 66: const hash = this.route.snapshot.data.params.substr(1) 67: const splitted = hash.split('&') 68: const params: any = {} >>> 69: for (let i = 0; i < splitted.length; i++) { 70: const param: string = splitted[i].split('=') 71: const key: string = param[0] 72: params[key] = param[1] </pre> </div>		
<div>L</div>	routes/resetPassword.ts (Line 66)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 63: challengeUtils.solveIf(challenges.geoStalkingMetaChallenge, () => { 64: const securityAnswer = (() => { 65: const memories = config.get<MemoryConfig[]>('memories') >>> 66: for (let i = 0; i < memories.length; i++) { 67: if (memories[i].geoStalkingMetaSecurityAnswer) { 68: return memories[i].geoStalkingMetaSecurityAnswer 69: } </pre> </div>		

<div>L</div>	routes/resetPassword.ts (Line 77)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 74: challengeUtils.solveIf(challenges.geoStalkingVisualChallenge, () => { 75: const securityAnswer = (() => { 76: const memories = config.get<MemoryConfig[]>('memories') >>> 77: for (let i = 0; i < memories.length; i++) { 78: if (memories[i].geoStalkingVisualSecurityAnswer) { 79: return memories[i].geoStalkingVisualSecurityAnswer 80: } </pre> </div>		
<div>L</div>	test/api/metricsApiSpec.ts (Line 18)	typescript:S6353 Use concise character class syntax `\\d` instead of `[0-9]`.
<div> <div>■ Problematic Code:</div> <pre> 15: return frisby.get(API_URL) 16: .expect('status', 200) 17: .expect('header', 'content-type', /text\\/plain/) >>> 18: .expect('bodyContains', /^.*_version_info{version="[0-9]+.[0-9]+.[0-9]+(-SNAPSHO T)?",major="[0-9]+",minor="[0-9]+",patch="[0-9]+",app=".*"} 1\$/gm) 19: .expect('bodyContains', /^.*_challenges_solved{difficulty="[1-6]",category=".*",app=".*"} [0-9]*\$/gm) 20: .expect('bodyContains', /^.*_challenges_total{difficulty="[1-6]",category=".*",app=".*"} [0-9]*\$/gm) 21: .expect('bodyContains', /^.*_cheat_score{app=".*"} [0-9.]*\$/gm) </pre> </div>		
<div>L</div>	test/api/metricsApiSpec.ts (Line 18)	typescript:S6353 Use concise character class syntax `\\d` instead of `[0-9]`.
<div> <div>■ Problematic Code:</div> <pre> 15: return frisby.get(API_URL) 16: .expect('status', 200) 17: .expect('header', 'content-type', /text\\/plain/) >>> 18: .expect('bodyContains', /^.*_version_info{version="[0-9]+.[0-9]+.[0-9]+(-SNAPSHO T)?",major="[0-9]+",minor="[0-9]+",patch="[0-9]+",app=".*"} 1\$/gm) 19: .expect('bodyContains', /^.*_challenges_solved{difficulty="[1-6]",category=".*",app=".*"} [0-9]*\$/gm) 20: .expect('bodyContains', /^.*_challenges_total{difficulty="[1-6]",category=".*",app=".*"} [0-9]*\$/gm) 21: .expect('bodyContains', /^.*_cheat_score{app=".*"} [0-9.]*\$/gm) </pre> </div>		
<div>L</div>	test/api/metricsApiSpec.ts (Line 18)	typescript:S6353 Use concise character class syntax `\\d` instead of `[0-9]`.

■ Problematic Code:

```

15:     return frisby.get(API_URL)
16:     .expect('status', 200)
17:     .expect('header', 'content-type', /text\/plain/)
>>> 18:     .expect('bodyContains', /^.*_version_info{version="[0-9]+\.[0-9]+\.[0-9]+(-SNAPSHO
T)?",major="[0-9]+",minor="[0-9]+",patch="[0-9]+",app=".*"} 1$/gm)
19:     .expect('bodyContains',
/^.*_challenges_solved{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
20:     .expect('bodyContains',
/^.*_challenges_total{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
21:     .expect('bodyContains', /^.*_cheat_score{app=".*"} [0-9.]*$/gm)

```

L

test/api/metricsApiSpec.ts
(Line 18)

typescript:S6353

Use concise character class syntax '\d' instead of '[0-9]'.

■ Problematic Code:

```

15:     return frisby.get(API_URL)
16:     .expect('status', 200)
17:     .expect('header', 'content-type', /text\/plain/)
>>> 18:     .expect('bodyContains', /^.*_version_info{version="[0-9]+\.[0-9]+\.[0-9]+(-SNAPSHO
T)?",major="[0-9]+",minor="[0-9]+",patch="[0-9]+",app=".*"} 1$/gm)
19:     .expect('bodyContains',
/^.*_challenges_solved{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
20:     .expect('bodyContains',
/^.*_challenges_total{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
21:     .expect('bodyContains', /^.*_cheat_score{app=".*"} [0-9.]*$/gm)

```

L

test/api/metricsApiSpec.ts
(Line 18)

typescript:S6353

Use concise character class syntax '\d' instead of '[0-9]'.

■ Problematic Code:

```

15:     return frisby.get(API_URL)
16:     .expect('status', 200)
17:     .expect('header', 'content-type', /text\/plain/)
>>> 18:     .expect('bodyContains', /^.*_version_info{version="[0-9]+\.[0-9]+\.[0-9]+(-SNAPSHO
T)?",major="[0-9]+",minor="[0-9]+",patch="[0-9]+",app=".*"} 1$/gm)
19:     .expect('bodyContains',
/^.*_challenges_solved{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
20:     .expect('bodyContains',
/^.*_challenges_total{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
21:     .expect('bodyContains', /^.*_cheat_score{app=".*"} [0-9.]*$/gm)

```

L

test/api/metricsApiSpec.ts
(Line 18)

typescript:S6353

Use concise character class syntax '\d' instead of '[0-9]'.

■ Problematic Code:

```

15:     return frisby.get(API_URL)
16:     .expect('status', 200)
17:     .expect('header', 'content-type', /text\/plain/)
>>> 18:     .expect('bodyContains', /^.*_version_info{version="[0-9]+\.[0-9]+\.[0-9]+(-SNAPSHOT)?",major="[0-9]+",minor="[0-9]+",patch="[0-9]+",app=".*"} 1$/gm)
19:     .expect('bodyContains',
/^.*_challenges_solved{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
20:     .expect('bodyContains',
/^.*_challenges_total{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
21:     .expect('bodyContains', /^.*_cheat_score{app=".*"} [0-9]*$/gm)

```

L

test/api/metricsApiSpec.ts
(Line 19)

typescript:S6353

Use concise character class syntax '\d' instead of '[0-9]'.

■ Problematic Code:

```

16:     .expect('status', 200)
17:     .expect('header', 'content-type', /text\/plain/)
18:     .expect('bodyContains', /^.*_version_info{version="[0-9]+\.[0-9]+\.[0-9]+(-SNAPSHOT)?",major="[0-9]+",minor="[0-9]+",patch="[0-9]+",app=".*"} 1$/gm)
>>> 19:     .expect('bodyContains',
/^.*_challenges_solved{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
20:     .expect('bodyContains',
/^.*_challenges_total{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
21:     .expect('bodyContains', /^.*_cheat_score{app=".*"} [0-9]*$/gm)
22:     .expect('bodyContains', /^.*_orders_placed_total{app=".*"} [0-9]*$/gm)

```

L

test/api/metricsApiSpec.ts
(Line 20)

typescript:S6353

Use concise character class syntax '\d' instead of '[0-9]'.

■ Problematic Code:

```

17:     .expect('header', 'content-type', /text\/plain/)
18:     .expect('bodyContains', /^.*_version_info{version="[0-9]+\.[0-9]+\.[0-9]+(-SNAPSHOT)?",major="[0-9]+",minor="[0-9]+",patch="[0-9]+",app=".*"} 1$/gm)
19:     .expect('bodyContains',
/^.*_challenges_solved{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
>>> 20:     .expect('bodyContains',
/^.*_challenges_total{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
21:     .expect('bodyContains', /^.*_cheat_score{app=".*"} [0-9]*$/gm)
22:     .expect('bodyContains', /^.*_orders_placed_total{app=".*"} [0-9]*$/gm)
23:     .expect('bodyContains', /^.*_users_registered{type="standard",app=".*"}
[0-9]*$/gm)

```

L

test/server
challengeTutorialSequenceSpec.ts
(Line 34)

typescript:S6653





Use 'Object.hasOwn()' instead of
'Object.prototype.hasOwnProperty.call()'.

■ Problematic Code:

```

31:     for (const { tutorial } of challenges) {
32:         if (tutorial) {
33:             const order: string = tutorial.order
>>> 34:             if (!Object.prototype.hasOwnProperty.call(tutorialOrderCounts, order)) {
35:                 tutorialOrderCounts[order] = 0
36:             }
37:             tutorialOrderCounts[order]++

```

	test/api/metricsApiSpec.ts (Line 61)	typescript:S6353 Use concise character class syntax '\d' instead of '[0-9]'.
<div> <div>■ Problematic Code:</div> <pre> 58: return frisby.get(API_URL) 59: .expect('status', 200) 60: .expect('header', 'content-type', /text\/plain/) >>> 61: .expect('bodyContains', /^file_upload_errors{file_type=".*",app=".*"} [0-9]*\$/gm) 62: }) 63: }) 64: }) </pre> </div>		
	test/api/metricsApiSpec.ts (Line 45)	typescript:S6353 Use concise character class syntax '\d' instead of '[0-9]'.
<div> <div>■ Problematic Code:</div> <pre> 42: return frisby.get(API_URL) 43: .expect('status', 200) 44: .expect('header', 'content-type', /text\/plain/) >>> 45: .expect('bodyContains', /^file_uploads_count{file_type=".*",app=".*"} [0-9]*\$/gm) 46: }) 47: }) 48: </pre> </div>		
	test/api/metricsApiSpec.ts (Line 30)	typescript:S6353 Use concise character class syntax '\d' instead of '[0-9]'.
<div> <div>■ Problematic Code:</div> <pre> 27: .expect('bodyContains', /^.*_user_social_interactions{type="review",app=".*"} [0-9]*\$/gm) 28: .expect('bodyContains', /^.*_user_social_interactions{type="feedback",app=".*"} [0-9]*\$/gm) 29: .expect('bodyContains', /^.*_user_social_interactions{type="complaint",app=".*"} [0-9]*\$/gm) >>> 30: .expect('bodyContains', /^http_requests_count{status_code="[0-9]XX",app=".*"} [0-9]*\$/gm) 31: }) 32: 33: xit('GET file upload metrics via public API', () => { // FIXME Flaky on CI/CD on at least Windows </pre> </div>		
	test/api/metricsApiSpec.ts (Line 30)	typescript:S6353 Use concise character class syntax '\d' instead of '[0-9]'.

	<div data-bbox="175 264 458 293" data-label="Section-Header"> Problematic Code: </div> <pre data-bbox="175 300 1404 674"> 27: .expect('bodyContains', /^.*_user_social_interactions{type="review",app=".*"} [0-9]*\$/gm) 28: .expect('bodyContains', /^.*_user_social_interactions{type="feedback",app=".*"} [0-9]*\$/gm) 29: .expect('bodyContains', /^.*_user_social_interactions{type="complaint",app=".*"} [0-9]*\$/gm) >>> 30: .expect('bodyContains', /^http_requests_count{status_code="[0-9]XX",app=".*"} [0-9]*\$/gm) 31: }) 32: 33: xit('GET file upload metrics via public API', () => { // FIXME Flaky on CI/CD on at least Windows </pre>	
<div data-bbox="165 712 213 763" data-label="Image"> </div>	<div data-bbox="276 712 529 775" data-label="Text"> test/api/metricsApiSpec.ts (Line 27) </div>	<div data-bbox="655 712 1323 779" data-label="Text"> typescript:S6353 Use concise character class syntax '\d' instead of '[0-9]'. </div>
	<div data-bbox="175 815 458 844" data-label="Section-Header"> Problematic Code: </div> <pre data-bbox="175 851 1404 1225"> 24: .expect('bodyContains', /^.*_users_registered{type="deluxe",app=".*"} [0-9]*\$/gm) 25: .expect('bodyContains', /^.*_users_registered_total{app=".*"} [0-9]*\$/gm) 26: .expect('bodyContains', /^.*_wallet_balance_total{app=".*"} [0-9]*\$/gm) >>> 27: .expect('bodyContains', /^.*_user_social_interactions{type="review",app=".*"} [0-9]*\$/gm) 28: .expect('bodyContains', /^.*_user_social_interactions{type="feedback",app=".*"} [0-9]*\$/gm) 29: .expect('bodyContains', /^.*_user_social_interactions{type="complaint",app=".*"} [0-9]*\$/gm) 30: .expect('bodyContains', /^http_requests_count{status_code="[0-9]XX",app=".*"} [0-9]*\$/gm) </pre>	
<div data-bbox="165 1265 213 1317" data-label="Image"> </div>	<div data-bbox="276 1265 529 1328" data-label="Text"> test/api/metricsApiSpec.ts (Line 28) </div>	<div data-bbox="655 1265 1323 1332" data-label="Text"> typescript:S6353 Use concise character class syntax '\d' instead of '[0-9]'. </div>
	<div data-bbox="175 1368 458 1397" data-label="Section-Header"> Problematic Code: </div> <pre data-bbox="175 1404 1404 1749"> 25: .expect('bodyContains', /^.*_users_registered_total{app=".*"} [0-9]*\$/gm) 26: .expect('bodyContains', /^.*_wallet_balance_total{app=".*"} [0-9]*\$/gm) 27: .expect('bodyContains', /^.*_user_social_interactions{type="review",app=".*"} [0-9]*\$/gm) >>> 28: .expect('bodyContains', /^.*_user_social_interactions{type="feedback",app=".*"} [0-9]*\$/gm) 29: .expect('bodyContains', /^.*_user_social_interactions{type="complaint",app=".*"} [0-9]*\$/gm) 30: .expect('bodyContains', /^http_requests_count{status_code="[0-9]XX",app=".*"} [0-9]*\$/gm) 31: }) </pre>	
<div data-bbox="165 1787 213 1839" data-label="Image"> </div>	<div data-bbox="276 1787 529 1850" data-label="Text"> test/api/metricsApiSpec.ts (Line 29) </div>	<div data-bbox="655 1787 1323 1854" data-label="Text"> typescript:S6353 Use concise character class syntax '\d' instead of '[0-9]'. </div>

■ Problematic Code:

```

26:         .expect('bodyContains', /^.*_wallet_balance_total{app=".*"} [0-9]*$/gm)
27:         .expect('bodyContains', /^.*_user_social_interactions{type="review",app=".*"}
[0-9]*$/gm)
28:         .expect('bodyContains', /^.*_user_social_interactions{type="feedback",app=".*"}
[0-9]*$/gm)
>>> 29:         .expect('bodyContains', /^.*_user_social_interactions{type="complaint",app=".*"}
[0-9]*$/gm)
30:         .expect('bodyContains', /^http_requests_count{status_code="[0-9]XX",app=".*"}
[0-9]*$/gm)
31:     })
32:

```

L

test/api/metricsApiSpec.ts
(Line 22)

typescript:S6353

Use concise character class syntax '\d' instead of '[0-9]'.

■ Problematic Code:

```

19:         .expect('bodyContains',
/^.*_challenges_solved{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
20:         .expect('bodyContains',
/^.*_challenges_total{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
21:         .expect('bodyContains', /^.*_cheat_score{app=".*"} [0-9.]*$/gm)
>>> 22:         .expect('bodyContains', /^.*_orders_placed_total{app=".*"} [0-9]*$/gm)
23:         .expect('bodyContains', /^.*_users_registered{type="standard",app=".*"}
[0-9]*$/gm)
24:         .expect('bodyContains', /^.*_users_registered{type="deluxe",app=".*"}
[0-9]*$/gm)
25:         .expect('bodyContains', /^.*_users_registered_total{app=".*"} [0-9]*$/gm)

```

L

test/api/metricsApiSpec.ts
(Line 23)

typescript:S6353

Use concise character class syntax '\d' instead of '[0-9]'.

■ Problematic Code:

```

20:         .expect('bodyContains',
/^.*_challenges_total{difficulty="[1-6]",category=".*",app=".*"} [0-9]*$/gm)
21:         .expect('bodyContains', /^.*_cheat_score{app=".*"} [0-9.]*$/gm)
22:         .expect('bodyContains', /^.*_orders_placed_total{app=".*"} [0-9]*$/gm)
>>> 23:         .expect('bodyContains', /^.*_users_registered{type="standard",app=".*"}
[0-9]*$/gm)
24:         .expect('bodyContains', /^.*_users_registered{type="deluxe",app=".*"}
[0-9]*$/gm)
25:         .expect('bodyContains', /^.*_users_registered_total{app=".*"} [0-9]*$/gm)
26:         .expect('bodyContains', /^.*_wallet_balance_total{app=".*"} [0-9]*$/gm)

```

L

test/api/metricsApiSpec.ts
(Line 24)

typescript:S6353

Use concise character class syntax '\d' instead of '[0-9]'.

	<p>■ Problematic Code:</p> <pre> 21: .expect('bodyContains', /^.*_cheat_score{app=".*" } [0-9.]*\$/gm) 22: .expect('bodyContains', /^.*_orders_placed_total{app=".*" } [0-9]*\$/gm) 23: .expect('bodyContains', /^.*_users_registered{type="standard",app=".*" } [0-9]*\$/gm) >>> 24: .expect('bodyContains', /^.*_users_registered{type="deluxe",app=".*" } [0-9]*\$/gm) 25: .expect('bodyContains', /^.*_users_registered_total{app=".*" } [0-9]*\$/gm) 26: .expect('bodyContains', /^.*_wallet_balance_total{app=".*" } [0-9]*\$/gm) 27: .expect('bodyContains', /^.*_user_social_interactions{type="review",app=".*" } [0-9]*\$/gm) </pre>	
<div>L</div>	test/api/metricsApiSpec.ts (Line 25)	typescript:S6353 Use concise character class syntax '\d' instead of '[0-9]'.
	<p>■ Problematic Code:</p> <pre> 22: .expect('bodyContains', /^.*_orders_placed_total{app=".*" } [0-9]*\$/gm) 23: .expect('bodyContains', /^.*_users_registered{type="standard",app=".*" } [0-9]*\$/gm) 24: .expect('bodyContains', /^.*_users_registered{type="deluxe",app=".*" } [0-9]*\$/gm) >>> 25: .expect('bodyContains', /^.*_users_registered_total{app=".*" } [0-9]*\$/gm) 26: .expect('bodyContains', /^.*_wallet_balance_total{app=".*" } [0-9]*\$/gm) 27: .expect('bodyContains', /^.*_user_social_interactions{type="review",app=".*" } [0-9]*\$/gm) 28: .expect('bodyContains', /^.*_user_social_interactions{type="feedback",app=".*" } [0-9]*\$/gm) </pre>	
<div>L</div>	test/api/metricsApiSpec.ts (Line 26)	typescript:S6353 Use concise character class syntax '\d' instead of '[0-9]'.
	<p>■ Problematic Code:</p> <pre> 23: .expect('bodyContains', /^.*_users_registered{type="standard",app=".*" } [0-9]*\$/gm) 24: .expect('bodyContains', /^.*_users_registered{type="deluxe",app=".*" } [0-9]*\$/gm) 25: .expect('bodyContains', /^.*_users_registered_total{app=".*" } [0-9]*\$/gm) >>> 26: .expect('bodyContains', /^.*_wallet_balance_total{app=".*" } [0-9]*\$/gm) 27: .expect('bodyContains', /^.*_user_social_interactions{type="review",app=".*" } [0-9]*\$/gm) 28: .expect('bodyContains', /^.*_user_social_interactions{type="feedback",app=".*" } [0-9]*\$/gm) 29: .expect('bodyContains', /^.*_user_social_interactions{type="complaint",app=".*" } [0-9]*\$/gm) </pre>	
<div>L</div>	routes/search.ts (Line 31)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.

■ Problematic Code:

```

28:         UserModel.findAll().then(data => {
29:             const users = utils.queryResultToJson(data)
30:             if (users.data?.length) {
>>> 31:                 for (let i = 0; i < users.data.length; i++) {
32:                     solved = solved && utils.containsOrEscaped(dataString,
users.data[i].email) && utils.contains(dataString, users.data[i].password)
33:                     if (!solved) {
34:                         break

```

L

server.ts
(Line 516)

typescript:S4138

Expected a `for-of` loop instead of a `for` loop with this simple iteration.

■ Problematic Code:

```

513:         // translate challenge descriptions on-the-fly
514:         if (name === 'Challenge') {
515:             resource.list.fetch.after((req: Request, res: Response, context: { instance:
string | any[], continue: any }) => {
>>> 516:                 for (let i = 0; i < context.instance.length; i++) {
517:                     let description = context.instance[i].description
518:                     if (utils.contains(description, '<em>(This challenge is <strong>')) {
519:                         const warning = description.substring(description.indexOf(' <em>(This
challenge is <strong>'))

```

L

lib/startup/validateConfig.ts
(Line 62)

typescript:S1874

The signature '(from: number, length?: number | undefined): string' of 'message.substr' is deprecated.

■ Problematic Code:

```

59:     if (schemaErrors.length !== 0) {
60:         logger.warn(`Config schema validation failed with ${schemaErrors.length} errors
(${colors.red('NOT OK')})`)
61:         schemaErrors.forEach(({ path, message }: { path: string, message: string }) => {
>>> 62:             logger.warn(`${path}:${colors.red(message.substr(message.indexOf(path) +
path.length))}`)
63:         })
64:         success = false
65:     }

```

L

routes/search.ts
(Line 64)

typescript:S4138

Expected a `for-of` loop instead of a `for` loop with this simple iteration.

■ Problematic Code:




```

61:         }
62:     })
63: } // vuln-code-snippet hide-end
>>> 64:     for (let i = 0; i < products.length; i++) {
65:         products[i].name = req.__(products[i].name)
66:         products[i].description = req.__(products[i].description)
67:     }

```


<div>L</div>	server.ts (Line 537)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 534: // translate security questions on-the-fly 535: if (name === 'SecurityQuestion') { 536: resource.list.fetch.after((req: Request, res: Response, context: { instance: string any[], continue: any }) => { >>> 537: for (let i = 0; i < context.instance.length; i++) { 538: context.instance[i].question = req.__(context.instance[i].question) 539: } 540: return context.continue </pre> </div>		
<div>L</div>	server.ts (Line 565)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 562: // translate product names and descriptions on-the-fly 563: if (name === 'Product') { 564: resource.list.fetch.after((req: Request, res: Response, context: { instance: any[], continue: any }) => { >>> 565: for (let i = 0; i < context.instance.length; i++) { 566: context.instance[i].name = req.__(context.instance[i].name) 567: context.instance[i].description = req.__(context.instance[i].description) 568: } </pre> </div>		
<div>L</div>	routes/currentUser.ts (Line 18)	typescript:S2486 Handle this exception or don't catch it at all.
<div> <div>■ Problematic Code:</div> <pre> 15: if (security.verify(req.cookies.token)) { 16: user = security.authenticatedUsers.get(req.cookies.token) 17: } >>> 18: } catch (err) { 19: user = undefined 20: } finally { 21: const response = { user: { id: (user?.data ? user.data.id : undefined), email: (user?.data ? user.data.email : undefined), lastLoginIp: (user?.data ? user.data.lastLoginIp : undefined), profileImage: (user?.data ? user.data.profileImage : undefined) } } </pre> </div>		
<div>L</div>	frontend/src/app/photo-wall mime-type.validator.ts (Line 23)	typescript:S4138 Expected a `for-of` loop instead of a `for` loop with this simple iteration.
<div> <div>■ Problematic Code:</div> <pre> 20: const arr = new Uint8Array(fileReader.result as ArrayBuffer).subarray(0, 4) 21: let header = '' 22: let isValid = false >>> 23: for (let i = 0; i < arr.length; i++) { 24: header += arr[i].toString(16) 25: } 26: switch (header) { </pre> </div>		

<div>L</div>	routes/languages.ts (Line 63)	typescript:S6653 Use 'Object.hasOwn()' instead of 'Object.prototype.hasOwnProperty.call()'.
<div> <div>■ Problematic Code:</div> <pre> 60: return await new Promise((resolve, reject) => { 61: try { 62: for (const key in fileContent) { >>> 63: if (Object.prototype.hasOwnProperty.call(fileContent, key) && fileContent[key] !== enContent[key]) { 64: differentStrings++ 65: } 66: } </pre> </div>		
<div>L</div>	test/server challengeCountryMappingSpec.ts (Line 41)	typescript:S6653 Use 'Object.hasOwn()' instead of 'Object.prototype.hasOwnProperty.call()'.
<div> <div>■ Problematic Code:</div> <pre> 38: for (const key of Object.keys(countryMapping)) { 39: const { code } = countryMapping[key] 40: >>> 41: if (!Object.prototype.hasOwnProperty.call(countryCodeCounts, code)) { 42: countryCodeCounts[code] = 0 43: } 44: countryCodeCounts[code]++ </pre> </div>		
<div>L</div>	frontend/src/app/search-result search-result.component.html (Line 40)	Web:S6842 Non-interactive elements should not be assigned interactive roles.
<div> <div>■ Problematic Code:</div> <pre> 37: } 38: <div class="product" (click)="showDetail(item)" aria-label="Click for more information about the product" 39: matTooltip="Click for more information" matTooltipPosition="above"> >>> 40: 42: <div class="info-box"> 43: <div class="item-name">{{item.name}}</div> </pre> </div>		
<div>L</div>	routes/countryMapping.ts (Line 19)	typescript:S2486 Handle this exception or don't catch it at all.
<div> <div>■ Problematic Code:</div> <pre> 16: } else { 17: res.send(countryMapping) 18: } >>> 19: } catch (err) { 20: logger.warn('Country mapping was requested but was not found in the selected config file. Take a look at the fbctf.yml config file to find out how to configure the country mappings required by FBCTF.') 21: res.status(500).send() 22: } </pre> </div>		

	routes/2fa.ts (Line 173)	typescript:S2486 Handle this exception or don't catch it at all.
<div> <div>■ Problematic Code:</div> <pre> 170: security.authenticatedUsers.updateFrom(req, utils.queryResultToJson(userModel)) 171: 172: res.status(200).send() >>> 173: } catch (error) { 174: res.status(401).send() 175: } 176: }</pre> </div>		
	routes/2fa.ts (Line 89)	typescript:S2486 Handle this exception or don't catch it at all.
<div> <div>■ Problematic Code:</div> <pre> 86: setup: true 87: }) 88: } >>> 89: } catch (error) { 90: res.status(401).send() 91: } 92: }</pre> </div>		
	routes/2fa.ts (Line 140)	typescript:S2486 Handle this exception or don't catch it at all.
<div> <div>■ Problematic Code:</div> <pre> 137: security.authenticatedUsers.updateFrom(req, utils.queryResultToJson(userModel)) 138: 139: res.status(200).send() >>> 140: } catch (error) { 141: res.status(401).send() 142: } 143: }</pre> </div>		
	lib/utlis.ts (Line 213)	typescript:S1874 The signature '(from: number, length?: number undefined): string' of 'ipv6.substr' is deprecated.
<div> <div>■ Problematic Code:</div> <pre> 210: 211: export const toSimpleIpAddress = (ipv6: string) => { 212: if (startsWith(ipv6, '::ffff:')) { >>> 213: return ipv6.substr(7) 214: } else if (ipv6 === ':::1') { 215: return '127.0.0.1' 216: } else {</pre> </div>		
	routes/2fa.ts (Line 54)	typescript:S2486 Handle this exception or don't catch it at all.

<p>■ Problematic Code:</p> <pre> 51: security.authenticatedUsers.put(token, plainUser) 52: 53: res.json({ authentication: { token, bid: basket.id, uemail: user.email } }) >>> 54: } catch (error) { 55: res.status(401).send() 56: } 57: }</pre>		
L	frontend/src/app/Services form-submit.service.ts (Line 22)	typescript:S1874 'keyCode' is deprecated.
<p>■ Problematic Code:</p> <pre> 19: form.addEventListener('keyup', function (event) { 20: event.preventDefault() 21: // eslint-disable-next-line import/no-deprecated >>> 22: if (event.keyCode === 13 && !submitButton.disabled) { 23: onSubmit() 24: } 25: })</pre>		
L	frontend/src/app/token-sale token-sale.component.html (Line 96)	Web:S6850 Headings must have content and the content must be accessible by a screen reader.
<p>■ Problematic Code:</p> <pre> 93: 94: <mat-card appearance="outlined" class="mat-elevation-z6"> 95: <div class="mdc-card"> >>> 96: <h5><i class='fas fa-comments fa-2x'></i> </h5> 97: <small class="text-justify">{{"ICO_FAQ_ANSWER" translate}}</small> 98: </div> 99: </mat-card></pre>		
L	routes/userProfile.ts (Line 63)	typescript:S2486 Handle this exception or don't catch it at all.
<p>■ Problematic Code:</p> <pre> 60: throw new Error('Username is null') 61: } 62: username = eval(code) // eslint-disable-line no-eval >>> 63: } catch (err) { 64: username = '\\\\' + username 65: } 66: } else {</pre>		
L	frontend/src/app/app.routing.ts (Line 236)	typescript:S1874 '(from: number, length?: number): string' is deprecated.




■ Problematic Code: <pre> 233: // vuln-code-snippet start tokenSaleChallenge 234: { 235: matcher: oauthMatcher, >>> 236: data: { params: (window.location.href).substr(window.location.href.indexOf('#')) }, 237: component: OAuthComponent 238: }, 239: { // vuln-code-snippet neutral-line tokenSaleChallenge </pre>		
L	frontend/src/main.ts (Line 7)	typescript:S1128 Remove this unused import of 'platformBrowserDynamic'.
■ Problematic Code: <pre> 4: */ 5: 6: import { enableProdMode, importProvidersFrom } from '@angular/core' >>> 7: import { platformBrowserDynamic } from '@angular/platform-browser-dynamic' 8: 9: import { AppModule, HttpLoaderFactory } from './app/app.module' 10: import { environment } from './environments/environment' </pre>		
L	lib/utils.ts (Line 65)	typescript:S1874 The signature '(from: number, length?: number undefined): string' of 'str.substr' is deprecated.
■ Problematic Code: <pre> 62: 63: export const trunc = function (str: string, length: number) { 64: str = str.replace(/(\r\n \n \r)/gm, '') >>> 65: return (str.length > length) ? str.substr(0, length - 1) + '...' : str 66: } 67: 68: export const version = (module?: string) => { </pre>		
I	routes/languages.ts (Line 10)	typescript:S1135 Complete the task associated to this "TODO" comment.
■ Problematic Code: <pre> 7: import fs from 'node:fs' 8: import { type Request, type Response, type NextFunction } from 'express' 9: >>> 10: export function getLanguageList () { // TODO Refactor and extend to also load backend translations from /i18n/*.json and calculate joint percentage/gauge 11: return (req: Request, res: Response, next: NextFunction) => { 12: const languages: Array<{ key: string, lang: any, icons: string[], shortKey: string, percentage: unknown, gauge: string }> = [] 13: let count = 0 </pre>		
I	routes/fileUpload.ts (Line 119)	typescript:S1135 Complete the task associated to this "TODO" comment.




<div>■ Problematic Code:</div> <pre>116: const yamlString = vm.runInContext('JSON.stringify(yaml.load(data))', sandbox, { timeout: 2000 }) 117: res.status(410) 118: next(new Error('B2B customer complaints via file upload have been deprecated for security reasons: ' + utils.trunc(yamlString, 400) + ' (' + file.originalname + ')')) >>> 119: } catch (err: any) { // TODO: Remove any 120: if (utils.contains(err.message, 'Invalid string length') utils.contains(err.message, 'Script execution timed out')) { 121: if (challengeUtils.notSolved(challenges.yamlBombChallenge)) { 122: challengeUtils.solve(challenges.yamlBombChallenge)</pre>		
<div>I</div>	<div>frontend/src/app/password-strength-info password-strength-info.component.spec.ts (Line 24)</div>	<div>typescript:S1135</div> <div>Complete the task associated to this "TODO" comment.</div>
<div>■ Problematic Code:</div> <pre>21: expect(component).toBeTruthy() 22: }) 23: >>> 24: // todo: unit test each conditional passwordLength message 25: }) 26:</pre>		
<div>I</div>	<div>frontend/src/app/app.module.ts (Line 16)</div>	<div>typescript:S1135</div> <div>Complete the task associated to this "TODO" comment.</div>
<div>■ Problematic Code:</div> <pre>13: return new TranslateHttpLoader(http, './assets/i18n/', '.json') 14: } 15: >>> 16: @NgModule(/* TODO(standalone-migration): clean up removed NgModule class manually. 17: { 18: declarations: [AppComponent], 19: imports: [</pre>		
<div>I</div>	<div>data/datacreator.ts (Line 76)</div>	<div>typescript:S1135</div> <div>Complete the task associated to this "TODO" comment.</div>
<div>■ Problematic Code:</div> <pre>73: 74: await Promise.all(75: challenges.map(async ({ name, category, description, difficulty, hints, mitigationUrl, key, disabledEnv, tutorial, tags }) => { >>> 76: // todo(@J12934) change this to use a proper challenge model or something 77: // eslint-disable-next-line @typescript-eslint/consistent-type-assertions 78: const { enabled: isChallengeEnabled, disabledBecause } = utils.getChallengeEnablementStatus({ disabledEnv: disabledEnv?.join(';') ?? '' } as ChallengeModel) 79: description = description.replace('juice-sh.op', config.get<string>('application.domain'))</pre>		

	data/datacreator.ts (Line 93)	typescript:S1135 Complete the task associated to this "TODO" comment.
<div> <div>■ Problematic Code:</div> <pre> 90: name, 91: category, 92: tags: (tags != null) ? tags.join(',') : undefined, >>> 93: // todo(@Jl2934) currently missing the 'not available' text. Needs changes to the model and utils functions 94: description: isChallengeEnabled ? description : (description + ' (This challenge is potentially harmful on ' + disabledBecause + '!)'), 95: difficulty, 96: solved: false, </pre> </div>		
	lib/config.types.ts (Line 2)	typescript:S1135 Complete the task associated to this "TODO" comment.
<div> <div>■ Problematic Code:</div> <pre> 1: // manually typed definitions for the config file >>> 2: // todo(@jannik.hollenbach) migrate the config schema to something which can automatically generate the types 3: 4: export interface ServerConfig { 5: port: number </pre> </div>		
	test/cypress/e2e/scoreBoard.spec.ts (Line 30)	typescript:S1135 Complete the task associated to this "TODO" comment.
<div> <div>■ Problematic Code:</div> <pre> 27: }) 28: }) 29: >>> 30: xdescribe('/#/score-board-legacy', () => { // TODO Replace with test based on new Score Board 31: describe('repeat notification', () => { 32: beforeEach(() => { 33: cy.visit('/#/score-board-legacy') </pre> </div>		
	lib/startup/validatePreconditions.ts (Line 94)	typescript:S1135 Complete the task associated to this "TODO" comment.
<div> <div>■ Problematic Code:</div> <pre> 91: domainDependencies[domain].forEach((dependency: string) => { 92: logger.warn(`\${colors.italic(dependency)} will not work as intended without access to \${colors.bold(domain)}`) 93: }) >>> 94: return true // TODO Consider switching to "false" with breaking release v16.0.0 or later 95: }) 96: } 97: </pre> </div>		
	lib/is-docker.ts (Line 26)	typescript:S1135 Complete the task associated to this "TODO" comment.

<div> <div>■ Problematic Code:</div> <pre> 23: } 24: 25: export default function isDocker () { >>> 26: // TODO: Use `??=` when targeting Node.js 16. 27: if (isDockerCached === undefined) { 28: isDockerCached = hasDockerEnv() hasDockerCGroup() 29: } </pre> </div>		
<div> <div>I</div> </div>	lib/codingChallenges.ts (Line 63)	typescript:S1135 Complete the task associated to this "TODO" comment.
<div> <div>■ Problematic Code:</div> <pre> 60: if (snippets == null) { 61: throw new BrokenBoundary('Broken code snippet boundaries for: ' + challengeKey) 62: } >>> 63: let snippet = snippets[0] // TODO Currently only a single code snippet is supported 64: snippet = snippet.replace(/\s?[/#]{0,2} vuln-code-snippet start.*[\r\n]{0,2}/g, '') 65: snippet = snippet.replace(/\s?[/#]{0,2} vuln-code-snippet end.*/g, '') 66: snippet = snippet.replace(/.*[/#]{0,2} vuln-code-snippet hide-line[\r\n]{0,2}/g, '') </pre> </div>		
<div> <div>I</div> </div>	test/cypress/e2e/restApi.spec.ts (Line 105)	typescript:S1135 Complete the task associated to this "TODO" comment.
<div> <div>■ Problematic Code:</div> <pre> 102: console.log('Success') 103: } 104: }) >>> 105: cy.expectChallengeSolved({ challenge: 'HTTP-Header XSS' }) // TODO Add missing check for alert presence 106: } 107: }) 108: }) </pre> </div>		
<div> <div>I</div> </div>	test/cypress/e2e/basket.spec.ts (Line 51)	typescript:S1135 Complete the task associated to this "TODO" comment.
<div> <div>■ Problematic Code:</div> <pre> 48: 49: cy.visit('/#/basket') 50: >>> 51: // TODO Verify functionally that it's not the basket of the admin 52: cy.expectChallengeSolved({ challenge: 'View Basket' }) 53: }) 54: }) </pre> </div>		
<div> <div>I</div> </div>	test/api/quantityApiSpec.ts (Line 185)	typescript:S1135 Complete the task associated to this "TODO" comment.

<div> <div>■ Problematic Code:</div> <pre> 182: }) 183: }) 184: >>> 185: xit('GET quantity of all items for accounting users from IP 123.456.789', () => { // TODO Check if possible to set IP in frisby tests 186: return frisby.post(`\${REST_URL}/user/login`, { 187: headers: jsonHeader, 188: body: { </pre> </div>		
<div> <div>I</div> </div>	test/api/quantityApiSpec.ts (Line 264)	typescript:S1135 Complete the task associated to this "TODO" comment.
<div> <div>■ Problematic Code:</div> <pre> 261: }) 262: }) 263: >>> 264: xit('PUT quantity as accounting user from IP 123.456.789', () => { // TODO Check if possible to set IP in frisby tests 265: return frisby.post(`\${REST_URL}/user/login`, { 266: headers: jsonHeader, 267: body: { </pre> </div>		
<div> <div>I</div> </div>	data/datacreator.ts (Line 391)	typescript:S1135 Complete the task associated to this "TODO" comment.
<div> <div>■ Problematic Code:</div> <pre> 388: persistedProduct) 389: }) 390: >>> 391: if (deletedDate) void deleteProduct(persistedProduct.id) // TODO Rename into "isDeleted" or "deletedFlag" in config for v14.x release 392: } else { 393: throw new Error('No persisted product found!') 394: } </pre> </div>		
<div> <div>I</div> </div>	routes/fileUpload.ts (Line 88)	typescript:S1135 Complete the task associated to this "TODO" comment.
<div> <div>■ Problematic Code:</div> <pre> 85: challengeUtils.solveIf(challenges.xxeFileDisclosureChallenge, () => { return (utils.matchesEtcPasswdFile(xmlString) utils.matchesSystemIniFile(xmlString)) }) 86: res.status(410) 87: next(new Error('B2B customer complaints via file upload have been deprecated for security reasons: ' + utils.trunc(xmlString, 400) + ' (' + file.originalname + ')')) >>> 88: } catch (err: any) { // TODO: Remove any 89: if (utils.contains(err.message, 'Script execution timed out')) { 90: if (challengeUtils.notSolved(challenges.xxeDosChallenge)) { 91: challengeUtils.solve(challenges.xxeDosChallenge) </pre> </div>		
<div> <div>I</div> </div>	lib/utlis.ts (Line 45)	typescript:S1135 Complete the task associated to this "TODO" comment.

<p>■ Problematic Code:</p> <pre> 42: 43: export const endsWith = (str?: string, suffix?: string) => (str && suffix) ? str.includes(suffix, str.length - suffix.length) : false 44: >>> 45: export const contains = (str: string, element: string) => str ? str.includes(element) : false // TODO Inline all usages as this function is not adding any functionality to String.includes 46: 47: export const containsEscaped = function (str: string, element: string) { 48: return contains(str, element.replace(/"/g, '\\\\"')) </pre>		
	data/static/codefixes adminSectionChallenge_1_correct.t s (Line 2)	typescript:S1135 Complete the task associated to this "TODO" comment.
<p>■ Problematic Code:</p> <pre> 1: const routes: Routes = [>>> 2: /* TODO: Externalize admin functions into separate application 3: that is only accessible inside corporate network. 4: */ 5: // { </pre>		
	lib/antiCheat.ts (Line 19)	typescript:S1135 Complete the task associated to this "TODO" comment.
<p>■ Problematic Code:</p> <pre> 16: import median from 'median' 17: import { type ChallengeKey } from 'models/challenge' 18: >>> 19: const coupledChallenges = { // TODO prevent also near-identical challenges (e.g. all null byte file access or dom xss + bonus payload etc.) from counting as cheating 20: loginAdminChallenge: ['weakPasswordChallenge'], 21: nullByteChallenge: ['easterEggLevelOneChallenge', 'forgottenDevBackupChallenge', 'forgottenBackupChallenge', 'misplacedSignatureFileChallenge'], 22: deprecatedInterfaceChallenge: ['uploadTypeChallenge', 'xxeFileDisclosureChallenge', 'xxeDosChallenge', 'yamlBombChallenge'], </pre>		
	frontend/src/hacking-instructor challenges/bonusPayload.ts (Line 21)	typescript:S1135 Complete the task associated to this "TODO" comment.

	<p>■ Problematic Code:</p> <pre> 18: 'Assuming you did the **DOM XSS** tutorial already, this one just uses a funnier payload on the _Search_ field.', 19: fixture: '#product-search-fixture', 20: unskippable: true, >>> 21: resolved: waitInMs(10000) // TODO Add check if "DOM XSS" is solved and if not recommend doing that first 22: }, 23: { 24: text: 'Enter or paste this payload into the _Search_ field: <code>&lt;iframe width=&quot;100%&quot; height=&quot;166&quot; scrolling=&quot;no&quot; frameborder=&quot;no&quot; allow=&quot;autoplay&quot; src=&quot;https://w.soundcloud.com/player /?url=https%3A//api.soundcloud.com/tracks/771984076&amp;color=%23ff5500&amp;auto&lowbar;play=tr ue&amp;hide&lowbar;related=false&amp;show&lowbar;comments=true&amp;show&lowbar;user=true&amp;sh ow&lowbar;reposts=false&amp;show&lowbar;teaser=true&quot;&gt;&lt;/iframe&gt;</code>.', </pre>	
	frontend/src/hacking-instructor challenges/passwordStrength.ts (Line 43)	typescript:S1135 Complete the task associated to this "TODO" comment.
	<p>■ Problematic Code:</p> <pre> 40: text: "Enter the admin's email address into the **email field**.", 41: fixture: '#email', 42: unskippable: true, >>> 43: resolved: waitForInputToHaveValue('#email', 'admin@juice-sh.op') // TODO Use domain from config instead 44: }, 45: { 46: text: 'Now for the password. Lucky for us, the admin chose a really, really, **really** stupid one. Just try any that comes to your mind!', </pre>	
	frontend/src/hacking-instructor challenges/loginBender.ts (Line 22)	typescript:S1135 Complete the task associated to this "TODO" comment.
	<p>■ Problematic Code:</p> <pre> 19: "To start this challenge, you'll have to log out first.", 20: fixture: '#navbarAccount', 21: unskippable: true, >>> 22: resolved: waitForLogout() // TODO Add check if "Login Admin" is solved and if not recommend doing that first 23: }, 24: { 25: text: </pre>	
	frontend/src/hacking-instructor challenges/loginJim.ts (Line 22)	typescript:S1135 Complete the task associated to this "TODO" comment.

■ Problematic Code:

```
19:         "To start this challenge, you'll have to log out first.",
20:         fixture: '#navbarAccount',
21:         unskippable: true,
>>> 22:         resolved: waitForLogout() // TODO Add check if "Login Admin" is solved and if not
recommend doing that first
23:     },
24:     {
25:         text:
```

Risk	File Path	Rule & Message
H	test/api/userApiSpec.ts (Line 302)	typescript:S6418 "Authorization" detected here, make sure this is not a hard-coded secret. Category: Authentication
<div>■ Security Hotspot Code:</div> <pre>299: }) 300: 301: it('GET who-am-i request returns nothing on expired auth token', () => { >>> 302: return frisby.get(`\${REST_URL}/user/whoami`, { headers: { Authorization: 'eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiaWF0YSI6eyIpZmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGU0IjoiJHVGplbiIsImxhc3Rmb2dpbkIWIjoicmc4wLjAuMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMDE5LTA4LTE5IDE1OjU2OjE1LjYyOSArMDA6MDAiLCJlcGRhdGVkQXQiOiIyMDE5LTA4LTE5IDE1OjU2OjE1LjYyOSArMDA6MDAiLCJkZWxldGVkQXQiOm5lbGx9LCJpYXQiOjE1bnVtYmZAYmJqSImV4cCI6MTU2NjI0ODIyNH0.Fl0kkcInY5sDMGKeLHfEOYDTQd3Bjr6_mK7Tcm_RH6iCLotTSRRoRxHpLkbtIQKqBFIt14J4BpLapkg7ppRWCEley5nego-4iFomXQvCBz5ISS3HdtM0saJnOe0agyVUen3huFp4F2UCth_y2Scjm_n_4AgW66cz8NSFFRVpC8g' } }) 303: .expect('status', 200) 304: .expect('header', 'content-type', /application\/json/)) 305: .expect('json', {</pre>		
H	test/server/verifySpec.ts (Line 263)	typescript:S6418 "authorization" detected here, make sure this is not a hard-coded secret. Category: Authentication
<div>■ Security Hotspot Code:</div> <pre>260: Header: { "alg": "none", "typ": "JWT" } 261: Payload: { "data": { "email": "jwt3d@juice-sh.op" }, "iat": 1508639612, "exp": 9999999999 } 262: */ >>> 263: req.headers = { authorization: `Bearer eyJhbGciOiJIub251IiwidHlwIjoiiSldUIIn0.eyJkYXRhIjp7ImVtYWlsIjoian0bjnkQGplaWNlLXNoLm9wIn0sImhhdCI6MTUwODYyOTYxMiwizXhwIjo5OTk5OTk5OTk5fQ.` } 264: 265: verify.jwtChallenges()(req, res, next) 266:</pre>		
H	test/server/verifySpec.ts (Line 275)	typescript:S6418 "authorization" detected here, make sure this is not a hard-coded secret. Category: Authentication





<p>■ Security Hotspot Code:</p> <pre> 272: Header: { "alg": "none", "typ": "JWT" } 273: Payload: { "data": { "email": "jwtn3d@" }, "iat": 1508639612, "exp": 9999999999 } 274: */ >>> 275: req.headers = { authorization: 'Bearer eyJhbGciOiJub25lIiwidHlwIjoisIldUIn0.eyJkYXRhIjp7ImVtYWlsIjoian0bjNkQCU9LCCjYpYXQiojE1MDg2Mzk2MTIsImV4cCI6OTk5OTk5OTk5OX0.' } 276: 277: verify.jwtChallenges()(req, res, next) 278: </pre>		
	test/server/verifySpec.ts (Line 297)	<p>typescript:S6418 "authorization" detected here, make sure this is not a hard-coded secret. Category: Authentication</p>
<p>■ Security Hotspot Code:</p> <pre> 294: Header: { "alg": "HS256", "typ": "JWT" } 295: Payload: { "data": { "email": "rsa_lord@juice-sh.op" }, "iat": 1508639612, "exp": 9999999999 } 296: */ >>> 297: req.headers = { authorization: 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRhIjp7ImVtYWlsIjoicnNhX2xvcmRAanVpY2Utc2gub3AifSwiaWF0IjoxNTgyMjI1NTc1fQ.ycFwtqh4ht4Pq9K5rh iPPY256F9YCTIecd4FHFuSEAg' } 298: 299: verify.jwtChallenges()(req, res, next) 300: </pre>		
	test/server/verifySpec.ts (Line 309)	<p>typescript:S6418 "authorization" detected here, make sure this is not a hard-coded secret. Category: Authentication</p>
<p>■ Security Hotspot Code:</p> <pre> 306: Header: { "alg": "HS256", "typ": "JWT" } 307: Payload: { "data": { "email": "rsa_lord@" }, "iat": 1508639612, "exp": 9999999999 } 308: */ >>> 309: req.headers = { authorization: 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRhIjp7ImVtYWlsIjoicnNhX2xvcmRAIn0sIm1hdCI6MTU4MjIyMTY3NX0.50f6VAIQk2Uzpf3sgH-1JVrrTuwudonm2 DKn2ec7Tg8' } 310: 311: verify.jwtChallenges()(req, res, next) 312: </pre>		
	frontend/src/app/Services two-factor-auth-service.spec.ts (Line 65)	<p>typescript:S2068 Review this potentially hard-coded password. Category: Authentication</p>

<p>■ Security Hotspot Code:</p> <pre> 62: tick() 63: 64: expect(req.request.method).toBe('POST') >>> 65: expect(req.request.body).toEqual({ password: 's3cr3t!', initialToken: 'initialToken', setupToken: 'setupToken' }) 66: expect(res).toBe(undefined) 67: httpMock.verify() 68: }) </pre>		
<div>H</div>	frontend/src/app/Services two-factor-auth-service.spec.ts (Line 81)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 78: tick() 79: 80: expect(req.request.method).toBe('POST') >>> 81: expect(req.request.body).toEqual({ password: 's3cr3t!' }) 82: expect(res).toBe(undefined) 83: httpMock.verify() 84: }) </pre>		
<div>H</div>	frontend/src/app/oauth oauth.component.spec.ts (Line 84)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 81: it('will create regular user account with base64 encoded reversed email as password', fakeAsync(() => { 82: userService.oauthLogin.and.returnValue(of({ email: 'test@test.com' }))) 83: component.ngOnInit() >>> 84: expect(userService.save).toHaveBeenCalledWith({ email: 'test@test.com', password: 'bw9jLnRzZXRADHNldA==', passwordRepeat: 'bw9jLnRzZXRADHNldA==' }) 85: })) 86: 87: it('logs in user even after failed account creation as account might already have existed from previous OAuth login', fakeAsync(() => { </pre>		
<div>H</div>	frontend/src/app/oauth oauth.component.spec.ts (Line 84)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 81: it('will create regular user account with base64 encoded reversed email as password', fakeAsync(() => { 82: userService.oauthLogin.and.returnValue(of({ email: 'test@test.com' }))) 83: component.ngOnInit() >>> 84: expect(userService.save).toHaveBeenCalledWith({ email: 'test@test.com', password: 'bw9jLnRzZXRADHNldA==', passwordRepeat: 'bw9jLnRzZXRADHNldA==' }) 85: })) 86: 87: it('logs in user even after failed account creation as account might already have existed from previous OAuth login', fakeAsync(() => { </pre>		


<div>H</div>	frontend/src/app/oauth oauth.component.spec.ts (Line 91)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 88: userService.oauthLogin.and.returnValue(of({ email: 'test@test.com' })) 89: userService.save.and.returnValue(throwError({ error: 'Account already exists' })) 90: component.ngOnInit() >>> 91: expect(userService.login).toHaveBeenCalledWith({ email: 'test@test.com', password: 'bw9jLnRzZXRAdHNldA==', oauth: true }) 92: }) 93: 94: it('removes authentication token and basket id on failed subsequent regular login attempt', fakeAsync(() => { </pre> </div>		
<div>H</div>	frontend/src/app/register register.component.spec.ts (Line 115)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 112: it('password should not be more than 20 characters', () => { 113: let password: string = '' 114: for (let i = 0; i < 41; i++) { >>> 115: password += 'a' 116: } 117: component.passwordControl.setValue(password) 118: expect(component.passwordControl.valid).toBeFalsy() </pre> </div>		
<div>H</div>	frontend/src/app/register register.component.spec.ts (Line 133)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 130: }) 131: 132: it('password and repeat password should be the same', () => { >>> 133: const password: string = 'aaaaa' 134: const passwordRepeat: string = 'aaaaa' 135: component.passwordControl.setValue(password) 136: component.repeatPasswordControl.setValue('bbbbbb') </pre> </div>		
<div>H</div>	frontend/src/app/register register.component.spec.ts (Line 134)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 131: 132: it('password and repeat password should be the same', () => { 133: const password: string = 'aaaaa' >>> 134: const passwordRepeat: string = 'aaaaa' 135: component.passwordControl.setValue(password) 136: component.repeatPasswordControl.setValue('bbbbbb') 137: expect(component.repeatPasswordControl.valid).toBeFalsy() </pre> </div>		

<div>H</div>	frontend/src/app/register register.component.spec.ts (Line 151)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 148: component.repeatPasswordControl.setValue('password') 149: component.securityQuestionControl.setValue(1) 150: component.securityAnswerControl.setValue('Answer') >>> 151: const user = { email: 'x@x.xx', password: 'password', passwordRepeat: 'password', securityQuestion: { id: 1, question: 'Wat is?' }, securityAnswer: 'Answer' } 152: const securityAnswerObject = { UserId: 1, answer: 'Answer', SecurityQuestionId: 1 } 153: component.save() 154: tick() </pre> </div>		
<div>H</div>	frontend/src/app/register register.component.spec.ts (Line 151)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 148: component.repeatPasswordControl.setValue('password') 149: component.securityQuestionControl.setValue(1) 150: component.securityAnswerControl.setValue('Answer') >>> 151: const user = { email: 'x@x.xx', password: 'password', passwordRepeat: 'password', securityQuestion: { id: 1, question: 'Wat is?' }, securityAnswer: 'Answer' } 152: const securityAnswerObject = { UserId: 1, answer: 'Answer', SecurityQuestionId: 1 } 153: component.save() 154: tick() </pre> </div>		
<div>H</div>	test/api/2faSpec.ts (Line 174)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 171: it('GET should indicate 2fa is setup for 2fa enabled users', async () => { 172: const { token } = await login({ 173: email: `wurstbrot@\${config.get<string>('application.domain')}`, >>> 174: password: 'EinBelegtesBrotMitSchinkenSCHINKEN!', 175: totpSecret: 'IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH' 176: }) 177: }) </pre> </div>		
<div>H</div>	test/api/2faSpec.ts (Line 200)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 197: it('GET should indicate 2fa is not setup for users with 2fa disabled', async () => { 198: const { token } = await login({ 199: email: `J12934@\${config.get<string>('application.domain')}`, >>> 200: password: '0Y8rMnww\$*9VFYE\$59-!Fg1L6t&6lB' 201: }) 202: }) 203: // @ts-expect-error FIXME promise return handling broken </pre> </div>		

H	test/api/2faSpec.ts (Line 236)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 233: describe('/rest/2fa/setup', () => { 234: it('POST should be able to setup 2fa for accounts without 2fa enabled', async () => { 235: const email = 'fooooo1@bar.com' >>> 236: const password = '123456' 237: 238: const secret = 'ASDVAJSDUASZGDIADBJS' 239: </pre> </div>		
H	test/api/2faSpec.ts (Line 282)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 279: 280: it('POST should fail if the password doesnt match', async () => { 281: const email = 'fooooo2@bar.com' >>> 282: const password = '123456' 283: 284: const secret = 'ASDVAJSDUASZGDIADBJS' 285: </pre> </div>		
H	test/api/2faSpec.ts (Line 311)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 308: 309: it('POST should fail if the initial token is incorrect', async () => { 310: const email = 'fooooo3@bar.com' >>> 311: const password = '123456' 312: 313: const secret = 'ASDVAJSDUASZGDIADBJS' 314: </pre> </div>		
H	test/api/2faSpec.ts (Line 340)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 337: 338: it('POST should fail if the token is of the wrong type', async () => { 339: const email = 'fooooo4@bar.com' >>> 340: const password = '123456' 341: 342: const secret = 'ASDVAJSDUASZGDIADBJS' 343: </pre> </div>		
H	test/api/2faSpec.ts (Line 369)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 366: 367: it('POST should fail if the account has already set up 2fa', async () => { 368: const email = `wurstbrot@\${config.get<string>('application.domain')}` >>> 369: const password = 'EinBelegtesBrotMitSchinkenSCHINKEN!' 370: const totpSecret = 'IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH' 371: 372: const { token } = await login({ email, password, totpSecret }) </pre>		
	test/api/2faSpec.ts (Line 398)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 395: describe('/rest/2fa/disable', () => { 396: it('POST should be able to disable 2fa for account with 2fa enabled', async () => { 397: const email = 'fooooodisablel@bar.com' >>> 398: const password = '123456' 399: const totpSecret = 'ASDVAJSDUASZGDIADBJS' 400: 401: await register({ email, password, totpSecret }) </pre>		
	test/api/2faSpec.ts (Line 435)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 432: 433: it('POST should not be possible to disable 2fa without the correct password', async () => { 434: const email = 'fooooodisablel@bar.com' >>> 435: const password = '123456' 436: const totpSecret = 'ASDVAJSDUASZGDIADBJS' 437: 438: await register({ email, password, totpSecret }) </pre>		
	test/api/addressApiSpec.ts (Line 20)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 17: headers: jsonHeader, 18: body: { 19: email: 'jim@juice-sh.op', >>> 20: password: 'ncc-1701' 21: } 22: }) 23: .expect('status', 200) </pre>		
	test/api/authenticatedUsersSpec.ts (Line 21)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<div> <div>■ Security Hotspot Code:</div> <pre> 18: return frisby.get(`\${REST_URL}/user/authentication-details`, { headers: authHeader }) 19: .expect('status', 200) 20: .expect('json', 'data.?', { >>> 21: password: '*****' 22: }) 23: }) 24: </pre> </div>		
<div> <div>H</div> </div>	test/api/authenticatedUsersSpec.ts (Line 30)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 27: headers: jsonHeader, 28: body: { 29: email: `jim\${config.get<string>('application.domain')}`, >>> 30: password: 'ncc-1701' 31: } 32: }).promise() 33: </pre> </div>		
<div> <div>H</div> </div>	test/api/basketApiSpec.ts (Line 25)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 22: headers: jsonHeader, 23: body: { 24: email: 'jim@juice-sh.op', >>> 25: password: 'ncc-1701' 26: } 27: }) 28: .expect('status', 200) </pre> </div>		
<div> <div>H</div> </div>	test/api/basketApiSpec.ts (Line 101)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 98: headers: jsonHeader, 99: body: { 100: email: 'bjoern.kimminich@gmail.com', >>> 101: password: 'bW9jLmxpYWlnQGhjaW5pbWlpay5ucmVvamI=' 102: } 103: }) 104: .expect('status', 200) </pre> </div>		
<div> <div>H</div> </div>	test/api/basketItemApiSpec.ts (Line 21)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication





<p>■ Security Hotspot Code:</p> <pre> 18: headers: jsonHeader, 19: body: { 20: email: 'jim@' + config.get<string>('application.domain'), >>> 21: password: 'ncc-1701' 22: } 23: }) 24: .expect('status', 200) </pre>		
	test/api/chatBotSpec.ts (Line 56)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 53: it('GET bot state for authenticated users contains request for username', async () => { 54: const { token } = await login({ 55: email: `J12934@\${config.get<string>('application.domain')}`, >>> 56: password: '0Y8rMnww\$*9VFYE\$59-!FglL6t&6lB' 57: }) 58: 59: await frisby.setup({ </pre>		
	test/api/chatBotSpec.ts (Line 77)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 74: it('Asks for username if not defined', async () => { 75: const { token } = await login({ 76: email: `J12934@\${config.get<string>('application.domain')}`, >>> 77: password: '0Y8rMnww\$*9VFYE\$59-!FglL6t&6lB' 78: }) 79: 80: const testCommand = trainingData.data[0].utterances[0] </pre>		
	test/api/chatBotSpec.ts (Line 108)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 105: } 106: const { token } = await login({ 107: email: 'bjoern.kimminich@gmail.com', >>> 108: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 109: }) 110: 111: bot.addUser('1337', 'bkimminich') </pre>		
	test/api/chatBotSpec.ts (Line 140)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication


<div> <div>■ Security Hotspot Code:</div> <pre> 137: } 138: const { token } = await login({ 139: email: 'bjoern.kimminich@gmail.com', >>> 140: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 141: }) 142: bot.addUser('12345', 'bkimminich') 143: const testCommand = trainingData.data[0].utterances[0] </pre> </div>		
<div>H</div>	test/api/chatBotSpec.ts (Line 174)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 171: it('Responds with product price when asked question with product name', async () => { 172: const { token } = await login({ 173: email: 'bjoern.kimminich@gmail.com', >>> 174: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 175: }) 176: const { json } = await frisby.get(API_URL + '/Products/1') 177: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/chatBotSpec.ts (Line 205)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 202: it('Greets back registered user after being told username', async () => { 203: const { token } = await login({ 204: email: `stan@\${config.get<string>('application.domain')}`, >>> 205: password: 'ship coffin krypt cross estate supply insurance asbestos souvenir' 206: }) 207: await frisby.setup({ 208: request: { </pre> </div>		
<div>H</div>	test/api/chatBotSpec.ts (Line 250)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 247: const functionTest = trainingData.data.filter(data => data.intent === 'queries.functionTest') 248: const { token } = await login({ 249: email: 'bjoern.kimminich@gmail.com', >>> 250: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 251: }) 252: const testCommand = functionTest[0].utterances[0] 253: const testResponse = '3be2e438b7f3d04c89d7749f727bb3bd' </pre> </div>		
<div>H</div>	test/api/chatBotSpec.ts (Line 287)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 284: }, 285: body: { 286: email: `chatbot-testuser@\${config.get<string>('application.domain')}`, >>> 287: password: 'testtesttest', 288: username: '', 289: role: 'admin' 290: } </pre>		
	test/api/chatBotSpec.ts (Line 295)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 292: 293: const { token } = await login({ 294: email: `chatbot-testuser@\${config.get<string>('application.domain')}`, >>> 295: password: 'testtesttest' 296: }) 297: 298: const functionTest = trainingData.data.filter(data => data.intent === 'queries.functionTest') </pre>		
	test/api/dataExportApiSpec.ts (Line 21)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 18: headers: jsonHeader, 19: body: { 20: email: 'bjoern.kimminich@gmail.com', >>> 21: password: 'bW9jLmxpYWlnQGhjaW5pbWlpay5ucmVvamI=' 22: } 23: }) 24: .expect('status', 200) </pre>		
	test/api/dataExportApiSpec.ts (Line 48)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 45: headers: jsonHeader, 46: body: { 47: email: 'bjoern.kimminich@gmail.com', >>> 48: password: 'bW9jLmxpYWlnQGhjaW5pbWlpay5ucmVvamI=' 49: } 50: }) 51: .expect('status', 200) </pre>		
	test/api/dataExportApiSpec.ts (Line 77)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication





<p>■ Security Hotspot Code:</p> <pre> 74: headers: jsonHeader, 75: body: { 76: email: 'bjoern.kimminich@gmail.com', >>> 77: password: 'bW9jLmxpYWlnQGhjaW5pbWlpay5ucmVvamI=' 78: } 79: }) 80: .expect('status', 200) </pre>		
H	test/api/dataExportApiSpec.ts (Line 112)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 109: headers: jsonHeader, 110: body: { 111: email: 'amy@' + config.get<string>('application.domain'), >>> 112: password: 'Klf.....' 113: } 114: }) 115: .expect('status', 200) </pre>		
H	test/api/dataExportApiSpec.ts (Line 152)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 149: headers: jsonHeader, 150: body: { 151: email: 'jim@' + config.get<string>('application.domain'), >>> 152: password: 'ncc-1701' 153: } 154: }) 155: .expect('status', 200) </pre>		
H	test/api/dataExportApiSpec.ts (Line 194)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 191: headers: jsonHeader, 192: body: { 193: email: 'jim@' + config.get<string>('application.domain'), >>> 194: password: 'ncc-1701' 195: } 196: }) 197: .expect('status', 200) </pre>		
H	test/api/dataExportApiSpec.ts (Line 234)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 231: headers: jsonHeader, 232: body: { 233: email: 'amy@' + config.get<string>('application.domain'), >>> 234: password: 'Klf.....' 235: } 236: }) 237: .expect('status', 200) </pre>		
H	test/api/dataExportApiSpec.ts (Line 282)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 279: headers: jsonHeader, 280: body: { 281: email: 'jim@' + config.get<string>('application.domain'), >>> 282: password: 'ncc-1701' 283: } 284: }) 285: .expect('status', 200) </pre>		
H	test/api/dataExportApiSpec.ts (Line 332)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 329: headers: jsonHeader, 330: body: { 331: email: 'jim@' + config.get<string>('application.domain'), >>> 332: password: 'ncc-1701' 333: } 334: }) 335: .expect('status', 200) </pre>		
H	test/api/deliveryApiSpec.ts (Line 23)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 20: headers: jsonHeader, 21: body: { 22: email: 'jim@' + config.get<string>('application.domain'), >>> 23: password: 'ncc-1701' 24: } 25: }) 26: .expect('status', 200) </pre>		
H	test/api/deliveryApiSpec.ts (Line 52)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication





<p>■ Security Hotspot Code:</p> <pre> 49: headers: jsonHeader, 50: body: { 51: email: 'ciso@' + config.get<string>('application.domain'), >>> 52: password: 'mDLx?94T~1CfVfZMzw@sJ9f?s3L6lbMqE70FfI8^54jbNikY5fymx7c!YbJb' 53: } 54: }) 55: .expect('status', 200) </pre>		
	test/api/deliveryApiSpec.ts (Line 83)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 80: headers: jsonHeader, 81: body: { 82: email: 'jim@' + config.get<string>('application.domain'), >>> 83: password: 'ncc-1701' 84: } 85: }) 86: .expect('status', 200) </pre>		
	test/api/deliveryApiSpec.ts (Line 111)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 108: headers: jsonHeader, 109: body: { 110: email: 'ciso@' + config.get<string>('application.domain'), >>> 111: password: 'mDLx?94T~1CfVfZMzw@sJ9f?s3L6lbMqE70FfI8^54jbNikY5fymx7c!YbJb' 112: } 113: }) 114: .expect('status', 200) </pre>		
	test/api/deluxeApiSpec.ts (Line 35)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 32: headers: jsonHeader, 33: body: { 34: email: 'bender@' + config.get<string>('application.domain'), >>> 35: password: 'OhG0dPleaseInsertLiquor!' 36: } 37: }) 38: .expect('status', 200) </pre>		
	test/api/deluxeApiSpec.ts (Line 53)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 50: headers: jsonHeader, 51: body: { 52: email: 'ciso@' + config.get<string>('application.domain'), >>> 53: password: 'mDLx?94T~1CfVfZMzw@sJ9f?s3L61bMqE70FfI8^54jbNikY5fymx7c!YbJb' 54: } 55: }) 56: .expect('status', 200) </pre>		
	test/api/deluxeApiSpec.ts (Line 71)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 68: headers: jsonHeader, 69: body: { 70: email: 'admin@' + config.get<string>('application.domain'), >>> 71: password: 'admin123' 72: } 73: }) 74: .expect('status', 200) </pre>		
	test/api/deluxeApiSpec.ts (Line 89)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 86: headers: jsonHeader, 87: body: { 88: email: 'accountant@' + config.get<string>('application.domain'), >>> 89: password: 'i am an awesome accountant' 90: } 91: }) 92: .expect('status', 200) </pre>		
	test/api/deluxeApiSpec.ts (Line 105)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 102: it('POST upgrade deluxe membership status for customers', async () => { 103: const { token } = await login({ 104: email: `bender@\${config.get<string>('application.domain')}`, >>> 105: password: 'OhG0dPleaseInsertLiquor!' 106: }) 107: 108: const { json } = await frisby.get(API_URL + '/Cards', { </pre>		
	test/api/deluxeApiSpec.ts (Line 129)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 126: it('POST deluxe membership status with wrong card id throws error', async () => { 127: const { token } = await login({ 128: email: `jim\${config.get<string>('application.domain')}` , >>> 129: password: 'ncc-1701' 130: }) 131: 132: await frisby.post(REST_URL + '/deluxe-membership', { </pre>		
H	test/api/deluxeApiSpec.ts (Line 149)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 146: headers: jsonHeader, 147: body: { 148: email: 'ciso@' + config.get<string>('application.domain'), >>> 149: password: 'mDLx?94T~1CfVfZMzw@sJ9f?s3L6lbMqE70FfI8^54jbNikY5fymx7c!YbJb' 150: } 151: }) 152: .expect('status', 200) </pre>		
H	test/api/deluxeApiSpec.ts (Line 170)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 167: headers: jsonHeader, 168: body: { 169: email: 'admin@' + config.get<string>('application.domain'), >>> 170: password: 'admin123' 171: } 172: }) 173: .expect('status', 200) </pre>		
H	test/api/deluxeApiSpec.ts (Line 191)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 188: headers: jsonHeader, 189: body: { 190: email: 'accountant@' + config.get<string>('application.domain'), >>> 191: password: 'i am an awesome accountant' 192: } 193: }) 194: .expect('status', 200) </pre>		
H	test/api/erasureRequestApiSpec.ts (Line 18)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication





<p>■ Security Hotspot Code:</p> <pre> 15: headers: jsonHeader, 16: body: { 17: email: 'bjoern@owasp.org', >>> 18: password: 'kitten lesser pooch karate buffoon indoors' 19: } 20: }) 21: .expect('status', 200) </pre>		
	test/api/erasureRequestApiSpec.ts (Line 37)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 34: headers: jsonHeader, 35: body: { 36: email: 'bjoern.kimminich@gmail.com', >>> 37: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 38: } 39: }) 40: .expect('status', 200) </pre>		
	test/api/erasureRequestApiSpec.ts (Line 64)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 61: headers: jsonHeader, 62: body: { 63: email: 'bjoern.kimminich@gmail.com', >>> 64: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 65: } 66: }) 67: .expect('status', 200) </pre>		
	test/api/erasureRequestApiSpec.ts (Line 80)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 77: headers: jsonHeader, 78: body: { 79: email: 'bjoern.kimminich@gmail.com', >>> 80: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 81: } 82: }) 83: .expect('status', 200) </pre>		
	test/api/erasureRequestApiSpec.ts (Line 99)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication





<p>■ Security Hotspot Code:</p> <pre> 96: headers: jsonHeader, 97: body: { 98: email: 'bjoern.kimminich@gmail.com', >>> 99: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 100: } 101: }) 102: .expect('status', 200) </pre>		
H	test/api/erasureRequestApiSpec.ts (Line 119)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 116: headers: jsonHeader, 117: body: { 118: email: 'bjoern.kimminich@gmail.com', >>> 119: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 120: } 121: }) 122: .expect('status', 200) </pre>		
H	test/api/erasureRequestApiSpec.ts (Line 140)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 137: headers: jsonHeader, 138: body: { 139: email: 'bjoern.kimminich@gmail.com', >>> 140: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 141: } 142: }) 143: .expect('status', 200) </pre>		
H	test/api/feedbackApiSpec.ts (Line 120)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 117: headers: jsonHeader, 118: body: { 119: email: 'bjoern.kimminich@gmail.com', >>> 120: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 121: } 122: }) 123: .expect('status', 200) </pre>		
H	test/api/feedbackApiSpec.ts (Line 153)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication





<p>■ Security Hotspot Code:</p> <pre> 150: headers: jsonHeader, 151: body: { 152: email: 'bjoern.kimminich@gmail.com', >>> 153: password: 'bW9jLmXpYWlnQGhjaW5pbWlpay5ucmVvamI=' 154: } 155: }) 156: .expect('status', 200) </pre>		
	test/api/loginApiSpec.ts (Line 21)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 18: headers: jsonHeader, 19: body: { 20: email: 'kalli@kasper.le', >>> 21: password: 'kallliiii' 22: } 23: }) 24: .expect('status', 201) </pre>		
	test/api/loginApiSpec.ts (Line 30)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 27: headers: jsonHeader, 28: body: { 29: email: 'kalli@kasper.le', >>> 30: password: 'kallliiii' 31: } 32: }) 33: .expect('status', 200) </pre>		
	test/api/loginApiSpec.ts (Line 46)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 43: it('POST login non-existing user', () => { 44: return frisby.post(REST_URL + '/user/login', { 45: email: 'otto@mei.er', >>> 46: password: 'ooootto' 47: }, { json: true }) 48: .expect('status', 401) 49: }) </pre>		
	test/api/loginApiSpec.ts (Line 64)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication





<p>■ Security Hotspot Code:</p> <pre> 61: headers: jsonHeader, 62: body: { 63: email: 'admin@' + config.get<string>('application.domain'), >>> 64: password: 'admin123' 65: } 66: }) 67: .expect('status', 200) </pre>		
H	test/api/loginApiSpec.ts (Line 79)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 76: headers: jsonHeader, 77: body: { 78: email: 'support@' + config.get<string>('application.domain'), >>> 79: password: 'J6aVjTgOpRs@?5l!Zkq2AYnCE@RF\$P' 80: } 81: }) 82: .expect('status', 200) </pre>		
H	test/api/loginApiSpec.ts (Line 94)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 91: headers: jsonHeader, 92: body: { 93: email: 'mc.safesearch@' + config.get<string>('application.domain'), >>> 94: password: 'Mr. N00dles' 95: } 96: }) 97: .expect('status', 200) </pre>		
H	test/api/loginApiSpec.ts (Line 109)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 106: headers: jsonHeader, 107: body: { 108: email: 'amy@' + config.get<string>('application.domain'), >>> 109: password: 'Klf.....' 110: } 111: }) 112: .expect('status', 200) </pre>		
H	test/api/loginApiSpec.ts (Line 124)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication





<div> <div>■ Security Hotspot Code:</div> <pre> 121: headers: jsonHeader, 122: body: { 123: email: 'wurstbrot@' + config.get<string>('application.domain'), >>> 124: password: 'EinBelegtesBrotMitSchinkenSCHINKEN!' 125: } 126: }) 127: .expect('status', 401) </pre> </div>		
<div>H</div>	test/api/loginApiSpec.ts (Line 142)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 139: headers: jsonHeader, 140: body: { 141: email: 'bjoern.kimminich@gmail.com', >>> 142: password: 'bW9jLmxpYWlnQGhjaW5pbWlpay5ucmVvamI=' 143: } 144: }) 145: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/loginApiSpec.ts (Line 245)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 242: headers: jsonHeader, 243: body: { 244: email: 'bjoern.kimminich@gmail.com', >>> 245: password: 'bW9jLmxpYWlnQGhjaW5pbWlpay5ucmVvamI=' 246: } 247: }) 248: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/loginApiSpec.ts (Line 266)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 263: headers: jsonHeader, 264: body: { 265: email: 'bjoern.kimminich@gmail.com', >>> 266: password: 'bW9jLmxpYWlnQGhjaW5pbWlpay5ucmVvamI=' 267: } 268: }) 269: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/memoryApiSpec.ts (Line 26)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication


<p>■ Security Hotspot Code:</p> <pre> 23: headers: jsonHeader, 24: body: { 25: email: 'jim@' + config.get<string>('application.domain'), >>> 26: password: 'ncc-1701' 27: } 28: }) 29: .expect('status', 200) </pre>		
	test/api/memoryApiSpec.ts (Line 64)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 61: headers: jsonHeader, 62: body: { 63: email: 'jim@' + config.get<string>('application.domain'), >>> 64: password: 'ncc-1701' 65: } 66: }) 67: .expect('status', 200) </pre>		
	test/api/memoryApiSpec.ts (Line 86)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 83: headers: jsonHeader, 84: body: { 85: email: 'jim@' + config.get<string>('application.domain'), >>> 86: password: 'ncc-1701' 87: } 88: }) 89: .expect('status', 200) </pre>		
	test/api/memoryApiSpec.ts (Line 112)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 109: headers: jsonHeader, 110: body: { 111: email: 'jim@' + config.get<string>('application.domain'), >>> 112: password: 'ncc-1701' 113: } 114: }) 115: .expect('status', 200) </pre>		
	test/api/memoryApiSpec.ts (Line 142)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication



<p>■ Security Hotspot Code:</p> <pre> 139: headers: jsonHeader, 140: body: { 141: email: 'jim@' + config.get<string>('application.domain'), >>> 142: password: 'ncc-1701' 143: } 144: }) 145: .expect('status', 200) </pre>		
	test/api/orderHistoryApiSpec.ts (Line 19)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 16: headers: jsonHeader, 17: body: { 18: email: 'admin@' + config.get<string>('application.domain'), >>> 19: password: 'admin123' 20: } 21: }) 22: .expect('status', 200) </pre>		
	test/api/orderHistoryApiSpec.ts (Line 56)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 53: headers: jsonHeader, 54: body: { 55: email: 'jim@' + config.get<string>('application.domain'), >>> 56: password: 'ncc-1701' 57: } 58: }) 59: .expect('status', 200) </pre>		
	test/api/orderHistoryApiSpec.ts (Line 73)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 70: headers: jsonHeader, 71: body: { 72: email: 'admin@' + config.get<string>('application.domain'), >>> 73: password: 'admin123' 74: } 75: }) 76: .expect('status', 200) </pre>		
	test/api/orderHistoryApiSpec.ts (Line 90)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication



<p>■ Security Hotspot Code:</p> <pre> 87: headers: jsonHeader, 88: body: { 89: email: 'accountant@' + config.get<string>('application.domain'), >>> 90: password: 'i am an awesome accountant' 91: } 92: }) 93: .expect('status', 200) </pre>		
	test/api/orderHistoryApiSpec.ts (Line 109)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 106: headers: jsonHeader, 107: body: { 108: email: 'admin@' + config.get<string>('application.domain'), >>> 109: password: 'admin123' 110: } 111: }) 112: .expect('status', 200) </pre>		
	test/api/orderHistoryApiSpec.ts (Line 129)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 126: headers: jsonHeader, 127: body: { 128: email: 'jim@' + config.get<string>('application.domain'), >>> 129: password: 'ncc-1701' 130: } 131: }) 132: .expect('status', 200) </pre>		
	test/api/orderHistoryApiSpec.ts (Line 149)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 146: headers: jsonHeader, 147: body: { 148: email: 'accountant@' + config.get<string>('application.domain'), >>> 149: password: 'i am an awesome accountant' 150: } 151: }) 152: .expect('status', 200) </pre>		
	test/api/passwordApiSpec.ts (Line 20)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 17: headers: jsonHeader, 18: body: { 19: email: 'kuni@be.rt', >>> 20: password: 'kunigunde' 21: } 22: }) 23: .expect('status', 201) </pre>		
	test/api/passwordApiSpec.ts (Line 29)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 26: headers: jsonHeader, 27: body: { 28: email: 'kuni@be.rt', >>> 29: password: 'kunigunde' 30: } 31: }) 32: .expect('status', 200) </pre>		
	test/api/passwordApiSpec.ts (Line 47)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 44: headers: jsonHeader, 45: body: { 46: email: 'bjoern@' + config.get<string>('application.domain'), >>> 47: password: 'monkey summer birthday are all bad passwords but work just fine in a long passphrase' 48: } 49: }) 50: .expect('status', 200) </pre>		
	test/api/passwordApiSpec.ts (Line 93)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 90: headers: jsonHeader, 91: body: { 92: email: 'bender@' + config.get<string>('application.domain'), >>> 93: password: 'OhG0dPleaseInsertLiquor!' 94: } 95: }) 96: .expect('status', 200) </pre>		
	test/api/paymentApiSpec.ts (Line 20)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 17: headers: jsonHeader, 18: body: { 19: email: 'jim@juice-sh.op', >>> 20: password: 'ncc-1701' 21: } 22: }) 23: .expect('status', 200) </pre>		
	test/api/productReviewApiSpec.ts (Line 112)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 109: headers: jsonHeader, 110: body: { 111: email: 'bjoern.kimminich@gmail.com', >>> 112: password: 'bW9jLmxpYWlnQGhjaW5pbWlpay5ucmVvamI=' 113: } 114: }) 115: .expect('status', 200) </pre>		
	test/api/productReviewApiSpec.ts (Line 132)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 129: headers: jsonHeader, 130: body: { 131: email: 'bjoern.kimminich@gmail.com', >>> 132: password: 'bW9jLmxpYWlnQGhjaW5pbWlpay5ucmVvamI=' 133: } 134: }) 135: .expect('status', 200) </pre>		
	test/api/profileImageUploadSpec.ts (Line 25)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 22: headers: jsonHeader, 23: body: { 24: email: `jim@\${config.get<string>('application.domain')}`, >>> 25: password: 'ncc-1701' 26: } 27: }) 28: .expect('status', 200) </pre>		
	test/api/profileImageUploadSpec.ts (Line 52)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 49: headers: jsonHeader, 50: body: { 51: email: `jim\${config.get<string>('application.domain')}`, >>> 52: password: 'ncc-1701' 53: } 54: }) 55: .expect('status', 200) </pre>		
	test/api/profileImageUploadSpec.ts (Line 97)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 94: headers: jsonHeader, 95: body: { 96: email: `jim\${config.get<string>('application.domain')}`, >>> 97: password: 'ncc-1701' 98: } 99: }) 100: .expect('status', 200) </pre>		
	test/api/profileImageUploadSpec.ts (Line 123)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 120: headers: jsonHeader, 121: body: { 122: email: `jim\${config.get<string>('application.domain')}`, >>> 123: password: 'ncc-1701' 124: } 125: }) 126: .expect('status', 200) </pre>		
	test/api/profileImageUploadSpec.ts (Line 164)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 161: headers: jsonHeader, 162: body: { 163: email: `jim\${config.get<string>('application.domain')}`, >>> 164: password: 'ncc-1701' 165: } 166: }) 167: .expect('status', 200) </pre>		
	test/api/quantityApiSpec.ts (Line 21)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 18: headers: jsonHeader, 19: body: { 20: email: `jim\${config.get<string>('application.domain')}`, >>> 21: password: 'ncc-1701' 22: } 23: }) 24: .expect('status', 200) </pre>		
	test/api/quantityApiSpec.ts (Line 38)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 35: headers: jsonHeader, 36: body: { 37: email: `admin\${config.get<string>('application.domain')}`, >>> 38: password: 'admin123' 39: } 40: }) 41: .expect('status', 200) </pre>		
	test/api/quantityApiSpec.ts (Line 55)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 52: headers: jsonHeader, 53: body: { 54: email: `accountant\${config.get<string>('application.domain')}`, >>> 55: password: 'i am an awesome accountant' 56: } 57: }) 58: .expect('status', 200) </pre>		
	test/api/quantityApiSpec.ts (Line 72)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 69: headers: jsonHeader, 70: body: { 71: email: `jim\${config.get<string>('application.domain')}`, >>> 72: password: 'ncc-1701' 73: } 74: }) 75: .expect('status', 200) </pre>		
	test/api/quantityApiSpec.ts (Line 93)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 90: headers: jsonHeader, 91: body: { 92: email: `admin@\${config.get<string>('application.domain')}`, >>> 93: password: 'admin123' 94: } 95: }) 96: .expect('status', 200) </pre>		
	test/api/quantityApiSpec.ts (Line 114)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 111: headers: jsonHeader, 112: body: { 113: email: `accountant@\${config.get<string>('application.domain')}`, >>> 114: password: 'i am an awesome accountant' 115: } 116: }) 117: .expect('status', 200) </pre>		
	test/api/quantityApiSpec.ts (Line 137)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 134: headers: jsonHeader, 135: body: { 136: email: `jim@\${config.get<string>('application.domain')}`, >>> 137: password: 'ncc-1701' 138: } 139: }) 140: .expect('status', 200) </pre>		
	test/api/quantityApiSpec.ts (Line 155)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 152: headers: jsonHeader, 153: body: { 154: email: `admin@\${config.get<string>('application.domain')}`, >>> 155: password: 'admin123' 156: } 157: }) 158: .expect('status', 200) </pre>		
	test/api/quantityApiSpec.ts (Line 173)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication





<div> <div>■ Security Hotspot Code:</div> <pre> 170: headers: jsonHeader, 171: body: { 172: email: `accountant@\${config.get<string>('application.domain')}`, >>> 173: password: 'i am an awesome accountant' 174: } 175: }) 176: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/quantityApiSpec.ts (Line 190)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 187: headers: jsonHeader, 188: body: { 189: email: `accountant@\${config.get<string>('application.domain')}`, >>> 190: password: 'i am an awesome accountant' 191: } 192: }) 193: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/quantityApiSpec.ts (Line 207)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 204: headers: jsonHeader, 205: body: { 206: email: `jim@\${config.get<string>('application.domain')}`, >>> 207: password: 'ncc-1701' 208: } 209: }) 210: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/quantityApiSpec.ts (Line 228)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 225: headers: jsonHeader, 226: body: { 227: email: `jim@\${config.get<string>('application.domain')}`, >>> 228: password: 'ncc-1701' 229: } 230: }) 231: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/quantityApiSpec.ts (Line 249)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<div> <div>■ Security Hotspot Code:</div> <pre> 246: headers: jsonHeader, 247: body: { 248: email: `accountant@\${config.get<string>('application.domain')}`, >>> 249: password: 'i am an awesome accountant' 250: } 251: }) 252: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/quantityApiSpec.ts (Line 269)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 266: headers: jsonHeader, 267: body: { 268: email: `accountant@\${config.get<string>('application.domain')}`, >>> 269: password: 'i am an awesome accountant' 270: } 271: }) 272: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/quantityApiSpec.ts (Line 292)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 289: headers: jsonHeader, 290: body: { 291: email: `accountant@\${config.get<string>('application.domain')}`, >>> 292: password: 'i am an awesome accountant' 293: } 294: }) 295: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/quantityApiSpec.ts (Line 309)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 306: headers: jsonHeader, 307: body: { 308: email: `admin@\${config.get<string>('application.domain')}`, >>> 309: password: 'admin123' 310: } 311: }) 312: .expect('status', 200) </pre> </div>		
<div>H</div>	test/api/quantityApiSpec.ts (Line 326)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 323: headers: jsonHeader, 324: body: { 325: email: `jim\${config.get<string>('application.domain')}`, >>> 326: password: 'ncc-1701' 327: } 328: }) 329: .expect('status', 200) </pre>		
H	test/api/securityAnswerApiSpec.ts (Line 44)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 41: it('POST security answer for a newly registered user', () => { 42: return frisby.post(`\${API_URL}/Users`, { 43: email: 'new.user@te.st', >>> 44: password: '12345' 45: }, { json: true }) 46: .expect('status', 201) 47: .then(({ json }) => { </pre>		
H	test/api/userApiSpec.ts (Line 45)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 42: headers: jsonHeader, 43: body: { 44: email: 'horst@horstma.nn', >>> 45: password: 'hooooorst' 46: } 47: }) 48: .expect('status', 201) </pre>		
H	test/api/userApiSpec.ts (Line 63)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 60: headers: jsonHeader, 61: body: { 62: email: 'horst2@horstma.nn', >>> 63: password: 'hooooorst', 64: role: 'admin' 65: } 66: }) </pre>		
H	test/api/userApiSpec.ts (Line 85)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<div> <div>■ Security Hotspot Code:</div> <pre> 82: headers: jsonHeader, 83: body: { 84: email: ' ', >>> 85: password: ' ' 86: } 87: }) 88: .expect('status', 201) </pre> </div>		
<div>H</div>	test/api/userApiSpec.ts (Line 103)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 100: headers: jsonHeader, 101: body: { 102: email: ' ', >>> 103: password: ' ' 104: } 105: }).post(`\${API_URL}/Users`, { 106: headers: jsonHeader, </pre> </div>		
<div>H</div>	test/api/userApiSpec.ts (Line 109)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 106: headers: jsonHeader, 107: body: { 108: email: ' ', >>> 109: password: ' ' 110: } 111: }) 112: .expect('status', 400) </pre> </div>		
<div>H</div>	test/api/userApiSpec.ts (Line 121)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 118: headers: jsonHeader, 119: body: { 120: email: ' test@gmail.com', >>> 121: password: ' test' 122: } 123: }) 124: .expect('status', 201) </pre> </div>		
<div>H</div>	test/api/userApiSpec.ts (Line 139)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 136: headers: jsonHeader, 137: body: { 138: email: 'horst3@horstma.nn', >>> 139: password: 'hooooorst', 140: role: 'deluxe' 141: } 142: }) </pre>		
H	test/api/userApiSpec.ts (Line 161)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 158: headers: jsonHeader, 159: body: { 160: email: 'horst4@horstma.nn', >>> 161: password: 'hooooorst', 162: role: 'accounting' 163: } 164: }) </pre>		
H	test/api/userApiSpec.ts (Line 183)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 180: headers: jsonHeader, 181: body: { 182: email: 'horst5@horstma.nn', >>> 183: password: 'hooooorst', 184: role: 'accountinguser' 185: } 186: }) </pre>		
H	test/api/userApiSpec.ts (Line 202)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 199: headers: jsonHeader, 200: body: { 201: email: '<iframe src="javascript:alert(`xss`)">', >>> 202: password: 'does.not.matter' 203: } 204: }) 205: .expect('status', 201) </pre>		
H	test/api/userApiSpec.ts (Line 256)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication




<p>■ Security Hotspot Code:</p> <pre> 253: headers: jsonHeader, 254: body: { 255: email: 'bjoern.kimminich@gmail.com', >>> 256: password: 'bW9jLmxpYWlnQGhjaW5pbWlpay5ucmVvamI=' 257: } 258: }) 259: .expect('status', 200) </pre>		
	test/api/userProfileSpec.ts (Line 19)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 16: headers: jsonHeader, 17: body: { 18: email: 'jim@juice-sh.op', >>> 19: password: 'ncc-1701' 20: } 21: }) 22: .expect('status', 200) </pre>		
	test/api/walletApiSpec.ts (Line 18)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 15: headers: jsonHeader, 16: body: { 17: email: 'demo', >>> 18: password: 'demo' 19: } 20: }) 21: .expect('status', 200) </pre>		
	test/cypress/e2e/administration.spec.ts (Line 5)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 2: beforeEach(() => { 3: cy.login({ 4: email: 'admin', >>> 5: password: 'admin123' 6: }) 7: }) 8: describe('challenge "adminSection"', () => { </pre>		
	test/cypress/e2e/b2bOrder.spec.ts (Line 6)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

	<p>■ Security Hotspot Code:</p> <pre> 3: it('an infinite loop deserialization payload should not bring down the server', () => { 4: cy.task('isDocker').then((isDocker) => { 5: if (!isDocker) { >>> 6: cy.login({ email: 'admin', password: 'admin123' }) 7: 8: cy.window().then(async () => { 9: const response = await fetch(</pre>	
<div>H</div>	<div>test/cypress/e2e/b2bOrder.spec.ts (Line 37)</div> <div> typescript:S2068 Review this potentially hard-coded password. Category: Authentication </div>	
	<p>■ Security Hotspot Code:</p> <pre> 34: it('should be possible to cause request timeout using a recursive regular expression payload', () => { 35: cy.task('isDocker').then((isDocker) => { 36: if (!isDocker) { >>> 37: cy.login({ email: 'admin', password: 'admin123' }) 38: cy.window().then(async () => { 39: const response = await fetch(40: `\${Cypress.config('baseUrl')}/b2b/v2/orders/`, </pre>	
<div>H</div>	<div>test/cypress/e2e/basket.spec.ts (Line 4)</div> <div> typescript:S2068 Review this potentially hard-coded password. Category: Authentication </div>	
	<p>■ Security Hotspot Code:</p> <pre> 1: describe('/#/basket', () => { 2: describe('as admin', () => { 3: beforeEach(() => { >>> 4: cy.login({ email: 'admin', password: 'admin123' }) 5: }) 6: 7: describe('challenge "negativeOrder"', () => { </pre>	
<div>H</div>	<div>test/cypress/e2e/basket.spec.ts (Line 76)</div> <div> typescript:S2068 Review this potentially hard-coded password. Category: Authentication </div>	
	<p>■ Security Hotspot Code:</p> <pre> 73: 74: describe('as jim', () => { 75: beforeEach(() => { >>> 76: cy.login({ email: 'jim', password: 'ncc-1701' }) 77: }) 78: describe('challenge "manipulateClock"', () => { 79: it('should be possible to enter WMNSDY2019 coupon & place order with this expired coupon', () => { </pre>	
<div>H</div>	<div>test/cypress/e2e/changePassword.spec.ts (Line 6)</div> <div> typescript:S2068 Review this potentially hard-coded password. Category: Authentication </div>	

<p>■ Security Hotspot Code:</p> <pre> 3: beforeEach(() => { 4: cy.login({ 5: email: 'morty', >>> 6: password: 'focusOnScienceMorty!focusOnScience' 7: }) 8: cy.visit('/#/privacy-security/change-password') 9: }) </pre>		
<div>H</div>	test/cypress/e2e/changePassword.spec.ts (Line 25)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 22: it('should be able to change password via XSS-powered attack on password change without passing current password', () => { 23: cy.login({ 24: email: 'bender', >>> 25: password: 'OhG0dPleaseInsertLiquor!' 26: }) 27: cy.visit(28: '/#/search?q=%3Ciframe%20src%3D%22javascript%3Axmlhttp%20%3D%20new%20XMLHttpRequest%28%29%3B%20xmlhttp.open%28%27GET%27%2C%20%27http%3A%2F%2Flocalhost%3A3000%2Frest%2Fuser%2Fchange-password%3Fnew%3DslurmCl4ssic%26amp%3Brepeat%3DslurmCl4ssic%27%29%3B%20xmlhttp.setRequestHeader%28%27Authorization%27%2C%60Bearer%3D%24%7BlocalStorage.getItem%28%27token%27%29%7D%60%29%3B%20xmlhttp.send%28%29%3B%22%3E' </pre>		
<div>H</div>	test/cypress/e2e/changePassword.spec.ts (Line 31)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 28: '/#/search?q=%3Ciframe%20src%3D%22javascript%3Axmlhttp%20%3D%20new%20XMLHttpRequest%28%29%3B%20xmlhttp.open%28%27GET%27%2C%20%27http%3A%2F%2Flocalhost%3A3000%2Frest%2Fuser%2Fchange-password%3Fnew%3DslurmCl4ssic%26amp%3Brepeat%3DslurmCl4ssic%27%29%3B%20xmlhttp.setRequestHeader%28%27Authorization%27%2C%60Bearer%3D%24%7BlocalStorage.getItem%28%27token%27%29%7D%60%29%3B%20xmlhttp.send%28%29%3B%22%3E' 29:) 30: cy.wait(2000) >>> 31: cy.login({ email: 'bender', password: 'slurmCl4ssic' }) 32: cy.url().should('match', /\//search/) 33: 34: cy.expectChallengeSolved({ challenge: "Change Bender's Password" }) </pre>		
<div>H</div>	test/cypress/e2e/chatbot.spec.ts (Line 3)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 1: describe('/chatbot', () => { 2: beforeEach(() => { >>> 3: cy.login({ email: 'admin', password: 'admin123' }) 4: }) 5: 6: describe('challenge "killChatbot"', () => { </pre>		

<div>H</div>	test/cypress/e2e/complain.spec.ts (Line 5)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 2: beforeEach(() => { 3: cy.login({ 4: email: 'admin', >>> 5: password: 'admin123' 6: }) 7: 8: cy.visit('/#/complain') </pre> </div>		
<div>H</div>	test/cypress/e2e/contact.spec.ts (Line 11)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 8: 9: describe('challenge "forgedFeedback"', () => { 10: beforeEach(() => { >>> 11: cy.login({ email: 'admin', password: 'admin123' }) 12: cy.visit('/#/contact') 13: solveNextCaptcha() 14: }) </pre> </div>		
<div>H</div>	test/cypress/e2e/contact.spec.ts (Line 47)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 44: 45: describe('challenge "persistedXssFeedback"', () => { 46: beforeEach(() => { >>> 47: cy.login({ email: 'admin', password: 'admin123' }) 48: cy.visit('/#/contact') 49: solveNextCaptcha() 50: }) </pre> </div>		
<div>H</div>	test/cypress/e2e/dataErasure.spec.ts (Line 3)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 1: describe('/dataaerasure', () => { 2: beforeEach(() => { >>> 3: cy.login({ email: 'admin', password: 'admin123' }) 4: }) 5: 6: describe('challenge "lfr"', () => { </pre> </div>		
<div>H</div>	test/cypress/e2e/dataExport.spec.ts (Line 24)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 21: }) 22: 23: it('should be possible to steal admin user data by causing email clash during export', () => { >>> 24: cy.login({ email: 'admun', password: 'admun123' }) 25: 26: cy.visit('/#/privacy-security/data-export') 27: cy.get('#formatControl').contains('JSON').click() </pre>		
<div>H</div>	test/cypress/e2e/deluxe.spec.ts (Line 4)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 1: describe('/#/deluxe-membership', () => { 2: describe('challenge "svgInjection"', () => { 3: it('should be possible to pass in a forgotten test parameter abusing the redirect-endpoint to load an external image', () => { >>> 4: cy.login({ email: 'jim', password: 'ncc-1701' }) 5: cy.location().then((loc) => { 6: cy.visit(7: `/#/deluxe-membership?testDecal=\${encodeURIComponent(</pre>		
<div>H</div>	test/cypress/e2e/deluxe.spec.ts (Line 21)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 18: it('should upgrade to deluxe for free by making a post request to /rest/deluxe-membership by setting the paymentMode parameter to null', () => { 19: cy.login({ 20: email: 'jim', >>> 21: password: 'ncc-1701' 22: }) 23: cy.visit('/#/') 24: cy.getCookie('token').then((token) => { </pre>		
<div>H</div>	test/cypress/e2e/noSql.spec.ts (Line 8)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 5: 6: describe('challenge "NoSQL DoS"', () => { 7: beforeEach(() => { >>> 8: cy.login({ email: 'admin', password: 'admin123' }) 9: }) 10: it('should be possible to inject a command into the get route', () => { 11: cy.task('isDocker').then((isDocker) => { </pre>		
<div>H</div>	test/cypress/e2e/noSql.spec.ts (Line 53)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 50: 51: describe('challenge "NoSQL Manipulation"', () => { 52: beforeEach(() => { >>> 53: cy.login({ email: 'admin', password: 'admin123' }) 54: }) 55: 56: it('should be possible to inject a selector into the update route', () => { </pre>		
	test/cypress/e2e/noSql.spec.ts (Line 76)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 73: 74: describe('challenge "Forged Review"', () => { 75: beforeEach(() => { >>> 76: cy.login({ email: 'mc.safesearch', password: 'Mr. N00dles' }) 77: }) 78: 79: it('should be possible to edit any existing review', () => { </pre>		
	test/cypress/e2e/noSql.spec.ts (Line 120)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 117: 118: describe('challenge "Multiple Likes"', () => { 119: beforeEach(() => { >>> 120: cy.login({ email: 'mc.safesearch', password: 'Mr. N00dles' }) 121: }) 122: 123: it('should be possible to like reviews multiple times', () => { </pre>		
	test/cypress/e2e/profile.spec.ts (Line 3)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 1: describe('/profile', () => { 2: beforeEach(() => { >>> 3: cy.login({ email: 'admin', password: 'admin123' }) 4: }) 5: describe('challenge "ssrf"', () => { 6: it('should be possible to request internal resources using image upload URL', () => { { </pre>		
	test/cypress/e2e/register.spec.ts (Line 10)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 7: beforeEach(() => { 8: cy.login({ 9: email: 'admin', >>> 10: password: 'admin123' 11: }) 12: }) 13: </pre>		
H	test/cypress/e2e/register.spec.ts (Line 28)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 25: }, 26: body: JSON.stringify({ 27: email: '<iframe src="javascript:alert(`xss`)">', >>> 28: password: 'XSSed', 29: passwordRepeat: 'XSSed', 30: role: 'admin' 31: }) </pre>		
H	test/cypress/e2e/register.spec.ts (Line 29)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 26: body: JSON.stringify({ 27: email: '<iframe src="javascript:alert(`xss`)">', 28: password: 'XSSed', >>> 29: passwordRepeat: 'XSSed', 30: role: 'admin' 31: }) 32: } </pre>		
H	test/cypress/e2e/register.spec.ts (Line 60)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 57: }, 58: body: JSON.stringify({ 59: email: 'testing@test.com', >>> 60: password: 'pwned', 61: passwordRepeat: 'pwned', 62: role: 'admin' 63: }) </pre>		
H	test/cypress/e2e/register.spec.ts (Line 61)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<div> <div>■ Security Hotspot Code:</div> <pre> 58: body: JSON.stringify({ 59: email: 'testing@test.com', 60: password: 'pwned', >>> 61: passwordRepeat: 'pwned', 62: role: 'admin' 63: }) 64: }) </pre> </div>		
<div>H</div>	test/cypress/e2e/register.spec.ts (Line 84)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 81: }, 82: body: JSON.stringify({ 83: email: 'uncle@bob.com', >>> 84: password: 'ThereCanBeOnlyOne' 85: }) 86: }) 87: if (response.status === 201) { </pre> </div>		
<div>H</div>	test/cypress/e2e/restApi.spec.ts (Line 4)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 1: describe('/api', () => { 2: describe('challenge "restfulXss"', () => { 3: beforeEach(() => { >>> 4: cy.login({ email: 'admin', password: 'admin123' }) 5: }) 6: 7: // Cypress alert bug </pre> </div>		
<div>H</div>	test/cypress/e2e/restApi.spec.ts (Line 82)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<div> <div>■ Security Hotspot Code:</div> <pre> 79: beforeEach(() => { 80: cy.login({ 81: email: 'admin', >>> 82: password: 'admin123' 83: }) 84: }) 85: </pre> </div>		
<div>H</div>	test/cypress/e2e/search.spec.ts (Line 56)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication

<p>■ Security Hotspot Code:</p> <pre> 53: beforeEach(() => { 54: cy.login({ 55: email: 'admin', >>> 56: password: 'admin123' 57: }) 58: }) 59: </pre>		
H	test/cypress/e2e/search.spec.ts (Line 83)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 80: beforeEach(() => { 81: cy.login({ 82: email: 'admin', >>> 83: password: 'admin123' 84: }) 85: }) 86: </pre>		
H	test/cypress/e2e/totpSetup.spec.ts (Line 6)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 3: beforeEach(() => { 4: cy.login({ 5: email: 'wurstbrot', >>> 6: password: 'EinBelegtesBrotMitSchinkenSCHINKEN!', 7: totpSecret: 'IFTXE3SPOEYVURT2MRYGI52TKJ4HC3KH' 8: }) 9: }) </pre>		
H	test/cypress/e2e/totpSetup.spec.ts (Line 20)	typescript:S2068 Review this potentially hard-coded password. Category: Authentication
<p>■ Security Hotspot Code:</p> <pre> 17: beforeEach(() => { 18: cy.login({ 19: email: 'amy', >>> 20: password: 'Klf.....' 21: }) 22: }) 23: </pre>		
H	Dockerfile (Line 38)	docker:S6504 Make sure no write permissions are assigned to the copied resource. Category: Authentication

■ Security Hotspot Code:

```

35:     org.opencontainers.image.revision=$VCS_REF \
36:     org.opencontainers.image.created=$BUILD_DATE
37: WORKDIR /juice-shop
>>> 38: COPY --from=installer --chown=65532:0 /juice-shop .
39: USER 65532
40: EXPOSE 3000
41: CMD ["/juice-shop/build/app.js"]

```

Denial of Service (DoS) (10 hotspots)

Risk	File Path	Rule & Message
M	server.ts (Line 682)	typescript:S5693 Make sure the content length limit is safe here. Category: Denial of Service (DoS)
<h3>■ Security Hotspot Code:</h3> <pre> 679: 'image/jpeg': 'jpg', 680: 'image/jpg': 'jpg' 681: } >>> 682: const uploadToDisk = multer({ 683: storage: multer.diskStorage({ 684: destination: (req: Request, file: any, cb: any) => { 685: const isValid = mimeTypeMap[file.mimetype] </pre>		
M	frontend/src/app/change-password/change-password.component.ts (Line 94)	typescript:S5852 Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service. Category: Denial of Service (DoS)
<h3>■ Security Hotspot Code:</h3> <pre> 91: if (92: localStorage.getItem('email')?.match(/support@.*/) && 93: !this.newPasswordControl.value.match(>>> 94: /(?!.*[a-z])(?!.*[A-Z])(?!.*\d)(?!.*[@\$_!*?&])[A-Za-z\d@\$_!*?&]{12,30}/ 95:) 96:) { 97: console.error(</pre>		
M	lib/codingChallenges.ts (Line 66)	typescript:S5852 Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service. Category: Denial of Service (DoS)

M	routes/profileImageUrlUpload.ts (Line 20)	typescript:S5852 Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service. Category: Denial of Service (DoS)
■ Security Hotspot Code: <pre> 17: return async (req: Request, res: Response, next: NextFunction) => { 18: if (req.body.imageUrl !== undefined) { 19: const url = req.body.imageUrl >>> 20: if (url.match(/(.)*solve\/challenges\/server-side(.)*\/) !== null) req.app.locals.abused_ssrf_bug = true 21: const loggedInUser = security.authenticatedUsers.get(req.cookies.token) 22: if (loggedInUser) { 23: try { </pre>		
M	server.ts (Line 246)	typescript:S5852 Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service. Category: Denial of Service (DoS)
■ Security Hotspot Code: <pre> 243: // @ts-expect-error FIXME assignment broken due to seemingly void return value 244: res.end = function () { 245: if (arguments.length) { >>> 246: const reqPath = req.originalUrl.replace(/\?.*\$/, '') 247: // eslint-disable-next-line @typescript-eslint/no-non-null-assertion 248: const currentFolder = reqPath.split('/').pop()! 249: arguments[0] = arguments[0].replace(/a href="([^\"]+?)"\/gi, function (matchString: string, matchedUrl: string) { </pre>		
M	test/api/metricsApiSpec.ts (Line 18)	typescript:S5852 Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service. Category: Denial of Service (DoS)
■ Security Hotspot Code: <pre> 15: return frisby.get(API_URL) 16: .expect('status', 200) 17: .expect('header', 'content-type', /text\/plain/) >>> 18: .expect('bodyContains', /^.*_version_info{version="[0-9]+\.[0-9]+\.[0-9]+(-SNAPSHO T)?",major="[0-9]+",minor="[0-9]+",patch="[0-9]+",app=".*"} 1\$/gm) 19: .expect('bodyContains', /^.*_challenges_solved{difficulty="[1-6]",category=".*",app=".*"} [0-9]*\$/gm) 20: .expect('bodyContains', /^.*_challenges_total{difficulty="[1-6]",category=".*",app=".*"} [0-9]*\$/gm) 21: .expect('bodyContains', /^.*_cheat_score{app=".*"} [0-9.]*\$/gm) </pre>		
M	test/cypress/support/commands.ts (Line 36)	typescript:S5852 Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service. Category: Denial of Service (DoS)

■ Security Hotspot Code:

```

33:   'login',
34:   (context: { email: string, password: string, totpSecret?: string }) => {
35:     cy.visit('/#/login')
>>> 36:     if (context.email.match(/\S+@\S+\.\S+/) != null) {
37:       cy.get('#email').type(context.email)
38:     } else {
39:       cy.task<string>('GetFromConfig', 'application.domain').then(

```

Encrypt Data (42 hotspots)

Risk	File Path	Rule & Message
L	data/static/codefixes redirectChallenge_1.ts (Line 6)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<h4>■ Security Hotspot Code:</h4> <pre> 3: 'https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm', 4: 'https://explorer.dash.org/address/Xr556RzuwX6hg5EGpkybbv5RanJoZN17kW', 5: 'https://etherscan.io/address/0x0f933ab9fcaaa782d0279c300d73750e1311eae6', >>> 6: 'http://shop.spreadshirt.com/juiceshop', 7: 'http://shop.spreadshirt.de/juiceshop', 8: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 9: 'http://leanpub.com/juice-shop' </pre>		
L	data/static/codefixes redirectChallenge_1.ts (Line 7)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<h4>■ Security Hotspot Code:</h4> <pre> 4: 'https://explorer.dash.org/address/Xr556RzuwX6hg5EGpkybbv5RanJoZN17kW', 5: 'https://etherscan.io/address/0x0f933ab9fcaaa782d0279c300d73750e1311eae6', 6: 'http://shop.spreadshirt.com/juiceshop', >>> 7: 'http://shop.spreadshirt.de/juiceshop', 8: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 9: 'http://leanpub.com/juice-shop' 10:]} </pre>		
L	data/static/codefixes redirectChallenge_1.ts (Line 9)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<h4>■ Security Hotspot Code:</h4> <pre> 6: 'http://shop.spreadshirt.com/juiceshop', 7: 'http://shop.spreadshirt.de/juiceshop', 8: 'https://www.stickeryou.com/products/owasp-juice-shop/794', >>> 9: 'http://leanpub.com/juice-shop' 10:]} 11: 12: export const isRedirectAllowed = (url: string) => { </pre>		

<div>L</div>	data/static/codefixes redirectChallenge_2.ts (Line 6)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<div> <div>■ Security Hotspot Code:</div> <pre> 3: 'https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm', 4: 'https://explorer.dash.org/address/Xr556RzuwX6hg5EGpkybbv5RanJoZN17kW', 5: 'https://etherscan.io/address/0x0f933ab9fcaaa782d0279c300d73750e1311eae6', >>> 6: 'http://shop.spreadshirt.com/juiceshop', 7: 'http://shop.spreadshirt.de/juiceshop', 8: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 9: 'http://leanpub.com/juice-shop' </pre> </div>		
<div>L</div>	data/static/codefixes redirectChallenge_2.ts (Line 7)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<div> <div>■ Security Hotspot Code:</div> <pre> 4: 'https://explorer.dash.org/address/Xr556RzuwX6hg5EGpkybbv5RanJoZN17kW', 5: 'https://etherscan.io/address/0x0f933ab9fcaaa782d0279c300d73750e1311eae6', 6: 'http://shop.spreadshirt.com/juiceshop', >>> 7: 'http://shop.spreadshirt.de/juiceshop', 8: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 9: 'http://leanpub.com/juice-shop' 10:]} </pre> </div>		
<div>L</div>	data/static/codefixes redirectChallenge_2.ts (Line 9)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<div> <div>■ Security Hotspot Code:</div> <pre> 6: 'http://shop.spreadshirt.com/juiceshop', 7: 'http://shop.spreadshirt.de/juiceshop', 8: 'https://www.stickeryou.com/products/owasp-juice-shop/794', >>> 9: 'http://leanpub.com/juice-shop' 10:]} 11: 12: export const isRedirectAllowed = (url: string) => { </pre> </div>		
<div>L</div>	data/static/codefixes redirectChallenge_3.ts (Line 6)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<div> <div>■ Security Hotspot Code:</div> <pre> 3: 'https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm', 4: 'https://explorer.dash.org/address/Xr556RzuwX6hg5EGpkybbv5RanJoZN17kW', 5: 'https://etherscan.io/address/0x0f933ab9fcaaa782d0279c300d73750e1311eae6', >>> 6: 'http://shop.spreadshirt.com/juiceshop', 7: 'http://shop.spreadshirt.de/juiceshop', 8: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 9: 'http://leanpub.com/juice-shop' </pre> </div>		
<div>L</div>	data/static/codefixes redirectChallenge_3.ts (Line 7)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data

<div> <div>■ Security Hotspot Code:</div> <pre> 6: 'http://shop.spreadshirt.com/juiceshop', 7: 'http://shop.spreadshirt.de/juiceshop', 8: 'https://www.stickeryou.com/products/owasp-juice-shop/794', >>> 9: 'http://leanpub.com/juice-shop' 10:]) 11: 12: export const isRedirectAllowed = (url: string) => { </pre> </div>		
<div> <div>L</div> </div>	<div> <div>data/static/codefixes</div> <div>redirectCryptoCurrencyChallenge_1</div> <div>.ts</div> <div>(Line 5)</div> </div>	<div> <div>typescript:S5332</div> <div>Using http protocol is insecure. Use https instead.</div> <div>Category: Encrypt Data</div> </div>
<div> <div>■ Security Hotspot Code:</div> <pre> 2: 'https://github.com/juice-shop/juice-shop', 3: 'https://explorer.dash.org/address/Xr556RzuwX6hg5EGpkybbv5RanJoZN17kW', 4: 'https://etherscan.io/address/0x0f933ab9fcaaa782d0279c300d73750e1311eae6', >>> 5: 'http://shop.spreadshirt.com/juiceshop', 6: 'http://shop.spreadshirt.de/juiceshop', 7: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 8: 'http://leanpub.com/juice-shop' </pre> </div>		
<div> <div>L</div> </div>	<div> <div>data/static/codefixes</div> <div>redirectCryptoCurrencyChallenge_1</div> <div>.ts</div> <div>(Line 6)</div> </div>	<div> <div>typescript:S5332</div> <div>Using http protocol is insecure. Use https instead.</div> <div>Category: Encrypt Data</div> </div>
<div> <div>■ Security Hotspot Code:</div> <pre> 3: 'https://explorer.dash.org/address/Xr556RzuwX6hg5EGpkybbv5RanJoZN17kW', 4: 'https://etherscan.io/address/0x0f933ab9fcaaa782d0279c300d73750e1311eae6', 5: 'http://shop.spreadshirt.com/juiceshop', >>> 6: 'http://shop.spreadshirt.de/juiceshop', 7: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 8: 'http://leanpub.com/juice-shop' 9:]) </pre> </div>		
<div> <div>L</div> </div>	<div> <div>data/static/codefixes</div> <div>redirectCryptoCurrencyChallenge_1</div> <div>.ts</div> <div>(Line 8)</div> </div>	<div> <div>typescript:S5332</div> <div>Using http protocol is insecure. Use https instead.</div> <div>Category: Encrypt Data</div> </div>
<div> <div>■ Security Hotspot Code:</div> <pre> 5: 'http://shop.spreadshirt.com/juiceshop', 6: 'http://shop.spreadshirt.de/juiceshop', 7: 'https://www.stickeryou.com/products/owasp-juice-shop/794', >>> 8: 'http://leanpub.com/juice-shop' 9:]) 10: 11: export const isRedirectAllowed = (url: string) => { </pre> </div>		
<div> <div>L</div> </div>	<div> <div>data/static/codefixes</div> <div>redirectCryptoCurrencyChallenge_2</div> <div>.ts</div> <div>(Line 5)</div> </div>	<div> <div>typescript:S5332</div> <div>Using http protocol is insecure. Use https instead.</div> <div>Category: Encrypt Data</div> </div>

<div> <div>■ Security Hotspot Code:</div> <pre> 2: 'https://github.com/juice-shop/juice-shop', 3: 'https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm', 4: 'https://etherscan.io/address/0x0f933ab9fcaaa782d0279c300d73750e1311eae6', >>> 5: 'http://shop.spreadshirt.com/juiceshop', 6: 'http://shop.spreadshirt.de/juiceshop', 7: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 8: 'http://leanpub.com/juice-shop' </pre> </div>		
<div> <div>L</div> </div>	<div> <div>data/static/codefixes</div> <div>redirectCryptoCurrencyChallenge_2</div> <div>.ts</div> <div>(Line 6)</div> </div>	<div> <div>typescript:S5332</div> <div>Using http protocol is insecure. Use https instead.</div> <div>Category: Encrypt Data</div> </div>
<div> <div>■ Security Hotspot Code:</div> <pre> 3: 'https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm', 4: 'https://etherscan.io/address/0x0f933ab9fcaaa782d0279c300d73750e1311eae6', 5: 'http://shop.spreadshirt.com/juiceshop', >>> 6: 'http://shop.spreadshirt.de/juiceshop', 7: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 8: 'http://leanpub.com/juice-shop' 9:]} </pre> </div>		
<div> <div>L</div> </div>	<div> <div>data/static/codefixes</div> <div>redirectCryptoCurrencyChallenge_2</div> <div>.ts</div> <div>(Line 8)</div> </div>	<div> <div>typescript:S5332</div> <div>Using http protocol is insecure. Use https instead.</div> <div>Category: Encrypt Data</div> </div>
<div> <div>■ Security Hotspot Code:</div> <pre> 5: 'http://shop.spreadshirt.com/juiceshop', 6: 'http://shop.spreadshirt.de/juiceshop', 7: 'https://www.stickeryou.com/products/owasp-juice-shop/794', >>> 8: 'http://leanpub.com/juice-shop' 9:]} 10: 11: export const isRedirectAllowed = (url: string) => { </pre> </div>		
<div> <div>L</div> </div>	<div> <div>data/static/codefixes</div> <div>redirectCryptoCurrencyChallenge_3</div> <div>_correct.ts</div> <div>(Line 3)</div> </div>	<div> <div>typescript:S5332</div> <div>Using http protocol is insecure. Use https instead.</div> <div>Category: Encrypt Data</div> </div>
<div> <div>■ Security Hotspot Code:</div> <pre> 1: export const redirectAllowlist = new Set([2: 'https://github.com/juice-shop/juice-shop', >>> 3: 'http://shop.spreadshirt.com/juiceshop', 4: 'http://shop.spreadshirt.de/juiceshop', 5: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 6: 'http://leanpub.com/juice-shop' </pre> </div>		
<div> <div>L</div> </div>	<div> <div>data/static/codefixes</div> <div>redirectCryptoCurrencyChallenge_3</div> <div>_correct.ts</div> <div>(Line 4)</div> </div>	<div> <div>typescript:S5332</div> <div>Using http protocol is insecure. Use https instead.</div> <div>Category: Encrypt Data</div> </div>

<p>■ Security Hotspot Code:</p> <pre> 1: export const redirectAllowlist = new Set([2: 'https://github.com/juice-shop/juice-shop', 3: 'http://shop.spreadshirt.com/juiceshop', >>> 4: 'http://shop.spreadshirt.de/juiceshop', 5: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 6: 'http://leanpub.com/juice-shop' 7:])</pre>		
L	data/static/codefixes redirectCryptoCurrencyChallenge_3 _correct.ts (Line 6)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<p>■ Security Hotspot Code:</p> <pre> 3: 'http://shop.spreadshirt.com/juiceshop', 4: 'http://shop.spreadshirt.de/juiceshop', 5: 'https://www.stickeryou.com/products/owasp-juice-shop/794', >>> 6: 'http://leanpub.com/juice-shop' 7:])</pre> <pre> 8: 9: export const isRedirectAllowed = (url: string) => {</pre>		
L	data/static/codefixes redirectCryptoCurrencyChallenge_4 .ts (Line 5)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<p>■ Security Hotspot Code:</p> <pre> 2: 'https://github.com/juice-shop/juice-shop', 3: 'https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm', 4: 'https://explorer.dash.org/address/Xr556RzuwX6hg5EGpkybbv5RanJoZN17kW', >>> 5: 'http://shop.spreadshirt.com/juiceshop', 6: 'http://shop.spreadshirt.de/juiceshop', 7: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 8: 'http://leanpub.com/juice-shop'</pre>		
L	data/static/codefixes redirectCryptoCurrencyChallenge_4 .ts (Line 6)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<p>■ Security Hotspot Code:</p> <pre> 3: 'https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm', 4: 'https://explorer.dash.org/address/Xr556RzuwX6hg5EGpkybbv5RanJoZN17kW', 5: 'http://shop.spreadshirt.com/juiceshop', >>> 6: 'http://shop.spreadshirt.de/juiceshop', 7: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 8: 'http://leanpub.com/juice-shop' 9:])</pre>		
L	data/static/codefixes redirectCryptoCurrencyChallenge_4 .ts (Line 8)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data

<p>■ Security Hotspot Code:</p> <pre> 5: 'http://shop.spreadshirt.com/juiceshop', 6: 'http://shop.spreadshirt.de/juiceshop', 7: 'https://www.stickeryou.com/products/owasp-juice-shop/794', >>> 8: 'http://leanpub.com/juice-shop' 9: }) 10: 11: export const isRedirectAllowed = (url: string) => { </pre>		
<p>L</p>	<p>frontend/src/app/order-completion order-completion.component.spec.ts (Line 136)</p>	<p>typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data</p>
<p>■ Security Hotspot Code:</p> <pre> 133: 134: it('should use configured URL as is if it is not a twitter URL', () => { 135: trackOrderService.find.and.returnValue(of({ data: [{ products: [] }] })) >>> 136: configurationService.getApplicationConfiguration.and.returnValue(of({ application: { social: { twitterUrl: 'http://localhost:42' } } })) 137: component.ngOnInit() 138: expect(component.tweetText).toBe('I just purchased%0afrom http://localhost:42') 139: }) </pre>		
<p>L</p>	<p>lib/insecurity.ts (Line 129)</p>	<p>typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data</p>
<p>■ Security Hotspot Code:</p> <pre> 126: 'https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm', // vuln-code-snippet vuln-line redirectCryptoCurrencyChallenge 127: 'https://explorer.dash.org/address/Xr556RzuwX6hg5EGpkybbv5RanJoZN17kW', // vuln-code-snippet vuln-line redirectCryptoCurrencyChallenge 128: 'https://etherscan.io/address/0x0f933ab9fcaaa782d0279c300d73750e1311eae6', // vuln-code-snippet vuln-line redirectCryptoCurrencyChallenge >>> 129: 'http://shop.spreadshirt.com/juiceshop', 130: 'http://shop.spreadshirt.de/juiceshop', 131: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 132: 'http://leanpub.com/juice-shop' </pre>		
<p>L</p>	<p>lib/insecurity.ts (Line 130)</p>	<p>typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data</p>
<p>■ Security Hotspot Code:</p> <pre> 127: 'https://explorer.dash.org/address/Xr556RzuwX6hg5EGpkybbv5RanJoZN17kW', // vuln-code-snippet vuln-line redirectCryptoCurrencyChallenge 128: 'https://etherscan.io/address/0x0f933ab9fcaaa782d0279c300d73750e1311eae6', // vuln-code-snippet vuln-line redirectCryptoCurrencyChallenge 129: 'http://shop.spreadshirt.com/juiceshop', >>> 130: 'http://shop.spreadshirt.de/juiceshop', 131: 'https://www.stickeryou.com/products/owasp-juice-shop/794', 132: 'http://leanpub.com/juice-shop' 133: }) </pre>		

<div>L</div>	lib/insecurity.ts (Line 132)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<div> <div>■ Security Hotspot Code:</div> <pre> 129: 'http://shop.spreadshirt.com/juiceshop', 130: 'http://shop.spreadshirt.de/juiceshop', 131: 'https://www.stickeryou.com/products/owasp-juice-shop/794', >>> 132: 'http://leanpub.com/juice-shop' 133:]) 134: 135: export const isRedirectAllowed = (url: string) => { </pre> </div>		
<div>L</div>	test/cypress/e2e/profile.spec.ts (Line 74)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<div> <div>■ Security Hotspot Code:</div> <pre> 71: describe('challenge "csrf"', () => { 72: // FIXME Only works on Chrome <80 but Protractor uses latest Chrome version. Test can probably never be turned on again. 73: xit('should be possible to perform a CSRF attack against the user profile page', () => { >>> 74: cy.visit('http://htmledit.squarefree.com') 75: /* The script executed below is equivalent to pasting this string into http://htmledit.squarefree.com: */ 76: /* <form action="http://localhost:3000/profile" method="POST"><input type="hidden" name="username" value="CSRF"/><input type="submit"/></form><script>document.forms[0].submit();</script> */ 77: let document: any </pre> </div>		
<div>L</div>	test/cypress/e2e/profile.spec.ts (Line 107)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<div> <div>■ Security Hotspot Code:</div> <pre> 104: headers: { 105: 'Content-type': 'application/x-www-form-urlencoded', 106: Authorization: `Bearer \${localStorage.getItem('token')}`, >>> 107: Origin: 'http://htmledit.squarefree.com', // FIXME Not allowed by browser due to "unsafe header not permitted" 108: Cookie: `token=\${localStorage.getItem('token')}` // FIXME Not allowed by browser due to "unsafe header not permitted" 109: }, 110: body: formData </pre> </div>		
<div>L</div>	test/server/redirectSpec.ts (Line 45)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data

<p>■ Security Hotspot Code:</p> <pre> 42: }) 43: 44: it('should raise error for URL not on allowlist', () => { >>> 45: req.query.to = 'http://kimminich.de' 46: 47: performRedirect()(req, res, next) 48: </pre>		
L	test/server/redirectSpec.ts (Line 81)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<p>■ Security Hotspot Code:</p> <pre> 78: }) 79: 80: it('tricking the allowlist should solve "redirectChallenge"', () => { >>> 81: req.query.to = 'http://kimminich.de?to=https://github.com/juice-shop/juice-shop' 82: challenges.redirectChallenge = { solved: false, save } as unknown as Challenge 83: 84: performRedirect()(req, res, next) </pre>		
L	test/server/utisSpec.ts (Line 38)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<p>■ Security Hotspot Code:</p> <pre> 35: }) 36: 37: it('returns filename from http:// URL', () => { >>> 38: expect(utis.extractFilename('http://bla.blubb/test.exe')).to.equal('test.exe') 39: }) 40: 41: it('ignores query part of http:// URL', () => { </pre>		
L	test/server/utisSpec.ts (Line 42)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<p>■ Security Hotspot Code:</p> <pre> 39: }) 40: 41: it('ignores query part of http:// URL', () => { >>> 42: expect(utis.extractFilename('http://bla.blubb/test.exe?bla=blubb&a=b')).to.equal('test.exe') 43: }) 44: 45: it('also works for file:// URLs', () => { </pre>		
L	test/server/verifySpec.ts (Line 88)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data

<p>■ Security Hotspot Code:</p> <pre> 85: describe('accessControlChallenges', () => { 86: it('"scoreBoardChallenge" is solved when the lpx.png transpixel is requested', () => { 87: challenges.scoreBoardChallenge = { solved: false, save } as unknown as Challenge >>> 88: req.url = 'http://juice-sh.op/public/images/padding/lpx.png' 89: 90: verify.accessControlChallenges()(req, res, next) 91: </pre>		
L	test/server/verifySpec.ts (Line 97)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<p>■ Security Hotspot Code:</p> <pre> 94: 95: it('"adminSectionChallenge" is solved when the 19px.png transpixel is requested', () => { 96: challenges.adminSectionChallenge = { solved: false, save } as unknown as Challenge >>> 97: req.url = 'http://juice-sh.op/public/images/padding/19px.png' 98: 99: verify.accessControlChallenges()(req, res, next) 100: </pre>		
L	test/server/verifySpec.ts (Line 106)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<p>■ Security Hotspot Code:</p> <pre> 103: 104: it('"tokenSaleChallenge" is solved when the 56px.png transpixel is requested', () => { 105: challenges.tokenSaleChallenge = { solved: false, save } as unknown as Challenge >>> 106: req.url = 'http://juice-sh.op/public/images/padding/56px.png' 107: 108: verify.accessControlChallenges()(req, res, next) 109: </pre>		
L	test/server/verifySpec.ts (Line 115)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<p>■ Security Hotspot Code:</p> <pre> 112: 113: it('"extraLanguageChallenge" is solved when the Klingon translation file is requested', () => { 114: challenges.extraLanguageChallenge = { solved: false, save } as unknown as Challenge >>> 115: req.url = 'http://juice-sh.op/public/i18n/tlh_AA.json' 116: 117: verify.accessControlChallenges()(req, res, next) 118: </pre>		

L	test/server/verifySpec.ts (Line 125)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<div> <div>■ Security Hotspot Code:</div> <pre> 122: it('"retrieveBlueprintChallenge" is solved when the blueprint file is requested', () => { 123: challenges.retrieveBlueprintChallenge = { solved: false, save } as unknown as Challenge 124: setRetrieveBlueprintChallengeFile('test.dxf') >>> 125: req.url = 'http://juice-sh.op/public/images/products/test.dxf' 126: 127: verify.accessControlChallenges()(req, res, next) 128: </pre> </div>		
L	test/server/verifySpec.ts (Line 134)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<div> <div>■ Security Hotspot Code:</div> <pre> 131: 132: it('"missingEncodingChallenge" is solved when the crazy cat photo is requested', () => { 133: challenges.missingEncodingChallenge = { solved: false, save } as unknown as Challenge >>> 134: req.url = 'http://juice-sh.op/public/images/uploads/%E1%93%9A%E1%98%8F%E1%97%A2- %23zatschi-%23whoneedsfourlegs-1572600969477.jpg' 135: 136: verify.accessControlChallenges()(req, res, next) 137: </pre> </div>		
L	test/server/verifySpec.ts (Line 143)	typescript:S5332 Using http protocol is insecure. Use https instead. Category: Encrypt Data
<div> <div>■ Security Hotspot Code:</div> <pre> 140: 141: it('"accessLogDisclosureChallenge" is solved when any server access log file is requested', () => { 142: challenges.accessLogDisclosureChallenge = { solved: false, save } as unknown as Challenge >>> 143: req.url = 'http://juice-sh.op/support/logs/access.log.2019-01-15' 144: 145: verify.accessControlChallenges()(req, res, next) 146: </pre> </div>		
L	test/smoke/Dockerfile (Line 7)	docker:S5332 Make sure that using clear-text protocols is safe here. Category: Encrypt Data
<div> <div>■ Security Hotspot Code:</div> <pre> 4: 5: COPY smoke-test.sh smoke-test.sh 6: >>> 7: CMD ["sh", "smoke-test.sh", "http://app:3000"] </pre> </div>		

Insecure Configuration (2 hotspots)

Risk	File Path	Rule & Message
L	server.ts (Line 182)	typescript:S5122 Make sure that enabling CORS is safe here. Category: Insecure Configuration
■ Security Hotspot Code: <pre> 179: 180: /* Bludgeon solution for possible CORS problems: Allow everything! */ 181: app.options('*', cors()) >>> 182: app.use(cors()) 183: 184: /* Security middleware */ 185: app.use(helmet.noSniff()) </pre>		
L	server.ts (Line 671)	typescript:S4507 Make sure this debug feature is deactivated before delivering the code in production. Category: Insecure Configuration
■ Security Hotspot Code: <pre> 668: 669: /* Error Handling */ 670: app.use(verify.errorHandlingChallenge()) >>> 671: app.use(errorhandler()) 672: }).catch((err) => { 673: console.error(err) 674: }) </pre>		

Others (14 hotspots)

Risk	File Path	Rule & Message
L	lib/insecurity.ts (Line 43)	typescript:S4790 Make sure this weak hash algorithm is not used in a sensitive context here. Category: Others

<p>■ Security Hotspot Code:</p> <pre> 40: updateFrom: (req: Request, user: ResponseWithUser) => any 41: } 42: >>> 43: export const hash = (data: string) => crypto.createHash('md5').update(data).digest('hex') 44: export const hmac = (data: string) => crypto.createHmac('sha256', 'pa4qacea4VK9t9nGv7yZtwmj').update(data).digest('hex') 45: 46: export const cutOffPoisonNullByte = (str: string) => { </pre>		
L	frontend/src/index.html (Line 15)	<p>Web:S5725</p> <p>Make sure not using resource integrity feature is safe here.</p> <p>Category: Others</p>
<p>■ Security Hotspot Code:</p> <pre> 12: <meta name="viewport" content="width=device-width, initial-scale=1"> 13: <link id="favicon" rel="icon" type="image/x-icon" href="assets/public/favicon_js.ico"> 14: <link rel="stylesheet" type="text/css" href="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.css" /> >>> 15: <script src="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js"></script> 16: <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script> 17: <script> 18: window.addEventListener("load", function(){ </pre>		
L	frontend/src/index.html (Line 16)	<p>Web:S5725</p> <p>Make sure not using resource integrity feature is safe here.</p> <p>Category: Others</p>
<p>■ Security Hotspot Code:</p> <pre> 13: <link id="favicon" rel="icon" type="image/x-icon" href="assets/public/favicon_js.ico"> 14: <link rel="stylesheet" type="text/css" href="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.css" /> 15: <script src="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js"></script> >>> 16: <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script> 17: <script> 18: window.addEventListener("load", function(){ 19: window.cookieconsent.initialise({ </pre>		
L	test/api/loginApiSpec.ts (Line 253)	<p>typescript:S1313</p> <p>Make sure using a hardcoded IP address 1.2.3.4 is safe here.</p> <p>Category: Others</p>
<p>■ Security Hotspot Code:</p> <pre> 250: return frisby.get(REST_URL + '/saveLoginIp', { 251: headers: { 252: Authorization: 'Bearer ' + jsonLogin.authentication.token, >>> 253: 'true-client-ip': '1.2.3.4' 254: } 255: }) 256: .expect('status', 200) </pre>		

L	test/api/loginApiSpec.ts (Line 257)	typescript:S1313 Make sure using a hardcoded IP address 1.2.3.4 is safe here. Category: Others
<div> <div>■ Security Hotspot Code:</div> <pre> 254: } 255: }) 256: .expect('status', 200) >>> 257: .expect('json', { lastLoginIp: '1.2.3.4' }) 258: }) 259: }) 260: </pre> </div>		
L	test/server/utillsSpec.ts (Line 16)	typescript:S1313 Make sure using a hardcoded IP address 2001:0db8:85a3:0000:0000:8a2e:0370:7334 is safe here. Category: Others
<div> <div>■ Security Hotspot Code:</div> <pre> 13: describe('utils', () => { 14: describe('toSimpleIpAddress', () => { 15: it('returns ipv6 address unchanged', () => { >>> 16: expect(utils.toSimpleIpAddress('2001:0db8:85a3:0000:0000:8a2e:0370:7334')).toEqual('2001:0db8:85a3:0000:0000:8a2e:0370:7334') 17: }) 18: }) 19: it('returns ipv4 address fully specified as ipv6 unchanged', () => { </pre> </div>		
L	test/server/utillsSpec.ts (Line 16)	typescript:S1313 Make sure using a hardcoded IP address 2001:0db8:85a3:0000:0000:8a2e:0370:7334 is safe here. Category: Others
<div> <div>■ Security Hotspot Code:</div> <pre> 13: describe('utils', () => { 14: describe('toSimpleIpAddress', () => { 15: it('returns ipv6 address unchanged', () => { >>> 16: expect(utils.toSimpleIpAddress('2001:0db8:85a3:0000:0000:8a2e:0370:7334')).toEqual('2001:0db8:85a3:0000:0000:8a2e:0370:7334') 17: }) 18: }) 19: it('returns ipv4 address fully specified as ipv6 unchanged', () => { </pre> </div>		
L	test/server/utillsSpec.ts (Line 20)	typescript:S1313 Make sure using a hardcoded IP address 0:0:0:0:ffff:7f00:1 is safe here. Category: Others

<p>■ Security Hotspot Code:</p> <pre> 17: }) 18: 19: it('returns ipv4 address fully specified as ipv6 unchanged', () => { >>> 20: expect(utils.toSimpleIpAddress('0:0:0:0:ffff:7f00:1')).to.equal('0:0:0:0:ffff:7f00:1') 21: }) 22: 23: it('returns ipv6 loopback address as ipv4 address', () => { </pre>		
<p>L</p>	<p>test/server/utilsSpec.ts (Line 20)</p>	<p>typescript:S1313 Make sure using a hardcoded IP address 0:0:0:0:ffff:7f00:1 is safe here. Category: Others</p>
<p>■ Security Hotspot Code:</p> <pre> 17: }) 18: 19: it('returns ipv4 address fully specified as ipv6 unchanged', () => { >>> 20: expect(utils.toSimpleIpAddress('0:0:0:0:ffff:7f00:1')).to.equal('0:0:0:0:ffff:7f00:1') 21: }) 22: 23: it('returns ipv6 loopback address as ipv4 address', () => { </pre>		
<p>L</p>	<p>test/server/utilsSpec.ts (Line 28)</p>	<p>typescript:S1313 Make sure using a hardcoded IP address ::ffff:192.0.2.128 is safe here. Category: Others</p>
<p>■ Security Hotspot Code:</p> <pre> 25: }) 26: 27: it('returns ipv4-mapped address as ipv4 address', () => { >>> 28: expect(utils.toSimpleIpAddress('::ffff:192.0.2.128')).to.equal('192.0.2.128') 29: }) 30: }) 31: </pre>		
<p>L</p>	<p>Dockerfile (Line 4)</p>	<p>docker:S6505 Omitting "--ignore-scripts" can lead to the execution of shell scripts. Make sure it is safe here. Category: Others</p>
<p>■ Security Hotspot Code:</p> <pre> 1: FROM node:22 AS installer 2: COPY . /juice-shop 3: WORKDIR /juice-shop >>> 4: RUN npm i -g typescript ts-node 5: RUN npm install --omit=dev --unsafe-perm 6: RUN npm dedupe --omit=dev 7: RUN rm -rf frontend/node_modules </pre>		

L	Dockerfile (Line 5)	docker:S6505 Omitting "--ignore-scripts" can lead to the execution of shell scripts. Make sure it is safe here. Category: Others
<div> <div>■ Security Hotspot Code:</div> <pre> 2: COPY . /juice-shop 3: WORKDIR /juice-shop 4: RUN npm i -g typescript ts-node >>> 5: RUN npm install --omit=dev --unsafe-perm 6: RUN npm dedupe --omit=dev 7: RUN rm -rf frontend/node_modules 8: RUN rm -rf frontend/.angular </pre> </div>		
L	Dockerfile (Line 19)	docker:S6505 Omitting "--ignore-scripts" can lead to the execution of shell scripts. Make sure it is safe here. Category: Others
<div> <div>■ Security Hotspot Code:</div> <pre> 16: RUN rm il8n/*.json true 17: 18: ARG CYCLONEDX_NPM_VERSION=latest >>> 19: RUN npm install -g @cyclonedx/cyclonedx-npm@\$CYCLONEDX_NPM_VERSION 20: RUN npm run sbom 21: 22: FROM gcr.io/distroless/nodejs22-debian12 </pre> </div>		
L	Gruntfile.js (Line 77)	javascript:S4790 Make sure this weak hash algorithm is not used in a sensitive context here. Category: Others
<div> <div>■ Security Hotspot Code:</div> <pre> 74: const crypto = require('node:crypto') 75: fs.readdirSync('dist/').forEach(file => { 76: const buffer = fs.readFileSync('dist/' + file) >>> 77: const md5 = crypto.createHash('md5') 78: md5.update(buffer) 79: const md5Hash = md5.digest('hex') 80: const md5FileName = 'dist/' + file + '.md5' </pre> </div>		

Permission (2 hotspots)

Risk	File Path	Rule & Message
M	Dockerfile (Line 2)	docker:S6470 Copying recursively might inadvertently add sensitive data to the container. Make sure it is safe here. Category: Permission

■ Security Hotspot Code: <pre> 1: FROM node:22 AS installer >>> 2: COPY . /juice-shop 3: WORKDIR /juice-shop 4: RUN npm i -g typescript ts-node 5: RUN npm install --omit=dev --unsafe-perm </pre>		
M	test/smoke/Dockerfile (Line 1)	docker:S6471 The "alpine" image runs with "root" as the default user. Make sure it is safe here. Category: Permission
■ Security Hotspot Code: <pre> >>> 1: FROM alpine 2: 3: RUN apk add curl 4: </pre>		

Remote Code Execution (3 hotspots)

Risk	File Path	Rule & Message
M	routes/captcha.ts (Line 23)	typescript:S1523 Make sure that this dynamic injection or execution of code is safe. Category: Remote Code Execution
■ Security Hotspot Code: <pre> 20: const secondOperator = operators[Math.floor((Math.random() * 3))] 21: 22: const expression = firstTerm.toString() + firstOperator + secondTerm.toString() + secondOperator + thirdTerm.toString() >>> 23: const answer = eval(expression).toString() // eslint-disable-line no-eval 24: 25: const captcha = { 26: captchaId, </pre>		
M	routes/userProfile.ts (Line 62)	typescript:S1523 Make sure that this dynamic injection or execution of code is safe. Category: Remote Code Execution
■ Security Hotspot Code: <pre> 59: if (!code) { 60: throw new Error('Username is null') 61: } >>> 62: username = eval(code) // eslint-disable-line no-eval 63: } catch (err) { 64: username = '\\\\' + username 65: } </pre>		

M	test/cypress/e2e/contact.spec.ts (Line 258)	typescript:S1523 Make sure that this dynamic injection or execution of code is safe. Category: Remote Code Execution
■ Security Hotspot Code: <pre> 255: .then((val) => { 256: cy.get('#captchaControl').clear() 257: // eslint-disable-next-line no-eval >>> 258: const answer = eval(val).toString() 259: cy.get('#captchaControl').type(answer) 260: }) 261: }</pre>		

Weak Cryptography (11 hotspots)

Risk	File Path	Rule & Message
M	data/datacreator.ts (Line 247)	typescript:S2245 Make sure that using this pseudorandom number generator is safe here. Category: Weak Cryptography
■ Security Hotspot Code: <pre> 244: let text = '' 245: const possible = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789' 246: >>> 247: for (let i = 0; i < length; i++) { text += possible.charAt(Math.floor(Math.random() * possible.length)) } 248: 249: return text 250: }</pre>		
M	data/datacreator.ts (Line 265)	typescript:S2245 Make sure that using this pseudorandom number generator is safe here. Category: Weak Cryptography
■ Security Hotspot Code: <pre> 262: config.get<ProductConfig[]>('products').map(async (product, index) => { 263: return await QuantityModel.create({ 264: ProductId: index + 1, >>> 265: quantity: product.quantity ?? Math.floor(Math.random() * 70 + 30), 266: limitPerUser: product.limitPerUser ?? null 267: }).catch((err: unknown) => { 268: logger.error(`Could not create quantity: \${utils.getErrorMessage(err)}`)</pre>		
M	data/datacreator.ts (Line 323)	typescript:S2245 Make sure that using this pseudorandom number generator is safe here. Category: Weak Cryptography

<div> <div>■ Security Hotspot Code:</div> <pre> 320: 321: async function createProducts () { 322: const products = structuredClone(config.get<ProductConfig[]>('products')).map((product) => { >>> 323: product.price = product.price ?? Math.floor(Math.random() * 9 + 1) 324: product.deluxePrice = product.deluxePrice ?? product.price 325: product.description = product.description 'Lorem ipsum dolor sit amet, consectetuer adipiscing elit.' 326: </pre> </div>		
<div>M</div>	data/datacreator.ts (Line 698)	typescript:S2245 Make sure that using this pseudorandom number generator is safe here. Category: Weak Cryptography
<div> <div>■ Security Hotspot Code:</div> <pre> 695: totalPrice: basket1Products[0].total + basket1Products[1].total, 696: bonus: basket1Products[0].bonus + basket1Products[1].bonus, 697: products: basket1Products, >>> 698: eta: Math.floor((Math.random() * 5) + 1).toString(), 699: delivered: false 700: }, 701: { </pre> </div>		
<div>M</div>	frontend/src/app/code-snippet code-snippet.component.ts (Line 167)	typescript:S2245 Make sure that using this pseudorandom number generator is safe here. Category: Weak Cryptography
<div> <div>■ Security Hotspot Code:</div> <pre> 164: 165: shuffle () { 166: this.randomFixes = this.fixes >>> 167: .map((fix, index) => ({ fix, index, sort: Math.random() })) 168: .sort((a, b) => a.sort - b.sort) 169: .map(({ fix, index }) => ({ fix, index })) 170: } </pre> </div>		
<div>M</div>	lib/insecurity.ts (Line 55)	typescript:S2245 Make sure that using this pseudorandom number generator is safe here. Category: Weak Cryptography
<div> <div>■ Security Hotspot Code:</div> <pre> 52: } 53: 54: export const isAuthorized = () => expressJwt(({ secret: publicKey }) as any) >>> 55: export const denyAll = () => expressJwt({ secret: '' + Math.random() } as any) 56: export const authorize = (user = {}) => jwt.sign(user, privateKey, { expiresIn: '6h', algorithm: 'RS256' }) 57: export const verify = (token: string) => token ? (jws.verify as ((token: string, secret: string) => boolean))(token, publicKey) : false 58: export const decode = (token: string) => { return jws.decode(token)?.payload } </pre> </div>		

M	routes/captcha.ts (Line 15)	typescript:S2245 Make sure that using this pseudorandom number generator is safe here. Category: Weak Cryptography
<div> <div>■ Security Hotspot Code:</div> <pre> 12: const captchaId = req.app.locals.captchaId++ 13: const operators = ['*', '+', '-'] 14: >>> 15: const firstTerm = Math.floor((Math.random() * 10) + 1) 16: const secondTerm = Math.floor((Math.random() * 10) + 1) 17: const thirdTerm = Math.floor((Math.random() * 10) + 1) 18: </pre> </div>		
M	routes/captcha.ts (Line 16)	typescript:S2245 Make sure that using this pseudorandom number generator is safe here. Category: Weak Cryptography
<div> <div>■ Security Hotspot Code:</div> <pre> 13: const operators = ['*', '+', '-'] 14: 15: const firstTerm = Math.floor((Math.random() * 10) + 1) >>> 16: const secondTerm = Math.floor((Math.random() * 10) + 1) 17: const thirdTerm = Math.floor((Math.random() * 10) + 1) 18: 19: const firstOperator = operators[Math.floor((Math.random() * 3))] </pre> </div>		
M	routes/captcha.ts (Line 17)	typescript:S2245 Make sure that using this pseudorandom number generator is safe here. Category: Weak Cryptography
<div> <div>■ Security Hotspot Code:</div> <pre> 14: 15: const firstTerm = Math.floor((Math.random() * 10) + 1) 16: const secondTerm = Math.floor((Math.random() * 10) + 1) >>> 17: const thirdTerm = Math.floor((Math.random() * 10) + 1) 18: 19: const firstOperator = operators[Math.floor((Math.random() * 3))] 20: const secondOperator = operators[Math.floor((Math.random() * 3))] </pre> </div>		
M	routes/captcha.ts (Line 19)	typescript:S2245 Make sure that using this pseudorandom number generator is safe here. Category: Weak Cryptography

■ Security Hotspot Code: <pre> 16: const secondTerm = Math.floor((Math.random() * 10) + 1) 17: const thirdTerm = Math.floor((Math.random() * 10) + 1) 18: >>> 19: const firstOperator = operators[Math.floor((Math.random() * 3))] 20: const secondOperator = operators[Math.floor((Math.random() * 3))] 21: 22: const expression = firstTerm.toString() + firstOperator + secondTerm.toString() + secondOperator + thirdTerm.toString() </pre>		
M	routes/captcha.ts (Line 20)	typescript:S2245 Make sure that using this pseudorandom number generator is safe here. Category: Weak Cryptography
■ Security Hotspot Code: <pre> 17: const thirdTerm = Math.floor((Math.random() * 10) + 1) 18: 19: const firstOperator = operators[Math.floor((Math.random() * 3))] >>> 20: const secondOperator = operators[Math.floor((Math.random() * 3))] 21: 22: const expression = firstTerm.toString() + firstOperator + secondTerm.toString() + secondOperator + thirdTerm.toString() 23: const answer = eval(expression).toString() // eslint-disable-line no-eval </pre>		

Cross-Site Scripting (XSS) (9 hotspots)

Risk	File Path	Rule & Message
H	frontend/src/app/about/about.component.ts (Line 121)	typescript:S6268 Make sure disabling Angular built-in sanitization is safe here. Category: Cross-Site Scripting (XSS)
■ Security Hotspot Code: <pre> 118: feedbacks[i].comment = `\${ 119: feedbacks[i].comment 120: }
 (\${this.stars[feedbacks[i].rating]})` >>> 121: feedbacks[i].comment = this.sanitizer.bypassSecurityTrustHtml(122: feedbacks[i].comment 123:) 124: </pre>		
H	frontend/src/app/administration/administration.component.ts (Line 57)	typescript:S6268 Make sure disabling Angular built-in sanitization is safe here. Category: Cross-Site Scripting (XSS)

<p>■ Security Hotspot Code:</p> <pre> 54: this.userDataSource = users 55: this.userDataSourceHidden = users 56: for (const user of this.userDataSource) { >>> 57: user.email = this.sanitizer.bypassSecurityTrustHtml(`\${user.email}`) 58: } 59: this.userDataSource = new MatTableDataSource(this.userDataSource) 60: this.userDataSource.paginator = this.paginatorUsers </pre>		
H	frontend/src/app/administration administration.component.ts (Line 75)	typescript:S6268 Make sure disabling Angular built-in sanitization is safe here. Category: Cross-Site Scripting (XSS)
<p>■ Security Hotspot Code:</p> <pre> 72: next: (feedbacks) => { 73: this.feedbackDataSource = feedbacks 74: for (const feedback of this.feedbackDataSource) { >>> 75: feedback.comment = this.sanitizer.bypassSecurityTrustHtml(feedback.comment) 76: } 77: this.feedbackDataSource = new MatTableDataSource(this.feedbackDataSource) 78: this.feedbackDataSource.paginator = this.paginatorFeedb </pre>		
H	frontend/src/app/data-export data-export.component.ts (Line 55)	typescript:S6268 Make sure disabling Angular built-in sanitization is safe here. Category: Cross-Site Scripting (XSS)
<p>■ Security Hotspot Code:</p> <pre> 52: 53: getNewCaptcha () { 54: this.imageCaptchaService.getCaptcha().subscribe((data: any) => { >>> 55: this.captcha = this.sanitizer.bypassSecurityTrustHtml(data.image) 56: }) 57: } 58: </pre>		
H	frontend/src/app/last-login-ip last-login-ip.component.ts (Line 38)	typescript:S6268 Make sure disabling Angular built-in sanitization is safe here. Category: Cross-Site Scripting (XSS)
<p>■ Security Hotspot Code:</p> <pre> 35: payload = jwtDecode(token) 36: if (payload.data.lastLoginIp) { 37: // eslint-disable-next-line @typescript-eslint/restrict-template-expressions >>> 38: this.lastLoginIp = this.sanitizer.bypassSecurityTrustHtml(`<small>\${payload.data.lastLoginIp}</small>`) 39: } 40: } 41: } </pre>		
H	frontend/src/app/score-board score-board.component.ts (Line 86)	typescript:S6268 Make sure disabling Angular built-in sanitization is safe here. Category: Cross-Site Scripting (XSS)

	<p>■ Security Hotspot Code:</p> <pre> 83: hintsAvailable: hints.filter((hint) => hint.ChallengeId === challenge.id).length, 84: tagList: challenge.tags ? challenge.tags.split(',').map((tag) => tag.trim()) : [], 85: originalDescription: challenge.description as string, >>> 86: description: this.sanitizer.bypassSecurityTrustHtml(challenge.description as string) 87: } 88: }) 89: </pre>	
<div>H</div>	<p>frontend/src/app/search-result search-result.component.ts (Line 135)</p>	<p>typescript:S6268 Make sure disabling Angular built-in sanitization is safe here. Category: Cross-Site Scripting (XSS)</p>
	<p>■ Security Hotspot Code:</p> <pre> 132: 133: trustProductDescription (tableData: any[]) { // vuln-code-snippet neutral-line restfulXssChallenge 134: for (let i = 0; i < tableData.length; i++) { // vuln-code-snippet neutral-line restfulXssChallenge >>> 135: tableData[i].description = this.sanitizer.bypassSecurityTrustHtml(tableData[i].description) // vuln-code-snippet vuln-line restfulXssChallenge 136: } // vuln-code-snippet neutral-line restfulXssChallenge 137: } // vuln-code-snippet neutral-line restfulXssChallenge 138: // vuln-code-snippet end restfulXssChallenge </pre>	
<div>H</div>	<p>frontend/src/app/search-result search-result.component.ts (Line 161)</p>	<p>typescript:S6268 Make sure disabling Angular built-in sanitization is safe here. Category: Cross-Site Scripting (XSS)</p>
	<p>■ Security Hotspot Code:</p> <pre> 158: this.io.socket().emit('verifyLocalXssChallenge', queryParam) 159: }) // vuln-code-snippet hide-end 160: this.dataSource.filter = queryParam.toLowerCase() >>> 161: this.searchValue = this.sanitizer.bypassSecurityTrustHtml(queryParam) // vuln-code-snippet vuln-line localXssChallenge xssBonusChallenge 162: this.gridDataSource.subscribe((result: any) => { 163: if (result.length === 0) { 164: this.emptyState = true </pre>	
<div>H</div>	<p>frontend/src/app/track-result track-result.component.ts (Line 45)</p>	<p>typescript:S6268 Make sure disabling Angular built-in sanitization is safe here. Category: Cross-Site Scripting (XSS)</p>

■ Security Hotspot Code:

```
42:   this.orderId = this.route.snapshot.queryParams.id
43:   this.trackOrderService.find(this.orderId).subscribe((results) => {
44:     // eslint-disable-next-line @typescript-eslint/restrict-template-expressions
>>> 45:     this.results.orderNo =
this.sanitizer.bypassSecurityTrustHtml(`<code>${results.data[0].orderId}</code>`)
46:     this.results.email = results.data[0].email
47:     this.results.totalPrice = results.data[0].totalPrice
48:     this.results.products = results.data[0].products
```

Rules Reference

This section contains 72 rules found in the analysis.

Sections of code should not be commented out

Key: Web:AvoidCommentedOutCodeCheck

Root Cause:

Commented-out code distracts the focus from the actual executed code. It creates a noise that increases maintenance code. And because it is never executed, it quickly becomes out of date and invalid.

Commented-out code should be deleted and can be retrieved from source control history if required.

Image, area and button with image elements should have an "alt" attribute

Key: Web:ImgWithoutAltCheck

How To Fix:

Add an alternative text to the HTML element.

Noncompliant code example

```
 <!-- missing `alt` attribute -->
<input type="image" src="bar.png" /> <!-- missing `alt` attribute -->
<input type="image" src="bar.png" alt="" /> <!-- empty `alt` attribute on <input> -->


<map id="map1" name="map1">
    <area shape="rect" coords="0,0,42,42"
        href="bedroom.html"/> <!-- missing `alt` attribute -->
    <area shape="rect" coords="0,0,21,21"
        href="lounge.html" alt="" /> <!-- empty `alt` attribute on <area> -->
</map>
```

Compliant solution

```

<input type="image" src="bar.png" alt="Textual description of bar.png" />


<map id="map1" name="map1">
    <area shape="rect" coords="0,0,42,42"
        href="bedroom.html" alt="Bedroom" />
    <area shape="rect" coords="0,0,21,21"
        href="lounge.html" alt="Lounge" />
</map>
```

Root Cause:

The `alt` attribute provides a textual alternative to an image.

It is used whenever the actual image cannot be rendered.

Common reasons for that include:

- The image can no longer be found
- Visually impaired users using a screen reader software
- Image loading is disabled, to reduce data consumption on mobile phones

It is also very important not to set an `alt` attribute to a non-informative value. For example, ``

is useless as it doesn't give any information to the user. In this case, as for any other decorative image, it is better to use a CSS background image instead of an `` tag. If using CSS `background-image` is not possible, an empty `alt=""` is tolerated. See Exceptions below.

This rule raises an issue when:

- An `<input type="image">` or `<area>` element has no `alt` attribute or it holds an empty string value.
- An `` element has no `alt` attribute.

Exceptions

`` elements with an empty string `alt=""` attribute won't raise any issue. However, this way should be used in two cases only:

When the image is decorative and it is not possible to use a CSS background image. For example, when the decorative `` is generated via javascript with a source image coming from a database, it is better to use an `` tag rather than generate CSS code.

```
<li *ngFor="let image of images">
  <img [src]="image" alt="">
</li>
```

When the image is not decorative but its `alt` text would repeat a nearby text. For example, images contained in links should not duplicate the link's text in their `alt` attribute, as it would make the screen reader repeat the text twice.

```
<a href="flowers.html">
  
  A blooming tulip
</a>
```

In all other cases you should use CSS background images.

Resources:

Documentation

- W3C - [W3C WAI Web Accessibility Tutorials](#)
- W3C - [Providing text alternatives for the area elements of image maps](#)

- W3C - [Using alt attributes on images used as submit buttons](#)
- W3C - [Using alt attributes on img elements](#)
- W3C - [Using null alt text and no title attribute on img elements for images that AT should ignore](#)
- W3C - [Combining adjacent image and text links for the same resource](#)
- W3C - [Non-text Content](#)
- W3C - [Link Purpose \(In Context\)](#)
- W3C - [Link Purpose \(Link Only\)](#)

Mouse events should have corresponding keyboard events

Key: Web:MouseEventWithoutKeyboardEquivalentCheck

Root Cause:

Offering the same experience with the mouse and the keyboard allow users to pick their preferred devices.

Additionally, users of assistive technology will also be able to browse the site even if they cannot use the mouse.

This rule raises an issue when:

- an HTML element with an `onmouseover` attribute doesn't also have an `onFocus` attribute.
- an HTML element with an `onmouseout` attribute doesn't also have an `onBlur` attribute.
- an HTML element with an `onClick` attribute doesn't also have one of the following attributes: `onKeyDown`, `onKeyUp`, `onKeyPress`.

Note that in the case of `onClick`, the equivalent keyboard handler should support both the "Enter" and "Space" keys as these are usually used by screen-readers.

Noncompliant code example

```
<div onClick="doSomething();" ...> <!-- Noncompliant - 'onKeyDown/onKeyUp/onKeyPress'
missing -->
<a onmouseover="doSomething();" ...> <!-- Noncompliant - 'onFocus' missing -->
<a onmouseout="doSomething();" ...> <!-- Noncompliant - 'onBlur' missing -->
```

Compliant solution

Note that setting the `tabindex` attribute is necessary to make the `<div>` element focusable.

```
<div onClick="doSomething();" onKeyDown="doSomething();" tabindex="0" ...> <!--
Compliant -->
<a onmouseover="doSomething();" onFocus="doSomething();" ...> <!-- Compliant -->
<a onmouseout="doSomething();" onBlur="doSomething();" ...> <!-- Compliant -->
```

Exceptions

For the following elements, [pressing a key will trigger the `onClick` attribute](#): `<input type="button">`, `<input type="submit">`, `<button>`, `<a>`. Thus no issue will be raised when an `onClick` attribute is found in these elements without a `onKeyDown/onKeyUp/onKeyPress`.

An issue will still be raised for [elements with the `role="button"` attribute](#) as they don't behave the same way.

Resources:

- [SCR2: Using redundant keyboard and mouse event handlers](#)

"<html>" element should have a language attribute

Key: Web:S5254

Resources:

- [WCAG2, H57](#) - Using language attributes on the html element
- [WCAG2, 3.1.1](#) - Language of Page

Root Cause:

The `<html>` element should provide the `lang` and/or `xml:lang` attribute in order to identify the default language of a document.

It enables assistive technologies, such as screen readers, to provide a comfortable reading experience by adapting the pronunciation and accent to the language. It also helps braille translation software, telling it to switch the control codes for accented characters for instance.

Other benefits of marking the language include assisting user agents in providing dictionary definitions or helping users benefit from translation tools.

Both the `lang` and the `xml:lang` attributes can take only one value.

Noncompliant code example

```
<!DOCTYPE html>
<html> <!-- Noncompliant -->
  <head>
    <title>A page written in english</title>
    <meta content="text/html; charset=utf-8" />
  </head>

  <body>
    ...
  </body>
</html>
```

Compliant solution

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>A page written in english</title>
    <meta content="text/html; charset=utf-8" />
  </head>

  <body>
    ...
  </body>
</html>
```

```
<!DOCTYPE html>
<html lang="en" xml:lang="en">
  <head>
    <title>A page written in english</title>
    <meta content="text/html; charset=utf-8" />
  </head>

  <body>
    ...
  </body>
</html>
```

Tables should have headers

Key: Web:S5256

Root Cause:

Table headers are essential to enhance the accessibility of a table's data, particularly for assistive technologies like screen readers. These headers provide the necessary context to transform data into information. Without headers, users get rapidly lost in the flow of data.

This rule raises an issue whenever a `<table>` does not contain any `<th>` elements.

Exceptions

No issue will be raised on `<table>` used for layout purpose, i.e. when it contains a `role` attribute set to `"presentation"` or `"none"`.

```
<table role="presentation">
  <tr>
    <td>Name</td>
    <td>Age</td>
  </tr>
  <tr>
    <td>John Doe</td>
    <td>42</td>
  </tr>
</table>
```

Note that [using `<table>` for layout purpose is a bad practice](#).

No issue will be raised on `<table>` containing an `aria-hidden` attribute set to `"true"`.

```
<table aria-hidden="true">
  <tr>
    <td>Name</td>
    <td>Age</td>
  </tr>
  <tr>
    <td>John Doe</td>
    <td>42</td>
  </tr>
</table>
```

Resources:

Documentation

- [WCAG2, 1.3.1](#) - Info and Relationships
- [WCAG2, H51](#) - Using table markup to present tabular information

How To Fix:

The first `<tr>` of the table should contain `<th>` elements, with the appropriate description of what the data in those columns represents.

Going the extra mile

Headers should be properly associated with the corresponding `<td>` cells by using either a `scope` attribute or `headers` and `id` attributes. See [W3C WAI Web Accessibility Tutorials](#) for more information.

Noncompliant code example

```
<table> <!-- Noncompliant -->
  <tr>
    <td>Name</td>
    <td>Age</td>
  </tr>
  <tr>
    <td>John Doe</td>
    <td>24</td>
  </tr>
  <tr>
    <td>Alice Doe</td>
    <td>54</td>
  </tr>
</table>
```

Compliant solution

```
<table>
```



```
<tr>
  <th scope="col">Name</th>
  <th scope="col">Age</th>
</tr>
<tr>
  <td>John Doe</td>
  <td>24</td>
</tr>
<tr>
  <td>Alice Doe</td>
  <td>54</td>
</tr>
</table>
```

Using remote artifacts without integrity checks is security-sensitive

Key: Web:S5725

Default:

Using remote artifacts without integrity checks can lead to the unexpected execution of malicious code in the application.

On the client side, where front-end code is executed, malicious code could:

- impersonate users' identities and take advantage of their privileges on the application.
- add quiet malware that monitors users' session and capture sensitive secrets.
- gain access to sensitive clients' personal data.
- deface, or otherwise affect the general availability of the application.
- mine cryptocurrencies in the background.

Likewise, a compromised software piece that would be deployed on a server-side application could badly affect the application's security. For example, server-side malware could:

- access and modify sensitive technical and business data.
- elevate its privileges on the underlying operating system.
- Use the compromised application as a pivot to attack the local network.

By ensuring that a remote artifact is exactly what it is supposed to be before using it, the application is protected from unexpected changes applied to it before it is downloaded.

Especially, integrity checks will allow for identifying an artifact replaced by malware on the publication website or that was legitimately changed by its author, in a more benign scenario.

Important note: downloading an artifact over HTTPS only protects it while in transit from one host to another. It provides authenticity and integrity checks **for the network stream** only. It does not ensure the authenticity or security of the artifact itself.

Ask Yourself Whether

- The artifact is a file intended to execute code.

- The artifact is a file that is intended to configure or affect running code in some way.

There is a risk if you answered yes to any of these questions.

Recommended Secure Coding Practices

To check the integrity of a remote artifact, hash verification is the most reliable solution. It does ensure that the file has not been modified since the fingerprint was computed.

In this case, the artifact's hash must:

- Be computed with a secure hash algorithm such as SHA512, SHA384 or SHA256.
- Be compared with a secure hash that was **not** downloaded from the same source.

To do so, the best option is to add the hash in the code explicitly, by following [Mozilla's official documentation on how to generate integrity strings](#).

Note: Use this fix together with version binding on the remote file. Avoid downloading files named "latest" or similar, so that the front-end pages do not break when the code of the latest remote artifact changes.

Sensitive Code Example

The following code sample uses neither integrity checks nor version pinning:

```
<script
  src="https://cdn.example.com/latest/script.js"
></script> <!-- Sensitive -->
```

Compliant Solution

```
<script
  src="https://cdn.example.com/v5.3.6/script.js"
  integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlGYl1kPzQh0lwx4JwY8wC"
></script>
```

See

- OWASP - [Top 10 2021 Category A8 - Software and Data Integrity Failures](#)
- CWE - [CWE-353 - Missing Support for Integrity Check](#)
- OWASP - [Top 10 2017 Category A6 - Security Misconfiguration](#)
- developer.mozilla.org - Subresource Integrity
- [Wikipedia, Watering Hole Attacks](#)

Assess The Problem:

Ask Yourself Whether

- The artifact is a file intended to execute code.
- The artifact is a file that is intended to configure or affect running code in some way.

There is a risk if you answered yes to any of these questions.

Sensitive Code Example

The following code sample uses neither integrity checks nor version pinning:

```
<script
  src="https://cdn.example.com/latest/script.js"
></script> <!-- Sensitive -->
```

How To Fix:

Recommended Secure Coding Practices

To check the integrity of a remote artifact, hash verification is the most reliable solution. It does ensure that the file has not been modified since the fingerprint was computed.

In this case, the artifact's hash must:

- Be computed with a secure hash algorithm such as SHA512, SHA384 or SHA256.
- Be compared with a secure hash that was **not** downloaded from the same source.

To do so, the best option is to add the hash in the code explicitly, by following [Mozilla's official documentation on how to generate integrity strings](#).

Note: Use this fix together with version binding on the remote file. Avoid downloading files named "latest" or similar, so that the front-end pages do not break when the code of the latest remote artifact changes.

Compliant Solution

```
<script
  src="https://cdn.example.com/v5.3.6/script.js"
  integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlGYl1kPzQh0lwx4JwY8wC"
></script>
```

See

- OWASP - [Top 10 2021 Category A8 - Software and Data Integrity Failures](#)
- CWE - [CWE-353 - Missing Support for Integrity Check](#)
- OWASP - [Top 10 2017 Category A6 - Security Misconfiguration](#)
- developer.mozilla.org - Subresource Integrity
- [Wikipedia, Watering Hole Attacks](#)

Root Cause:

Using remote artifacts without integrity checks can lead to the unexpected execution of malicious code in the application.

On the client side, where front-end code is executed, malicious code could:

- impersonate users' identities and take advantage of their privileges on the application.
- add quiet malware that monitors users' session and capture sensitive secrets.
- gain access to sensitive clients' personal data.
- deface, or otherwise affect the general availability of the application.
- mine cryptocurrencies in the background.

Likewise, a compromised software piece that would be deployed on a server-side application could badly affect the application's security. For example, server-side malware could:

- access and modify sensitive technical and business data.
- elevate its privileges on the underlying operating system.
- Use the compromised application as a pivot to attack the local network.

By ensuring that a remote artifact is exactly what it is supposed to be before using it, the application is protected from unexpected changes applied to it before it is downloaded.

Especially, integrity checks will allow for identifying an artifact replaced by malware on the publication website or that was legitimately changed by its author, in a more benign scenario.

Important note: downloading an artifact over HTTPS only protects it while in transit from one host to another. It provides authenticity and integrity checks **for the network stream** only. It does not ensure the authenticity or security of the artifact itself.

Prefer tag over ARIA role

Key: Web:S6819

Resources:

Documentation

- MDN web docs - [Using ARIA: Roles, states, and properties](#)
- MDN web docs - [ARIA roles \(Reference\)](#)

Standards

- W3C - [Accessible Rich Internet Applications \(WAI-ARIA\) 1.2](#)

Root Cause:

ARIA (Accessible Rich Internet Applications) roles are used to make web content and web applications more accessible to people with disabilities.

However, you should not use an ARIA role on a generic element (like `span` or `div`) if there is a semantic HTML tag with similar functionality, just use that tag instead.

For example, instead of using a div element with a button role (`<div role="button">Click me</div>`), you should just use a button element (`<button>Click me</button>`).

Semantic HTML tags are generally preferred over ARIA roles for accessibility due to their built-in functionality, universal support by browsers and assistive technologies, simplicity, and maintainability. They come with inherent behaviors and keyboard interactions, reducing the need for additional JavaScript. Semantic HTML also enhances SEO by helping search engines better understand the content and structure of web pages. While ARIA roles are useful, they should be considered a last resort when no suitable HTML element can provide the required behavior or semantics.

```
<div role="button" onClick="myFunction()">Click me</div> <!-- Noncompliant -->
```

Replace the ARIA role with an appropriate HTML tag.

```
<button onClick="myFunction()">Click me</button>
```

Non-interactive DOM elements should not have interactive ARIA roles

Key: Web:S6842

Resources:

Documentation

- WCAG - [Name, Role, Value](#)
- WCAG - [WAI-ARIA Roles](#)
- MDN web docs - [WAI-ARIA Roles](#)

Root Cause:

Non-interactive DOM elements are those that do not have built-in interactivity or do not require user interaction. Examples include

`<div>`, `<p>`, ``, `<h1>` to `<h6>`, and

``, among others. These elements are typically used to structure content and layout but do not have any inherent interactive behavior.

Interactive ARIA roles, on the other hand, are used to make elements interactive and accessible. These roles include `button`,

`link`, `checkbox`, `menuitem`, `tab`, and others. They are used to communicate the type of interaction that users can expect from an element.

Non-interactive DOM elements should not use interactive ARIA roles because it can confuse assistive technologies and their users. For example, if a

`<div>` (a non-interactive element) is given an interactive role like `"button"`, assistive technologies like screen readers will announce it as a button. However, since `<div>` doesn't have the inherent behavior of a button, it can confuse users who expect it to behave like a button when interacted with.

This can lead to a poor user experience and can make the website less accessible for users relying on assistive technologies. Therefore, it's important to ensure that non-interactive DOM elements are not given interactive ARIA roles.

How To Fix:

Ensure that non-interactive DOM elements do not use interactive ARIA roles.

Noncompliant code example

```
<li role="button">Foo</li> <!--Noncompliant; "li" isn't interactive, but "button" is-->
```

Compliant solution

```
<li role="listitem">Foo</li>
```

Heading elements should have accessible content

Key: Web:S6850

Root Cause:

Heading elements are represented by the tags `<h1>` through `<h6>`, with `<h1>` being the highest level and `<h6>` being the lowest. These elements are used to structure the content of the page and create a hierarchical outline that can be followed by users and search engines alike.

In the context of accessibility, heading elements play a crucial role. They provide a way for users, especially those using assistive technologies like screen readers, to navigate through the content of a webpage. By reading out the headings, screen readers can give users an overview of the content and allow them to jump to the section they're interested in.

If a heading element is empty, it can lead to confusion as it doesn't provide any information about the content that follows. This can make navigation difficult for users relying on screen readers, resulting in a poor user experience and making the website less accessible for people with visual impairments.

Therefore, to ensure your website is accessible to all users, it's important to always include meaningful content in your heading elements.

Resources:

Documentation

- MDN web docs - [Heading elements](#)
- MDN web docs - [aria-hidden](#)
- WCAG - [Headings and Labels](#)

How To Fix:

Do not leave empty your heading elements.

Noncompliant code example

```
<h1>JavaScript Programming Guide</h1>
<p>An introduction to JavaScript programming and its applications.</p>

<h2>JavaScript Basics</h2>
<p>Understanding the basic concepts in JavaScript programming.</p>

<h3>Variables</h3>
<p>Explanation of what variables are and how to declare them in JavaScript.</p>
```

```
<h3 aria-hidden>Data Types</h3> <!-- Noncompliant -->
<p>Overview of the different data types in JavaScript.</p>

<h3 /> <!-- Noncompliant -->
<p>Understanding how to declare and use functions in JavaScript.</p>
```

Compliant solution

```
<h1>JavaScript Programming Guide</h1>
<p>An introduction to JavaScript programming and its applications.</p>

<h2>JavaScript Basics</h2>
<p>Understanding the basic concepts in JavaScript programming.</p>

<h3>Variables</h3>
<p>Explanation of what variables are and how to declare them in JavaScript.</p>

<h3>Data Types</h3>
<p>Overview of the different data types in JavaScript.</p>

<h3>Functions</h3>
<p>Understanding how to declare and use functions in JavaScript.</p>
```

Images should have a non-redundant alternate description

Key: Web:S6851

Root Cause:

`alt` attributes, also known as "alt tags" or "alt descriptions," are used to specify alternative text that is rendered when an image cannot be displayed. They are crucial for improving web accessibility, as they provide a text description of images for users who rely on screen readers.

Screen readers announce the presence of an `` element and read its `alt` attribute aloud to describe the image. If the `alt` attribute includes words like "image", "picture", or "photo", it leads to redundancy as the screen reader would repeat "image".

For instance, an `alt` attribute like "image of a sunrise" would be read as "Image, image of a sunrise", unnecessarily repeating "image".

Instead, the `alt` attribute should focus on describing the content of the image, not the fact that it is an image. This makes the browsing experience more efficient and enjoyable for users of screen readers, as they receive a concise and meaningful description of the image without unnecessary repetition.

How To Fix:

To fix this issue, you should revise the `alt` attribute of your `` elements to remove any instances of the words

"image", "picture", or "photo". Instead, provide a concise and accurate description of the image content that adds value for users who cannot see the image.

Noncompliant code example

```
 <!-- Noncompliant: "Image, image of
```

```
a sunrise" -->
```

Compliant solution

```

```

Resources:

Documentation

- MDN web docs - [img element](#)
- MDN web docs - [alt property](#)
- WebAIM - [Alternative Text](#)

Label elements should have a text label and an associated control

Key: Web:S6853

Resources:

Documentation

- MDN web docs - [The Label element](#)
- W3C - [Info and Relationships](#)
- W3C - [Labels or Instructions](#)

Root Cause:

When a label element lacks a text label or an associated control, it can lead to several issues:

- **Poor Accessibility:** Screen readers rely on correctly associated labels to describe the function of the form control. If the label is not properly associated with a control, it can make the form difficult or impossible for visually impaired users to understand or interact with.
- **Confusing User Interface:** Labels provide users with clear instructions about what information is required in a form control. Without a properly associated label, users might not understand what input is expected, leading to confusion and potential misuse of the form.
- **Code Maintainability:** Properly structured and labeled code is easier to read, understand, and maintain. When labels are not correctly associated, it can make the code more difficult to navigate and debug, especially for new developers or those unfamiliar with the codebase.

Control elements are: * <input> * <meter> * <output> * <progress> *
<select> * <textarea>

Exceptions

Custom components may contain control elements, therefore label elements containing custom elements do not raise issues.

Introduction:

A `<label>` element should wrap a control element or have an `<htmlFor>` attribute referencing a control and text content.

How To Fix:

If you have a pair of control and `<label>` elements, make sure that the `<label>` wraps the control element. If you lack a control element, add one.

It is strongly recommended to avoid using generated `ids` since they must be deterministic.

Noncompliant code example

```
<input type="text" />
<label>Favorite food</label>
```

Compliant solution

```
<label>
  <input type="text" />
  Favorite food
</label>
```

Properties should not be duplicated

Key: `css:S4656`

Root Cause:

CSS allows duplicate property names but only the last instance of a duplicated name determines the actual value that will be used for it.

Therefore, changing values of other occurrences of a duplicated name will have no effect and may cause misunderstandings and bugs.

This rule ignores `$sass`, `@less`, and `var(--custom-property)` variable syntaxes.

Noncompliant code example

```
a {
  color: pink;
  background: orange;
  color: orange
}
```

Compliant solution

```
a {
  color: pink;
  background: orange
}
```

Selectors should not be duplicated

Key: css:S4666

Root Cause:

In CSS, when selectors are duplicated, the browser applies them in cascade. This means that if two selectors are identical, the second one takes precedence. However, if the declarations within the selectors are not conflicting, they will be combined.

This behavior can lead to unexpected results and make debugging more difficult, especially in larger stylesheets. Therefore, it's generally recommended to avoid duplicating selectors.

The rule detects the following kinds of duplications:

- within a list of selectors in a single rule set,
- for duplicated selectors in different rule sets within a single stylesheet.

How To Fix:

To fix your code, either remove the duplicated selector or merge all declarations.

Noncompliant code example

```
p {  
  color: blue;  
  font-size: 16px;  
}  
  
p { /* Noncompliant: duplicated selector 'p', overwrites property 'color' */  
  color: red;  
}
```

Compliant solution

```
p {  
  color: red;  
  font-size: 16px;  
}
```

Resources:

Documentation

- MDN - [CSS selectors](#)
- MDN - [Cascade, specificity, and inheritance](#)

Using clear-text protocols is security-sensitive

Key: docker:S5332

Root Cause:

Clear-text protocols such as `ftp`, `telnet`, or `http` lack encryption of transported data, as well as the capability to build an authenticated connection. It means that an attacker able to sniff traffic from the

network can read, modify, or corrupt the transported content. These protocols are not secure as they expose applications to an extensive range of risks:

- sensitive data exposure
- traffic redirected to a malicious endpoint
- malware-infected software update or installer
- execution of client-side code
- corruption of critical information

Even in the context of isolated networks like offline environments or segmented cloud environments, the insider threat exists. Thus, attacks involving communications being sniffed or tampered with can still happen.

For example, attackers could successfully compromise prior security layers by:

- bypassing isolation mechanisms
- compromising a component of the network
- getting the credentials of an internal IAM account (either from a service account or an actual person)

In such cases, encrypting communications would decrease the chances of attackers to successfully leak data or steal credentials from other network components. By layering various security practices (segmentation and encryption, for example), the application will follow the *defense-in-depth* principle.

Note that using the `http` protocol is being deprecated by [major web browsers](#).

In the past, it has led to the following vulnerabilities:

- [CVE-2019-6169](#)
- [CVE-2019-12327](#)
- [CVE-2019-11065](#)

Default:

Clear-text protocols such as `ftp`, `telnet`, or `http` lack encryption of transported data, as well as the capability to build an authenticated connection. It means that an attacker able to sniff traffic from the network can read, modify, or corrupt the transported content. These protocols are not secure as they expose applications to an extensive range of risks:

- sensitive data exposure
- traffic redirected to a malicious endpoint
- malware-infected software update or installer
- execution of client-side code
- corruption of critical information

Even in the context of isolated networks like offline environments or segmented cloud environments, the

insider threat exists. Thus, attacks involving communications being sniffed or tampered with can still happen.

For example, attackers could successfully compromise prior security layers by:

- bypassing isolation mechanisms
- compromising a component of the network
- getting the credentials of an internal IAM account (either from a service account or an actual person)

In such cases, encrypting communications would decrease the chances of attackers to successfully leak data or steal credentials from other network components. By layering various security practices (segmentation and encryption, for example), the application will follow the *defense-in-depth* principle.

Note that using the `http` protocol is being deprecated by [major web browsers](#).

In the past, it has led to the following vulnerabilities:

- [CVE-2019-6169](#)
- [CVE-2019-12327](#)
- [CVE-2019-11065](#)

Ask Yourself Whether

- Application data needs to be protected against tampering or leaks when transiting over the network.
- Application data transits over an untrusted network.
- Compliance rules require the service to encrypt data in transit.
- OS-level protections against clear-text traffic are deactivated.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Make application data transit over a secure, authenticated and encrypted protocol like TLS or SSH. Here are a few alternatives to the most common clear-text protocols:

Use `sftp`, `scp`, or `ftps` instead of `ftp`.

- Use `https` instead of `http`.

It is recommended to secure all transport channels, even on local networks, as it can take a single non-secure connection to compromise an entire application or system.

Sensitive Code Example

RUN curl http://www.example.com/

Compliant Solution

RUN curl https://www.example.com/

See

Documentation

- AWS Documentation - [Listeners for your Application Load Balancers](#)
- AWS Documentation - [href="https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-kinesis-stream-streamencryption.html">Stream Encryption](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-kinesis-stream-streamencryption.html)

Articles & blog posts

- Google - [Moving towards more secure web](#)
- Mozilla - [Deprecating non secure http](#)

Standards

- CWE - [CWE-200 - Exposure of Sensitive Information to an Unauthorized Actor](#)
- CWE - [CWE-319 - Cleartext Transmission of Sensitive Information](#)
- STIG Viewer - [Application Security and Development: V-222397](#) - The application must implement cryptographic mechanisms to protect the integrity of remote access sessions.
- STIG Viewer - [Application Security and Development: V-222534](#) - Service-Oriented Applications handling non-releasable data must authenticate endpoint devices via mutual SSL/TLS.
- STIG Viewer - [Application Security and Development: V-222562](#) - Applications used for non-local maintenance must implement cryptographic mechanisms to protect the integrity of maintenance and diagnostic communications.
- STIG Viewer - [Application Security and Development: V-222563](#) - Applications used for non-local maintenance must implement cryptographic mechanisms to protect the confidentiality of maintenance and diagnostic communications.
- STIG Viewer - [Application Security and Development: V-222577](#) - The application must not expose session IDs.
- STIG Viewer - [Application Security and Development: V-222596](#) - The application must protect the confidentiality and integrity of transmitted information.

- STIG Viewer - [Application Security and Development: V-222597](#) - The application must implement cryptographic mechanisms to prevent unauthorized disclosure of information and/or detect changes to information during transmission.
- STIG Viewer - [Application Security and Development: V-222598](#) - The application must maintain the confidentiality and integrity of information during preparation for transmission.
- STIG Viewer - [Application Security and Development: V-222599](#) - The application must maintain the confidentiality and integrity of information during reception.

How To Fix:

Recommended Secure Coding Practices

- Make application data transit over a secure, authenticated and encrypted protocol like TLS or SSH. Here are a few alternatives to the most common clear-text protocols:

Use `sftp`, `scp`, or `ftps` instead of `ftp`.

- Use `https` instead of `http`.

It is recommended to secure all transport channels, even on local networks, as it can take a single non-secure connection to compromise an entire application or system.

Compliant Solution

```
RUN curl https://www.example.com/
```

See

Documentation

- AWS Documentation - [Listeners for your Application Load Balancers](#)
- AWS Documentation - [href="https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-kinesis-stream-streamencryption.html">Stream Encryption](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-kinesis-stream-streamencryption.html)

Articles & blog posts

- Google - [Moving towards more secure web](#)
- Mozilla - [Deprecating non secure http](#)

Standards

- CWE - [CWE-200 - Exposure of Sensitive Information to an Unauthorized Actor](#)
- CWE - [CWE-319 - Cleartext Transmission of Sensitive Information](#)
- STIG Viewer - [Application Security and Development: V-222397](#) - The application must implement cryptographic mechanisms to protect the integrity of remote access sessions.
- STIG Viewer - [Application Security and Development: V-222534](#) - Service-Oriented Applications handling non-releasable data must authenticate endpoint devices via mutual SSL/TLS.
- STIG Viewer - [Application Security and Development: V-222562](#) - Applications used for non-local maintenance must implement cryptographic mechanisms to protect the integrity of maintenance and diagnostic communications.
- STIG Viewer - [Application Security and Development: V-222563](#) - Applications used for non-local maintenance must implement cryptographic mechanisms to protect the confidentiality of maintenance and diagnostic communications.
- STIG Viewer - [Application Security and Development: V-222577](#) - The application must not expose session IDs.
- STIG Viewer - [Application Security and Development: V-222596](#) - The application must protect the confidentiality and integrity of transmitted information.
- STIG Viewer - [Application Security and Development: V-222597](#) - The application must implement cryptographic mechanisms to prevent unauthorized disclosure of information and/or detect changes to information during transmission.
- STIG Viewer - [Application Security and Development: V-222598](#) - The application must maintain the confidentiality and integrity of information during preparation for transmission.
- STIG Viewer - [Application Security and Development: V-222599](#) - The application must maintain the confidentiality and integrity of information during reception.

Assess The Problem:

Ask Yourself Whether

- Application data needs to be protected against tampering or leaks when transiting over the network.
- Application data transits over an untrusted network.
- Compliance rules require the service to encrypt data in transit.
- OS-level protections against clear-text traffic are deactivated.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

```
RUN curl http://www.example.com/
```

Recursively copying context directories is security-sensitive

Key: docker:S6470

How To Fix:

Recommended Secure Coding Practices

- Limit the usage of globbing in the `COPY` and `ADD` sources definition.
- Avoid copying the entire context directory to the image filesystem.
- Prefer providing an explicit list of files and directories that are required for the image to properly run.

Compliant Solution

```
FROM ubuntu:22.04
COPY ./example1 /example1
COPY ./example2 /example2
COPY ./run.sh /
CMD /run.sh
```

See

- [Dockerfile reference](#) - `COPY` directive
- [Dockerfile reference](#) - `ADD` directive
- CWE - [CWE-668 - Exposure of Resource to Wrong Sphere](#)
- CWE - [CWE-497 - Exposure of Sensitive System Information to an Unauthorized Control Sphere](#)

Default:

When building a Docker image from a Dockerfile, a context directory is used and sent to the Docker daemon before the actual build starts. This context directory usually contains the Dockerfile itself, along with all the files that will be necessary for the build to succeed. This generally includes:

- the source code of applications to set up in the container.
- configuration files for other software components.
- other necessary packages or components.

The `COPY` and `ADD` directives in the Dockerfiles are then used to actually copy content from the context directory to the image file system.

When `COPY` or `ADD` are used to recursively copy entire top-level directories or multiple items whose names are determined at build-time, unexpected files might get copied to the image filesystem. It could affect their confidentiality.

Ask Yourself Whether

- The copied files and directories might contain sensitive data that should be kept confidential.
- The context directory contains files and directories that have no functional purpose for the final container image.

There is a risk if you answered yes to any of those questions.

Keep in mind that the content of the context directory might change depending on the build environment and over time.

Recommended Secure Coding Practices

- Limit the usage of globbing in the `COPY` and `ADD` sources definition.
- Avoid copying the entire context directory to the image filesystem.
- Prefer providing an explicit list of files and directories that are required for the image to properly run.

Sensitive Code Example

Copying the complete context directory:

```
FROM ubuntu:22.04
# Sensitive
COPY . .
CMD /run.sh
```

Copying multiple files and directories whose names are expanded at build time:

```
FROM ubuntu:22.04
# Sensitive
COPY ./example* /
COPY ./run.sh /
CMD /run.sh
```

Compliant Solution

```
FROM ubuntu:22.04
COPY ./example1 /example1
COPY ./example2 /example2
COPY ./run.sh /
CMD /run.sh
```

See

- [Dockerfile reference](#) - `COPY` directive
- [Dockerfile reference](#) - `ADD` directive
- CWE - [CWE-668 - Exposure of Resource to Wrong Sphere](#)

- CWE - [CWE-497 - Exposure of Sensitive System Information to an Unauthorized Control Sphere](#)

Assess The Problem:

Ask Yourself Whether

- The copied files and directories might contain sensitive data that should be kept confidential.
- The context directory contains files and directories that have no functional purpose for the final container image.

There is a risk if you answered yes to any of those questions.

Keep in mind that the content of the context directory might change depending on the build environment and over time.

Sensitive Code Example

Copying the complete context directory:

```
FROM ubuntu:22.04
# Sensitive
COPY . .
CMD /run.sh
```

Copying multiple files and directories whose names are expanded at build time:

```
FROM ubuntu:22.04
# Sensitive
COPY ./example* /
COPY ./run.sh /
CMD /run.sh
```

Root Cause:

When building a Docker image from a Dockerfile, a context directory is used and sent to the Docker daemon before the actual build starts. This context directory usually contains the Dockerfile itself, along with all the files that will be necessary for the build to succeed. This generally includes:

- the source code of applications to set up in the container.
- configuration files for other software components.
- other necessary packages or components.

The `COPY` and `ADD` directives in the Dockerfiles are then used to actually copy content from the context directory to the image file system.

When `COPY` or `ADD` are used to recursively copy entire top-level directories or multiple items whose names are determined at build-time, unexpected files might get copied to the image filesystem. It could affect their confidentiality.

Running containers as a privileged user is security-sensitive

Key: docker:S6471

How To Fix:

Recommended Secure Coding Practices

In the Dockerfile:

- Create a new default user and use it with the `USER` statement.

Some container maintainers create a specific user to be used without explicitly setting it as default, such as `postgresql` or `zookeeper`. It is recommended to use these users instead of `root`.

- On Windows containers, the `ContainerUser` is available for this purpose.

Or, at launch time:

- Use the `user` argument when calling Docker or in the docker-compose file.
- Add fine-grained Linux capabilities to perform specific actions that require root privileges.

If this image is already explicitly set to launch with a non-privileged user, you can add it to the safe images list rule property of your SonarQube instance, without the tag.

Compliant Solution

For Linux-based images and scratch-based images that untar a Linux distribution:

```
FROM alpine

RUN addgroup -S nonroot \
    && adduser -S nonroot -G nonroot

USER nonroot

ENTRYPOINT ["id"]
```

For Windows-based images, you can use `ContainerUser` or create a new user:

```
FROM mcr.microsoft.com/windows/servercore:ltsc2019

RUN net user /add nonroot

USER nonroot
```

For multi-stage builds, the non-root user should be on the last stage:

```
FROM alpine as builder
```

```
COPY Makefile ./src /
RUN make build

FROM alpine as runtime
RUN addgroup -S nonroot \
    && adduser -S nonroot -G nonroot
COPY --from=builder bin/production /app
USER nonroot
ENTRYPOINT ["/app/production"]
```

For images that use `scratch` as their base, it is not possible to add non-privileged users by default. To do this, add an additional build stage to add the group and user, and later copy `/etc/passwd`.

Here is an example that uses `adduser` in the first stage to generate a user and add it to the `/etc/passwd` file. In the next stage, this user is added by copying that file over from the previous stage:

```
FROM alpine:latest as security_provider
RUN addgroup -S nonroot \
    && adduser -S nonroot -G nonroot

FROM scratch as production
COPY --from=security_provider /etc/passwd /etc/passwd
USER nonroot
COPY production_binary /app
ENTRYPOINT ["/app/production_binary"]
```

See

- CWE - [CWE-284 - Improper Access Control](#)
- [nginxinc/nginx-unprivileged: Example of a non-root container by default](#)
- <https://learn.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/container-security#when-to-use-containeradmin-and-containeruser-user-accounts>>Microsoft docs, When to use ContainerAdmin and ContainerUser user accounts

Default:

Running containers as a privileged user weakens their runtime security, allowing any user whose code runs on the container to perform administrative actions.

In Linux containers, the privileged user is usually named `root`. In Windows containers, the equivalent is `ContainerAdministrator`.

A malicious user can run code on a system either thanks to actions that could be deemed legitimate - depending on internal business logic or operational management shells - or thanks to malicious actions. For example, with arbitrary code execution after exploiting a service that the container hosts.

Suppose the container is not hardened to prevent using a shell, interpreter, or [Linux capabilities](#). In this case, the malicious user can read and exfiltrate any file (including Docker volumes), open new network connections, install malicious software, or, worse, break out of the container's isolation context by exploiting other components.

This means giving the possibility to attackers to steal important infrastructure files, intellectual property, or personal data.

Depending on the infrastructure's resilience, attackers may then extend their attack to other services, such as Kubernetes clusters or cloud providers, in order to maximize their reach.

Ask Yourself Whether

This container:

- Serves services accessible from the Internet.
- Does not require a privileged user to run.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

In the Dockerfile:

- Create a new default user and use it with the `USER` statement.

Some container maintainers create a specific user to be used without explicitly setting it as default, such as `postgresql` or `zookeeper`. It is recommended to use these users instead of `root`.

- On Windows containers, the `ContainerUser` is available for this purpose.

Or, at launch time:

- Use the `user` argument when calling Docker or in the `docker-compose` file.
- Add fine-grained Linux capabilities to perform specific actions that require root privileges.

If this image is already explicitly set to launch with a non-privileged user, you can add it to the `safe images` list rule property of your SonarQube instance, without the tag.

Sensitive Code Example

For any image that does not provide a user by default, regardless of their underlying operating system:

```
# Sensitive
FROM alpine

ENTRYPOINT ["id"]
```

For multi-stage builds, the last stage is non-compliant if it does not contain the `USER` instruction with a non-root user:

```
FROM alpine AS builder
COPY Makefile ./src /
RUN make build
USER nonroot
```

```
# Sensitive, previous user settings are dropped
FROM alpine AS runtime
COPY --from=builder bin/production /app
ENTRYPOINT ["/app/production"]
```

Compliant Solution

For Linux-based images and scratch-based images that untar a Linux distribution:

```
FROM alpine

RUN addgroup -S nonroot \
    && adduser -S nonroot -G nonroot

USER nonroot

ENTRYPOINT ["id"]
```

For Windows-based images, you can use `ContainerUser` or create a new user:

```
FROM mcr.microsoft.com/windows/servercore:ltsc2019

RUN net user /add nonroot

USER nonroot
```

For multi-stage builds, the non-root user should be on the last stage:

```
FROM alpine as builder
COPY Makefile ./src /
RUN make build

FROM alpine as runtime
RUN addgroup -S nonroot \
    && adduser -S nonroot -G nonroot
COPY --from=builder bin/production /app
USER nonroot
ENTRYPOINT ["/app/production"]
```

For images that use `scratch` as their base, it is not possible to add non-privileged users by default. To do this, add an additional build stage to add the group and user, and later copy `/etc/passwd`.

Here is an example that uses `adduser` in the first stage to generate a user and add it to the `/etc/passwd` file. In the next stage, this user is added by copying that file over from the previous stage:

```
FROM alpine:latest as security_provider
RUN addgroup -S nonroot \
    && adduser -S nonroot -G nonroot

FROM scratch as production
COPY --from=security_provider /etc/passwd /etc/passwd
```

```
USER nonroot
COPY production_binary /app
ENTRYPOINT [ "/app/production_binary" ]
```

See

- CWE - [CWE-284 - Improper Access Control](#)
- [nginxinc/nginx-unprivileged: Example of a non-root container by default](#)
- <https://learn.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/container-security#when-to-use-containeradmin-and-containeruser-user-accounts>>Microsoft docs, When to use ContainerAdmin and ContainerUser user accounts

Root Cause:

Running containers as a privileged user weakens their runtime security, allowing any user whose code runs on the container to perform administrative actions.

In Linux containers, the privileged user is usually named `root`. In Windows containers, the equivalent is `ContainerAdministrator`.

A malicious user can run code on a system either thanks to actions that could be deemed legitimate - depending on internal business logic or operational management shells - or thanks to malicious actions. For example, with arbitrary code execution after exploiting a service that the container hosts.

Suppose the container is not hardened to prevent using a shell, interpreter, or [Linux capabilities](#). In this case, the malicious user can read and exfiltrate any file (including Docker volumes), open new network connections, install malicious software, or, worse, break out of the container's isolation context by exploiting other components.

This means giving the possibility to attackers to steal important infrastructure files, intellectual property, or personal data.

Depending on the infrastructure's resilience, attackers may then extend their attack to other services, such as Kubernetes clusters or cloud providers, in order to maximize their reach.

Assess The Problem:

Ask Yourself Whether

This container:

- Serves services accessible from the Internet.
- Does not require a privileged user to run.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

For any image that does not provide a user by default, regardless of their underlying operating system:

```
# Sensitive
FROM alpine
```

```
ENTRYPOINT [ "id" ]
```

For multi-stage builds, the last stage is non-compliant if it does not contain the `USER` instruction with a non-root user:

```
FROM alpine AS builder
COPY Makefile ./src /
RUN make build
USER nonroot

# Sensitive, previous user settings are dropped
FROM alpine AS runtime
COPY --from=builder bin/production /app
ENTRYPOINT [ "/app/production" ]
```

Allowing non-root users to modify resources copied to an image is security-sensitive

Key: docker:S6504

How To Fix:

Recommended Secure Coding Practices

- Use `--chmod` to change the permissions so that only root users can write to files.
- Use `--chown` to change the file/directory owner to a root user.
- Be mindful of the container immutability principle.

Compliant Solution

```
FROM example

COPY --chown=root:root --chmod=755 src.py dst.py
```

See

- [Dockerfile reference](#) - ADD instruction
- [Dockerfile reference](#) - COPY instruction
- CWE - [CWE-732 - Incorrect Permission Assignment for Critical Resource](#)
- [Google Cloud, Immutability](#) - Best practices for operating containers

Default:

Ownership or write permissions for a file or directory copied to the Docker image have been assigned to a user other than root.

Write permissions enable malicious actors, who have a foothold on the container, to tamper with the resource and thus potentially manipulate the container's expected behavior.

Manipulating files could disrupt services or aid in escalating privileges inside the container.

This also breaches the container immutability principle as it facilitates container changes during its life. Immutability, a container best practice, allows for a more reliable and reproducible behavior of Docker containers.

If a user is given ownership on a file but no write permissions, the user can still modify it by using his ownership to change the file permissions first. This is why both ownership and write permissions should be avoided.

Ask Yourself Whether

- A non-root user owns the resource.
- A non-root user has been granted write permissions for the resource.

There is a risk if you answered yes to any of these questions.

Recommended Secure Coding Practices

- Use `--chmod` to change the permissions so that only root users can write to files.
- Use `--chown` to change the file/directory owner to a root user.
- Be mindful of the container immutability principle.

Sensitive Code Example

```
FROM example

RUN useradd exampleuser
# Sensitive
COPY --chown=exampleuser:exampleuser src.py dst.py
```

Compliant Solution

```
FROM example

COPY --chown=root:root --chmod=755 src.py dst.py
```

See

- [Dockerfile reference](#) - ADD instruction
- [Dockerfile reference](#) - COPY instruction
- CWE - [CWE-732 - Incorrect Permission Assignment for Critical Resource](#)
- [Google Cloud, Immutability](#) - Best practices for operating containers

Assess The Problem:

Ask Yourself Whether

- A non-root user owns the resource.
- A non-root user has been granted write permissions for the resource.

There is a risk if you answered yes to any of these questions.

Sensitive Code Example

```
FROM example

RUN useradd exampleuser
# Sensitive
COPY --chown=exampleuser:exampleuser src.py dst.py
```

Root Cause:

Ownership or write permissions for a file or directory copied to the Docker image have been assigned to a user other than root.

Write permissions enable malicious actors, who have a foothold on the container, to tamper with the resource and thus potentially manipulate the container's expected behavior. Manipulating files could disrupt services or aid in escalating privileges inside the container.

This also breaches the container immutability principle as it facilitates container changes during its life. Immutability, a container best practice, allows for a more reliable and reproducible behavior of Docker containers.

If a user is given ownership on a file but no write permissions, the user can still modify it by using his ownership to change the file permissions first. This is why both ownership and write permissions should be avoided.

Allowing shell scripts execution during package installation is security-sensitive

Key: docker:S6505

Assess The Problem:

Ask Yourself Whether

- The execution of dependency installation scripts is required for the application to function correctly.

There is a risk if you answered no to the question.

Sensitive Code Example

```
FROM node:latest

# Sensitive
RUN npm install
```

```
FROM node:latest

# Sensitive
RUN yarn install
```

How To Fix:

Recommended Secure Coding Practices

Execution of third-party scripts should be disabled if not strictly necessary for dependencies to work correctly. Doing this will reduce the attack surface and block a well-known supply chain attack vector.

Commands that are subject to this issue are: `npm install`, `yarn install` and `yarn` (`yarn` without an explicit command will execute `install`).

Compliant Solution

```
FROM node:latest
```

```
RUN npm install --ignore-scripts
```

```
FROM node:latest
```

```
RUN yarn install --ignore-scripts
```

See

- CWE - [CWE-506 - Embedded Malicious Code](#)
- CWE - [CWE-829 - Inclusion of Functionality from Untrusted Control Sphere](#)
- [ESLint blog](#) - Postmortem for Malicious Packages
Published on July 12th, 2018

Default:

When installing dependencies, package managers like `npm` will automatically execute shell scripts distributed along with the source code. Post-install scripts, for example, are a common way to execute malicious code at install time whenever a package is compromised.

Ask Yourself Whether

- The execution of dependency installation scripts is required for the application to function correctly.

There is a risk if you answered no to the question.

Recommended Secure Coding Practices

Execution of third-party scripts should be disabled if not strictly necessary for dependencies to work correctly. Doing this will reduce the attack surface and block a well-known supply chain attack vector.

Commands that are subject to this issue are: `npm install`, `yarn install` and `yarn` (`yarn` without an explicit command will execute `install`).

Sensitive Code Example

```
FROM node:latest
```

```
# Sensitive
```

```
RUN npm install
```

```
FROM node:latest
```

```
# Sensitive  
RUN yarn install
```

Compliant Solution

```
FROM node:latest
```

```
RUN npm install --ignore-scripts
```

```
FROM node:latest
```

```
RUN yarn install --ignore-scripts
```

See

- CWE - [CWE-506 - Embedded Malicious Code](#)
- CWE - [CWE-829 - Inclusion of Functionality from Untrusted Control Sphere](#)
- [ESLint blog](#) - Postmortem for Malicious Packages
Published on July 12th, 2018

Root Cause:

When installing dependencies, package managers like `npm` will automatically execute shell scripts distributed along with the source code. Post-install scripts, for example, are a common way to execute malicious code at install time whenever a package is compromised.

Double quote to prevent globbing and word splitting

Key: docker:S6570

Root Cause:

Within the command, variable references and command substitutions go through word splitting and pathname expansion (globbing).

This causes issues if the variable contains whitespaces or shell pathname expansion (glob) characters like `*.`

What is the potential impact?

This issue can lead to bugs if the variable contains sensitive characters, which may be interpreted incorrectly and thus lead to undesired behavior.

How To Fix:

Surround variable reference with double quotes.

Noncompliant code example

This example demonstrates pathname expansion using the `echo` command:

```
RUN test="command t*.sh" && echo $test
```

Suppose this code is executed in a directory that contains two files: `temp1.sh` and `temp2.sh`. This code will print

`"command temp1.sh temp2.sh"`, as `*` is substituted with matching files in the current folder.

This example demonstrates word splitting using the `echo` command:

```
RUN test=" Hello World " && echo $test
```

This code will print `"Hello World"`, omitting the leading and trailing whitespaces.

Compliant solution

This example demonstrates pathname expansion using the `echo` command, which will print `"command t*.sh"` as intended:

```
RUN test="command t*.sh" && echo "$test"
```

This example demonstrates word splitting using the `echo` command, which will print `" Hello World "` as intended:

```
RUN test=" Hello World " && echo "$test"
```

Introduction:

Variable references should be encapsulated with double quotes to avoid globbing and word splitting.

Resources:

Documentation

- [Linux Documentation Project - Globbing](#)
- [Filenames and Pathnames in Shell: How to do it Correctly](#)

Cache should be cleaned after package installation

Key: docker:S6587

How To Fix:

Noncompliant code example

For apk:

```
RUN apk add nginx
```

For apt-get:

```
RUN apt-get update \
```

```
&& apt-get install nginx
```

For aptitude:

```
RUN aptitude update \  
  && aptitude install nginx
```

For apt:

```
RUN apt update \  
  && apt install nginx
```

For apt-get, without a cache mount:

```
RUN apt-get update \  
  && apt-get install nginx
```

Compliant solution

For apk:

```
RUN apk --no-cache add nginx
```

```
RUN apk add nginx \  
  && apk cache clean
```

```
RUN apk add nginx \  
  && rm -rf /var/cache/apk/*
```

```
# This cache location is only used in specific distributions / configurations  
RUN apk add nginx \  
  && rm -rf /etc/apk/cache/*
```

For apt-get:

```
RUN apt-get update \  
  && apt-get install nginx \  
  && apt-get clean
```

```
RUN apt-get update \  
  && apt-get install nginx \  
  && rm -rf /var/lib/apt/lists/* /var/cache/apt/archives/*
```

For aptitude:

```
RUN aptitude update \  
  && aptitude install nginx \  
  && aptitude clean
```

```
RUN aptitude update \  
  && aptitude install nginx
```

```
&& aptitude install nginx \  
&& rm -rf /var/lib/apt/lists/* /var/cache/apt/archives/*
```

For apt:

```
RUN apt update \  
&& apt install nginx \  
&& apt clean
```

```
RUN apt update \  
&& apt install nginx \  
&& rm -rf /var/lib/apt/lists/* /var/cache/apt/archives/*
```

For apt-get, with a cache mount:

```
RUN \  
--mount=type=cache,target=/var/cache/apt,sharing=locked \  
--mount=type=cache,target=/var/lib/apt,sharing=locked \  
apt-get update \  
&& apt-get install nginx
```

How does this work?

When installing packages using `apt-get`, `aptitude` or `apt` these package managers store an index in the Docker image layer in `/var/lib/apt/lists`. Using `apk`, it will store an index in `/var/cache/apk/`. In some distributions and configurations, the cache will be created in `/etc/apk/cache`.

This index is not needed after installation, so it can be removed. To do that, execute the `clean` command, or run `rm -rf` `<location>` for the cache location of your package manager tool.

Additionally, for `apt-get`, `aptitude` and `apt` some lock files are stored in `/var/cache/apt/archives`, which can also be removed safely. They are not removed by the `clean` command, so they need to be removed manually.

Alternatively, store the cache in a dedicated cache mount. A cache mount can be created by adding a flag `--mount type=cache` to the `RUN` command.

This will store the cache in a Docker volume, which will be persisted between builds making the build faster.

Also, each `RUN` instruction creates a new layer, and any changes made in one layer are not visible in the next. Thus, the cache should be removed in the same layer (i.e., the same `RUN` instruction) as the installation.

The following code incorrectly cleans the cache:

```
RUN apt-get install nginx  
RUN apt-get clean
```

It should be written as:

```
RUN apt-get install nginx && apt-get clean
```

Root Cause:

Docker images should only contain the necessary data. The package index is redundant for the correct operation of the installed software. Storing an index also increases the size of the Docker image. It should be reduced to speed up deployments and reduce storage and bandwidth.

Exceptions

In multi-stage builds, the rule only scans instructions that are part of the final image.

Introduction:

In Docker, when packages are installed via a package manager, an index is cached locally by default. This index should either be cleaned up or stored in a dedicated cache mount.

Resources:

Documentation

- Dockerfile Best Practices - [RUN - Best practices for writing Dockerfiles](#)
- [apk man](#)
- [apt-get man](#)
- [aptitude man](#)
- Ask Ubuntu - [How do I remove the apt package index?](#)
- Docker Build Cache - [Use the dedicated RUN cache](#)

Specific version tag for image should be used

Key: docker:S6596

Root Cause:

While using always the latest version may seem convenient, the build cannot be repeated because it is not clear which was the last version. In addition, it can lead to unpredictability and issues such as version mismatch and potential security vulnerabilities.

What is the potential impact?

For example, if a developer builds and deploys an application using `my-image:latest`, they may unknowingly be using a different version of the image than another developer who also built and deployed the same application using `my-image:latest`. This can lead to version mismatches, which can cause bugs or compatibility issues.

In addition, using `latest` as the tag for Dockerfile images can potentially introduce security vulnerabilities. For instance, if a security vulnerability is discovered in an image and a new version is released to fix it, using `latest` as the tag means that the application will automatically use the updated image, even if it has not been properly tested and vetted for compatibility with the application.

Resources:

Documentation

- [Dockerfile reference - FROM](#)
- [Docker development best practices](#)
- [Pull an image by digest \(immutable identifier\)](#)

How To Fix:

To avoid these issues, it is recommended to use specific version tags for Dockerfile images.

This can be done by appending the version number or tag to the Dockerfile image name. For example, instead of `my-image:latest`, it is better to use `my-image:1.2.3-alpine` or `my-image:1.2.3`.

For even more control and traceability, it is also possible to specify your image by digest using the sha256 of the image. This will pin your image to a specific version in time, but will also exclude it from eventual security updates. An example would be using

```
my-image@sha256:26c68657ccce2cb0a31b330cb0be2b5e108d467f641c62e13ab40cbec258c68d.
```

More information can be found in the documentation at the end.

Noncompliant code example

```
FROM my-image
FROM my-image:latest
```

Compliant solution

```
FROM my-image:1.2.3
FROM my-image:1.2.3-alpine
```

How does this work?

This way, the same version of the Dockerfile image is used every time the application is built, deployed, or run, ensuring consistency and predictability across different environments. It is also not enough to use the latest tag, as this version also changes with each release.

Going the extra mile

Adhering to this can also make it easier to track which version of the Dockerfile image is being used, which can be useful for debugging and troubleshooting purposes.

Introduction:

When a Dockerfile image is not tagged with a specific version, it is referred to as `latest`. This means that every time the image is built, deployed, or run, it will always use the latest version of the image.

Using weak hashing algorithms is security-sensitive

Key: javascript:S4790

Root Cause:

Cryptographic hash algorithms such as MD2, MD4, MD5, MD6, HAVAL-128, DSA (which uses SHA-1), RIPEMD, RIPEMD-128, RIPEMD-160 and SHA-1 are no longer considered secure,

because it is possible to have collisions (little computational effort is enough to find two or more different inputs that produce the same hash).

Message authentication code (MAC) algorithms such as HMAC-MD5 or HMAC-SHA1 use weak hash functions as building blocks.

Although they are not all proven to be weak, they are considered legacy algorithms and should be avoided.

Assess The Problem:

Ask Yourself Whether

The hashed value is used in a security context like:

- User-password storage.
- Security token generation (used to confirm e-mail when registering on a website, reset password, etc ...■).
- To compute some message integrity.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

```
const crypto = require("crypto");

const hash = crypto.createHash('sha1'); // Sensitive
```

Default:

Cryptographic hash algorithms such as MD2, MD4, MD5, MD6, HAVAL-128, DSA (which uses SHA-1), RIPEMD, RIPEMD-128, RIPEMD-160 and SHA-1 are no longer considered secure, because it is possible to have collisions (little computational effort is enough to find two or more different inputs that produce the same hash).

Message authentication code (MAC) algorithms such as HMAC-MD5 or HMAC-SHA1 use weak hash functions as building blocks.

Although they are not all proven to be weak, they are considered legacy algorithms and should be avoided.

Ask Yourself Whether

The hashed value is used in a security context like:

- User-password storage.
- Security token generation (used to confirm e-mail when registering on a website, reset password, etc ...■).
- To compute some message integrity.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

Safer alternatives, such as SHA-256, SHA-512, SHA-3 are recommended, and for password hashing, it's

even better to use algorithms that do not compute too "quickly", like `bcrypt`, `scrypt`, `argon2` or `pbkdf2` because it slows down brute force attacks.

Sensitive Code Example

```
const crypto = require("crypto");

const hash = crypto.createHash('sha1'); // Sensitive
```

Compliant Solution

```
const crypto = require("crypto");

const hash = crypto.createHash('sha512'); // Compliant
```

See

- OWASP - [Top 10 2021 Category A2 - Cryptographic Failures](#)
- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)
- OWASP - [Top 10 2017 Category A6 - Security Misconfiguration](#)
- CWE - [CWE-1240 - Use of a Risky Cryptographic Primitive](#)

How To Fix:

Recommended Secure Coding Practices

Safer alternatives, such as SHA-256, SHA-512, SHA-3 are recommended, and for password hashing, it's even better to use algorithms that do not compute too "quickly", like `bcrypt`, `scrypt`, `argon2` or `pbkdf2` because it slows down brute force attacks.

Compliant Solution

```
const crypto = require("crypto");

const hash = crypto.createHash('sha512'); // Compliant
```

See

- OWASP - [Top 10 2021 Category A2 - Cryptographic Failures](#)
- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)
- OWASP - [Top 10 2017 Category A6 - Security Misconfiguration](#)
- CWE - [CWE-1240 - Use of a Risky Cryptographic Primitive](#)

The "print" statement should not be used

Key: python:PrintStatementUsage

Root Cause:

The `print` statement was removed in Python 3.0. The built-in function should be used instead.

Noncompliant code example

```
print '1' # Noncompliant
```

Compliant solution

```
print('1')
```

Introduction:

This rule raises an issue when the print statement is used.

Cryptographic private keys should not be disclosed

Key: secrets:S6706

Introduction:

Secret leaks often occur when a sensitive piece of authentication data is stored with the source code of an application. Considering the source code is intended to be deployed across multiple assets, including source code repositories or application hosting servers, the secrets might get exposed to an unintended audience.

Root Cause:

In most cases, trust boundaries are violated when a secret is exposed in a source code repository or an uncontrolled deployment environment.

Unintended people who don't need to know the secret might get access to it. They might then be able to use it to gain unwanted access to associated services or resources.

The trust issue can be more or less severe depending on the people's role and entitlement.

What is the potential impact?

In most cases, trust boundaries are violated when a secret is exposed in a source code repository or an uncontrolled deployment environment.

Unintended people who don't need to know the secret might get access to it. They might then be able to use it to gain unwanted access to associated services or resources.

The trust issue can be more or less severe depending on the people's role and entitlement.

How To Fix:

Revoke the secret

Revoke any leaked secrets and remove them from the application source code.

Before revoking the secret, ensure that no other applications or processes are using it. Other usages of the secret will also be impacted when the secret is revoked.

In most cases, if the key is used as part of a larger trust model (X509, PGP, etc), it is necessary to issue and publish a revocation certificate.

Doing so will ensure that all people and assets that rely on this key for security operations are aware of its compromise and stop trusting it.

Analyze recent secret use

When available, analyze authentication logs to identify any unintended or malicious use of the secret since its disclosure date. Doing this will allow determining if an attacker took advantage of the leaked secret and to what extent.

This operation should be part of a global incident response process.

Use a secret vault

A secret vault should be used to generate and store the new secret. This will ensure the secret's security and prevent any further unexpected disclosure.

Depending on the development platform and the leaked secret type, multiple solutions are currently available.

Noncompliant code example

```
private_key = "-----BEGIN EC PRIVATE KEY-----" \
    "MF8CAQEEGEfVxjrMPigNhGP6DqH6DPeUZPbaoaCCXaAKBggqhkjOPQMBAAe0AzIA" \
    "BCIxho34upZyXDi/AUy/TBisGeh4yKJN7pit9Z+nKs4QajVy97X8W9JdySlbWeRt" \
    "2w==" \
    "-----END EC PRIVATE KEY-----"
```

Compliant solution

```
with open("/path/to/private.key","r") as key_file:
    private_key = key_file.read()
```

Resources:

Standards

- OWASP - [Top 10 2021 Category A7 - Identification and Authentication Failures](#)
- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)
- CWE - [CWE-798 - Use of Hard-coded Credentials](#)
- CWE - [CWE-259 - Use of Hard-coded Password](#)
- STIG Viewer - [Application Security and Development: V-222642](#) - The application must not contain embedded authentication data.
- OWASP - [Mobile Top 10 2024 Category M1 - Improper Credential Usage](#)
- OWASP - [Mobile Top 10 2024 Category M10 -](#)

[Insufficient Cryptography](#)

Class names should comply with a naming convention

Key: typescript:S101

Root Cause:

Shared naming conventions allow teams to collaborate efficiently.

This rule raises an issue when a class name (or an interface for TypeScript) does not match a provided regular expression.

For example, with the default provided regular expression `^[A-Z][a-zA-Z0-9]*$`, the class:

```
class my_class {...} // Noncompliant
```

should be renamed to

```
class MyClass {...}
```

Unnecessary imports should be removed

Key: typescript:S1128

Root Cause:

Unnecessary imports refer to importing modules, libraries, or dependencies that are not used or referenced anywhere in the code. These imports do not contribute to the functionality of the application and only add extra weight to the JavaScript bundle, leading to potential performance and maintainability issues.

```
import A from 'a'; // Noncompliant: The imported symbol 'A' isn't used
import { B1 } from 'b';
```

```
console.log(B1);
```

To mitigate the problems associated with unnecessary imports, you should regularly review and remove any imports that are not being used. Modern JavaScript build tools and bundlers often provide features like tree shaking, which eliminates unused code during the bundling process, resulting in a more optimized bundle size.

```
import { B1 } from 'b';
```

```
console.log(B1);
```

Resources:

Documentation

- MDN web docs - [import](#)

Related rules

- [S1481](#) - Unused local variables and functions should be removed

Track uses of "FIXME" tags

Key: typescript:S1134

Resources:

Documentation

- CWE - [CWE-546 - Suspicious Comment](#)

Root Cause:

`FIXME` tags are commonly used to mark places where a bug is suspected, but which the developer wants to deal with later.

Sometimes the developer will not have the time or will simply forget to get back to that tag.

This rule is meant to track those tags and to ensure that they do not go unnoticed.

```
function divide(numerator, denominator) {  
    return numerator / denominator; // FIXME denominator value might be 0  
}
```

Track uses of "TODO" tags

Key: typescript:S1135

Resources:

- CWE - [CWE-546 - Suspicious Comment](#)

Root Cause:

Developers often use `TODO` tags to mark areas in the code where additional work or improvements are needed but are not implemented immediately. However, these `TODO` tags sometimes get overlooked or forgotten, leading to incomplete or unfinished code. This rule aims to identify and address unattended `TODO` tags to ensure a clean and maintainable codebase. This description explores why this is a problem and how it can be fixed to improve the overall code quality.

What is the potential impact?

Unattended `TODO` tags in code can have significant implications for the development process and the overall codebase.

Incomplete Functionality: When developers leave `TODO` tags without implementing the corresponding code, it results in incomplete functionality within the software. This can lead to unexpected behavior or missing features, adversely affecting the end-user experience.

Missed Bug Fixes: If developers do not promptly address `TODO` tags, they might overlook critical bug fixes and security updates.

Delayed bug fixes can result in more severe issues and increase the effort required to resolve them later.

Impact on Collaboration: In team-based development environments, unattended `TODO` tags can hinder collaboration. Other team members might not be aware of the intended changes, leading to conflicts or

redundant efforts in the codebase.

Codebase Bloat: The accumulation of unattended `TODO` tags over time can clutter the codebase and make it difficult to distinguish between work in progress and completed code. This bloat can make it challenging to maintain an organized and efficient codebase.

Addressing this code smell is essential to ensure a maintainable, readable, reliable codebase and promote effective collaboration among developers.

Noncompliant code example

```
function doSomething() {  
    // TODO  
}
```

Sections of code should not be commented out

Key: typescript:S125

Root Cause:

Commented-out code distracts the focus from the actual executed code. It creates a noise that increases maintenance code. And because it is never executed, it quickly becomes out of date and invalid.

Commented-out code should be deleted and can be retrieved from source control history if required.

"if" statements should be preferred over "switch" when simpler

Key: typescript:S1301

Resources:

Documentation

- MDN web docs - [switch](#)
- MDN web docs - [if...else](#)

Root Cause:

A `switch` statement is a control flow statement that allows you to execute different blocks of code based on the value of an expression. It provides a more concise way to handle multiple conditions compared to using multiple `if-else` statements.

If you only have a single condition to check, using an `if` statement is simpler and more concise. `switch` statements are designed for handling multiple cases, so using them for a single condition can be overkill and less readable.

This rule raises an issue when a `switch` statement has only one `case` clause and possibly a `default` one.

```
switch (condition) { // Noncompliant: The switch has only one case and a default  
    case 0:  
        doSomething();  
        break;  
    default:  
        doSomethingElse();
```



```
        break;
    }
```

Use a `switch` statement when you have multiple cases to handle and an `if` statement when you have only one condition to check.

```
if (condition === 0) {
    doSomething();
} else {
    doSomethingElse();
}
```

Using hardcoded IP addresses is security-sensitive

Key: typescript:S1313

How To Fix:

Recommended Secure Coding Practices

Don't hard-code the IP address in the source code, instead make it configurable with environment variables, configuration files, or a similar approach. Alternatively, if confidentiality is not required a domain name can be used since it allows to change the destination quickly without having to rebuild the software.

Compliant Solution

```
ip = process.env.IP_ADDRESS; // Compliant

const net = require('net');
var client = new net.Socket();
client.connect(80, ip, function() {
    // ...
});
```

See

- OWASP - [Top 10 2021 Category A1 - Broken Access Control](#)
- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)

Assess The Problem:

Ask Yourself Whether

The disclosed IP address is sensitive, e.g.:

- Can give information to an attacker about the network topology.
- It's a personal (assigned to an identifiable person) IP address.

There is a risk if you answered yes to any of these questions.

Sensitive Code Example

```
ip = "192.168.12.42"; // Sensitive

const net = require('net');
var client = new net.Socket();
client.connect(80, ip, function() {
  // ...
});
```

Root Cause:

Hardcoding IP addresses is security-sensitive. It has led in the past to the following vulnerabilities:

- [CVE-2006-5901](#)
- [CVE-2005-3725](#)

Today's services have an ever-changing architecture due to their scaling and redundancy needs. It is a mistake to think that a service will always have the same IP address. When it does change, the hardcoded IP will have to be modified too. This will have an impact on the product development, delivery, and deployment:

- The developers will have to do a rapid fix every time this happens, instead of having an operation team change a configuration file.
- It misleads to use the same address in every environment (dev, sys, qa, prod).

Last but not least it has an effect on application security. Attackers might be able to decompile the code and thereby discover a potentially sensitive address. They can perform a Denial of Service attack on the service, try to get access to the system, or try to spoof the IP address to bypass security checks. Such attacks can always be possible, but in the case of a hardcoded IP address solving the issue will take more time, which will increase an attack's impact.

Exceptions

No issue is reported for the following cases because they are not considered sensitive:

- Loopback addresses 127.0.0.0/8 in CIDR notation (from 127.0.0.0 to 127.255.255.255)
- Broadcast address 255.255.255.255
- Non routable address 0.0.0.0
- Strings of the form 2.5.<number>.<number> as they [often match Object Identifiers](#) (OID).
- Addresses in the ranges 192.0.2.0/24, 198.51.100.0/24, 203.0.113.0/24, reserved for documentation purposes by [href="https://datatracker.ietf.org/doc/html/rfc5737">RFC 5737](https://datatracker.ietf.org/doc/html/rfc5737)
- Addresses in the 2001:db8::/32 range, reserved for documentation purposes by [RFC 3849](#)

Default:

Hardcoding IP addresses is security-sensitive. It has led in the past to the following vulnerabilities:

- [CVE-2006-5901](#)

- [CVE-2005-3725](#)

Today's services have an ever-changing architecture due to their scaling and redundancy needs. It is a mistake to think that a service will always have the same IP address. When it does change, the hardcoded IP will have to be modified too. This will have an impact on the product development, delivery, and deployment:

- The developers will have to do a rapid fix every time this happens, instead of having an operation team change a configuration file.
- It misleads to use the same address in every environment (dev, sys, qa, prod).

Last but not least it has an effect on application security. Attackers might be able to decompile the code and thereby discover a potentially sensitive address. They can perform a Denial of Service attack on the service, try to get access to the system, or try to spoof the IP address to bypass security checks. Such attacks can always be possible, but in the case of a hardcoded IP address solving the issue will take more time, which will increase an attack's impact.

Ask Yourself Whether

The disclosed IP address is sensitive, e.g.:

- Can give information to an attacker about the network topology.
- It's a personal (assigned to an identifiable person) IP address.

There is a risk if you answered yes to any of these questions.

Recommended Secure Coding Practices

Don't hard-code the IP address in the source code, instead make it configurable with environment variables, configuration files, or a similar approach. Alternatively, if confidentiality is not required a domain name can be used since it allows to change the destination quickly without having to rebuild the software.

Sensitive Code Example

```
ip = "192.168.12.42"; // Sensitive

const net = require('net');
var client = new net.Socket();
client.connect(80, ip, function() {
  // ...
});
```

Compliant Solution

```
ip = process.env.IP_ADDRESS; // Compliant

const net = require('net');
var client = new net.Socket();
client.connect(80, ip, function() {
  // ...
});
```

Exceptions

No issue is reported for the following cases because they are not considered sensitive:

- Loopback addresses 127.0.0.0/8 in CIDR notation (from 127.0.0.0 to 127.255.255.255)
- Broadcast address 255.255.255.255
- Non routable address 0.0.0.0
- Strings of the form `2.5.<number>.<number>` as they [often match Object Identifiers](#) (OID).
- Addresses in the ranges 192.0.2.0/24, 198.51.100.0/24, 203.0.113.0/24, reserved for documentation purposes by [href="https://datatracker.ietf.org/doc/html/rfc5737">RFC 5737](https://datatracker.ietf.org/doc/html/rfc5737)
- Addresses in the 2001:db8::/32 range, reserved for documentation purposes by [RFC 3849](#)

See

- OWASP - [Top 10 2021 Category A1 - Broken Access Control](#)
- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)

Dynamically executing code is security-sensitive

Key: typescript:S1523

Root Cause:

Executing code dynamically is security-sensitive. It has led in the past to the following vulnerabilities:

- [CVE-2017-9807](#)
- [CVE-2017-9802](#)

Some APIs enable the execution of dynamic code by providing it as strings at runtime. These APIs might be useful in some very specific meta-programming use-cases. However most of the time their use is frowned upon as they also increase the risk of [Injected Code](#). Such attacks can either run on the server or in the client (example: XSS attack) and have a huge impact on an application's security.

This rule raises issues on calls to `eval` and `Function` constructor. This rule does not detect code injections. It only highlights the use of APIs which should be used sparingly and very carefully. The goal is to guide security code reviews.

The rule also flags string literals starting with `javascript:` as the code passed in `javascript:` URLs is evaluated the same way as calls to `eval` or `Function` constructor.

Exceptions

This rule will not raise an issue when the argument of the `eval` or `Function` is a literal string as it is reasonably safe.

Assess The Problem:

Ask Yourself Whether

- the executed code may come from an untrusted source and hasn't been sanitized.
- you really need to run code dynamically.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

```
let value = eval('obj.' + propName); // Sensitive
let func = Function('obj' + propName); // Sensitive
location.href = 'javascript:void(0)'; // Sensitive
```

Default:

Executing code dynamically is security-sensitive. It has led in the past to the following vulnerabilities:

- [CVE-2017-9807](#)
- [CVE-2017-9802](#)

Some APIs enable the execution of dynamic code by providing it as strings at runtime. These APIs might be useful in some very specific meta-programming use-cases. However most of the time their use is frowned upon as they also increase the risk of [Injected Code](#). Such attacks can either run on the server or in the client (example: XSS attack) and have a huge impact on an application's security.

This rule raises issues on calls to `eval` and `Function` constructor. This rule does not detect code injections. It only highlights the use of APIs which should be used sparingly and very carefully. The goal is to guide security code reviews.

The rule also flags string literals starting with `javascript:` as the code passed in `javascript:` URLs is evaluated the same way as calls to `eval` or `Function` constructor.

Ask Yourself Whether

- the executed code may come from an untrusted source and hasn't been sanitized.
- you really need to run code dynamically.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

Regarding the execution of unknown code, the best solution is to not run code provided by an untrusted source. If you really need to do it, run the code in a [sandboxed](#) environment. Use jails, firewalls and whatever means your operating system and programming language provide (example: [Security Managers](#) in java, [iframes](#) and [same-origin policy](#) for javascript in a web browser).

Do not try to create a blacklist of dangerous code. It is impossible to cover all attacks that way.

Avoid using dynamic code APIs whenever possible. Hard-coded code is always safer.

Sensitive Code Example

```
let value = eval('obj.' + propName); // Sensitive
let func = Function('obj' + propName); // Sensitive
location.href = 'javascript:void(0)'; // Sensitive
```

Exceptions

This rule will not raise an issue when the argument of the `eval` or `Function` is a literal string as it is reasonably safe.

See

- OWASP - [Top 10 2021 Category A3 - Injection](#)
- OWASP - [Top 10 2017 Category A1 - Injection](#)
- CWE - [CWE-95 - Improper Neutralization of Directives in Dynamically Evaluated Code \('Eval Injection'\)](#)

How To Fix:

Recommended Secure Coding Practices

Regarding the execution of unknown code, the best solution is to not run code provided by an untrusted source. If you really need to do it, run the code in a [sandboxed](#) environment. Use jails, firewalls and whatever means your operating system and programming language provide (example: [Security Managers](#) in java, [iframes](#) and [same-origin policy](#) for javascript in a web browser).

Do not try to create a blacklist of dangerous code. It is impossible to cover all attacks that way.

Avoid using dynamic code APIs whenever possible. Hard-coded code is always safer.

See

- OWASP - [Top 10 2021 Category A3 - Injection](#)
- OWASP - [Top 10 2017 Category A1 - Injection](#)
- CWE - [CWE-95 - Improper Neutralization of Directives in Dynamically Evaluated Code \('Eval Injection'\)](#)

Objects should not be created to be dropped immediately without being used

Key: typescript:S1848

Root Cause:

Creating an object without assigning it to a variable or using it in any function means the object is essentially created for no reason and may be dropped immediately without being used. Most of the time, this is due to a missing piece of code and could lead to an unexpected behavior.

If it's intended because the constructor has side effects, that side effect should be moved into a separate method and called directly. This can help to improve the performance and readability of the code.

```
new MyConstructor(); // Noncompliant: object may be dropped
```

Determine if the objects are necessary for the code to function correctly. If they are not required, remove them from the code. Otherwise, assign them to a variable for later use.

```
let something = new MyConstructor();
```

Exceptions

- Creating new objects inside a `try` block is ignored.

```
try {  
    new MyConstructor();  
} catch (e) {  
    /* ... */  
}
```

- Known constructors with side effects like `Notification` or `Vue` are also ignored.

Resources:

Documentation

- MDN web docs - [href="https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/constructor">Object.prototype.constructor](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/constructor)
- MDN web docs - [constructor](#)
- MDN web docs - [new operator](#)

Unused assignments should be removed

Key: typescript:S1854

How To Fix:

Remove the unnecessary assignment, then test the code to make sure that the right-hand side of a given assignment had no side effects (e.g. a method that writes certain data to a file and returns the number of written bytes).

Noncompliant code example

```
function foo(y) {  
    let x = 100; // Noncompliant: dead store  
    x = 150; // Noncompliant: dead store  
    x = 200;  
    return x + y;  
}
```

Compliant solution

```
function foo(y) {  
    let x = 200; // Compliant: no unnecessary assignment
```

```
    return x + y;  
}
```

Root Cause:

Dead stores refer to assignments made to local variables that are subsequently never used or immediately overwritten. Such assignments are unnecessary and don't contribute to the functionality or clarity of the code. They may even negatively impact performance. Removing them enhances code cleanliness and readability. Even if the unnecessary operations do not do any harm in terms of the program's correctness, they are - at best - a waste of computing resources.

Exceptions

The rule ignores

- Initializations to `-1`, `0`, `1`, `undefined`, `[]`, `{}`, `true`, `false` and `""`.
- Variables that start with an underscore (e.g. `_unused`) are ignored.
- Assignment of `null` is ignored because it is sometimes used to help garbage collection
- Increment and decrement expressions are ignored because they are often used idiomatically instead of `x+1`
- This rule also ignores variables declared with object destructuring using rest syntax (used to exclude some properties from object)

```
let {a, b, ...rest} = obj; // 'a' and 'b' are compliant  
doSomething(rest);
```

```
let [x1, x2, x3] = arr; // 'x1' is noncompliant, as omitting syntax can be used: "let  
[, x2, x3] = arr;"  
doSomething(x2, x3);
```

Resources:

Standards

- CWE - [CWE-563 - Assignment to Variable without Use \('Unused Variable'\)](#)

Related rules

- [S1763](#) - All code should be reachable
- [S2589](#) - Boolean expressions should not be gratuitous
- [S3516](#) - Function returns should not be invariant
- [S3626](#) - Jump statements should not be redundant

Two branches in a conditional structure should not have exactly the same implementation

Key: typescript:S1871

Root Cause:

When the same code is duplicated in two or more separate branches of a conditional, it can make the code harder to understand, maintain, and can potentially introduce bugs if one instance of the code is changed but others are not.

Having two cases in a switch statement or two branches in an if chain with the same implementation is at best duplicate code, and at worst a coding error.

```
if (a >= 0 && a < 10) {
    doFirstThing();
    doTheThing();
}
else if (a >= 10 && a < 20) {
    doTheOtherThing();
}
else if (a >= 20 && a < 50) {
    doFirstThing();
    doTheThing(); // Noncompliant; duplicates first condition
}
else {
    doTheRest();
}
```

```
switch (i) {
    case 1:
        doFirstThing();
        doSomething();
        break;
    case 2:
        doSomethingDifferent();
        break;
    case 3: // Noncompliant; duplicates case 1's implementation
        doFirstThing();
        doSomething();
        break;
    default:
        doTheRest();
}
```

If the same logic is truly needed for both instances, then:

- in an if chain they should be combined

```
if ((a >= 0 && a < 10) || (a >= 20 && a < 50)) { // Compliant
    doFirstThing();
    doTheThing();
}
else if (a >= 10 && a < 20) {
    doTheOtherThing();
}
else {
    doTheRest();
}
```

- for a `switch`, one should fall through to the other

```
switch (i) {  
  case 1:  
  case 3: // Compliant  
    doFirstThing();  
    doSomething();  
    break;  
  case 2:  
    doSomethingDifferent();  
    break;  
  default:  
    doTheRest();  
}
```

When all blocks are identical, either this rule will trigger if there is no default clause or rule [S3923](#) will raise if there is a default clause.

Exceptions

Unless all blocks are identical, blocks in an `if` chain that contain a single line of code are ignored. The same applies to blocks in a `switch` statement that contains a single line of code with or without a following `break`.

```
if (a == 1) {  
  doSomething(); // Compliant, usually this is done on purpose to increase the  
  readability  
} else if (a == 2) {  
  doSomethingElse();  
} else {  
  doSomething();  
}
```

Resources:

Related rules

- [S3923](#) - All branches in a conditional structure should not have exactly the same implementation

Deprecated APIs should not be used

Key: typescript:S1874

Resources:

Documentation

- CWE - [CWE-477 - Use of Obsolete Functions](#)

Root Cause:

Code is sometimes annotated as deprecated by developers maintaining libraries or APIs to indicate that the method, class, or other programming element is no longer recommended for use. This is typically due to the introduction of a newer or more effective alternative. For example, when a better solution has been

identified, or when the existing code presents potential errors or security risks.

Deprecation is a good practice because it helps to phase out obsolete code in a controlled manner, without breaking existing software that may still depend on it. It is a way to warn other developers not to use the deprecated element in new code, and to replace it in existing code when possible.

Deprecated classes, interfaces, and their members should not be used, inherited or extended because they will eventually be removed. The deprecation period allows you to make a smooth transition away from the aging, soon-to-be-retired technology.

Check the documentation or the deprecation message to understand why the code was deprecated and what the recommended alternative is.

```
/**
 * @deprecated Use newFunction instead.
 */
function oldFunction() {
    console.log("This is the old function.");
}

function newFunction() {
    console.log("This is the new function.");
}

oldFunction(); // Noncompliant: "oldFunction is deprecated"
```

Functions should not be nested too deeply

Key: typescript:S2004

Root Cause:

Nested functions refer to the practice of defining a function within another function. These inner functions have access to the variables and parameters of the outer function, creating a closure.

While nesting functions is a common practice in JavaScript, deeply nested functions can make the code harder to read and understand, especially if the functions are long or if there are many levels of nesting.

This can make it difficult for other developers or even yourself to understand and maintain the code.

Noncompliant code example

With the default threshold of 4 levels:

```
function f() {
    function f_inner() {
        function f_inner_inner() {
            function f_inner_inner_inner() {
                function f_inner_inner_inner_inner() { // Noncompliant
                }
            }
        }
    }
}
```

Resources:

Documentation

- MDN web docs - [Nested functions and closures](#)

Hard-coded passwords are security-sensitive

Key: typescript:S2068

Assess The Problem:

Ask Yourself Whether

- Passwords allow access to a sensitive component like a database, a file storage, an API or a service.
- Passwords are used in production environments.
- Application re-distribution is required before updating the passwords.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

```
const mysql = require('mysql');

const connection = mysql.createConnection({
  host: 'localhost',
  user: "admin",
  database: "project",
  password: "mypassword", // sensitive
  multipleStatements: true
});

connection.connect();
```

Default:

Because it is easy to extract strings from an application source code or binary, passwords should not be hard-coded. This is particularly true for applications that are distributed or that are open-source.

In the past, it has led to the following vulnerabilities:

- [CVE-2019-13466](#)
- [CVE-2018-15389](#)

Passwords should be stored outside of the code in a configuration file, a database, or a management service for passwords.

This rule flags instances of hard-coded passwords used in database and LDAP connections. It looks for hard-coded passwords in connection strings, and for variable names that match any of the patterns from the provided list.

Ask Yourself Whether

- Passwords allow access to a sensitive component like a database, a file storage, an API or a service.

- Passwords are used in production environments.
- Application re-distribution is required before updating the passwords.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Store the passwords in a configuration file that is not pushed to the code repository.
- Store the passwords in a database.
- Use your cloud provider's service for managing passwords.
- If a password has been disclosed through the source code: change it.

Sensitive Code Example

```
const mysql = require('mysql');

const connection = mysql.createConnection({
  host: 'localhost',
  user: "admin",
  database: "project",
  password: "mypassword", // sensitive
  multipleStatements: true
});

connection.connect();
```

Compliant Solution

```
const mysql = require('mysql');

const connection = mysql.createConnection({
  host: process.env.MYSQL_URL,
  user: process.env.MYSQL_USERNAME,
  password: process.env.MYSQL_PASSWORD,
  database: process.env.MYSQL_DATABASE
});

connection.connect();
```

See

- OWASP - [Top 10 2021 Category A7 - Identification and Authentication Failures](#)
- OWASP - [Top 10 2017 Category A2 - Broken Authentication](#)
- CWE - [CWE-259 - Use of Hard-coded Password](#)
- Derived from FindSecBugs rule [Hard Coded Password](#)

How To Fix:

Recommended Secure Coding Practices

- Store the passwords in a configuration file that is not pushed to the code repository.
- Store the passwords in a database.
- Use your cloud provider's service for managing passwords.
- If a password has been disclosed through the source code: change it.

Compliant Solution

```
const mysql = require('mysql');

const connection = mysql.createConnection({
  host: process.env.MYSQL_URL,
  user: process.env.MYSQL_USERNAME,
  password: process.env.MYSQL_PASSWORD,
  database: process.env.MYSQL_DATABASE
});
connection.connect();
```

See

- OWASP - [Top 10 2021 Category A7 - Identification and Authentication Failures](#)
- OWASP - [Top 10 2017 Category A2 - Broken Authentication](#)
- CWE - [CWE-259 - Use of Hard-coded Password](#)
- Derived from FindSecBugs rule [Hard Coded Password](#)

Root Cause:

Because it is easy to extract strings from an application source code or binary, passwords should not be hard-coded. This is particularly true for applications that are distributed or that are open-source.

In the past, it has led to the following vulnerabilities:

- [CVE-2019-13466](#)
- [CVE-2018-15389](#)

Passwords should be stored outside of the code in a configuration file, a database, or a management service for passwords.

This rule flags instances of hard-coded passwords used in database and LDAP connections. It looks for hard-coded passwords in connection strings, and for variable names that match any of the patterns from the provided list.

Using pseudorandom number generators (PRNGs) is security-sensitive

Key: typescript:S2245

Root Cause:

PRNGs are algorithms that produce sequences of numbers that only approximate true randomness. While they are suitable for applications like simulations or modeling, they are not appropriate for security-sensitive contexts because their outputs can be predictable if the internal state is known.

In contrast, cryptographically secure pseudorandom number generators (CSPRNGs) are designed to be secure against prediction attacks. CSPRNGs use cryptographic algorithms to ensure that the generated sequences are not only random but also unpredictable, even if part of the sequence or the internal state becomes known. This unpredictability is crucial for security-related tasks such as generating encryption keys, tokens, or any other values that must remain confidential and resistant to guessing attacks.

For example, the use of non-cryptographic PRNGs has led to vulnerabilities such as:

- [CVE-2013-6386](#)
- [CVE-2006-3419](#)
- [CVE-2008-4102](#)

When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information. Therefore, it is critical to use CSPRNGs in any security-sensitive application to ensure the robustness and security of the system.

As the `Math.random()` function relies on a weak pseudorandom number generator, this function should not be used for security-critical applications or for protecting sensitive data. In such context, a cryptographically strong pseudorandom number generator (CSPRNG) should be used instead.

Assess The Problem:

Ask Yourself Whether

- the code using the generated value requires it to be unpredictable. It is the case for all encryption mechanisms or when a secret value, such as a password, is hashed.
- the function you use is a non-cryptographic PRNG.
- the generated value is used multiple times.
- an attacker can access the generated value.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

```
const val = Math.random(); // Sensitive
// Check if val is used in a security context.
```

Default:

PRNGs are algorithms that produce sequences of numbers that only approximate true randomness. While they are suitable for applications like simulations or modeling, they are not appropriate for security-sensitive contexts because their outputs can be predictable if the internal state is known.

In contrast, cryptographically secure pseudorandom number generators (CSPRNGs) are designed to be secure against prediction attacks. CSPRNGs use cryptographic algorithms to ensure that the generated sequences are not only random but also unpredictable, even if part of the sequence or the internal state becomes known. This unpredictability is crucial for security-related tasks such as generating encryption keys, tokens, or any other values that must remain confidential and resistant to guessing attacks.

For example, the use of non-cryptographic PRNGs has led to vulnerabilities such as:

- [CVE-2013-6386](#)
- [CVE-2006-3419](#)
- [CVE-2008-4102](#)

When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information. Therefore, it is critical to use CSPRNGs in any security-sensitive application to ensure the robustness and security of the system.

As the `Math.random()` function relies on a weak pseudorandom number generator, this function should not be used for security-critical applications or for protecting sensitive data. In such context, a cryptographically strong pseudorandom number generator (CSPRNG) should be used instead.

Ask Yourself Whether

- the code using the generated value requires it to be unpredictable. It is the case for all encryption mechanisms or when a secret value, such as a password, is hashed.
- the function you use is a non-cryptographic PRNG.
- the generated value is used multiple times.
- an attacker can access the generated value.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Use a cryptographically secure pseudorandom number generator (CSPRNG) like `crypto.getRandomValues()`.
- Use the generated random values only once.
- You should not expose the generated random value. If you have to store it, make sure that the database or file is secure.

Sensitive Code Example

```
const val = Math.random(); // Sensitive
// Check if val is used in a security context.
```

Compliant Solution


```
// === Client side ===
const crypto = window.crypto || window.msCrypto;
var array = new Uint32Array(1);
crypto.getRandomValues(array);

// === Server side ===
const crypto = require('crypto');
const buf = crypto.randomBytes(1);
```

See

- OWASP - [Secure Random Number Generation Cheat Sheet](#)
- OWASP - [Top 10 2021 Category A2 - Cryptographic Failures](#)
- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)
- CWE - [CWE-338 - Use of Cryptographically Weak Pseudo-Random Number Generator \(PRNG\)](#)
- CWE - [CWE-330 - Use of Insufficiently Random Values](#)
- CWE - [CWE-326 - Inadequate Encryption Strength](#)
- CWE - [CWE-1241 - Use of Predictable Algorithm in Random Number Generator](#)

How To Fix:

Recommended Secure Coding Practices

- Use a cryptographically secure pseudorandom number generator (CSPRNG) like `crypto.getRandomValues()`.
- Use the generated random values only once.
- You should not expose the generated random value. If you have to store it, make sure that the database or file is secure.

Compliant Solution

```
// === Client side ===
const crypto = window.crypto || window.msCrypto;
var array = new Uint32Array(1);
crypto.getRandomValues(array);

// === Server side ===
const crypto = require('crypto');
const buf = crypto.randomBytes(1);
```

See

- OWASP - [Secure Random Number Generation Cheat Sheet](#)

- OWASP - [Top 10 2021 Category A2 - Cryptographic Failures](#)
- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)
- CWE - [CWE-338 - Use of Cryptographically Weak Pseudo-Random Number Generator \(PRNG\)](#)
- CWE - [CWE-330 - Use of Insufficiently Random Values](#)
- CWE - [CWE-326 - Inadequate Encryption Strength](#)
- CWE - [CWE-1241 - Use of Predictable Algorithm in Random Number Generator](#)

Exceptions should not be ignored

Key: typescript:S2486

Root Cause:

When exceptions occur, it is usually a bad idea to simply ignore them. Instead, it is better to handle them properly, or at least to log them.

Noncompliant code example

```
function f() {  
    try {  
        doSomething();  
    } catch (err) {  
    }  
}
```

Compliant solution

```
function f() {  
    try {  
        doSomething();  
    } catch (err) {  
        console.log(`Exception while doing something: ${err}`);  
    }  
}
```

Resources:

- OWASP - [Top 10 2021 Category A9 - Security Logging and Monitoring Failures](#)
- OWASP - [Top 10 2017 Category A10 - Insufficient Logging & Monitoring](#)
- CWE - [CWE-390 - Detection of Error Condition Without Action](#)

Ternary operators should not be nested

Key: typescript:S3358

Root Cause:

Nested ternaries are hard to read and can make the order of operations complex to understand.

```
function getReadableStatus(job) {  
    return job.isRunning() ? "Running" : job.hasErrors() ? "Failed" : "Succeeded "; //  
Noncompliant  
}
```

Instead, use another line to express the nested operation in a separate statement.

```
function getReadableStatus(job) {  
    if (job.isRunning()) {  
        return "Running";  
    }  
    return job.hasErrors() ? "Failed" : "Succeeded";  
}
```

Exceptions

This rule does not apply in JSX expressions to support conditional rendering and conditional attributes as long as the nesting happens in separate JSX expression containers, i.e. JSX elements embedding JavaScript code, as shown below:

```
return (  
<>  
    {isLoading ? (  
        <Loader active />  
    ) : (  
        <Panel label={isEditing ? 'Open' : 'Not open'}>  
            <a>{isEditing ? 'Close now' : 'Start now'}</a>  
            <Checkbox onClick={!saving ? setSaving(saving => !saving) : null} />  
        </Panel>  
    )}  
</>  
);
```

If you have nested ternaries in the same JSX expression container, refactor your logic into a separate function like that:

```
function myComponent(condition) {  
    if (condition < 0) {  
        return '<DownSign>it is negative</DownSign>';  
    } else if (condition > 0) {  
        return '<UpSign>it is positive</UpSign>';  
    } else {  
        return '<BarSign>it is zero</BarSign>';  
    }  
}  
  
return (  
    {myComponent(foo)}  
);
```

```
);
```

Resources:

Articles & blog posts

- Sonar - [Stop nesting ternaries in JavaScript](#)

Variables should be declared with "let" or "const"

Key: typescript:S3504

Root Cause:

Variables declared with `var` are function-scoped, meaning they are accessible within the entire function in which they are defined. If a variable is declared using `var` outside of any function, it becomes a global variable and is accessible throughout the entire JavaScript program.

`let` and `const` were introduced in ECMAScript 6 (ES6) as a block-scoped alternative to `var`. Variables declared with `let` have block scope, meaning they are limited to the block of code in which they are defined. A block is typically delimited by curly braces `{}`.

Variables declared with `const` are also block-scoped, similar to `let`. However, `const` variables are immutable, meaning their value cannot be changed after assignment. This applies to the binding between the variable name and its value, but it does not mean the value itself is immutable if it is an object or an array.

A variable declared with `let` or `const` is said to be in a "temporal dead zone", meaning the period between entering a scope and declaring a `let` or `const` variable. During this phase, accessing the variable results in a

`ReferenceError`. This helps catch potential errors and encourages proper variable declaration.

Unlike `let` and `const`, variables declared with `var` are subject to "hoisting", which means that they are moved to the top of their scope during the compilation phase, even if the actual declaration is placed lower in the code.

Hoisting can sometimes lead to unexpected behavior. For example, variables declared with `var` are accessible before they are declared, although they will have the value `undefined` until the declaration is reached.

The distinction between the variable types created by `var` and by `let` is significant, and a switch to `let` will help alleviate many of the variable scope issues which have caused confusion in the past.

Because these new keywords create more precise variable types, they are preferred in environments that support ECMAScript 2015. However, some refactoring may be required by the switch from `var` to `let`, and you should be aware that they raise `SyntaxErrors` in pre-ECMAScript 2015 environments.

This rule raises an issue when `var` is used instead of `const` or `let`.

```
var color = "blue"; // Noncompliant
var size = 4; // Noncompliant
```

You should declare your variables with either `const` or `let` depending on whether you are going to modify them afterwards.

```
const color = "blue";  
let size = 4;
```

Resources:

Documentation

- MDN web docs - [var](#)
- MDN web docs - [let](#)
- MDN web docs - [const](#)
- MDN web docs - [Scope](#)
- MDN web docs - [Hoisting](#)
- MDN web docs - [Temporal dead zone \(TDZ\)](#)

"void" should not be used

Key: typescript:S3735

Root Cause:

The `void` operator evaluates its argument and always returns `undefined`. `void` allows using any expression where an `undefined` is expected. However, using `void` makes code more difficult to understand, as the intent is often unclear.

```
if (parameter === void 42) { // Noncompliant  
    // ...  
}  
doSomethingElse(void doSomething()); // Noncompliant
```

Instead of using `void` to get the `undefined` value, use the `undefined` global property. In ECMAScript5 and newer environments, `undefined` cannot be reassigned. In other cases, remove the `void` operator to avoid confusion for maintainers.

```
if (parameter === undefined) {  
    // ...  
}  
doSomething();  
doSomethingElse();
```

Exceptions

- `void 0` (or the equivalent `void(0)`) is allowed as it was a conventional way to obtain the `undefined` value in environments before ECMAScript 5.

```
if (parameter === void 0) {  
    // ...  
}
```

- `void` is allowed with immediately invoked function expressions.

```
void function() {  
    // ...  
}();
```

- `void` is allowed with Promise-like objects to mark a promise as intentionally not awaited, as advised by [@typescript-eslint/no-floating-promises](https://github.com/typescript-eslint/typescript-eslint/blob/main/packages/eslint-plugin/docs/rules/no-floating-promises.md).

```
const runPromise = () => Promise.resolve();  
void runPromise();
```

Resources:

Documentation

- MDN web docs - [void operator](#)
- MDN web docs - [undefined](#)
- MDN web docs - [IIFE \(Immediately Invoked Function Expression\)](#)

Cognitive Complexity of functions should not be too high

Key: typescript:S3776

Resources:

Documentation

- Sonar - [Cognitive Complexity](#)

Articles & blog posts

- Sonar Blog - [5 Clean Code Tips for Reducing Cognitive Complexity](#)

How To Fix:

Reducing cognitive complexity can be challenging.
Here are a few suggestions:

- **Extract complex conditions in a new function.**

Mixed operators in condition will increase complexity. Extracting the condition in a new function with an appropriate name will reduce cognitive load.

- **Break down large functions.**

Large functions can be hard to understand and maintain. If a function is doing too many things, consider breaking it down into smaller, more manageable functions. Each function should have a single responsibility.

- **Avoid deep nesting by returning early.**

To avoid the nesting of conditions, process exceptional cases first and return early.

- **Use null-safe operations (if available in the language).**

When available the `?.?` or `??` operator replaces multiple tests and simplifies the flow.

Extraction of a complex condition in a new function.

Noncompliant code example

The code is using a complex condition and has a cognitive cost of 3.

```
function calculateFinalPrice(user, cart) {  
  let total = calculateTotal(cart);  
  if (user.hasMembership // +1 (if)  
    && user.orders > 10 // +1 (more than one condition)  
    && user.accountActive  
    && !user.hasDiscount  
    || user.orders === 1) { // +1 (change of operator in condition)  
    total = applyDiscount(user, total);  
  }  
  return total;  
}
```

Compliant solution

Even if the cognitive complexity of the whole program did not change, it is easier for a reader to understand the code of the

`calculateFinalPrice` function, which now only has a cognitive cost of 1.

```
function calculateFinalPrice(user, cart) {  
  let total = calculateTotal(cart);  
  if (isEligibleForDiscount(user)) { // +1 (if)  
    total = applyDiscount(user, total);  
  }  
  return total;  
}  
  
function isEligibleForDiscount(user) {  
  return user.hasMembership  
    && user.orders > 10 // +1 (more than one condition)  
    && user.accountActive  
    && !user.hasDiscount  
    || user.orders === 1 // +1 (change of operator in condition)  
}
```

Break down large functions.

Noncompliant code example

For example, consider a function that calculates the total price of a shopping cart, including sales tax and shipping.

Note: The code is simplified here, to illustrate the purpose. Please imagine there is more happening in the for loops.

```
function calculateTotal(cart) {  
  let total = 0;  
  for (let i = 0; i < cart.length; i++) { // +1 (for)  
    total += cart[i].price;  
  }  
  
  // calculateSalesTax  
  for (let i = 0; i < cart.length; i++) { // +1 (for)  
    total += 0.2 * cart[i].price;  
  }  
  
  //calculateShipping  
  total += 5 * cart.length;  
  
  return total;  
}
```

This function could be refactored into smaller functions: The complexity is spread over multiple functions and the complex `calculateTotal` has now a complexity score of zero.

Compliant solution

```
function calculateTotal(cart) {  
  let total = calculateSubtotal(cart);  
  total += calculateSalesTax(cart);  
  total += calculateShipping(cart);  
  return total;  
}  
  
function calculateSubtotal(cart) {  
  let subTotal = 0;  
  for (const item of cart) { // +1 (for)  
    subTotal += item.price;  
  }  
  return subTotal;  
}  
  
function calculateSalesTax(cart) {  
  let salesTax = 0;  
  for (const item of cart) { // +1 (for)  
    salesTax += 0.2 * item.price;  
  }  
  return salesTax;  
}  
  
function calculateShipping(cart) {  
  return 5 * cart.length;  
}
```

Avoid deep nesting by returning early.

Noncompliant code example

The below code has a cognitive complexity of 6.

```
function calculateDiscount(price, user) {
  if (isEligibleForDiscount(user)) { // +1 ( if )
    if (user?.hasMembership) { // +2 ( nested if )
      return price * 0.9;
    } else if (user?.orders === 1) { // +1 ( else )
      return price * 0.95;
    } else { // +1 ( else )
      return price;
    }
  } else { // +1 ( else )
    return price;
  }
}
```

Compliant solution

Checking for the edge case first flattens the `if` statements and reduces the cognitive complexity to 3.

```
function calculateDiscount(price, user) {
  if (!isEligibleForDiscount(user)) { // +1 ( if )
    return price;
  }
  if (user?.hasMembership) { // +1 ( if )
    return price * 0.9;
  }
  if (user?.orders === 1) { // +1 ( if )
    return price * 0.95;
  }
  return price;
}
```

Use the optional chaining operator to access data.

In the below code, the cognitive complexity is increased due to the multiple checks required to access the manufacturer's name. This can be simplified using the optional chaining operator.

Noncompliant code example

```
let manufacturerName = null;

if (product && product.details && product.details.manufacturer) { // +1 (if) +1
  (multiple condition)
  manufacturerName = product.details.manufacturer.name;
}
if (manufacturerName) { // +1 (if)
  console.log(manufacturerName);
} else {
  console.log('Manufacturer name not found');
```

```
}
```

Compliant solution

The optional chaining operator will return `undefined` if any reference in the chain is `undefined` or `null`, avoiding multiple checks:

```
let manufacturerName = product?.details?.manufacturer?.name;

if (manufacturerName) { // +1 (if)
  console.log(manufacturerName);
} else {
  console.log('Manufacturer name not found');
}
```

Pitfalls

As this code is complex, ensure that you have unit tests that cover the code before refactoring.

Introduction:

This rule raises an issue when the code cognitive complexity of a function is above a certain threshold.

Root Cause:

Cognitive Complexity is a measure of how hard it is to understand the control flow of a unit of code. Code with high cognitive complexity is hard to read, understand, test, and modify.

As a rule of thumb, high cognitive complexity is a sign that the code should be refactored into smaller, easier-to-manage pieces.

Which syntax in code does impact cognitive complexity score?

Here are the core concepts:

- **Cognitive complexity is incremented each time the code breaks the normal linear reading flow.**

This concerns, for example, loop structures, conditionals, catches, switches, jumps to labels, and conditions mixing multiple operators.

- **Each nesting level increases complexity.**

During code reading, the deeper you go through nested layers, the harder it becomes to keep the context in mind.

- **Method calls are free**

A well-picked method name is a summary of multiple lines of code. A reader can first explore a high-level view of what the code is performing then go deeper and deeper by looking at called functions content.

Note: This does not apply to recursive calls, those will increment cognitive score.

The method of computation is fully detailed in the pdf linked in the resources.

Note that the calculation of cognitive complexity at function level deviates from the documented process. Given the functional nature of JavaScript, nesting functions is a prevalent practice, especially within frameworks like React.js. Consequently, the cognitive complexity of functions remains independent from one another. This means that the complexity of a nesting function does not increase with the complexity of

nested functions.

What is the potential impact?

Developers spend more time reading and understanding code than writing it. High cognitive complexity slows down changes and increases the cost of maintenance.

Exceptions

Cognitive complexity calculations exclude logical expressions using the `||` and `??` operators.

```
function greet(name) {  
  name = name || 'Guest';  
  console.log('Hello, ' + name + '!');  
}
```

Array-mutating methods should not be used misleadingly

Key: typescript:S4043

Resources:

Documentation

- MDN web docs - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#copying_methods_and_mutating_methods
- MDN web docs - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/reverse
- MDN web docs - [Array.prototype.sort\(\)](#)
- MDN web docs - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/toReversed
- MDN web docs - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/toSorted
- MDN web docs - [Spread syntax \(...\)](#)
- MDN web docs - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/slice

Root Cause:

In JavaScript, some `Array` methods do not mutate the existing array that the method was called on, but instead return a new array.

Other methods mutate the array, and their return value differs depending on the method.

`reverse` and `sort` are mutating methods and, in addition, return the altered version. This rule raises an issue when the return values of these methods are assigned, which could lead maintainers to overlook the fact that the original array has been modified.

```
const reversed = a.reverse(); // Noncompliant: mutating method, no need to assign
return value
const sorted = b.sort(); // Noncompliant: mutating method, no need to assign return
value
```

Remove the assignment, so that the intent of mutating the original array is clear.

```
a.reverse();
b.sort();
```

Or use non-mutating alternatives `toReversed` and `toSorted`.

```
const reversed = a.toReversed();
const sorted = b.toSorted();
```

Alternatively, change a mutating method into a non-mutating alternative using the spread syntax (`...`).

```
const reversed = [...a].reverse();
const sorted = [...b].sort();
```

Or `slice()` to create a copy first.

```
const reversed = a.slice().reverse();
const sorted = b.slice().sort();
```

"for of" should be used with Iterables

Key: typescript:S4138

Root Cause:

`for...of` statements are used to iterate over the values of an iterable object. [Iterables](#) are objects implementing the `@@iterator` method, which returns an object conforming to the [iterator protocol](#). JavaScript provides many [built-in iterables](#) that can and should be used with this looping statement.

The use of the `for...of` statement is recommended over the `for` statement when iterating through iterable objects as simplifies the syntax and eliminates the need for a counter variable.

```
const arr = [4, 3, 2, 1];

for (let i = 0; i < arr.length; i++) { // Noncompliant: arr is an iterable object
  console.log(arr[i]);
}
```

When looping over an iterable, use the `for...of` for better readability.

```
const arr = [4, 3, 2, 1];

for (let value of arr) {
  console.log(value);
}
```

```
}
```

Resources:

Documentation

- MDN web docs - [for...of](#)
- MDN web docs - [iterator protocol](#)

Assignments should not be redundant

Key: typescript:S4165

Root Cause:

The transitive property says that if $a == b$ and $b == c$, then $a == c$. In such cases, there's no point in assigning a to c or vice versa because they're already equivalent.

This rule raises an issue when an assignment is useless because the assigned-to variable already holds the value on all execution paths.

Noncompliant code example

```
a = b;  
c = a;  
b = c; // Noncompliant: c and b are already the same
```

Compliant solution

```
a = b;  
c = a;
```

Delivering code in production with debug features activated is security-sensitive

Key: typescript:S4507

How To Fix:

Recommended Secure Coding Practices

Do not enable debugging features on production servers or applications distributed to end users.

Compliant Solution

[errorhandler Express.js middleware](#) used only in development mode:

```
const express = require('express');  
const errorHandler = require('errorhandler');  
  
let app = express();  
  
if (process.env.NODE_ENV === 'development') {
```

```
    app.use(errorhandler());  
}
```

See

- OWASP - [Top 10 2021 Category A5 - Security Misconfiguration](#)
- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)
- CWE - [CWE-489 - Active Debug Code](#)
- CWE - [CWE-215 - Information Exposure Through Debug Information](#)

Default:

Development tools and frameworks usually have options to make debugging easier for developers. Although these features are useful during development, they should never be enabled for applications deployed in production. Debug instructions or error messages can leak detailed information about the system, like the application's path or file names.

Ask Yourself Whether

- The code or configuration enabling the application debug features is deployed on production servers or distributed to end users.
- The application runs by default with debug features activated.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

Do not enable debugging features on production servers or applications distributed to end users.

Sensitive Code Example

[errorhandler Express.js middleware](#) should not be used in production:

```
const express = require('express');  
const errorHandler = require('errorhandler');  
  
let app = express();  
app.use(errorHandler()); // Sensitive
```

Compliant Solution

[errorhandler Express.js middleware](#) used only in development mode:

```
const express = require('express');  
const errorHandler = require('errorhandler');  
  
let app = express();  
  
if (process.env.NODE_ENV === 'development') {  
    app.use(errorHandler());  
}
```

```
}
```

See

- OWASP - [Top 10 2021 Category A5 - Security Misconfiguration](#)
- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)
- CWE - [CWE-489 - Active Debug Code](#)
- CWE - [CWE-215 - Information Exposure Through Debug Information](#)

Assess The Problem:

Ask Yourself Whether

- The code or configuration enabling the application debug features is deployed on production servers or distributed to end users.
- The application runs by default with debug features activated.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

[errorhandler Express.js middleware](#) should not be used in production:

```
const express = require('express');
const errorHandler = require('errorhandler');

let app = express();
app.use(errorHandler()); // Sensitive
```

Root Cause:

Development tools and frameworks usually have options to make debugging easier for developers. Although these features are useful during development, they should never be enabled for applications deployed in production. Debug instructions or error messages can leak detailed information about the system, like the application's path or file names.

"undefined" should not be passed as the value of optional parameters

Key: typescript:S4623

Root Cause:

In TypeScript, optional and default parameters are features that enhance the flexibility and usability of functions by allowing you to define parameters that can be omitted during function calls or have default values assigned to them when not provided explicitly:

- Optional parameters are denoted by adding a question mark (?) after the parameter name in the function declaration. These parameters can be omitted when calling the function, and TypeScript will assign them a value of undefined if they are not provided.
- Default parameters are used to assign a default value to a parameter in case the caller does not provide

a value for that parameter. Default

values are specified in the function declaration using the assignment operator (=) followed by the default value.

When a parameter is defined as optional, it automatically allows that parameter to be omitted during function calls. If you explicitly pass

`undefined` as the value for an optional parameter, it contradicts the purpose of making the parameter optional, making the code less readable and maintainable.

Similarly, when a parameter has a default value, it means that if the caller omits the argument during function calls, the default value will be used automatically. There is no need to pass `undefined` explicitly for default parameters, except when they come before a required parameter.

This rule checks that the last argument of a function call is not redundant with regard to the function's signature.

```
function foo(x: number, y: string = "default", z?: number) {  
    // ...  
}  
  
foo(42, undefined); // Noncompliant: 'undefined' is redundant  
foo(42, undefined, undefined); // Noncompliant: Both 'undefined' are redundant  
foo(42, undefined, 5); // Compliant: 'undefined' is required to get the second  
parameter's default value
```

Instead of explicitly passing `undefined`, simply omit the optional argument when calling the function to let TypeScript handle the default value correctly.

```
function foo(x: number, y: string = "default", z?: number) {  
    // ...  
}  
  
foo(42);
```

Resources:

Documentation

- TypeScript Documentation - [Optional Parameters](#)

Template literals should not be nested

Key: typescript:S4624

Root Cause:

Template literals, also known as template strings, allow for string interpolation and multiline strings in JavaScript. They provide a more convenient and flexible way to work with strings compared to traditional string concatenation or manipulation.

Template literals are delimited with the backtick (``) character. They are a convenient way to include variables or expressions within a string using placeholders ``${expression}`` in the string and evaluate them dynamically.

However, nesting template literals can make the code less readable. With each nested template literal, the code becomes more complex and harder to understand. It can be challenging to keep track of the opening

and closing backticks and properly escape characters if needed.

```
const color = "red";
const count = 3;
const message = `I have ${color ? `${count} ${color}` : count} apples`; //
Noncompliant: nested template strings not easy to read
```

In such situations, moving the nested template into a separate statement is preferable.

```
const color = "red";
const count = 3;
const apples = color ? `${count} ${color}` : count;
const message = `I have ${apples} apples`;
```

The rule makes an exception for nested template literals spanning multiple lines. It allows you to visually separate different sections and gives you more freedom in formatting your text, including line breaks, indentation, and other formatting elements, enhancing readability.

```
const name = 'John';
const age = 42;

const message = `Hello ${name}!
You are ${age} years old.
${`This is a nested template literal.`}
It can span multiple lines.`;
```

Resources:

Documentation

- MDN web docs - [Template literals \(Template strings\)](#)

Using weak hashing algorithms is security-sensitive

Key: typescript:S4790

Root Cause:

Cryptographic hash algorithms such as MD2, MD4, MD5, MD6, HAVAL-128, DSA (which uses SHA-1), RIPEMD, RIPEMD-128, RIPEMD-160 and SHA-1 are no longer considered secure, because it is possible to have collisions (little computational effort is enough to find two or more different inputs that produce the same hash).

Message authentication code (MAC) algorithms such as HMAC-MD5 or HMAC-SHA1 use weak hash functions as building blocks.

Although they are not all proven to be weak, they are considered legacy algorithms and should be avoided.

Assess The Problem:

Ask Yourself Whether

The hashed value is used in a security context like:

- User-password storage.

- Security token generation (used to confirm e-mail when registering on a website, reset password, etc ...■).
- To compute some message integrity.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

```
const crypto = require("crypto");

const hash = crypto.createHash('sha1'); // Sensitive
```

Default:

Cryptographic hash algorithms such as MD2, MD4, MD5, MD6, HAVAL-128, DSA (which uses SHA-1), RIPEMD, RIPEMD-128, RIPEMD-160 and SHA-1 are no longer considered secure, because it is possible to have collisions (little computational effort is enough to find two or more different inputs that produce the same hash).

Message authentication code (MAC) algorithms such as HMAC-MD5 or HMAC-SHA1 use weak hash functions as building blocks.

Although they are not all proven to be weak, they are considered legacy algorithms and should be avoided.

Ask Yourself Whether

The hashed value is used in a security context like:

- User-password storage.
- Security token generation (used to confirm e-mail when registering on a website, reset password, etc ...■).
- To compute some message integrity.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

Safer alternatives, such as SHA-256, SHA-512, SHA-3 are recommended, and for password hashing, it's even better to use algorithms that do not compute too "quickly", like bcrypt, scrypt, argon2 or pbkdf2 because it slows down brute force attacks.

Sensitive Code Example

```
const crypto = require("crypto");

const hash = crypto.createHash('sha1'); // Sensitive
```

Compliant Solution

```
const crypto = require("crypto");
```

```
const hash = crypto.createHash('sha512'); // Compliant
```

See

- OWASP - [Top 10 2021 Category A2 - Cryptographic Failures](#)
- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)
- OWASP - [Top 10 2017 Category A6 - Security Misconfiguration](#)
- CWE - [CWE-1240 - Use of a Risky Cryptographic Primitive](#)

How To Fix:

Recommended Secure Coding Practices

Safer alternatives, such as SHA-256, SHA-512, SHA-3 are recommended, and for password hashing, it's even better to use algorithms that do not compute too "quickly", like bcrypt, scrypt, argon2 or pbkdf2 because it slows down brute force attacks.

Compliant Solution

```
const crypto = require("crypto");

const hash = crypto.createHash('sha512'); // Compliant
```

See

- OWASP - [Top 10 2021 Category A2 - Cryptographic Failures](#)
- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)
- OWASP - [Top 10 2017 Category A6 - Security Misconfiguration](#)
- CWE - [CWE-1240 - Use of a Risky Cryptographic Primitive](#)

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive

Key: typescript:S5122

How To Fix:

Recommended Secure Coding Practices

- The Access-Control-Allow-Origin header should be set only for a trusted origin and for specific resources.
- Allow only selected, trusted domains in the Access-Control-Allow-Origin header. Prefer whitelisting domains over blacklisting or allowing any domain (do not use * wildcard nor blindly return the Origin header content without any checks).

Compliant Solution

[nodejs http](#) built-in module:

```
const http = require('http');
const srv = http.createServer((req, res) => {
  res.writeHead(200, { 'Access-Control-Allow-Origin': 'trustedwebsite.com' }); //
  Compliant
  res.end('ok');
});
srv.listen(3000);
```

[Express.js](#) framework with [cors middleware](#):

```
const cors = require('cors');

let corsOptions = {
  origin: 'trustedwebsite.com' // Compliant
};

let app = express();
app.use(cors(corsOptions));
```

User-controlled origin validated with an allow-list:

```
function (req, res) {
  const origin = req.headers.origin;

  if (origin === 'trustedwebsite.com') {
    res.setHeader('Access-Control-Allow-Origin', origin);
  }
};
```

See

- OWASP - [Top 10 2021 Category A5 - Security Misconfiguration](#)
- OWASP - [Top 10 2021 Category A7 - Identification and Authentication Failures](#)
- [developer.mozilla.org](#) - CORS
- [developer.mozilla.org](#) - Same origin policy
- OWASP - [Top 10 2017 Category A6 - Security Misconfiguration](#)
- [OWASP HTML5 Security Cheat Sheet](#) - Cross Origin Resource Sharing
- CWE - [CWE-346 - Origin Validation Error](#)

- CWE - [CWE-942 - Overly Permissive Cross-domain Whitelist](#)

Assess The Problem:

Ask Yourself Whether

- You don't trust the origin specified, example: `Access-Control-Allow-Origin: untrustedwebsite.com`.
- Access control policy is entirely disabled: `Access-Control-Allow-Origin: *`
- Your access control policy is dynamically defined by a user-controlled input like [href="https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Origin">origin](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Origin) header.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

[nodejs http](#) built-in module:

```
const http = require('http');
const srv = http.createServer((req, res) => {
  res.writeHead(200, { 'Access-Control-Allow-Origin': '*' }); // Sensitive
  res.end('ok');
});
srv.listen(3000);
```

[Express.js](#) framework with [cors middleware](#):

```
const cors = require('cors');

let app1 = express();
app1.use(cors()); // Sensitive: by default origin is set to *

let corsOptions = {
  origin: '*' // Sensitive
};

let app2 = express();
app2.use(cors(corsOptions));
```

User-controlled origin:

```
function (req, res) {
  const origin = req.headers.origin;
  res.setHeader('Access-Control-Allow-Origin', origin); // Sensitive
};
```

Root Cause:

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive. It has led in the past to the following vulnerabilities:

- [CVE-2018-0269](#)

- [CVE-2017-14460](#)

[Same origin policy](#) in browsers prevents, by default and for security-reasons, a javascript frontend to perform a cross-origin HTTP request to a resource that has a different origin (domain, protocol, or port) from its own. The requested target can append additional HTTP headers in response, called [CORS](#), that act like directives for the browser and change the access control policy / relax the same origin policy.

Default:

Having a permissive Cross-Origin Resource Sharing policy is security-sensitive. It has led in the past to the following vulnerabilities:

- [CVE-2018-0269](#)
- [CVE-2017-14460](#)

[Same origin policy](#) in browsers prevents, by default and for security-reasons, a javascript frontend to perform a cross-origin HTTP request to a resource that has a different origin (domain, protocol, or port) from its own. The requested target can append additional HTTP headers in response, called [CORS](#), that act like directives for the browser and change the access control policy / relax the same origin policy.

Ask Yourself Whether

- You don't trust the origin specified, example: `Access-Control-Allow-Origin: untrustedwebsite.com`.
- Access control policy is entirely disabled: `Access-Control-Allow-Origin: *`
- Your access control policy is dynamically defined by a user-controlled input like [href="https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Origin">origin](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Origin) header.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- The `Access-Control-Allow-Origin` header should be set only for a trusted origin and for specific resources.
- Allow only selected, trusted domains in the `Access-Control-Allow-Origin` header. Prefer whitelisting domains over blacklisting or allowing any domain (do not use `*` wildcard nor blindly return the `Origin` header content without any checks).

Sensitive Code Example

[nodejs http](#) built-in module:

```
const http = require('http');
const srv = http.createServer((req, res) => {
  res.writeHead(200, { 'Access-Control-Allow-Origin': '*' }); // Sensitive
  res.end('ok');
});
srv.listen(3000);
```

[Express.js](#) framework with [cors middleware](#):

```
const cors = require('cors');

let app1 = express();
app1.use(cors()); // Sensitive: by default origin is set to *

let corsOptions = {
  origin: '*' // Sensitive
};

let app2 = express();
app2.use(cors(corsOptions));
```

User-controlled origin:

```
function (req, res) {
  const origin = req.headers.origin;
  res.setHeader('Access-Control-Allow-Origin', origin); // Sensitive
};
```

Compliant Solution

[nodejs http](#) built-in module:

```
const http = require('http');
const srv = http.createServer((req, res) => {
  res.writeHead(200, { 'Access-Control-Allow-Origin': 'trustedwebsite.com' }); //
Compliant
  res.end('ok');
});
srv.listen(3000);
```

[Express.js](#) framework with [cors middleware](#):

```
const cors = require('cors');

let corsOptions = {
  origin: 'trustedwebsite.com' // Compliant
};

let app = express();
app.use(cors(corsOptions));
```

User-controlled origin validated with an allow-list:

```
function (req, res) {
  const origin = req.headers.origin;

  if (origin === 'trustedwebsite.com') {
```

```
    res.setHeader('Access-Control-Allow-Origin', origin);  
  }  
};
```

See

- OWASP - [Top 10 2021 Category A5 - Security Misconfiguration](#)
- OWASP - [Top 10 2021 Category A7 - Identification and Authentication Failures](#)
- [developer.mozilla.org](#) - CORS
- [developer.mozilla.org](#) - Same origin policy
- OWASP - [Top 10 2017 Category A6 - Security Misconfiguration](#)
- [OWASP HTML5 Security Cheat Sheet](#) - Cross Origin Resource Sharing
- CWE - [CWE-346 - Origin Validation Error](#)
- CWE - [CWE-942 - Overly Permissive Cross-domain Whitelist](#)

Using clear-text protocols is security-sensitive

Key: typescript:S5332

Root Cause:

Clear-text protocols such as `ftp`, `telnet`, or `http` lack encryption of transported data, as well as the capability to build an authenticated connection. It means that an attacker able to sniff traffic from the network can read, modify, or corrupt the transported content. These protocols are not secure as they expose applications to an extensive range of risks:

- sensitive data exposure
- traffic redirected to a malicious endpoint
- malware-infected software update or installer
- execution of client-side code
- corruption of critical information

Even in the context of isolated networks like offline environments or segmented cloud environments, the insider threat exists. Thus, attacks involving communications being sniffed or tampered with can still happen.

For example, attackers could successfully compromise prior security layers by:

- bypassing isolation mechanisms
- compromising a component of the network

- getting the credentials of an internal IAM account (either from a service account or an actual person)

In such cases, encrypting communications would decrease the chances of attackers to successfully leak data or steal credentials from other network components. By layering various security practices (segmentation and encryption, for example), the application will follow the *defense-in-depth* principle.

Note that using the `http` protocol is being deprecated by [major web browsers](#).

In the past, it has led to the following vulnerabilities:

- [CVE-2019-6169](#)
- [CVE-2019-12327](#)
- [CVE-2019-11065](#)

Exceptions

No issue is reported for the following cases because they are not considered sensitive:

- Insecure protocol scheme followed by loopback addresses like `127.0.0.1` or `localhost`.

How To Fix:

Recommended Secure Coding Practices

- Make application data transit over a secure, authenticated and encrypted protocol like TLS or SSH. Here are a few alternatives to the most common clear-text protocols:

Use `ssh` as an alternative to `telnet`.

- Use `sftp`, `scp`, or `ftps` instead of `ftp`.
- Use `https` instead of `http`.
- Use SMTP over SSL/TLS or SMTP with STARTTLS instead of clear-text SMTP.
- Enable encryption of cloud components communications whenever it is possible.
- Configure your application to block mixed content when rendering web pages.
- If available, enforce OS-level deactivation of all clear-text traffic.

It is recommended to secure all transport channels, even on local networks, as it can take a single non-secure connection to compromise an entire application or system.

Compliant Solution

```
url = "https://example.com";  
url = "sftp://anonymous@example.com";  
url = "ssh://anonymous@example.com";
```

For [nodemailer](#) one of the following options must be set:

```
const nodemailer = require("nodemailer");
let transporter = nodemailer.createTransport({
  secure: true,
  requireTLS: true,
  port: 465,
  secured: true
});
```

For [ftp](#):

```
var Client = require('ftp');
var c = new Client();
c.connect({
  'secure': true
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.ApplicationLoadBalancer](#):

```
import { ApplicationLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

const alb = new ApplicationLoadBalancer(this, 'ALB', {
  vpc: vpc,
  internetFacing: true
});

alb.addListener('listener-https-explicit', {
  protocol: ApplicationProtocol.HTTPS,
  port: 8080,
  open: true,
  certificates: [certificate]
});

alb.addListener('listener-https-implicit', {
  port: 8080,
  open: true,
  certificates: [certificate]
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.ApplicationListener](#):

```
import { ApplicationListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

new ApplicationListener(this, 'listener-https-explicit', {
  loadBalancer: loadBalancer,
  protocol: ApplicationProtocol.HTTPS,
  port: 8080,
  open: true,
  certificates: [certificate]
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.NetworkLoadBalancer](#):

```
import { NetworkLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

const nlb = new NetworkLoadBalancer(this, 'nlb', {
  vpc: vpc,
  internetFacing: true
});

nlb.addListener('listener-tls-explicit', {
  protocol: Protocol.TLS,
  port: 1234,
  certificates: [certificate]
});

nlb.addListener('listener-tls-implicit', {
  port: 1234,
  certificates: [certificate]
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.NetworkListener](#):

```
import { NetworkListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

new NetworkListener(this, 'listener-tls-explicit', {
  loadBalancer: loadBalancer,
  protocol: Protocol.TLS,
  port: 8080,
  certificates: [certificate]
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.CfnListener](#):

```
import { CfnListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

new CfnListener(this, 'listener-https', {
  defaultActions: defaultActions,
  loadBalancerArn: loadBalancerArn,
  protocol: "HTTPS",
  port: 80
  certificates: [certificate]
});

new CfnListener(this, 'listener-tls', {
  defaultActions: defaultActions,
  loadBalancerArn: loadBalancerArn,
  protocol: "TLS",
  port: 80
  certificates: [certificate]
});
```

For [aws-cdk-lib.aws-elasticloadbalancing.CfnLoadBalancer](#):

```
import { CfnLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancing';

new CfnLoadBalancer(this, 'elb-ssl', {
  listeners: [{
    instancePort: '1000',
    loadBalancerPort: '1000',
    protocol: 'ssl',
    sslCertificateId: sslCertificateId
  }]
});

new CfnLoadBalancer(this, 'elb-https', {
  listeners: [{
    instancePort: '1000',
    loadBalancerPort: '1000',
    protocol: 'https',
    sslCertificateId: sslCertificateId
  }]
});
```

For [aws-cdk-lib.aws-elasticloadbalancing.LoadBalancer](#):

```
import { LoadBalancer, LoadBalancingProtocol } from
'aws-cdk-lib/aws-elasticloadbalancing';

const lb = new LoadBalancer(this, 'elb-ssl', {
  vpc,
  internetFacing: true,
  healthCheck: {
    port: 80,
  },
  listeners: [
    {
      externalPort:10000,
      externalProtocol:LoadBalancingProtocol.SSL,
      internalPort:10000
    }
  ]
});

lb.addListener({
  externalPort:10001,
  externalProtocol:LoadBalancingProtocol.SSL,
  internalPort:10001
});

lb.addListener({
  externalPort:10002,
  externalProtocol:LoadBalancingProtocol.HTTPS,
  internalPort:10002
});
```

For [aws-cdk-lib.aws-elasticache.CfnReplicationGroup](#):

```
import { CfnReplicationGroup } from 'aws-cdk-lib/aws-elasticache';

new CfnReplicationGroup(this, 'encrypted-explicit', {
  replicationGroupDescription: 'example',
  transitEncryptionEnabled: true
});
```

For [aws-cdk-lib.aws-kinesis.Stream](#):

```
import { Stream } from 'aws-cdk-lib/aws-kinesis';

new Stream(this, 'stream-implicit-encrypted');

new Stream(this, 'stream-explicit-encrypted-selfmanaged', {
  encryption: StreamEncryption.KMS,
  encryptionKey: encryptionKey,
});

new Stream(this, 'stream-explicit-encrypted-managed', {
  encryption: StreamEncryption.MANAGED
});
```

For [aws-cdk-lib.aws-kinesis.CfnStream](#):

```
import { CfnStream } from 'aws-cdk-lib/aws-kinesis';

new CfnStream(this, 'cfnstream-explicit-encrypted', {
  streamEncryption: {
    encryptionType: encryptionType,
    keyId: encryptionKey.keyId,
  }
});
```

See

Documentation

- AWS Documentation - [Listeners for your Application Load Balancers](#)
- AWS Documentation - [href="https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-p
roperities-kinesis-stream-streamencryption.html">Stream Encryption](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-kinesis-stream-streamencryption.html)

Articles & blog posts

- Google - [Moving towards more secure web](#)
- Mozilla - [Deprecating non secure http](#)

Standards

- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)

- OWASP - [Top 10 2021 Category A2 - Cryptographic Failures](#)
- CWE - [CWE-200 - Exposure of Sensitive Information to an Unauthorized Actor](#)
- CWE - [CWE-319 - Cleartext Transmission of Sensitive Information](#)
- STIG Viewer - [Application Security and Development: V-222397](#) - The application must implement cryptographic mechanisms to protect the integrity of remote access sessions.
- STIG Viewer - [Application Security and Development: V-222534](#) - Service-Oriented Applications handling non-releasable data must authenticate endpoint devices via mutual SSL/TLS.
- STIG Viewer - [Application Security and Development: V-222562](#) - Applications used for non-local maintenance must implement cryptographic mechanisms to protect the integrity of maintenance and diagnostic communications.
- STIG Viewer - [Application Security and Development: V-222563](#) - Applications used for non-local maintenance must implement cryptographic mechanisms to protect the confidentiality of maintenance and diagnostic communications.
- STIG Viewer - [Application Security and Development: V-222577](#) - The application must not expose session IDs.
- STIG Viewer - [Application Security and Development: V-222596](#) - The application must protect the confidentiality and integrity of transmitted information.
- STIG Viewer - [Application Security and Development: V-222597](#) - The application must implement cryptographic mechanisms to prevent unauthorized disclosure of information and/or detect changes to information during transmission.
- STIG Viewer - [Application Security and Development: V-222598](#) - The application must maintain the confidentiality and integrity of information during preparation for transmission.
- STIG Viewer - [Application Security and Development: V-222599](#) - The application must maintain the confidentiality and integrity of information during reception.

Assess The Problem:

Ask Yourself Whether

- Application data needs to be protected against falsifications or leaks when transiting over the network.
- Application data transits over an untrusted network.
- Compliance rules require the service to encrypt data in transit.
- Your application renders web pages with a relaxed mixed content policy.

- OS-level protections against clear-text traffic are deactivated.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

```
url = "http://example.com"; // Sensitive
url = "ftp://anonymous@example.com"; // Sensitive
url = "telnet://anonymous@example.com"; // Sensitive
```

For [nodemailer](#):

```
const nodemailer = require("nodemailer");
let transporter = nodemailer.createTransport({
  secure: false, // Sensitive
  requireTLS: false // Sensitive
});

const nodemailer = require("nodemailer");
let transporter = nodemailer.createTransport({}); // Sensitive
```

For [ftp](#):

```
var Client = require('ftp');
var c = new Client();
c.connect({
  'secure': false // Sensitive
});
```

For [telnet-client](#):

```
const Telnet = require('telnet-client'); // Sensitive
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.ApplicationLoadBalancer](#):

```
import { ApplicationLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

const alb = new ApplicationLoadBalancer(this, 'ALB', {
  vpc: vpc,
  internetFacing: true
});

alb.addListener('listener-http-default', {
  port: 8080,
  open: true
}); // Sensitive

alb.addListener('listener-http-explicit', {
  protocol: ApplicationProtocol.HTTP, // Sensitive
```

```
    port: 8080,  
    open: true  
  });
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.ApplicationListener](#):

```
import { ApplicationListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';  
  
new ApplicationListener(this, 'listener-http-explicit-constructor', {  
  loadBalancer: alb,  
  protocol: ApplicationProtocol.HTTP, // Sensitive  
  port: 8080,  
  open: true  
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.NetworkLoadBalancer](#):

```
import { NetworkLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancingv2';  
  
const nlb = new NetworkLoadBalancer(this, 'nlb', {  
  vpc: vpc,  
  internetFacing: true  
});  
  
var listenerNLB = nlb.addListener('listener-tcp-default', {  
  port: 1234  
}); // Sensitive  
  
listenerNLB = nlb.addListener('listener-tcp-explicit', {  
  protocol: Protocol.TCP, // Sensitive  
  port: 1234  
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.NetworkListener](#):

```
import { NetworkListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';  
  
new NetworkListener(this, 'listener-tcp-explicit-constructor', {  
  loadBalancer: nlb,  
  protocol: Protocol.TCP, // Sensitive  
  port: 8080  
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.CfnListener](#):

```
import { CfnListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';  
  
new CfnListener(this, 'listener-http', {  
  defaultActions: defaultActions,  
  loadBalancerArn: alb.loadBalancerArn,  
  protocol: "HTTP", // Sensitive  
  port: 80  
});
```



```
});  
  
new CfnListener(this, 'listener-tcp', {  
  defaultActions: defaultActions,  
  loadBalancerArn: alb.loadBalancerArn,  
  protocol: "TCP", // Sensitive  
  port: 80  
});
```

For [aws-cdk-lib.aws-elasticloadbalancing.CfnLoadBalancer](#):

```
import { CfnLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancing';  
  
new CfnLoadBalancer(this, 'elb-tcp', {  
  listeners: [{  
    instancePort: '1000',  
    loadBalancerPort: '1000',  
    protocol: 'tcp' // Sensitive  
  }]  
});  
  
new CfnLoadBalancer(this, 'elb-http', {  
  listeners: [{  
    instancePort: '1000',  
    loadBalancerPort: '1000',  
    protocol: 'http' // Sensitive  
  }]  
});
```

For [aws-cdk-lib.aws-elasticloadbalancing.LoadBalancer](#):

```
import { LoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancing';  
  
const loadBalancer = new LoadBalancer(this, 'elb-tcp-dict', {  
  vpc,  
  internetFacing: true,  
  healthCheck: {  
    port: 80,  
  },  
  listeners: [  
    {  
      externalPort: 10000,  
      externalProtocol: LoadBalancingProtocol.TCP, // Sensitive  
      internalPort: 10000  
    }  
  ]  
});  
  
loadBalancer.addListener({  
  externalPort: 10001,  
  externalProtocol: LoadBalancingProtocol.TCP, // Sensitive  
  internalPort: 10001  
});  
loadBalancer.addListener({  
  externalPort: 10002,
```

```
    externalProtocol:LoadBalancingProtocol.HTTP, // Sensitive
    internalPort:10002
  });
```

For [aws-cdk-lib.aws-elasticache.CfnReplicationGroup](#):

```
import { CfnReplicationGroup } from 'aws-cdk-lib/aws-elasticache';

new CfnReplicationGroup(this, 'unencrypted-implicit', {
  replicationGroupDescription: 'exampleDescription'
}); // Sensitive

new CfnReplicationGroup(this, 'unencrypted-explicit', {
  replicationGroupDescription: 'exampleDescription',
  transitEncryptionEnabled: false // Sensitive
});
```

For [aws-cdk-lib.aws-kinesis.CfnStream](#):

```
import { CfnStream } from 'aws-cdk-lib/aws-kinesis';

new CfnStream(this, 'cfnstream-implicit-unencrytped', undefined); // Sensitive

new CfnStream(this, 'cfnstream-explicit-unencrytped', {
  streamEncryption: undefined // Sensitive
});
```

For [aws-cdk-lib.aws-kinesis.Stream](#):

```
import { Stream } from 'aws-cdk-lib/aws-kinesis';

new Stream(this, 'stream-explicit-unencrypted', {
  encryption: StreamEncryption.UNENCRYPTED // Sensitive
});
```

Default:

Clear-text protocols such as ftp, telnet, or http lack encryption of transported data, as well as the capability to build an authenticated connection. It means that an attacker able to sniff traffic from the network can read, modify, or corrupt the transported content. These protocols are not secure as they expose applications to an extensive range of risks:

- sensitive data exposure
- traffic redirected to a malicious endpoint
- malware-infected software update or installer
- execution of client-side code
- corruption of critical information

Even in the context of isolated networks like offline environments or segmented cloud environments, the insider threat exists. Thus, attacks involving communications being sniffed or tampered with can still

happen.

For example, attackers could successfully compromise prior security layers by:

- bypassing isolation mechanisms
- compromising a component of the network
- getting the credentials of an internal IAM account (either from a service account or an actual person)

In such cases, encrypting communications would decrease the chances of attackers to successfully leak data or steal credentials from other network components. By layering various security practices (segmentation and encryption, for example), the application will follow the *defense-in-depth* principle.

Note that using the `http` protocol is being deprecated by [major web browsers](#).

In the past, it has led to the following vulnerabilities:

- [CVE-2019-6169](#)
- [CVE-2019-12327](#)
- [CVE-2019-11065](#)

Ask Yourself Whether

- Application data needs to be protected against falsifications or leaks when transiting over the network.
- Application data transits over an untrusted network.
- Compliance rules require the service to encrypt data in transit.
- Your application renders web pages with a relaxed mixed content policy.
- OS-level protections against clear-text traffic are deactivated.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Make application data transit over a secure, authenticated and encrypted protocol like TLS or SSH. Here are a few alternatives to the most common clear-text protocols:

Use `ssh` as an alternative to `telnet`.

- Use `sftp`, `scp`, or `ftps` instead of `ftp`.
- Use `https` instead of `http`.
- Use `SMTP over SSL/TLS` or `SMTP with STARTTLS` instead of clear-text SMTP.
- Enable encryption of cloud components communications whenever it is possible.

- Configure your application to block mixed content when rendering web pages.
- If available, enforce OS-level deactivation of all clear-text traffic.

It is recommended to secure all transport channels, even on local networks, as it can take a single non-secure connection to compromise an entire application or system.

Sensitive Code Example

```
url = "http://example.com"; // Sensitive
url = "ftp://anonymous@example.com"; // Sensitive
url = "telnet://anonymous@example.com"; // Sensitive
```

For [nodemailer](#):

```
const nodemailer = require("nodemailer");
let transporter = nodemailer.createTransport({
  secure: false, // Sensitive
  requireTLS: false // Sensitive
});

const nodemailer = require("nodemailer");
let transporter = nodemailer.createTransport({}); // Sensitive
```

For [ftp](#):

```
var Client = require('ftp');
var c = new Client();
c.connect({
  'secure': false // Sensitive
});
```

For [telnet-client](#):

```
const Telnet = require('telnet-client'); // Sensitive
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.ApplicationLoadBalancer](#):

```
import { ApplicationLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

const alb = new ApplicationLoadBalancer(this, 'ALB', {
  vpc: vpc,
  internetFacing: true
});

alb.addListener('listener-http-default', {
  port: 8080,
  open: true
}); // Sensitive
```

```
alb.addListener('listener-http-explicit', {
  protocol: ApplicationProtocol.HTTP, // Sensitive
  port: 8080,
  open: true
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.ApplicationListener](#):

```
import { ApplicationListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

new ApplicationListener(this, 'listener-http-explicit-constructor', {
  loadBalancer: alb,
  protocol: ApplicationProtocol.HTTP, // Sensitive
  port: 8080,
  open: true
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.NetworkLoadBalancer](#):

```
import { NetworkLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

const nlb = new NetworkLoadBalancer(this, 'nlb', {
  vpc: vpc,
  internetFacing: true
});

var listenerNLB = nlb.addListener('listener-tcp-default', {
  port: 1234
}); // Sensitive

listenerNLB = nlb.addListener('listener-tcp-explicit', {
  protocol: Protocol.TCP, // Sensitive
  port: 1234
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.NetworkListener](#):

```
import { NetworkListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

new NetworkListener(this, 'listener-tcp-explicit-constructor', {
  loadBalancer: nlb,
  protocol: Protocol.TCP, // Sensitive
  port: 8080
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.CfnListener](#):

```
import { CfnListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

new CfnListener(this, 'listener-http', {
  defaultActions: defaultActions,
```

```
    loadBalancerArn: alb.loadBalancerArn,  
    protocol: "HTTP", // Sensitive  
    port: 80  
  });
```

```
new CfnListener(this, 'listener-tcp', {  
  defaultActions: defaultActions,  
  loadBalancerArn: alb.loadBalancerArn,  
  protocol: "TCP", // Sensitive  
  port: 80  
});
```

For [aws-cdk-lib.aws-elasticloadbalancing.CfnLoadBalancer](#):

```
import { CfnLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancing';
```

```
new CfnLoadBalancer(this, 'elb-tcp', {  
  listeners: [{  
    instancePort: '1000',  
    loadBalancerPort: '1000',  
    protocol: 'tcp' // Sensitive  
  }]  
});
```

```
new CfnLoadBalancer(this, 'elb-http', {  
  listeners: [{  
    instancePort: '1000',  
    loadBalancerPort: '1000',  
    protocol: 'http' // Sensitive  
  }]  
});
```

For [aws-cdk-lib.aws-elasticloadbalancing.LoadBalancer](#):

```
import { LoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancing';
```

```
const loadBalancer = new LoadBalancer(this, 'elb-tcp-dict', {  
  vpc,  
  internetFacing: true,  
  healthCheck: {  
    port: 80,  
  },  
  listeners: [  
    {  
      externalPort: 10000,  
      externalProtocol: LoadBalancingProtocol.TCP, // Sensitive  
      internalPort: 10000  
    }  
  ]  
});
```

```
loadBalancer.addListener({  
  externalPort: 10001,  
  externalProtocol: LoadBalancingProtocol.TCP, // Sensitive  
  internalPort: 10001  
});
```

```
});  
loadBalancer.addListener({  
  externalPort:10002,  
  externalProtocol:LoadBalancingProtocol.HTTP, // Sensitive  
  internalPort:10002  
});
```

For [aws-cdk-lib.aws-elasticache.CfnReplicationGroup](#):

```
import { CfnReplicationGroup } from 'aws-cdk-lib/aws-elasticache';  
  
new CfnReplicationGroup(this, 'unencrypted-implicit', {  
  replicationGroupDescription: 'exampleDescription'  
}); // Sensitive  
  
new CfnReplicationGroup(this, 'unencrypted-explicit', {  
  replicationGroupDescription: 'exampleDescription',  
  transitEncryptionEnabled: false // Sensitive  
});
```

For [aws-cdk-lib.aws-kinesis.CfnStream](#):

```
import { CfnStream } from 'aws-cdk-lib/aws-kinesis';  
  
new CfnStream(this, 'cfnstream-implicit-unencrytped', undefined); // Sensitive  
  
new CfnStream(this, 'cfnstream-explicit-unencrytped', {  
  streamEncryption: undefined // Sensitive  
});
```

For [aws-cdk-lib.aws-kinesis.Stream](#):

```
import { Stream } from 'aws-cdk-lib/aws-kinesis';  
  
new Stream(this, 'stream-explicit-unencrypted', {  
  encryption: StreamEncryption.UNENCRYPTED // Sensitive  
});
```

Compliant Solution

```
url = "https://example.com";  
url = "sftp://anonymous@example.com";  
url = "ssh://anonymous@example.com";
```

For [nodemailer](#) one of the following options must be set:

```
const nodemailer = require("nodemailer");  
let transporter = nodemailer.createTransport({  
  secure: true,  
  requireTLS: true,  
  port: 465,
```

```
    secured: true  
  });
```

For [ftp](#):

```
var Client = require('ftp');  
var c = new Client();  
c.connect({  
  'secure': true  
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.ApplicationLoadBalancer](#):

```
import { ApplicationLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancingv2';  
  
const alb = new ApplicationLoadBalancer(this, 'ALB', {  
  vpc: vpc,  
  internetFacing: true  
});  
  
alb.addListener('listener-https-explicit', {  
  protocol: ApplicationProtocol.HTTPS,  
  port: 8080,  
  open: true,  
  certificates: [certificate]  
});  
  
alb.addListener('listener-https-implicit', {  
  port: 8080,  
  open: true,  
  certificates: [certificate]  
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.ApplicationListener](#):

```
import { ApplicationListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';  
  
new ApplicationListener(this, 'listener-https-explicit', {  
  loadBalancer: loadBalancer,  
  protocol: ApplicationProtocol.HTTPS,  
  port: 8080,  
  open: true,  
  certificates: [certificate]  
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.NetworkLoadBalancer](#):

```
import { NetworkLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancingv2';  
  
const nlb = new NetworkLoadBalancer(this, 'nlb', {  
  vpc: vpc,  
  internetFacing: true  
});
```



```
});

nlb.addListener('listener-tls-explicit', {
  protocol: Protocol.TLS,
  port: 1234,
  certificates: [certificate]
});

nlb.addListener('listener-tls-implicit', {
  port: 1234,
  certificates: [certificate]
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.NetworkListener](#):

```
import { NetworkListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

new NetworkListener(this, 'listener-tls-explicit', {
  loadBalancer: loadBalancer,
  protocol: Protocol.TLS,
  port: 8080,
  certificates: [certificate]
});
```

For [aws-cdk-lib.aws-elasticloadbalancingv2.CfnListener](#):

```
import { CfnListener } from 'aws-cdk-lib/aws-elasticloadbalancingv2';

new CfnListener(this, 'listener-https', {
  defaultActions: defaultActions,
  loadBalancerArn: loadBalancerArn,
  protocol: "HTTPS",
  port: 80
  certificates: [certificate]
});

new CfnListener(this, 'listener-tls', {
  defaultActions: defaultActions,
  loadBalancerArn: loadBalancerArn,
  protocol: "TLS",
  port: 80
  certificates: [certificate]
});
```

For [aws-cdk-lib.aws-elasticloadbalancing.CfnLoadBalancer](#):

```
import { CfnLoadBalancer } from 'aws-cdk-lib/aws-elasticloadbalancing';

new CfnLoadBalancer(this, 'elb-ssl', {
  listeners: [{
    instancePort: '1000',
    loadBalancerPort: '1000',
    protocol: 'ssl',
```

```
        sslCertificateId: sslCertificateId
      }]
    });

new CfnLoadBalancer(this, 'elb-https', {
  listeners: [{
    instancePort: '1000',
    loadBalancerPort: '1000',
    protocol: 'https',
    sslCertificateId: sslCertificateId
  }]
});
```

For [aws-cdk-lib.aws-elasticloadbalancing.LoadBalancer](#):

```
import { LoadBalancer, LoadBalancingProtocol } from
'aws-cdk-lib/aws-elasticloadbalancing';

const lb = new LoadBalancer(this, 'elb-ssl', {
  vpc,
  internetFacing: true,
  healthCheck: {
    port: 80,
  },
  listeners: [
    {
      externalPort: 10000,
      externalProtocol: LoadBalancingProtocol.SSL,
      internalPort: 10000
    }
  ]
});

lb.addListener({
  externalPort: 10001,
  externalProtocol: LoadBalancingProtocol.SSL,
  internalPort: 10001
});
lb.addListener({
  externalPort: 10002,
  externalProtocol: LoadBalancingProtocol.HTTPS,
  internalPort: 10002
});
```

For [aws-cdk-lib.aws-elasticache.CfnReplicationGroup](#):

```
import { CfnReplicationGroup } from 'aws-cdk-lib/aws-elasticache';

new CfnReplicationGroup(this, 'encrypted-explicit', {
  replicationGroupDescription: 'example',
  transitEncryptionEnabled: true
});
```

For [aws-cdk-lib.aws-kinesis.Stream](#):

```
import { Stream } from 'aws-cdk-lib/aws-kinesis';

new Stream(this, 'stream-implicit-encrypted');

new Stream(this, 'stream-explicit-encrypted-selfmanaged', {
  encryption: StreamEncryption.KMS,
  encryptionKey: encryptionKey,
});

new Stream(this, 'stream-explicit-encrypted-managed', {
  encryption: StreamEncryption.MANAGED
});
```

For [aws-cdk-lib.aws-kinesis.CfnStream](#):

```
import { CfnStream } from 'aws-cdk-lib/aws-kinesis';

new CfnStream(this, 'cfnstream-explicit-encrypted', {
  streamEncryption: {
    encryptionType: encryptionType,
    keyId: encryptionKey.keyId,
  }
});
```

Exceptions

No issue is reported for the following cases because they are not considered sensitive:

- Insecure protocol scheme followed by loopback addresses like 127.0.0.1 or localhost.

See

Documentation

- AWS Documentation - [Listeners for your Application Load Balancers](#)
- AWS Documentation - [href="https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-kinesis-stream-streamencryption.html">Stream Encryption](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-kinesis-stream-streamencryption.html)

Articles & blog posts

- Google - [Moving towards more secure web](#)
- Mozilla - [Deprecating non secure http](#)

Standards

- OWASP - [Top 10 2017 Category A3 - Sensitive Data Exposure](#)
- OWASP - [Top 10 2021 Category A2 - Cryptographic Failures](#)

- CWE - [CWE-200 - Exposure of Sensitive Information to an Unauthorized Actor](#)
- CWE - [CWE-319 - Cleartext Transmission of Sensitive Information](#)
- STIG Viewer - [Application Security and Development: V-222397](#) - The application must implement cryptographic mechanisms to protect the integrity of remote access sessions.
- STIG Viewer - [Application Security and Development: V-222534](#) - Service-Oriented Applications handling non-releasable data must authenticate endpoint devices via mutual SSL/TLS.
- STIG Viewer - [Application Security and Development: V-222562](#) - Applications used for non-local maintenance must implement cryptographic mechanisms to protect the integrity of maintenance and diagnostic communications.
- STIG Viewer - [Application Security and Development: V-222563](#) - Applications used for non-local maintenance must implement cryptographic mechanisms to protect the confidentiality of maintenance and diagnostic communications.
- STIG Viewer - [Application Security and Development: V-222577](#) - The application must not expose session IDs.
- STIG Viewer - [Application Security and Development: V-222596](#) - The application must protect the confidentiality and integrity of transmitted information.
- STIG Viewer - [Application Security and Development: V-222597](#) - The application must implement cryptographic mechanisms to prevent unauthorized disclosure of information and/or detect changes to information during transmission.
- STIG Viewer - [Application Security and Development: V-222598](#) - The application must maintain the confidentiality and integrity of information during preparation for transmission.
- STIG Viewer - [Application Security and Development: V-222599](#) - The application must maintain the confidentiality and integrity of information during reception.

Allowing requests with excessive content length is security-sensitive

Key: typescript:S5693

How To Fix:

Recommended Secure Coding Practices

- For most of the features of an application, it is recommended to limit the size of requests to:

lower or equal to 8mb for file uploads.

- lower or equal to 2mb for other requests.

It is recommended to customize the rule with the limit values that correspond to the web application.

Compliant Solution

[formidable](#) file upload module:

```
const form = new Formidable();
form.maxFileSize = 8000000; // Compliant: 8MB
```

[multer](#) (Express.js middleware) file upload module:

```
let diskUpload = multer({
  storage: diskStorage,
  limits: {
    fileSize: 8000000 // Compliant: 8MB
  }
});
```

[body-parser](#) module:

```
let jsonParser = bodyParser.json(); // Compliant, when the limit is not defined, the
default value is set to 100kb
let urlencodedParser = bodyParser.urlencoded({ extended: false, limit: "2mb" }); //
Compliant
```

See

- OWASP - [Top 10 2021 Category A5 - Security Misconfiguration](#)
- [Owasp Cheat Sheet](#) - Owasp Denial of Service Cheat Sheet
- OWASP - [Top 10 2017 Category A6 - Security Misconfiguration](#)
- CWE - [CWE-770 - Allocation of Resources Without Limits or Throttling](#)
- CWE - [CWE-400 - Uncontrolled Resource Consumption](#)

Assess The Problem:

Ask Yourself Whether

- size limits are not defined for the different resources of the web application.
- the web application is not protected by [rate limiting](#) features.
- the web application infrastructure has limited resources.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

[formidable](#) file upload module:

```
const form = new Formidable();
form.maxFileSize = 10000000; // Sensitive: 10MB is more than the recommended limit of 8MB
```

```
const formDefault = new Formidable(); // Sensitive, the default value is 200MB
```

[multer](#) (Express.js middleware) file upload module:

```
let diskUpload = multer({
  storage: diskStorage,
  limits: {
    fileSize: 10000000; // Sensitive: 10MB is more than the recommended limit of 8MB
  }
});
```

```
let diskUploadUnlimited = multer({ // Sensitive: the default value is no limit
  storage: diskStorage,
});
```

[body-parser](#) module:

```
// 4MB is more than the recommended limit of 2MB for non-file-upload requests
let jsonParser = bodyParser.json({ limit: "4mb" }); // Sensitive
let urlencodedParser = bodyParser.urlencoded({ extended: false, limit: "4mb" }); // Sensitive
```

Default:

Rejecting requests with significant content length is a good practice to control the network traffic intensity and thus resource consumption in order to prevent DoS attacks.

Ask Yourself Whether

- size limits are not defined for the different resources of the web application.
- the web application is not protected by [rate limiting](#) features.
- the web application infrastructure has limited resources.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- For most of the features of an application, it is recommended to limit the size of requests to:

lower or equal to 8mb for file uploads.

- lower or equal to 2mb for other requests.

It is recommended to customize the rule with the limit values that correspond to the web application.

Sensitive Code Example

[formidable](#) file upload module:

```
const form = new Formidable();
form.maxFileSize = 10000000; // Sensitive: 10MB is more than the recommended limit of 8MB
```

```
const formDefault = new Formidable(); // Sensitive, the default value is 200MB
```

[multer](#) (Express.js middleware) file upload module:

```
let diskUpload = multer({
  storage: diskStorage,
  limits: {
    fileSize: 10000000; // Sensitive: 10MB is more than the recommended limit of 8MB
  }
});
```

```
let diskUploadUnlimited = multer({ // Sensitive: the default value is no limit
  storage: diskStorage,
});
```

[body-parser](#) module:

```
// 4MB is more than the recommended limit of 2MB for non-file-upload requests
let jsonParser = bodyParser.json({ limit: "4mb" }); // Sensitive
let urlencodedParser = bodyParser.urlencoded({ extended: false, limit: "4mb" }); // Sensitive
```

Compliant Solution

[formidable](#) file upload module:

```
const form = new Formidable();
form.maxFileSize = 8000000; // Compliant: 8MB
```

[multer](#) (Express.js middleware) file upload module:

```
let diskUpload = multer({
  storage: diskStorage,
  limits: {
    fileSize: 8000000 // Compliant: 8MB
  }
});
```

```
});
```

[body-parser](#) module:

```
let jsonParser = bodyParser.json(); // Compliant, when the limit is not defined, the
default value is set to 100kb
let urlencodedParser = bodyParser.urlencoded({ extended: false, limit: "2mb" }); //
Compliant
```

See

- OWASP - [Top 10 2021 Category A5 - Security Misconfiguration](#)
- [Owasp Cheat Sheet](#) - Owasp Denial of Service Cheat Sheet
- OWASP - [Top 10 2017 Category A6 - Security Misconfiguration](#)
- CWE - [CWE-770 - Allocation of Resources Without Limits or Throttling](#)
- CWE - [CWE-400 - Uncontrolled Resource Consumption](#)

Root Cause:

Rejecting requests with significant content length is a good practice to control the network traffic intensity and thus resource consumption in order to prevent DoS attacks.

Using slow regular expressions is security-sensitive

Key: typescript:S5852

Root Cause:

Most of the regular expression engines use backtracking to try all possible execution paths of the regular expression when evaluating an input, in some cases it can cause performance issues, called **catastrophic backtracking** situations. In the worst case, the complexity of the regular expression is exponential in the size of the input, this means that a small carefully-crafted input (like 20 chars) can trigger catastrophic backtracking and cause a denial of service of the application. Super-linear regex complexity can lead to the same impact too with, in this case, a large carefully-crafted input (thousands chars).

This rule determines the runtime complexity of a regular expression and informs you if it is not linear.

Assess The Problem:

Ask Yourself Whether

- The input is user-controlled.
- The input size is not restricted to a small number of characters.
- There is no timeout in place to limit the regex evaluation time.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

The regex evaluation will never end:

```
/(a+)+$/ .test(
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa "+
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa "+
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa "+
"aaaaaaaaaaaaaaaaaaaaa!")
); // Sensitive
```

How To Fix:

Recommended Secure Coding Practices

To avoid catastrophic backtracking situations, make sure that none of the following conditions apply to your regular expression.

In all of the following cases, catastrophic backtracking can only happen if the problematic part of the regex is followed by a pattern that can fail, causing the backtracking to actually happen.

- If you have a repetition r^* or $r^*?$, such that the regex r could produce different possible matches (of possibly different lengths) on the same input, the worst case matching time can be exponential. This can be the case if r contains optional parts, alternations or additional repetitions (but not if the repetition is written in such a way that there's only one way to match it).
- If you have multiple repetitions that can match the same contents and are consecutive or are only separated by an optional separator or a separator that can be matched by both of the repetitions, the worst case matching time can be polynomial ($O(n^c)$ where c is the number of problematic repetitions). For example a^*b^* is not a problem because a^* and b^* match different things and $a^*_a^*$ is not a problem because the repetitions are separated by a `'_'` and can't match that `'_'`. However, a^*a^* and $.*_.*$ have quadratic runtime.
- If the regex is not anchored to the beginning of the string, quadratic runtime is especially hard to avoid because whenever a match fails, the regex engine will try again starting at the next index. This means that any unbounded repetition, if it's followed by a pattern that can fail, can cause quadratic runtime on some inputs. For example `str.split(/\s*,/)` will run in quadratic time on strings that consist entirely of spaces (or at least contain large sequences of spaces, not followed by a comma).

In order to rewrite your regular expression without these patterns, consider the following strategies:

- If applicable, define a maximum number of expected repetitions using the bounded quantifiers, like `{1,5}` instead of `+` for instance.
- Refactor nested quantifiers to limit the number of way the inner group can be matched by the outer quantifier, for instance this nested quantifier situation `(ba+)+` doesn't cause performance issues, indeed, the inner group can be matched only if there exists exactly one `b` char per repetition of the group.
- Optimize regular expressions by emulating *possessive quantifiers* and *atomic grouping*.

- Use negated character classes instead of `.` to exclude separators where applicable. For example the quadratic regex `.*_.*` can be made linear by changing it to `[^_]*_.*`

Sometimes it's not possible to rewrite the regex to be linear while still matching what you want it to match. Especially when the regex is not anchored to the beginning of the string, for which it is quite hard to avoid quadratic runtimes. In those cases consider the following approaches:

- Solve the problem without regular expressions
- Use an alternative non-backtracking regex implementations such as Google's [RE2](https://github.com/uhop/node-re2/) or [node-re2](https://github.com/uhop/node-re2/).
- Use multiple passes. This could mean pre- and/or post-processing the string manually before/after applying the regular expression to it or using multiple regular expressions. One example of this would be to replace `str.split(/\s*,\s*/)` with `str.split(",")` and then trimming the spaces from the strings as a second step.
- It is often possible to make the regex infallible by making all the parts that could fail optional, which will prevent backtracking. Of course this means that you'll accept more strings than intended, but this can be handled by using capturing groups to check whether the optional parts were matched or not and then ignoring the match if they weren't. For example the regex `x*y` could be replaced with `x*(y)?` and then the call to `str.match(regex)` could be replaced with `matched = str.match(regex)` and `matched[1] !== undefined`.

Compliant Solution

Possessive quantifiers do not keep backtracking positions, thus can be used, if possible, to avoid performance issues. Unfortunately, they are not supported in JavaScript, but one can still mimic them using lookahead assertions and backreferences:

```
/(?(?=(a+))\2)+$/ .test(
  "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
  "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
  "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
  "aaaaaaaaaaaaaaaaaaaaa!"
); // Compliant
```

See

- OWASP - [Top 10 2017 Category A1 - Injection](#)
- CWE - [CWE-400 - Uncontrolled Resource Consumption](#)
- CWE - [CWE-1333 - Inefficient Regular Expression Complexity](#)
- [owasp.org](#) - OWASP Regular expression Denial of Service - ReDoS
- <https://web.archive.org/web/20220506215733/https://stackstatus.net/post/147710624694/outage-postmortem-july-20-2016> - Outage Postmortem - July 20, 2016

- regular-expressions.info - Runaway Regular Expressions: Catastrophic Backtracking
- <https://docs.microsoft.com/en-us/dotnet/standard/base-types/backtracking-in-regular-expressions#backtracking-with-nested-optional-quantifiers> - Backtracking with Nested Optional Quantifiers

Default:

Most of the regular expression engines use backtracking to try all possible execution paths of the regular expression when evaluating an input, in some cases it can cause performance issues, called **catastrophic backtracking** situations. In the worst case, the complexity of the regular expression is exponential in the size of the input, this means that a small carefully-crafted input (like 20 chars) can trigger catastrophic backtracking and cause a denial of service of the application. Super-linear regex complexity can lead to the same impact too with, in this case, a large carefully-crafted input (thousands chars).

This rule determines the runtime complexity of a regular expression and informs you if it is not linear.

Ask Yourself Whether

- The input is user-controlled.
- The input size is not restricted to a small number of characters.
- There is no timeout in place to limit the regex evaluation time.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

To avoid catastrophic backtracking situations, make sure that none of the following conditions apply to your regular expression.

In all of the following cases, catastrophic backtracking can only happen if the problematic part of the regex is followed by a pattern that can fail, causing the backtracking to actually happen.

- If you have a repetition r^* or $r^*?$, such that the regex r could produce different possible matches (of possibly different lengths) on the same input, the worst case matching time can be exponential. This can be the case if r contains optional parts, alternations or additional repetitions (but not if the repetition is written in such a way that there's only one way to match it).
- If you have multiple repetitions that can match the same contents and are consecutive or are only separated by an optional separator or a separator that can be matched by both of the repetitions, the worst case matching time can be polynomial ($O(n^c)$ where c is the number of problematic repetitions). For example a^*b^* is not a problem because a^* and b^* match different things and $a^*_a^*$ is not a problem because the repetitions are separated by a `'_'` and can't match that `'_'`. However, a^*a^* and $.*_.*$ have quadratic runtime.
- If the regex is not anchored to the beginning of the string, quadratic runtime is especially hard to avoid because whenever a match fails, the regex engine will try again starting at the next index. This means that any unbounded repetition, if it's followed by a pattern that can fail, can cause quadratic runtime on some inputs. For example `str.split(/\s*,/)` will run in quadratic time on

strings that consist entirely of spaces (or at least contain large sequences of spaces, not followed by a comma).

In order to rewrite your regular expression without these patterns, consider the following strategies:

- If applicable, define a maximum number of expected repetitions using the bounded quantifiers, like `{1,5}` instead of `+` for instance.
- Refactor nested quantifiers to limit the number of way the inner group can be matched by the outer quantifier, for instance this nested quantifier situation `(ba+)+` doesn't cause performance issues, indeed, the inner group can be matched only if there exists exactly one `b` char per repetition of the group.
- Optimize regular expressions by emulating *possessive quantifiers* and *atomic grouping*.
- Use negated character classes instead of `.` to exclude separators where applicable. For example the quadratic regex `.*_.*` can be made linear by changing it to `[^_]*_.*`

Sometimes it's not possible to rewrite the regex to be linear while still matching what you want it to match. Especially when the regex is not anchored to the beginning of the string, for which it is quite hard to avoid quadratic runtimes. In those cases consider the following approaches:

- Solve the problem without regular expressions
- Use an alternative non-backtracking regex implementations such as Google's [RE2](https://github.com/uhop/node-re2/) or [node-re2](https://github.com/uhop/node-re2/).
- Use multiple passes. This could mean pre- and/or post-processing the string manually before/after applying the regular expression to it or using multiple regular expressions. One example of this would be to replace `str.split(/\s*,\s*/)` with `str.split(",")` and then trimming the spaces from the strings as a second step.
- It is often possible to make the regex infallible by making all the parts that could fail optional, which will prevent backtracking. Of course this means that you'll accept more strings than intended, but this can be handled by using capturing groups to check whether the optional parts were matched or not and then ignoring the match if they weren't. For example the regex `x*y` could be replaced with `x*(y)?` and then the call to `str.match(regex)` could be replaced with `matched = str.match(regex)` and `matched[1] !== undefined`.

Sensitive Code Example

The regex evaluation will never end:

```
/(a+)+$/ .test(
  "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
  "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
  "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" +
  "aaaaaaaaaaaaaaaaaaaaa!"
); // Sensitive
```

Compliant Solution

Possessive quantifiers do not keep backtracking positions, thus can be used, if possible, to avoid performance issues. Unfortunately, they are not supported in JavaScript, but one can still mimick them using lookahead assertions and backreferences:

```
// ((?=(a+))\2)+$/ .test(  
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"+  
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"+  
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"+  
"aaaaaaaaaaaaaaaaaaaaa!"  
); // Compliant
```

See

- OWASP - [Top 10 2017 Category A1 - Injection](#)
- CWE - [CWE-400 - Uncontrolled Resource Consumption](#)
- CWE - [CWE-1333 - Inefficient Regular Expression Complexity](#)
- [owasp.org](#) - OWASP Regular expression Denial of Service - ReDoS
- <https://web.archive.org/web/20220506215733/https://stackstatus.net/post/147710624694/outage-postmortem-july-20-2016>>[stackstatus.net\(archived\)](#) - Outage Postmortem - July 20, 2016
- [regular-expressions.info](#) - Runaway Regular Expressions: Catastrophic Backtracking
- <https://docs.microsoft.com/en-us/dotnet/standard/base-types/backtracking-in-regular-expressions#backtracking-with-nested-optional-quantifiers>>[docs.microsoft.com](#) - Backtracking with Nested Optional Quantifiers

Character classes in regular expressions should not contain the same character twice

Key: typescript:S5869

Resources:

Documentation

- MDN web docs - [Character classes](#)
- MDN web docs - [Character class escape](#)

Root Cause:

Character classes in regular expressions are a convenient way to match one of several possible characters by listing the allowed characters or ranges of characters. If the same character is listed twice in the same character class or if the character class contains overlapping ranges, this has no effect.

Thus duplicate characters in a character class are either a simple oversight or a sign that a range in the character class matches more than is intended or that the author misunderstood how character classes work and wanted to match more than one character. A common example of the latter mistake is trying to use a range like `[0-99]` to match numbers of up to two digits, when in fact it is equivalent to `[0-9]`. Another common cause is forgetting to escape the `-` character, creating an unintended range that overlaps with other characters in the character class.

Character ranges can also create duplicates when used with character class escapes. These are a type of escape sequence used in regular expressions to represent a specific set of characters. They are denoted by a backslash followed by a specific letter, such as `\d` for digits, `\w` for word characters, or `\s` for whitespace characters. For example, the character class escape `\d` is equivalent to the character range `[0-9]`, and the escape `\w` is equivalent to `[a-zA-Z0-9_]`.

How To Fix:

Remove the extra character, character range, or character class escape.

Noncompliant code example

```
/[0-99]/ // Noncompliant, this won't actually match strings with two digits
/[0-9.-_]/ // Noncompliant, .-_ is a range that already contains 0-9 (as well as
various other characters such as capital letters)
/[a-z0-9\d]/ // Noncompliant, \d matches a digit and is equivalent to [0-9]
```

Compliant solution

```
/[0-9]{1,2}/
/[0-9.\-_]/
/[a-z\d]/
```

Disabling Angular built-in sanitization is security-sensitive

Key: typescript:S6268

How To Fix:

Recommended Secure Coding Practices

- Avoid including dynamic executable code and thus disabling Angular's built-in sanitization unless it's absolutely necessary. Try instead to rely as much as possible on static templates and Angular built-in sanitization to define web page content.
- Make sure to understand how the value to consider as trusted is constructed and never concatenate it with user-controlled data.
- Make sure to choose the correct [DomSanitizer](#) "bypass" method based on the context. For instance, only use `bypassSecurityTrustUrl` to trust urls in an `href` attribute context.

Compliant Solution

```
import { Component, OnInit } from '@angular/core';
import { DomSanitizer } from '@angular/platform-browser';
import { ActivatedRoute } from '@angular/router';
```

```
@Component({
  template: '<div id="hello"><h1>Hello {{name}}</h1></div>',
})
export class HelloComponent implements OnInit {
  name: string;

  constructor(private sanitizer: DomSanitizer, private route: ActivatedRoute) { }

  ngOnInit(): void {
    this.name = this.route.snapshot.queryParams.name;
  }
}
```

See

- OWASP - [Top 10 2021 Category A3 - Injection](#)
- OWASP - [Top 10 2017 Category A7 - Cross-Site Scripting \(XSS\)](#)
- CWE - [CWE-79 - Improper Neutralization of Input During Web Page Generation \('Cross-site Scripting'\)](#)
- [Angular - Best Practices - Security](#)

Root Cause:

Angular prevents XSS vulnerabilities by treating all values as untrusted by default. Untrusted values are systematically sanitized by the framework before they are inserted into the DOM.

Still, developers have the ability to manually mark a value as trusted if they are sure that the value is already sanitized. Accidentally trusting malicious data will introduce an XSS vulnerability in the application and enable a wide range of serious attacks like accessing/modifying sensitive information or impersonating other users.

Default:

Angular prevents XSS vulnerabilities by treating all values as untrusted by default. Untrusted values are systematically sanitized by the framework before they are inserted into the DOM.

Still, developers have the ability to manually mark a value as trusted if they are sure that the value is already sanitized. Accidentally trusting malicious data will introduce an XSS vulnerability in the application and enable a wide range of serious attacks like accessing/modifying sensitive information or impersonating other users.

Ask Yourself Whether

- The value for which sanitization has been disabled is user-controlled.
- It's difficult to understand how this value is constructed.

There is a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Avoid including dynamic executable code and thus disabling Angular's built-in sanitization unless it's

absolutely necessary. Try instead to rely as much as possible on static templates and Angular built-in sanitization to define web page content.

- Make sure to understand how the value to consider as trusted is constructed and never concatenate it with user-controlled data.
- Make sure to choose the correct [DomSanitizer](#) "bypass" method based on the context. For instance, only use `bypassSecurityTrustUrl` to trust urls in an `href` attribute context.

Sensitive Code Example

```
import { Component, OnInit } from '@angular/core';
import { DomSanitizer, SafeHtml } from "@angular/platform-browser";
import { ActivatedRoute } from '@angular/router';

@Component({
  template: '<div id="hello" [innerHTML]="hello"></div>'
})
export class HelloComponent implements OnInit {
  hello: SafeHtml;

  constructor(private sanitizer: DomSanitizer, private route: ActivatedRoute) { }

  ngOnInit(): void {
    let name = this.route.snapshot.queryParams.name;
    let html = "<h1>Hello " + name + "</h1>";
    this.hello = this.sanitizer.bypassSecurityTrustHtml(html); // Sensitive
  }
}
```

Compliant Solution

```
import { Component, OnInit } from '@angular/core';
import { DomSanitizer } from "@angular/platform-browser";
import { ActivatedRoute } from '@angular/router';

@Component({
  template: '<div id="hello"><h1>Hello {{name}}</h1></div>',
})
export class HelloComponent implements OnInit {
  name: string;

  constructor(private sanitizer: DomSanitizer, private route: ActivatedRoute) { }

  ngOnInit(): void {
    this.name = this.route.snapshot.queryParams.name;
  }
}
```

See

- OWASP - [Top 10 2021 Category A3 - Injection](#)
- OWASP - [Top 10 2017 Category A7 - Cross-Site Scripting \(XSS\)](#)

- [CWE - CWE-79 - Improper Neutralization of Input During Web Page Generation \('Cross-site Scripting'\)](#)
- [Angular - Best Practices - Security](#)

Assess The Problem:

Ask Yourself Whether

- The value for which sanitization has been disabled is user-controlled.
- It's difficult to understand how this value is constructed.

There is a risk if you answered yes to any of those questions.

Sensitive Code Example

```
import { Component, OnInit } from '@angular/core';
import { DomSanitizer, SafeHtml } from "@angular/platform-browser";
import { ActivatedRoute } from '@angular/router';

@Component({
  template: '<div id="hello" [innerHTML]="hello"></div>'
})
export class HelloComponent implements OnInit {
  hello: SafeHtml;

  constructor(private sanitizer: DomSanitizer, private route: ActivatedRoute) { }

  ngOnInit(): void {
    let name = this.route.snapshot.queryParams.name;
    let html = "<h1>Hello " + name + "</h1>";
    this.hello = this.sanitizer.bypassSecurityTrustHtml(html); // Sensitive
  }
}
```

Regular expression quantifiers and character classes should be used concisely

Key: typescript:S6353

Root Cause:

A regular expression is a sequence of characters that specifies a match pattern in text. Among the most important concepts are:

- Character classes: defines a set of characters, any one of which can occur in an input string for a match to succeed.
- Quantifiers: used to specify how many instances of a character, group, or character class must be present in the input for a match.
- Wildcard (.): matches all characters except line terminators (also matches them if the `s` flag is set).

Many of these features include shortcuts of widely used expressions, so there is more than one way to construct a regular expression to achieve the same results. For example, to match a two-digit number, one

could write `[0-9]{2,2}` or `\d{2}`. The latter is not only shorter but easier to read and thus to maintain.

This rule recommends replacing some quantifiers and character classes with more concise equivalents:

- `\d` for `[0-9]` and `\D` for `[^0-9]`
- `\w` for `[A-Za-z0-9_]` and `\W` for `[^A-Za-z0-9_]`
- `.` for character classes matching everything (e.g. `[\w\W]`, `[\d\D]`, or `[\s\S]` with `s` flag)
- `x?` for `x{0,1}`, `x*` for `x{0,}`, `x+` for `x{1,}`, `x{N}` for `x{N,N}`

```
/a{1,}/; // Noncompliant, '{1,}' quantifier is the same as '+'  
/[A-Za-z0-9_]/; // Noncompliant, '\w' is equivalent
```

Use the more concise version to make the regex expression more readable.

```
/a+/;  
/\w/;
```

Character classes in regular expressions should not contain only one character

Key: typescript:S6397

Root Cause:

Character classes in regular expressions are a convenient way to match one of several possible characters by listing the allowed characters or ranges of characters. If a character class contains only one character, the effect is the same as just writing the character without a character class.

Thus, having only one character in a character class is usually a simple oversight that remained after removing other characters of the class.

Noncompliant code example

```
/a[b]c/  
/[\^]/
```

Compliant solution

```
/abc/  
/\^/  
/[*]c/ // Compliant, see Exceptions
```

Exceptions

This rule does not raise when the character inside the class is a metacharacter. This notation is sometimes used to avoid escaping (e.g., `[.]{3}` to match three dots).

Hard-coded secrets are security-sensitive

Key: typescript:S6418

Assess The Problem:

Ask Yourself Whether

- The secret allows access to a sensitive component like a database, a file storage, an API, or a service.
- The secret is used in a production environment.
- Application re-distribution is required before updating the secret.

There would be a risk if you answered yes to any of those questions.

Sensitive Code Example

```
const API_KEY = "1234567890abcdef" // Hard-coded secret (bad practice)

const response = await fetch("https://api.my-service/v1/users", {
  headers: {
    Authorization: `Bearer ${API_KEY}`,
  },
});
```

Root Cause:

Because it is easy to extract strings from an application source code or binary, secrets should not be hard-coded. This is particularly true for applications that are distributed or that are open-source.

In the past, it has led to the following vulnerabilities:

- [CVE-2022-25510](#)
- [CVE-2021-42635](#)

Secrets should be stored outside of the source code in a configuration file or a management service for secrets.

This rule detects variables/fields having a name matching a list of words (secret, token, credential, auth, api[_.-]?key) being assigned a pseudorandom hard-coded value. The pseudorandomness of the hard-coded value is based on its entropy and the probability to be human-readable. The randomness sensibility can be adjusted if needed. Lower values will detect less random values, raising potentially more false positives.

How To Fix:

Recommended Secure Coding Practices

- Store the secret in a configuration file that is not pushed to the code repository.
- Use your cloud provider's service for managing secrets.
- If a secret has been disclosed through the source code: revoke it and create a new one.

Compliant Solution

```
const API_KEY = process.env.API_KEY;

const response = await fetch("https://api.my-service/v1/users", {
  headers: {
    Authorization: `Bearer ${API_KEY}`,
  },
});
```

See

- OWASP - [Top 10 2021 Category A7 - Identification and Authentication Failures](#)
- OWASP - [Top 10 2017 Category A2 - Broken Authentication](#)
- CWE - [CWE-798 - Use of Hard-coded Credentials](#)
- MSC - [MSC03-J - Never hard code sensitive information](#)

Default:

Because it is easy to extract strings from an application source code or binary, secrets should not be hard-coded. This is particularly true for applications that are distributed or that are open-source.

In the past, it has led to the following vulnerabilities:

- [CVE-2022-25510](#)
- [CVE-2021-42635](#)

Secrets should be stored outside of the source code in a configuration file or a management service for secrets.

This rule detects variables/fields having a name matching a list of words (secret, token, credential, auth, api[_.-]?key) being assigned a pseudorandom hard-coded value. The pseudorandomness of the hard-coded value is based on its entropy and the probability to be human-readable. The randomness sensibility can be adjusted if needed. Lower values will detect less random values, raising potentially more false positives.

Ask Yourself Whether

- The secret allows access to a sensitive component like a database, a file storage, an API, or a service.
- The secret is used in a production environment.
- Application re-distribution is required before updating the secret.

There would be a risk if you answered yes to any of those questions.

Recommended Secure Coding Practices

- Store the secret in a configuration file that is not pushed to the code repository.
- Use your cloud provider's service for managing secrets.

- If a secret has been disclosed through the source code: revoke it and create a new one.

Sensitive Code Example

```
const API_KEY = "1234567890abcdef" // Hard-coded secret (bad practice)

const response = await fetch("https://api.my-service/v1/users", {
  headers: {
    Authorization: `Bearer ${API_KEY}`,
  },
});
```

Compliant Solution

```
const API_KEY = process.env.API_KEY;

const response = await fetch("https://api.my-service/v1/users", {
  headers: {
    Authorization: `Bearer ${API_KEY}`,
  },
});
```

See

- OWASP - [Top 10 2021 Category A7 - Identification and Authentication Failures](#)
- OWASP - [Top 10 2017 Category A2 - Broken Authentication](#)
- CWE - [CWE-798 - Use of Hard-coded Credentials](#)
- MSC - [MSC03-J - Never hard code sensitive information](#)

Ends of strings should be checked with "startsWith()" and "endsWith()"

Key: typescript:S6557

How To Fix:

One should use `String#startsWith` to check the start of a string and `String#endsWith` to check the end.

Noncompliant code example

```
const str = 'abc';

str[0] === 'a';
str.charAt(0) === 'a';
str.indexOf('abc') === 0;
str.slice(0, 3) === 'abc';
str.substring(0, 3) === 'abc';
str.match(/^abc/) != null;
```

```
/^abc/.test(str);
```

Compliant solution

```
str.startsWith('abc');
```

Noncompliant code example

```
const str = 'abc';

str[str.length - 1] === 'c';
str.charAt(str.length - 1) === 'c';
str.lastIndexOf('abc') === str.length - 3;
str.slice(-3) === 'abc';
str.substring(str.length - 3) === 'abc';
str.match(/abc$/) !== null;
/abc$/.test(str);
```

Compliant solution

```
str.endsWith('abc');
```

Root Cause:

When writing code, it is quite common to test patterns against string ends. For a long time, JavaScript did not provide proper support for this use case. As a result, developers have been relying on various programming subtleties to check the start or end of a string. Examples are getting the index of a substring, slicing the beginning of a string, extracting a substring from the head, matching a regular expression beginning or ending with a pattern, and so on.

While these approaches are all technically valid, they look more like hacking than anything else, blur the developer's intent, but more importantly affect code readability.

Since ES2015, JavaScript provides `String#startsWith` and `String#endsWith`, which are the preferred ways to test patterns against string ends.

Resources:

Documentation

- MDN web docs - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/startsWith
- MDN web docs - [String#endsWith](#)

Type constituents of unions and intersections should not be redundant

Key: typescript:S6571

Resources:

Documentation

- TypeScript Documentation - [Union Types](#)

- TypeScript Documentation - [Intersection Types](#)

How To Fix:

The redundant and overridden types should be removed.

Noncompliant code example

```
type UnionWithAny = any | 'redundant'; // Noncompliant
type UnionWithNever = never | 'override'; // Noncompliant
type UnionWithLiteral = number | 1; // Noncompliant

type IntersectionWithAny = any & 'redundant'; // Noncompliant
type IntersectionWithUnknown = string & unknown; // Noncompliant
type IntersectionWithLiteral = string & 'override'; // Noncompliant
```

Compliant solution

```
type UnionWithAny = any;
type UnionWithNever = never;
type UnionWithLiteral = number;

type IntersectionWithAny = any;
type IntersectionWithUnknown = string;
type IntersectionWithLiteral = 'override';
```

Root Cause:

When defining a union or intersection in TypeScript, it is possible to mistakenly include type constituents that encompass other constituents, that don't have any effect, or that are more restrictive. For instance,

- The type `something` in `any | something` is redundant because `any` covers all possible types, whatever `something` is.
- The types `never` in unions like `never | something` or `unknown` in intersections like `unknown & something` are effectless.
- More restrictive types in intersections like the literal type `1` in `1 & number` reduce the set of possible values to specific ones.

Eliminating redundant types from a union or intersection type simplifies the code and enhances its readability. Moreover, it provides a clearer representation of the actual values that a variable can hold.

Introduction:

In unions and intersections, redundant types should not be used.

"RegExp.exec()" should be preferred over "String.match()"

Key: typescript:S6594

Root Cause:

`String.match()` behaves the same way as `RegExp.exec()` when the regular expression does not include the global flag

g. While they work the same, `RegExp.exec()` can be slightly faster than `String.match()`. Therefore, it should be preferred for better performance.

The rule reports an issue on a call to `String.match()` whenever it can be replaced with semantically equivalent

`RegExp.exec()`.

```
'foo'.match(/bar/);
```

Rewrite the pattern matching from `string.match(regex)` to `regex.exec(string)`.

```
/bar/.exec('foo');
```

Resources:

Documentation

- MDN web docs - [String.prototype.match\(\)](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/String/match)
- MDN web docs - [RegExp.prototype.exec\(\)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp/exec)

Nullish coalescing should be preferred

Key: typescript:S6606

Root Cause:

The nullish coalescing operator `??` allows providing a default value when dealing with `null` or `undefined`. It only coalesces when the original value is `null` or `undefined`. Therefore, it is safer and shorter than relying upon chaining logical `||` expressions or testing against `null` or `undefined` explicitly.

This rule reports when disjunctions (`||`) and conditionals (`?`) can be safely replaced with coalescing (`??`).

The TSConfig needs to set `strictNullChecks` to `true` for the rule to work properly.

How To Fix:

Rewrite the logical expression `||` using `??` on the unchecked operands.

Noncompliant code example

```
function either(x: number | undefined, y: number) {  
    return x || y;  
}
```

Compliant solution

```
function either(x: number | undefined, y: number) {  
    return x ?? y;  
}
```


Noncompliant code example

```
function either(x: number | undefined, y: number) {  
  return x !== undefined ? x : y;  
}
```

Compliant solution

```
function either(x: number | undefined, y: number) {  
  return x ?? y;  
}
```

Resources:

- [MDN - Nullish coalescing](#)

Use Object.hasOwn static method instead of hasOwnProperty

Key: typescript:S6653

Resources:

Documentation

- MDN web docs - [Object.hasOwn\(\)](#)
- MDN web docs - [href="https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/hasOwnProperty">Object.prototype.hasOwnProperty\(\)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/hasOwnProperty)

Root Cause:

The `Object.hasOwn()` method was introduced in ES2022 as a replacement for the more verbose `Object.prototype.hasOwnProperty.call()`. These methods return `true` if the specified property of an object exists as its *own* property. If the property is only available further down the prototype chain or does not exist at all - the methods return `false`.

If you are still using the old method - replace it with a simpler and more concise alternative.

You should also avoid calling the `obj.hasOwnProperty()` method directly, without using `Object.prototype` as a source. This can lead to a runtime error if `obj.prototype` is `null` and therefore `obj.hasOwnProperty` is `undefined`. The static method `Object.hasOwn()` does not depend on the `obj.prototype` and is therefore safe to use in such cases.

```
Object.prototype.hasOwnProperty.call(obj, "propertyName"); // Noncompliant  
Object.hasOwnProperty.call(obj, "propertyName"); // Noncompliant  
({}).hasOwnProperty.call(obj, "propertyName"); // Noncompliant
```

To fix the code replace `hasOwnProperty()` with `Object.hasOwn()`

```
Object.hasOwn(obj, "propertyName");
```

If statements should not be the only statement in else blocks

Key: typescript:S6660

Root Cause:

When `if` is the only statement in the `else` block, it is better to use `else if` because it simplifies the code and makes it more readable.

When using nested `if` statements, it can be difficult to keep track of the logic and understand the flow of the code. Using `else`

`if` makes the code more concise and easier to follow.

```
if (condition1) {  
    // ...  
} else {  
    if (condition2) { // Noncompliant: 'if' statement is the only statement in the  
        'else' block  
        // ...  
    }  
}  
  
if (condition3) {  
    // ...  
} else {  
    if (condition4) { // Noncompliant: 'if' statement is the only statement in the  
        'else' block  
        // ...  
    } else {  
        // ...  
    }  
}
```

Fix your code by using `else if` if the nested `if` is the only statement in the `else` block.

```
if (condition1) {  
    // ...  
} else if (condition2) {  
    // ...  
}  
  
if (condition3) {  
    // ...  
} else if (condition4) {  
    // ...  
} else {  
    // ...  
}
```

Resources:

Documentation

- MDN web docs - [if...else](#)

Spread syntax should be used instead of "apply()"

Key: typescript:S6666

Root Cause:

The spread operator is a more concise and more readable way to pass arguments to a function that takes a variable number of arguments (variadic function). Prior to ES2015, the only way to call such functions with a variable number of arguments was to use the `.apply()` method.

```
foo.apply(undefined, args); // Noncompliant: use spread syntax instead of .apply()
foo.apply(null, args); // Noncompliant: use spread syntax instead of .apply()
obj.foo.apply(obj, args); // Noncompliant: use spread syntax instead of .apply()
```

Using `.apply()` is no longer necessary in such cases - replace it with a spread operator applied to the array of arguments.

```
foo( ...args);
foo( ...args);
obj.foo( ...args);
```

Resources:

Documentation

- MDN web docs - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax#spread_in_function_calls>spread syntax
- MDN web docs - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/apply>apply()

Literals should not be used for promise rejection

Key: typescript:S6671

Resources:

Documentation

- MDN web docs - [Promise](#)
- MDN web docs - [Error](#)

Related rules

- [S3696](#) - Literals should not be thrown

Root Cause:

The use of literals (primitive values such as strings, numbers, booleans, etc.) for promise rejection is generally discouraged. While it is syntactically valid to provide literals as a rejected promise value, it is considered best practice to use instances of the `Error` class or its subclasses instead.

Using an instance of the `Error` class allows you to provide more meaningful information about the error. The `Error` class and its subclasses provide properties such as `message` and `stack` that can be used to convey useful details about the error, such as a description of the problem, the context in which it occurred, or a stack trace for debugging.

```
new Promise(function(resolve, reject) {  
  reject(); // Noncompliant: use Error object to provide rejection reason  
});  
  
new Promise(function(resolve, reject) {  
  reject('Something went wrong'); // Noncompliant: use Error object instead of literal  
});
```

To fix your code provide an instance of the Error class to the promise reject function.

```
new Promise(function(resolve, reject) {  
  reject(new Error('Network timeout'));  
});  
  
new Promise(function(resolve, reject) {  
  reject(new Error('Something went wrong'));  
});
```

Mutable variables should not be exported

Key: typescript:S6861

Resources:

Documentation

- MDN web docs - [let](#)
- MDN web docs - [const](#)
- MDN web docs - [Mutable](#)
- MDN web docs - [Immutable](#)

Root Cause:

In JavaScript, a mutable variable is one whose value can be changed after it has been initially set. This is in contrast to immutable variables, whose values cannot be changed once they are set.

Exporting mutable variables can lead to unpredictable behavior and bugs in your code. This is because any module that imports the variable can change its value. If multiple modules import and change the value of the same variable, it can become difficult to track what the current value of the variable is and which module changed it last.

How To Fix:

If the value of the variable does not need to change, you can declare it as a constant using the `const` keyword. Alternatively, if you have a group of related variables that need to be mutable, consider using a class to encapsulate them. You can then export an instance of the class, or a factory function that creates instances of the class.

Noncompliant code example

```
let mutableVar = "initial value";
```

```
export { mutableVar }; // Noncompliant
```

Compliant solution

```
const immutableVar = "constant value";  
export { immutableVar };
```

or

```
class MyClass {  
  constructor() {  
    this.mutableVar = "initial value";  
  }  
}  
  
export function createMyClass() {  
  return new MyClass();  
}
```

Input bindings should not be aliased

Key: typescript:S7649

Root Cause:

Aliasing input bindings creates confusion between the template usage and the component implementation. When an input property has a different name in templates than in the component class, it becomes harder for developers to understand the relationship between the two.

This inconsistency makes code maintenance more difficult, especially when multiple developers work on the same codebase. It also breaks the principle of least surprise, where developers expect the template binding name to match the property name.

The practice can lead to debugging difficulties when developers search for property usage but cannot find template references due to the name mismatch.

What is the potential impact?

Code becomes harder to understand and maintain. Developers may struggle to trace the connection between template bindings and component properties, leading to increased development time and potential bugs during refactoring.

How to fix in Angular?

Remove the alias and use the property name directly in both the component and template. This creates a clear, consistent naming convention.

Non-compliant code example

```
@Component({})  
class MyComponent {  
  @Input('userName') name: string; // Noncompliant
```

```
}
```

Compliant code example

```
@Component({})  
class MyComponent {  
  @Input() userName: string;  
}
```

For the new input() function, remove the alias option and use the property name directly.

Non-compliant code example

```
@Component({})  
class MyComponent {  
  name = input('', { alias: 'userName' }); // Noncompliant  
}
```

Compliant code example

```
@Component({})  
class MyComponent {  
  userName = input('');  
}
```

In the inputs array, use the property name without aliasing syntax.

Non-compliant code example

```
@Component({  
  inputs: ['name: userName'] // Noncompliant  
})  
class MyComponent {}
```

Compliant code example

```
@Component({  
  inputs: ['userName']  
})  
class MyComponent {}
```

Documentation

- Angular Input Decorator - [Official documentation for the @Input decorator](#)
- Angular Component Inputs - [Guide on using inputs in Angular components](#)

Introduction:

This rule raises an issue when Angular input properties are given different names (aliases) than their actual property names.

Output bindings should not be named "on" or prefixed with "on"

Key: typescript:S7652

Introduction:

This rule raises an issue when Angular output bindings (using `@Output()` decorator or `output()` function) are named exactly "on" or start with "on".

Root Cause:

Angular follows specific naming conventions for component outputs to maintain consistency and avoid confusion.

Outputs named "on" or prefixed with "on" (like `onClick`, `onSubmit`) conflict with Angular's recommended patterns. These names suggest when something happens rather than what happened, which goes against Angular's event naming philosophy.

Using "on" prefixes can also create confusion with native DOM event handlers, making code harder to understand and maintain. Angular's style guide recommends descriptive event names that clearly indicate the action or state change that occurred.

What is the potential impact?

Using improper output naming can lead to confusion between component events and DOM events, making code less readable and harder to maintain. It also violates Angular's established conventions, potentially causing issues in team environments where consistent coding standards are important.

How to fix in Angular?

Rename outputs to use descriptive names without the "on" prefix. Focus on what happened rather than when it happened.

Non-compliant code example

```
@Component()  
class MyComponent {  
  @Output() onClick = new EventEmitter(); // Noncompliant  
  @Output() onSubmit = new EventEmitter(); // Noncompliant  
}
```

Compliant code example

```
@Component()  
class MyComponent {  
  @Output() click = new EventEmitter();  
  @Output() submit = new EventEmitter();  
}
```

For the newer `output()` function, avoid "on" prefixes in both property names and aliases.

Non-compliant code example

```
@Component()  
class MyComponent {
```

```
onClick = output(); // Noncompliant
change = output({ alias: 'onChange' }); // Noncompliant
}
```

Compliant code example

```
@Component()
class MyComponent {
  click = output();
  change = output({ alias: 'change' });
}
```

In component metadata, ensure output declarations don't use "on" prefixes.

Non-compliant code example

```
@Component({
  outputs: ['onClick', 'onSubmit'] // Noncompliant
})
class MyComponent {}
```

Compliant code example

```
@Component({
  outputs: ['click', 'submit']
})
class MyComponent {}
```

Documentation

- Angular Component Outputs Guide - [Official Angular documentation on component outputs and naming conventions](#)
- Angular Style Guide - Event Names - [Angular's official style guide recommendations for event naming](#)

Angular classes should implement lifecycle interfaces for their lifecycle methods

Key: typescript:S7655

Root Cause:

Angular provides lifecycle interfaces like `OnInit`, `OnDestroy`, and `DoBootstrap` that correspond to lifecycle methods such as `ngOnInit()`, `ngOnDestroy()`, and `ngDoBootstrap()`.

When a class implements lifecycle methods without the corresponding interfaces, several problems arise:

- **Type safety is reduced:** TypeScript cannot verify that the method signature matches the expected interface
- **IDE support is limited:** Code completion and refactoring tools work less effectively
- **Code clarity suffers:** Other developers cannot easily see which lifecycle hooks the class uses

- **Angular style guide violations:** The official Angular style guide recommends implementing lifecycle interfaces

The Angular compiler and runtime will still work without the interfaces, but the code becomes less maintainable and more error-prone.

What is the potential impact?

Without lifecycle interfaces, developers may accidentally implement lifecycle methods with incorrect signatures, leading to runtime issues where the methods are not called as expected. This can cause initialization logic to fail silently or cleanup code to not execute properly.

How to fix in Angular?

Import and implement the corresponding lifecycle interface for each lifecycle method in your class.

Non-compliant code example

```
@Component()  
class MyComponent {  
  ngOnInit() { // Noncompliant  
    console.log('Component initialized');  
  }  
  
  ngOnDestroy() { // Noncompliant  
    console.log('Component destroyed');  
  }  
}
```

Compliant code example

```
import { OnInit, OnDestroy } from '@angular/core';  
  
@Component()  
class MyComponent implements OnInit, OnDestroy {  
  ngOnInit() {  
    console.log('Component initialized');  
  }  
  
  ngOnDestroy() {  
    console.log('Component destroyed');  
  }  
}
```

When extending a base class that already implements the interface, use the `override` keyword instead of implementing the interface again.

Non-compliant code example

```
@Component()  
class BaseComponent implements OnInit {  
  ngOnInit() {
```

```
        console.log('Base initialization');
    }
}

@Component()
class DerivedComponent extends BaseComponent {
    ngOnInit() { // Noncompliant
        super.ngOnInit();
        console.log('Derived initialization');
    }
}
```

Compliant code example

```
@Component()
class BaseComponent implements OnInit {
    ngOnInit() {
        console.log('Base initialization');
    }
}

@Component()
class DerivedComponent extends BaseComponent {
    override ngOnInit() {
        super.ngOnInit();
        console.log('Derived initialization');
    }
}
```

Documentation

- Angular Lifecycle Hooks - [Official Angular documentation on lifecycle hooks and interfaces](#)
- Angular Style Guide - Lifecycle Interfaces - [Angular style guide recommendation for implementing lifecycle interfaces](#)

Introduction:

This rule raises an issue when an Angular class defines lifecycle methods without implementing the corresponding lifecycle interfaces.