GDL API Scan Report

Project Name	GDL API
Scan Start	Thursday, April 25, 2019 4:28:13 PM
Preset	Checkmarx Default
Scan Time	01h:59m:00s
Lines Of Code Scanned	91376
Files Scanned	24
Report Creation Time	Thursday, April 25, 2019 5:27:27 PM
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557
Team	GlobalDataLayer
Checkmarx Version	8.8.0.72 HF8
Scan Type	Full
Source Origin	LocalPath
Density	2/10000 (Vulnerabilities/LOC)
Visibility	Public

Filter Settings

Severity

Included: High, Medium, Low, Information

Excluded: None

Result State

Included: Confirmed, Not Exploitable, To Verify, Urgent, Proposed Not Exploitable

Excluded: None

Assigned to

Included: All

Categories

Included:

Uncategorized	All
Custom	All
PCI DSS v3.2	All
OWASP Top 10 2013	All
FISMA 2014	All
NIST SP 800-53	All
OWASP Top 10 2017	All
OWASP Mobile Top 10 2016	All

Excluded:

Uncategorized	None
Custom	None
PCI DSS v3.2	None
OWASP Top 10 2013	None
FISMA 2014	None

NIST SP 800-53	None
OWASP Top 10 2017	None
OWASP Mobile Top 10 2016	None

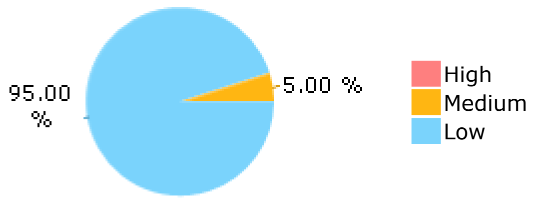
Results Limit

Results limit per query was set to 50

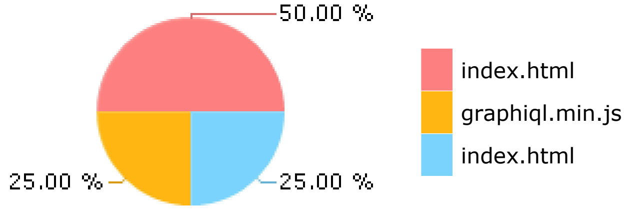
Selected Queries

Selected queries are listed in Result Summary

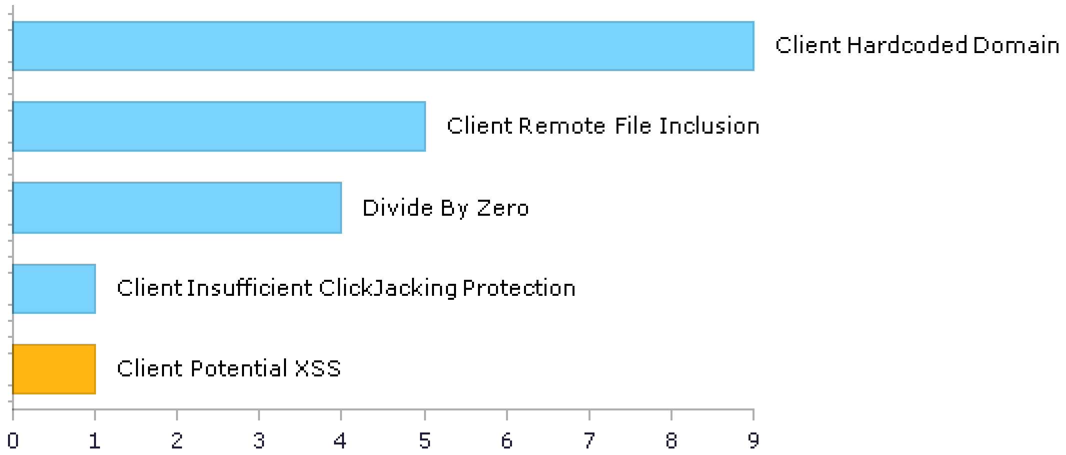
Result Summary



Most Vulnerable Files



Top 5 Vulnerabilities



Scan Summary - OWASP Top 10 2017

Further details and elaboration about vulnerabilities and risks can be found at: OWASP Top 10 2017

Category	Threat Agent	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection	App. Specific	EASY	COMMON	EASY	SEVERE	App. Specific	5	5
A2-Broken Authentication	App. Specific	EASY	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A3-Sensitive Data Exposure	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	App. Specific	0	0
A4-XML External Entities (XXE)	App. Specific	AVERAGE	COMMON	EASY	SEVERE	App. Specific	0	0
A5-Broken Access Control*	App. Specific	AVERAGE	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A6-Security Misconfiguration	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A7-Cross-Site Scripting (XSS)*	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A8-Insecure Deserialization	App. Specific	DIFFICULT	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A9-Using Components with Known Vulnerabilities	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	MODERATE	App. Specific	0	0
A10-Insufficient Logging & Monitoring	App. Specific	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	App. Specific	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - OWASP Top 10 2013

Further details and elaboration about vulnerabilities and risks can be found at: OWASP Top 10 2013

Category	Threat Agent	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	AVERAGE	SEVERE	ALL DATA	0	0
A2-Broken Authentication and Session Management	EXTERNAL, INTERNAL USERS	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	AFFECTED DATA AND FUNCTIONS	0	0
A3-Cross-Site Scripting (XSS)*****	EXTERNAL, INTERNAL, ADMIN USERS	AVERAGE	VERY WIDESPREAD	EASY	MODERATE	AFFECTED DATA AND SYSTEM	0	0
A4-Insecure Direct Object References*****	SYSTEM USERS	EASY	COMMON	EASY	MODERATE	EXPOSED DATA	0	0
A5-Security Misconfiguration	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	EASY	MODERATE	ALL DATA AND SYSTEM	0	0
A6-Sensitive Data Exposure	EXTERNAL, INTERNAL, ADMIN USERS, USERS BROWSERS	DIFFICULT	UNCOMMON	AVERAGE	SEVERE	EXPOSED DATA	0	0
A7-Missing Function Level Access Control	EXTERNAL, INTERNAL USERS	EASY	COMMON	AVERAGE	MODERATE	EXPOSED DATA AND FUNCTIONS	0	0
A8-Cross-Site Request Forgery (CSRF)*****	USERS BROWSERS	AVERAGE	COMMON	EASY	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A9-Using Components with Known Vulnerabilities	EXTERNAL USERS, AUTOMATED TOOLS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A10-Unvalidated Redirects and Forwards	USERS BROWSERS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0

***** Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - PCI DSS v3.2

Category	Issues Found	Best Fix Locations
PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection	5	5
PCI DSS (3.2) - 6.5.2 - Buffer overflows	0	0
PCI DSS (3.2) - 6.5.3 - Insecure cryptographic storage	0	0
PCI DSS (3.2) - 6.5.4 - Insecure communications	0	0
PCI DSS (3.2) - 6.5.5 - Improper error handling	0	0
PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)	0	0
PCI DSS (3.2) - 6.5.8 - Improper access control	0	0
PCI DSS (3.2) - 6.5.9 - Cross-site request forgery*	0	0
PCI DSS (3.2) - 6.5.10 - Broken authentication and session management	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - FISMA 2014

Category	Description	Issues Found	Best Fix Locations
Access Control	Organizations must limit information system access to authorized users, processes acting on behalf of authorized users, or devices (including other information systems) and to the types of transactions and functions that authorized users are permitted to exercise.	0	0
Audit And Accountability	Organizations must: (i) create, protect, and retain information system audit records to the extent needed to enable the monitoring, analysis, investigation, and reporting of unlawful, unauthorized, or inappropriate information system activity; and (ii) ensure that the actions of individual information system users can be uniquely traced to those users so they can be held accountable for their actions.	0	0
Configuration Management	Organizations must: (i) establish and maintain baseline configurations and inventories of organizational information systems (including hardware, software, firmware, and documentation) throughout the respective system development life cycles; and (ii) establish and enforce security configuration settings for information technology products employed in organizational information systems.	0	0
Identification And Authentication	Organizations must identify information system users, processes acting on behalf of users, or devices and authenticate (or verify) the identities of those users, processes, or devices, as a prerequisite to allowing access to organizational information systems.	0	0
Media Protection	Organizations must: (i) protect information system media, both paper and digital; (ii) limit access to information on information system media to authorized users; and (iii) sanitize or destroy information system media before disposal or release for reuse.	0	0
System And Communications Protection	Organizations must: (i) monitor, control, and protect organizational communications (i.e., information transmitted or received by organizational information systems) at the external boundaries and key internal boundaries of the information systems; and (ii) employ architectural designs, software development techniques, and systems engineering principles that promote effective information security within organizational information systems.	0	0
System And Information Integrity*****	Organizations must: (i) identify, report, and correct information and information system flaws in a timely manner; (ii) provide protection from malicious code at appropriate locations within organizational information systems; and (iii) monitor information system security alerts and advisories and take appropriate actions in response.	0	0

***** Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - NIST SP 800-53

Category	Issues Found	Best Fix Locations
AC-12 Session Termination (P2)	0	0
AC-3 Access Enforcement (P1)	0	0
AC-4 Information Flow Enforcement (P1)	0	0
AC-6 Least Privilege (P1)	0	0
AU-9 Protection of Audit Information (P1)	0	0
CM-6 Configuration Settings (P2)	0	0
IA-5 Authenticator Management (P1)	0	0
IA-6 Authenticator Feedback (P2)	0	0
IA-8 Identification and Authentication (Non-Organizational Users) (P1)	0	0
SC-12 Cryptographic Key Establishment and Management (P1)	0	0
SC-13 Cryptographic Protection (P1)	0	0
SC-17 Public Key Infrastructure Certificates (P1)	0	0
SC-18 Mobile Code (P2)	10	10
SC-23 Session Authenticity (P1)*****	0	0
SC-28 Protection of Information at Rest (P1)	0	0
SC-4 Information in Shared Resources (P1)	0	0
SC-5 Denial of Service Protection (P1)	0	0
SC-8 Transmission Confidentiality and Integrity (P1)	0	0
SI-10 Information Input Validation (P1)*****	0	0
SI-11 Error Handling (P2)	0	0
SI-15 Information Output Filtering (P0)*****	0	0
SI-16 Memory Protection (P1)	0	0

***** Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - OWASP Mobile Top 10 2016

Category	Description	Issues Found	Best Fix Locations
M1-Improper Platform Usage	This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security control that is part of the mobile operating system. There are several ways that mobile apps can experience this risk.	0	0
M2-Insecure Data Storage*	This category covers insecure data storage and unintended data leakage.	0	0
M3-Insecure Communication	This category covers poor handshaking, incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc.	0	0
M4-Insecure Authentication	This category captures notions of authenticating the end user or bad session management. This can include: -Failing to identify the user at all when that should be required -Failure to maintain the user's identity when it is required -Weaknesses in session management	0	0
M5-Insufficient Cryptography	The code applies cryptography to a sensitive information asset. However, the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3. Also, if the app fails to use cryptography at all when it should, that probably belongs in M2. This category is for issues where cryptography was attempted, but it wasnt done correctly.	0	0
M6-Insecure Authorization	This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced browsing, etc.). It is distinct from authentication issues (e.g., device enrolment, user identification, etc.). If the app does not authenticate users at all in a situation where it should (e.g., granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure.	0	0
M7-Client Code Quality*	This category is the catch-all for code-level implementation problems in the mobile client. That's distinct from server-side coding mistakes. This would capture things like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device.	0	0
M8-Code Tampering*	This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or	0	0

	modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.		
M9-Reverse Engineering	This category includes analysis of the final core binary to determine its source code, libraries, algorithms, and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker insight into the inner workings of the application. This may be used to exploit other nascent vulnerabilities in the application, as well as revealing information about back end servers, cryptographic constants and ciphers, and intellectual property.	0	0
M10-Extraneous Functionality	Often, developers include hidden backdoor functionality or other internal development security controls that are not intended to be released into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid app. Another example includes disabling of 2-factor authentication during testing.	0	0

***** Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - Custom

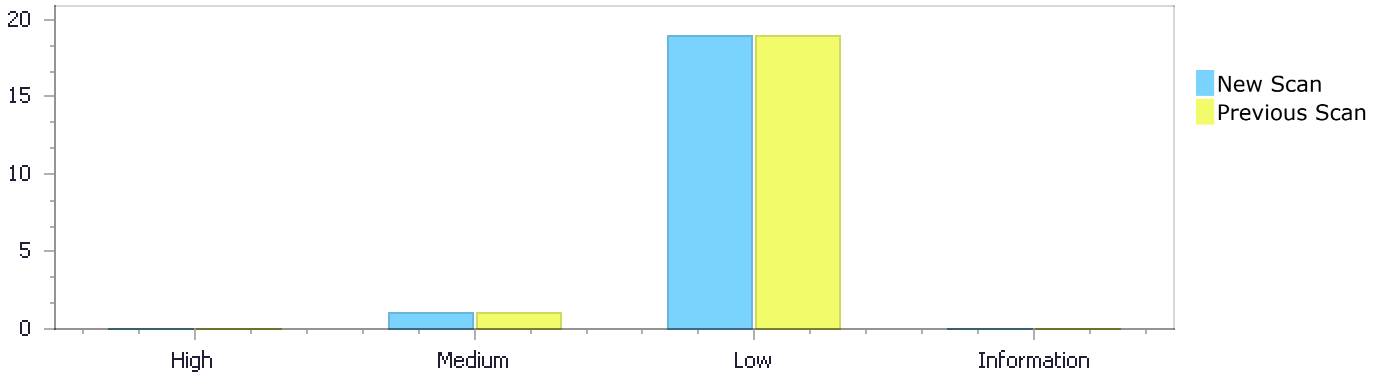
Category	Issues Found	Best Fix Locations
Must audit	0	0
Check	0	0
Optional	0	0

Results Distribution By Status

Compared to project scan from 4/25/2019 4:24 PM

	High	Medium	Low	Information	Total
New Issues	0	0	0	0	0
Recurrent Issues	0	0	10	0	10
Total	0	0	10	0	10

Fixed Issues	0	0	0	0	0



Results Distribution By State

	High	Medium	Low	Information	Total
Confirmed	0	0	0	0	0
Not Exploitable	0	1	9	0	10
To Verify	0	0	10	0	10
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	0	1	19	0	20

Result Summary

Vulnerability Type	Occurrences	Severity
Client Potential XSS	1	Medium
Client Hardcoded Domain	9	Low
Client Remote File Inclusion	5	Low
Divide By Zero	4	Low
Client Insufficient ClickJacking Protection	1	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
graphiql/graphiql.min.js	1

Scan Results Details

Client Potential XSS

Query Path:
JavaScript\Cx\JavaScript Medium Threat\Client Potential XSS Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)
OWASP Top 10 2013: A3-Cross-Site Scripting (XSS)
FISMA 2014: Access Control
NIST SP 800-53: SI-15 Information Output Filtering (P0)
OWASP Top 10 2017: A7-Cross-Site Scripting (XSS)

Description

Client Potential XSS\Path 1:

Severity	Medium
Result State	Not Exploitable
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=7
Status	Recurrent

Method insertLineContent at line 8484 of graphiql/graphiql.min.js gets user input for the text element. This element's value then flows through the code without being properly sanitized or validated and is eventually displayed to the user in method buildCollapsedSpan at line 8480 of graphiql/graphiql.min.js. This may enable a Cross-Site-Scripting attack.

	Source	Destination
File	graphiql/graphiql.min.js	graphiql/graphiql.min.js
Line	8486	8482
Object	text	content

Code Snippet
File Name graphiql/graphiql.min.js
Method function insertLineContent(line, builder, styles) {

```
....
8486.                              allText = line.text,
```

File Name graphiql/graphiql.min.js
Method function buildCollapsedSpan(builder, size, marker, ignoreWidget) {

```
....
8482.                           widget && builder.map.push(builder.pos,
builder.pos + size, widget), !ignoreWidget &&
builder.cm.display.input.needsContentAttribute && (widget || (widget =
builder.content.appendChild(document.createElement("span"))),
widget.setAttribute("cm-marker", marker.id)), widget &&
(builder.cm.display.input.setUneditable(widget),
builder.content.appendChild(widget)), builder.pos += size,
builder.trailingSpace = !1
```

Client Hardcoded Domain

Query Path:
JavaScript\Cx\JavaScript Low Visibility\Client Hardcoded Domain Version:1

Categories

NIST SP 800-53: SC-18 Mobile Code (P2)

Description

Client Hardcoded Domain\Path 1:

Severity	Low
Result State	Not Exploitable
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=12
Status	Recurrent

The JavaScript file imported in ""//cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js"" in graphiql/index.html at line 30 is from a remote domain, which may allow attackers to replace its contents with malicious code.

	Source	Destination
File	graphiql/index.html	graphiql/index.html
Line	30	30
Object	""//cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js""	""//cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js""

Code Snippet
File Name graphiql/index.html
Method <script src="//cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js"></script>

```
....
30.      <script src="//cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js"></script>
```

Client Hardcoded Domain\Path 2:

Severity	Low
Result State	Not Exploitable
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=13
Status	Recurrent

The JavaScript file imported in ""//cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js"" in graphiql/index.html at line 31 is from a remote domain, which may allow attackers to replace its contents with malicious code.

	Source	Destination
File	graphiql/index.html	graphiql/index.html
Line	31	31
Object	""//cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js""	""//cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js""

Code Snippet

File Name	graphiql/index.html
Method	<script src="//cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js"></script>

```
....
31.       <script
src="//cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js"></script>
```

Client Hardcoded Domain\Path 3:

Severity	Low
Result State	Not Exploitable
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=14
Status	Recurrent

The JavaScript file imported in ""//cdn.jsdelivr.net/react/15.4.2/react.min.js"" in graphiql/index.html at line 32 is from a remote domain, which may allow attackers to replace its contents with malicious code.

	Source	Destination
File	graphiql/index.html	graphiql/index.html
Line	32	32
Object	""//cdn.jsdelivr.net/react/15.4.2/react.min.js""	""//cdn.jsdelivr.net/react/15.4.2/react.min.js""

Code Snippet

File Name	graphiql/index.html
Method	<script src="//cdn.jsdelivr.net/react/15.4.2/react.min.js"></script>

```
....
32.       <script
src="//cdn.jsdelivr.net/react/15.4.2/react.min.js"></script>
```

Client Hardcoded Domain\Path 4:

Severity	Low
Result State	Not Exploitable
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=15
Status	Recurrent

The JavaScript file imported in ""//cdn.jsdelivr.net/react/15.4.2/react-dom.min.js"" in graphiql/index.html at line 33 is from a remote domain, which may allow attackers to replace its contents with malicious code.

	Source	Destination
File	graphiql/index.html	graphiql/index.html
Line	33	33
Object	""//cdn.jsdelivr.net/react/15.4.2/react-dom.min.js""	""//cdn.jsdelivr.net/react/15.4.2/react-dom.min.js""

Code Snippet

File Name	graphiql/index.html
Method	<script src="//cdn.jsdelivr.net/react/15.4.2/react-dom.min.js"></script>

```
....
33.      <script src="//cdn.jsdelivr.net/react/15.4.2/react-
dom.min.js"></script>
```

Client Hardcoded Domain\Path 5:

Severity	Low
Result State	To Verify
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=16
Status	Recurrent

The JavaScript file imported in ""https://cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js"" in voayger/index.html at line 23 is from a remote domain, which may allow attackers to replace its contents with malicious code.

	Source	Destination
File	voayger/index.html	voayger/index.html
Line	23	23
Object	""https://cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js""	""https://cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js""

Code Snippet

File Name	voayger/index.html
Method	<script src="https://cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js"></script>

```
....
23.      <script src="https://cdn.jsdelivr.net/es6-promise/4.0.5/es6-
promise.auto.min.js"></script>
```

Client Hardcoded Domain\Path 6:

Severity	Low
Result State	To Verify
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=17
Status	Recurrent

The JavaScript file imported in ""https://cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js"" in voayger/index.html at line 24 is from a remote domain, which may allow attackers to replace its contents with malicious code.

	Source	Destination
File	voayger/index.html	voayger/index.html
Line	24	24
Object	""https://cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js""	""https://cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js""

Code Snippet	
File Name	voayger/index.html
Method	<script src="https://cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js"></script>

```
....
24.      <script 
src="https://cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js"></script>
```

Client Hardcoded Domain\Path 7:

Severity	Low
Result State	To Verify
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=18
Status	Recurrent

The JavaScript file imported in ""https://cdn.jsdelivr.net/npm/react@16/umd/react.production.min.js"" in voayger/index.html at line 25 is from a remote domain, which may allow attackers to replace its contents with malicious code.

	Source	Destination
File	voayger/index.html	voayger/index.html
Line	25	25
Object	""https://cdn.jsdelivr.net/npm/react@16/umd/react.production.min.js""	""https://cdn.jsdelivr.net/npm/react@16/umd/react.production.min.js""

Code Snippet	
File Name	voayger/index.html
Method	<script src="https://cdn.jsdelivr.net/npm/react@16/umd/react.production.min.js"></script>

```
....
25.      <script 
src="https://cdn.jsdelivr.net/npm/react@16/umd/react.production.min.js">
</script>
```

Client Hardcoded Domain\Path 8:

Severity	Low
Result State	To Verify
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=19
Status	Recurrent

The JavaScript file imported in ""https://cdn.jsdelivr.net/npm/react-dom@16/umd/react-dom.production.min.js"" in voayger/index.html at line 26 is from a remote domain, which may allow attackers to replace its contents with malicious code.

	Source	Destination
File	voayger/index.html	voayger/index.html

Line	26	26
Object	""https://cdn.jsdelivr.net/npm/react-dom@16/umd/react-dom.production.min.js""	""https://cdn.jsdelivr.net/npm/react-dom@16/umd/react-dom.production.min.js""

Code Snippet
File Name voayger/index.html
Method <script src="https://cdn.jsdelivr.net/npm/react-dom@16/umd/react-dom.production.min.js"></script>

```
....
26.      <script src="https://cdn.jsdelivr.net/npm/react-
dom@16/umd/react-dom.production.min.js"></script>
```

Client Hardcoded Domain\Path 9:

Severity	Low
Result State	To Verify
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=20
Status	Recurrent

The JavaScript file imported in ""https://cdn.jsdelivr.net/npm/graphql-voyager/dist/voyager.min.js"" in voayger/index.html at line 33 is from a remote domain, which may allow attackers to replace its contents with malicious code.

	Source	Destination
File	voayger/index.html	voayger/index.html
Line	33	33
Object	""https://cdn.jsdelivr.net/npm/graphql-voyager/dist/voyager.min.js""	""https://cdn.jsdelivr.net/npm/graphql-voyager/dist/voyager.min.js""

Code Snippet
File Name voayger/index.html
Method <script src="https://cdn.jsdelivr.net/npm/graphql-voyager/dist/voyager.min.js"></script>

```
....
33.      <script src="https://cdn.jsdelivr.net/npm/graphql-
voyager/dist/voyager.min.js"></script>
```

Client Remote File Inclusion

Query Path:
JavaScript\Cx\JavaScript Low Visibility\Client Remote File Inclusion Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection
NIST SP 800-53: SC-18 Mobile Code (P2)
OWASP Top 10 2017: A1-Injection

Description

Client Remote File Inclusion\Path 1:

Severity	Low
Result State	To Verify
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=1
Status	Recurrent

The application loads an external library or source code file using ""https://cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js"", at line 23 of voayger/index.html. An attacker might be able to exploit this and cause the application to load arbitrary code.

	Source	Destination
File	voayger/index.html	voayger/index.html
Line	23	23
Object	""https://cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js""	""https://cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js""

Code Snippet
File Name voayger/index.html
Method <script src="https://cdn.jsdelivr.net/es6-promise/4.0.5/es6-promise.auto.min.js"></script>

```
....
23.       <script src="https://cdn.jsdelivr.net/es6-promise/4.0.5/es6-
promise.auto.min.js"></script>
```

Client Remote File Inclusion\Path 2:

Severity	Low
Result State	To Verify
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=2
Status	Recurrent

The application loads an external library or source code file using ""https://cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js"", at line 24 of voayger/index.html. An attacker might be able to exploit this and cause the application to load arbitrary code.

	Source	Destination
File	voayger/index.html	voayger/index.html
Line	24	24
Object	""https://cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js""	""https://cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js""

Code Snippet
File Name voayger/index.html
Method <script src="https://cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js"></script>

```
....
24.       <script
src="https://cdn.jsdelivr.net/fetch/0.9.0/fetch.min.js"></script>
```

Client Remote File Inclusion\Path 3:

Severity	Low
Result State	To Verify
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=3
Status	Recurrent

The application loads an external library or source code file using ""https://cdn.jsdelivr.net/npm/react@16/umd/react.production.min.js"", at line 25 of voayger/index.html. An attacker might be able to exploit this and cause the application to load arbitrary code.

	Source	Destination
File	voayger/index.html	voayger/index.html
Line	25	25
Object	""https://cdn.jsdelivr.net/npm/react@16/umd/react.production.min.js""	""https://cdn.jsdelivr.net/npm/react@16/umd/react.production.min.js""

Code Snippet

File Name	voayger/index.html
Method	<script src="https://cdn.jsdelivr.net/npm/react@16/umd/react.production.min.js"></script>

```
....
25.       <script
src="https://cdn.jsdelivr.net/npm/react@16/umd/react.production.min.js">
</script>
```

Client Remote File Inclusion\Path 4:

Severity	Low
Result State	To Verify
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=4
Status	Recurrent

The application loads an external library or source code file using ""https://cdn.jsdelivr.net/npm/react-dom@16/umd/react-dom.production.min.js"", at line 26 of voayger/index.html. An attacker might be able to exploit this and cause the application to load arbitrary code.

	Source	Destination
File	voayger/index.html	voayger/index.html
Line	26	26
Object	""https://cdn.jsdelivr.net/npm/react-dom@16/umd/react-dom.production.min.js""	""https://cdn.jsdelivr.net/npm/react-dom@16/umd/react-dom.production.min.js""

Code Snippet	
File Name	voayger/index.html
Method	<script src="https://cdn.jsdelivr.net/npm/react-dom@16/umd/react-dom.production.min.js"></script>

```
....
26.       <script src="https://cdn.jsdelivr.net/npm/react-
dom@16/umd/react-dom.production.min.js"></script>
```

Client Remote File Inclusion\Path 5:

Severity	Low
Result State	To Verify
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=5
Status	Recurrent

The application loads an external library or source code file using ""https://cdn.jsdelivr.net/npm/graphql-voyager/dist/voyager.min.js"", at line 33 of voayger/index.html. An attacker might be able to exploit this and cause the application to load arbitrary code.

	Source	Destination
File	voayger/index.html	voayger/index.html
Line	33	33
Object	""https://cdn.jsdelivr.net/npm/graphql-voyager/dist/voyager.min.js""	""https://cdn.jsdelivr.net/npm/graphql-voyager/dist/voyager.min.js""

Code Snippet	
File Name	voayger/index.html
Method	<script src="https://cdn.jsdelivr.net/npm/graphql-voyager/dist/voyager.min.js"></script>

```
....
33.       <script src="https://cdn.jsdelivr.net/npm/graphql-
voyager/dist/voyager.min.js"></script>
```

Divide By Zero

Query Path:
JavaScript\Cx\JavaScript Server Side Vulnerabilities\Divide By Zero Version:2

Description

Divide By Zero\Path 1:

Severity	Low
Result State	Not Exploitable
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=8
Status	Recurrent

The application performs an illegal operation in insertLineContent, in graphiql/graphiql.min.js. In line 8484, the program attempts to divide by 0, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input 0 in insertLineContent of graphiql/graphiql.min.js, at line 8484.

	Source	Destination
File	graphiql/graphiql.min.js	graphiql/graphiql.min.js
Line	8490	8490
Object	0	0

Code Snippet
File Name graphiql/graphiql.min.js
Method function insertLineContent(line, builder, styles) {

```
....
8490.                                   spanStyle = spanEndStyle =
spanStartStyle = title = css = "", collapsed = null, nextChange = 1 / 0;
```

Divide By Zero\Path 2:

Severity	Low
Result State	Not Exploitable
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=9
Status	Recurrent

The application performs an illegal operation in History, in graphiql/graphiql.min.js. In line 9831, the program attempts to divide by 0, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input 0 in History of graphiql/graphiql.min.js, at line 9831.

	Source	Destination
File	graphiql/graphiql.min.js	graphiql/graphiql.min.js
Line	9832	9832
Object	0	0

Code Snippet
File Name graphiql/graphiql.min.js
Method function History(startGen) {

```
....
9832.                          this.done = [], this.undone = [],
this.undoDepth = 1 / 0, this.lastModTime = this.lastSelTime = 0,
this.lastOp = this.lastSelOp = null, this.lastOrigin =
this.lastSelOrigin = null, this.generation = this.maxGeneration =
startGen || 1
```

Divide By Zero\Path 3:

Severity	Low
Result State	Not Exploitable
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=10
Status	Recurrent

The application performs an illegal operation in done, in graphiql/graphiql.min.js. In line 10686, the program attempts to divide by 0, which might be evaluate to 0 (zero) at time of division. This value could be a hardcoded zero value, or received from external, untrusted input 0 in done of graphiql/graphiql.min.js, at line 10686.

	Source	Destination
File	graphiql/graphiql.min.js	graphiql/graphiql.min.js
Line	10687	10687
Object	0	0

Code Snippet
File Name graphiql/graphiql.min.js
Method function done(e) {

```
....
10687.                              cm.state.selectingText = !1, counter = 1
/ 0, e_preventDefault(e), display.input.focus(), off(document,
"mousemove", move), off(document, "mouseup", up),
doc.history.lastSelOrigin = null
```

Divide By Zero\Path 4:

Severity	Low
Result State	Not Exploitable
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=11
Status	Recurrent

The application performs an illegal operation in CodeMirror.defineExtension, in graphiql/graphiql.min.js. In line 5526, the program attempts to divide by 0, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input 0 in CodeMirror.defineExtension of graphiql/graphiql.min.js, at line 5526.

	Source	Destination
File	graphiql/graphiql.min.js	graphiql/graphiql.min.js
Line	5528	5528
Object	0	0

Code Snippet
File Name graphiql/graphiql.min.js
Method }, CodeMirror.defineExtension("toggleComment", function(options) {

```
....
5528.                        for (var cm = this, minLine = 1 / 0, ranges =
this.listSelections(), mode = null, i = ranges.length - 1; i >= 0; i--)
{
```

Client Insufficient ClickJacking Protection

Query Path:
JavaScript\Cx\JavaScript Low Visibility\Client Insufficient ClickJacking Protection Version:3

Categories

FISMA 2014: Configuration Management

NIST SP 800-53: SC-8 Transmission Confidentiality and Integrity (P1)

Description

Client Insufficient ClickJacking Protection\Path 1:

Severity	Low
Result State	Not Exploitable
Online Results	https://codescan.seceng.rackspace.net/CxWebClient/ViewerMain.aspx?scanid=1033817&projectid=557&pathid=6
Status	Recurrent

The application does not protect the web page graphiql/index.html from clickjacking attacks in legacy browsers, by using framebusting scripts.

	Source	Destination
File	graphiql/index.html	graphiql/index.html
Line	1	1
Object	CxJSNS_526823029	CxJSNS_526823029

Code Snippet

File Name	graphiql/index.html
Method	<!--

```
....
1.   <!--
```

Client Potential XSS

Risk

What might happen

An attacker could use social engineering to cause a user to send the website engineered input, rewriting web pages and inserting malicious scripts. The attacker can then pretend to be the original website, which would enable the attacker to steal the user's password, request the user’s credit card information, provide false information, or run malware. From the victim’s point of view, this is the original website, and the victim would blame the site for incurred damage.

Cause

How does it happen

The application creates web pages that include data from previous user input. The user input is embedded directly in the page's HTML, causing the browser to display it as part of the web page. If the input includes HTML fragments or JavaScript, these are displayed too, and the user cannot tell that this is not the intended page. The vulnerability is the result of embedding arbitrary user input without first encoding it in a format that would prevent the browser from treating it like HTML instead of plain text.

General Recommendations

How to avoid it

1. Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
    - Data type
    - Size

  - Range
  - Format
  - Expected values
2. Fully encode all dynamic data before embedding it in output.
3. Encoding should be context-sensitive. For example:
   - HTML encoding for HTML content
   - HTML Attribute encoding for data output to attribute values
   - JavaScript encoding for server-generated JavaScript.
4. Consider using either the ESAPI encoding library, or the built-in platform functions. For earlier versions of ASP.NET, consider using the AntiXSS library.
5. In the Content-Type HTTP response header, explicitly define character encoding (charset) for the entire page.
6. Set the httpOnly flag on the session cookie, to prevent XSS exploits from stealing the cookie.

Source Code Examples

CSharp

Bad - The application uses the "Referer" field string to construct the HttpResponse

```
public class ReflectedXssAllClients
{
        public static void foo(HttpRequest Request, HttpResponse Response)
    {
                string Referer = Request.QueryString["Referer"];
                Response.BinaryWrite(Referer);
        }
}
```

Good - The "Referer" field string is HTML encoded before use

```
public class ReflectedXssAllClientsFixed
{
        public static void foo(HttpRequest Request, HttpResponse Response,
AntiXss.AntiXssEncoder encoder)
    {
                string Referer = Request.QueryString["Referer"];
                Response.BinaryWrite(encoder.HtmlEncode(Referer, true));
        }
}
```

Bad - User input is written to a TextBox displayed on the screen enabling a user to inject a script

```
public class ReflectedXSSSpecificClients
{
        public void foo(TextBox tb)
    {
```

```
                string input = Console.ReadLine();
                tb.Text = input;
        }
}
```

Good - The user input is HTML encoded before being displayed on the screen

```
public class ReflectedXSSSpecificClientsFixed
{
        public void foo(TextBox tb, AntiXssEncoder encode)
     {
                string input = Console.ReadLine();
                tb.Text = encode.HtmlEncode(input);
        }
}
```

Bad - The application uses the "filename" field string from an HttpRequest construct an HttpResponse

```
public class UTF7XSS
{
        public void foo(HttpRequest Request, HttpResponse Response
     {
         Response.Charset("UTF-7");
                string filename = Request.QueryString["filename"];
                Response.BinaryWrite(AntiXss.HtmlEncode(filename));
        }
}
```

Good - The "filename" string is converted to an int and using a switch case the new "filename" string is constructed

```
public class UTF7XSSFixed
{
        public static void foo(HttpRequest Request, HttpResponse Response)
     {
                Response.Charset("UTF-7");
                string filename = Request.QueryString["fileNum"];
                int fileNum = Convert.ToInt32(filename);

                switch(fileNum)
                {
                        case 1:
                                filename = "File1.txt";
                                break;
                        default:
                                filename = "File2.txt";
                                break;
                }

                Response.BinaryWrite(AntiXss.HtmlEncode(filename));
        }
}
```

Java

User input is written to a label displayed on the screen enabling a user to inject a script

```java
public class ReflectedXSSAllClients {
      public static void XSSExample(TextArea name) {
            Label label = new Label();
            label.setText("Hello " + name.getText());
      }
}
```

Switch case is used in order to assemble the label's text value and manage wrong user input

```java
public class ReflectedXSSAllClientsFixed {
      public static void XSSExample(TextArea name) {
            Label label = new Label();
            switch (name) {
            case "Joan":
                  label.setText("Hello Joan");
                  break;
            case "Jim":
                  label.setText("Hello Jim");
                  break;
            case "James":
                  label.setText("Hello James");
                  break;
            default:
                  System.out.println("Wrong Input");
            }
      }
}
```

Client Remote File Inclusion

Risk

What might happen

If an attacker can select the name of the library, or the location of the code file that is loaded by the application, they would be able to cause the application to execute arbitrary code. This effectively allows the attacker to control the code run by the application.

Cause

How does it happen

The application uses untrusted data to specify the library or code file, without proper sanitization. This causes the application to load any arbitrary code, as specified. The loaded code will then be executed.

General Recommendations

How to avoid it

- Do not dynamically load code libraries, especially not based on user input.
- If it is necessary to use untrusted data to select the library to be loaded, verify the selected library name matches a predefined set of whitelisted library names. Alternatively, use the value as an identifier to select from the whitelisted libraries.
- Validate any untrusted data used to load or process libraries or code files.

Source Code Examples

Client Insufficient ClickJacking Protection

Risk

What might happen

Clickjacking attacks allow an attacker to "hijack" a user's mouse clicks on a webpage, by invisibly framing the application, and superimposing it in front of a bogus site. When the user is convinced to click on the bogus website, e.g. on a link or a button, the user's mouse is actually clicking on the target webpage, despite being invisible.

This could allow the attacker to craft an overlay that, when clicked, would lead the user to perform undesirable actions in the vulnerable application, e.g. enabling the user's webcam, deleting all the user's records, changing the user's settings, or causing clickfraud.

Cause

How does it happen

The root cause of vulnerability to a clickjacking attack, is that the application's web pages can be loaded into a frame of another website. The application does not implement a proper frame-busting script, that would prevent the page from being loaded into another frame. Note that there are many types of simplistic redirection scripts that still leave the application vulnerable to clickjacking techniques, and should not be used.

When dealing with modern browsers, applications mitigate this vulnerability by issuing appropriate Content-Security-Policy or X-Frame-Options headers to indicate to the browser to disallow framing. However, many legacy browsers do not support this feature, and require a more manual approach by implementing a mitigation in Javascript. To ensure legacy support, a framebusting script is required.

General Recommendations

How to avoid it

Generic Guidance:

- Define and implement a a Content Security Policy (CSP) on the server side, including a frame-ancestors directive. Enforce the CSP on all relevant webpages.
- If certain webpages are required to be loaded into a frame, define a specific, whitelisted target URL.
- Alternatively, return a "X-Frame-Options" header on all HTTP responses. If it is necessary to allow a particular webpage to be loaded into a frame, define a specific, whitelisted target URL.
- For legacy support, implement framebusting code using Javascript and CSS to ensure that, if a page is framed, it is never displayed, and attempt to navigate into the frame to prevent attack. Even if navigation fails, the page is not displayed and is therefore not interactive, mitigating potential clickjacking attacks.

Specific Recommendations:

- Implement a proper framebuster script on the client, that is not vulnerable to frame-buster-busting attacks.
    - Code should first disable the UI, such that even if frame-busting is successfully evaded, the UI cannot be clicked. This can be done by setting the CSS value of the "display" attribute to "none" on either the "body" or "html" tags. This is done because, if a frame attempts to redirect and become the parent, the malicious parent can still prevent redirection via various techniques.
    - Code should then determine whether no framing occurs by comparing self === top; if the result is true, can the UI be enabled. If it is false, attempt to navigate away from the framing page by setting the top.location attribute to self.location.

Source Code Examples

JavaScript

Clickjackable Webpage

```
<html>
    <body>
    <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

Bustable Framebuster

```
<html>
    <head>
    <script>
            if ( window.self.location != window.top.location ) {
                   window.top.location = window.self.location;
            }
        </script>
    </head>

    <body>
        <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

Proper Framebusterbusterbusting

```
<html>
    <head>
    <style> html {display : none; } </style>
        <script>
            if ( self === top ) {
                   document.documentElement.style.display = 'block';
            }
            else {
                   top.location = self.location;
            }
        </script>
    </head>

    <body>
        <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

Divide By Zero

Risk

What might happen

When a program divides a number by zero, an exception will be raised. If this exception is not handled by the application, unexpected results may occur, including crashing the application. This can be considered a DoS (Denial of Service) attack, if an external user has control of the value of the denominator or can cause this error to occur.

Cause

How does it happen

The program receives an unexpected value, and uses it for division without filtering, validation, or verifying that the value is not zero. The application does not explicitly handle this error or prevent division by zero from occuring.

General Recommendations

How to avoid it

- Before dividing by an unknown value, validate the number and explicitly ensure it does not evaluate to zero.
- Validate all untrusted input from all sources, in particular verifying that it is not zero before dividing with it.
- Verify output of methods, calculations, dictionary lookups, and so on, and ensure it is not zero before dividing with the result.
- Ensure divide-by-zero errors are caught and handled appropriately.

Source Code Examples

Java

Divide by Zero

```
public float getAverage(HttpServletRequest req) {
      int total = Integer.parseInt(req.getParameter("total"));
      int count = Integer.parseInt(req.getParameter("count"));

      return total / count;
}
```

Checked Division

```
public float getAverage(HttpServletRequest req) {
      int total = Integer.parseInt(req.getParameter("total"));
      int count = Integer.parseInt(req.getParameter("count"));
```

```
        if (count > 0)
                return total / count;
        else
                return 0;
}
```

Client Hardcoded Domain

Risk

What might happen

An externally imported Javascript file may leave users vulnerable to attack - if the Javascript's host is compromised, if communications with the host are intercepted or if the host itself is not trustworthy, then the contents of the Javascript file may change to have malicious code, which could result in a Cross-Site Scripting (XSS) attack.

Cause

How does it happen

Javascript files can be imported dynamically from remote hosts when they are embedded into HTML. However, this reliance on a remote host for these scripts may diminish security, as web-application's users are only ever as secure as the remote host serving these Javascript files.

General Recommendations

How to avoid it

Where possible, host all script files locally, rather than remotely. Ensure that locally hosted 3rd party script files are constantly updated and maintained.

Source Code Examples

JavaScript

Remote Importation of A Script File

```
<script src="https://example.com/scripts/jquery.js" />
```

Local Importation of A Script File

```
<script src="/scripts/jquery.js" />
```

Scanned Languages

Language	Hash Number	Change Date
JavaScript	3602822811217894	2/6/2019
VbScript	1349101913133594	10/3/2017
Python	0153239019201885	2/6/2019
Typescript	1939975091058023	2/6/2019
Common	0206692917308612	2/6/2019