Threat Modeling Report

Project: hospitalportal-master

Property	Value
Project Name	hospitalportal-master
Analysis Date	2025-10-18T21:45:22.501142
Methodology	STRIDE
Total Findings	82
Risk Level	CRITICAL

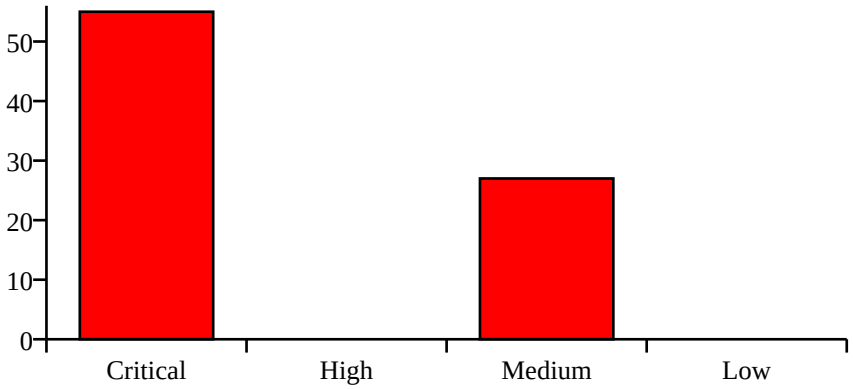
Executive Summary

This security assessment of **hospitalportal-master** identified **82** potential security threats using the STRIDE methodology. The overall risk level is assessed as **CRITICAL**.

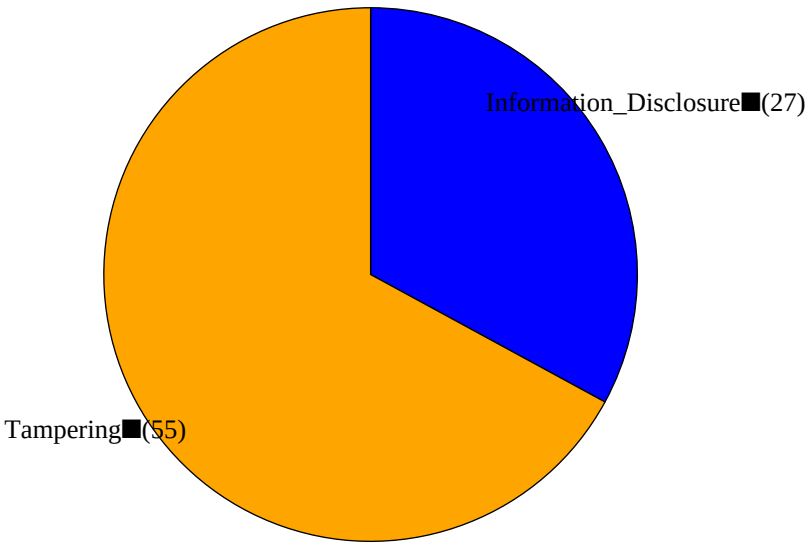
Key Findings:
• Critical vulnerabilities: 55
• High-severity issues: 0
• Medium-priority concerns: 27
• Low-priority items: 0

Immediate attention is required for all critical and high-severity vulnerabilities to prevent potential security breaches.

Threat Severity Distribution



STRIDE Category Distribution



STRIDE Methodology Overview

STRIDE is a threat modeling methodology developed by Microsoft that categorizes security threats into six main areas:

S - Spoofing Identity: Impersonating someone or something else to gain unauthorized access
T - Tampering with Data: Malicious modification of data or code
R - Repudiation: Users denying they performed an action without the system being able to prove otherwise
I - Information Disclosure: Exposure of information to individuals who shouldn't have access
D - Denial of Service: Attacks that deny or degrade service for legitimate users
E - Elevation of Privilege: A user gains capabilities without proper authorization

Each identified threat is categorized into one of these areas and assessed for severity and impact.

Project Architecture Analysis

Code Analysis Summary:
• Files analyzed: 15
• Programming languages: JavaScript
• Threat detection patterns: STRIDE-based security analysis
• Analysis depth: Source code static analysis with context awareness

Detailed Security Findings

Finding #1: Code injection via eval

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.80
File Location	hospitalportal-master/HospitalPortal/Content/js/jquery.validate.unobtrusive.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via eval detected in JavaScript code

Code Evidence:

```
** Copyright (C) Microsoft Corporation. All rights reserved. */ >>>
(function(a){var d=a.validator,b,e="unobtrusiveValidation";function
c(a,b,c){a.rules[b]=c;if(a.message)a.messages[b]=a.message}function
j(a){return a.replace(/^\s+|\s+$/g,"").split(/\s*,\s*/g)}function f(a){return
a.replace(/([!"#$%&'()*+,./:;<=>?@\[\\\]^`{|}~])/g,"\\$1")}function
h(a){return a.substr(0,a.lastIndexOf(".")+1)}function
g(a,b){if(a.indexOf("*.")===0)a=a.replace("*.",b);return a}function
m(c,e){var b=a(this).find("[data-valmsg-for='"+f(e[0].name)+"']"),d=b.attr("d
ata-valmsg-replace"),g=d?a.parseJSON(d)!==false:null;b.removeClass("field-val
idation-valid").addClass("field-validation-error");c.data("unobtrusiveContain
er",b);if(g){b.empty();c.removeClass("input-validation-error").appendTo(b)}el
se c.hide()}function l(e,d){var c=a(this).find("[data-valmsg-summary=true]"),
b=c.find("ul");if(b&&b.length;&&d.errorList.length;){b.empty();c.addClass("va
lidation-summary-errors").removeClass("valida...
```

Proof of Concept:

Steps to Reproduce:
1. Locate the vulnerable eval() function in the source code
2. Identify user input that reaches the eval() function
3. Craft malicious JavaScript payload
4. Execute payload through the vulnerable input vector
5. Observe code execution in the application context

Impact: Remote code execution, full application compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #2: Code injection via eval

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.80
File Location	hospitalportal-master/HospitalPortal/Content/js/jquery.validate.unobtrusive.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via eval detected in JavaScript code

Code Evidence:

```
** Copyright (C) Microsoft Corporation. All rights reserved. */ >>>
(function(a){var d=a.validator,b,e="unobtrusiveValidation";function
c(a,b,c){a.rules[b]=c;if(a.message)a.messages[b]=a.message}function
j(a){return a.replace(/^\s+|\s+$/g,"").split(/\s*,\s*/g)}function f(a){return
a.replace(/([!"#$%&'()*+,./:;<=>?@\[\\\]^`{|}~])/g,"\\$1")}function
h(a){return a.substr(0,a.lastIndexOf(".")+1)}function
g(a,b){if(a.indexOf("*.")===0)a=a.replace("*.",b);return a}function
m(c,e){var b=a(this).find("[data-valmsg-for='"+f(e[0].name)+"']"),d=b.attr("d
ata-valmsg-replace"),g=d?a.parseJSON(d)!==false:null;b.removeClass("field-val
idation-valid").addClass("field-validation-error");c.data("unobtrusiveContain
er",b);if(g){b.empty();c.removeClass("input-validation-error").appendTo(b)}el
se c.hide()}function l(e,d){var c=a(this).find("[data-valmsg-summary=true]"),
b=c.find("ul");if(b&&b.length;&&d.errorList.length;){b.empty();c.addClass("va
lidation-summary-errors").removeClass("valida...
```

Proof of Concept:

Steps to Reproduce:
1. Locate the vulnerable eval() function in the source code
2. Identify user input that reaches the eval() function
3. Craft malicious JavaScript payload
4. Execute payload through the vulnerable input vector
5. Observe code execution in the application context

Impact: Remote code execution, full application compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #3: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.80
File Location	hospitalportal-master/HospitalPortal/Content/js/jquery.validate.unobtrusive.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
** Copyright (C) Microsoft Corporation. All rights reserved. */ >>>
(function(a){var d=a.validator,b,e="unobtrusiveValidation";function
c(a,b,c){a.rules[b]=c;if(a.message)a.messages[b]=a.message}function
j(a){return a.replace(/^\s+|\s+$/g,"").split(/\s*,\s*/g)}function f(a){return
a.replace(/([!"#$%&'()*+,./:;<=>?@\[\\\]^`{|}~])/g,"\\$1")}function
h(a){return a.substr(0,a.lastIndexOf(".")+1)}function
g(a,b){if(a.indexOf("*.")===0)a=a.replace("*.",b);return a}function
m(c,e){var b=a(this).find("[data-valmsg-for='"+f(e[0].name)+"']"),d=b.attr("d
ata-valmsg-replace"),g=d?a.parseJSON(d)!==false:null;b.removeClass("field-val
idation-valid").addClass("field-validation-error");c.data("unobtrusiveContain
er",b);if(g){b.empty();c.removeClass("input-validation-error").appendTo(b)}el
se c.hide()}function l(e,d){var c=a(this).find("[data-valmsg-summary=true]"),
b=c.find("ul");if(b&&b.length;&&d.errorList.length;){b.empty();c.addClass("va
lidation-summary-errors").removeClass("valida...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #4: Code injection via eval

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.90
File Location	hospitalportal-master/HospitalPortal/Content/js/EditTable.js:494
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via eval detected in JavaScript code

Code Evidence:

```
/* If it is string assume it is json. */ if (String == data.constructor) { >>> eval('var json = ' + data); } else { /* Otherwise assume it is a hash already. */
```

Proof of Concept:

Steps to Reproduce:
1. Locate the vulnerable eval() function in the source code
2. Identify user input that reaches the eval() function
3. Craft malicious JavaScript payload
4. Execute payload through the vulnerable input vector
5. Observe code execution in the application context

Impact: Remote code execution, full application compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #5: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.90
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.flot.js:32
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
* produce a color rather than just crashing. */ >>>
(function($){$.color={};$.color.make=function(r,g,b,a){var
o={};o.r=r||0;o.g=g||0;o.b=b||0;o.a=a!=null?a:1;o.add=function(c,d){for(var
i=0;i<c.length;++i)o[c.charAt(i)]+=d;return
o.normalize()};o.scale=function(c,f){for(var
i=0;i<c.length;++i)o[c.charAt(i)]*=f;return o.normalize()};o.toString=functio
n(){if(o.a>=1){return"rgb("+[o.r,o.g,o.b].join(",")+")"}else{return"rgba("+[o
.r,o.g,o.b,o.a].join(",")+")"}};o.normalize=function(){function
clamp(min,value,max){return value<min?min:value>max?max:value}o.r=clamp(0,par
seInt(o.r),255);o.g=clamp(0,parseInt(o.g),255);o.b=clamp(0,parseInt(o.b),255)
;o.a=clamp(0,o.a,1);return o};o.clone=function(){return
$.color.make(o.r,o.b,o.g,o.a)};return
o.normalize()};$.color.extract=function(elem,css){var c;do{c=elem.css(css).to
LowerCase();if(c!=""&&c;!="transparent")break;elem=elem.parent()}while(elem.l
ength&&!$.nodeName(elem.get(0),"body"));if(c=="rgba(0, 0, 0, 0)")c="tra...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #6: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.90
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.flot.js:32
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
* produce a color rather than just crashing. */ >>>
(function($){$.color={};$.color.make=function(r,g,b,a){var
o={};o.r=r||0;o.g=g||0;o.b=b||0;o.a=a!=null?a:1;o.add=function(c,d){for(var
i=0;i<c.length;++i)o[c.charAt(i)]+=d;return
o.normalize()};o.scale=function(c,f){for(var
i=0;i<c.length;++i)o[c.charAt(i)]*=f;return o.normalize()};o.toString=functio
n(){if(o.a>=1){return"rgb("+[o.r,o.g,o.b].join(",")+")"}else{return"rgba("+[o
.r,o.g,o.b,o.a].join(",")+")"}};o.normalize=function(){function
clamp(min,value,max){return value<min?min:value>max?max:value}o.r=clamp(0,par
seInt(o.r),255);o.g=clamp(0,parseInt(o.g),255);o.b=clamp(0,parseInt(o.b),255)
;o.a=clamp(0,o.a,1);return o};o.clone=function(){return
$.color.make(o.r,o.b,o.g,o.a)};return
o.normalize()};$.color.extract=function(elem,css){var c;do{c=elem.css(css).to
LowerCase();if(c!=""&&c;!="transparent")break;elem=elem.parent()}while(elem.l
ength&&!$.nodeName(elem.get(0),"body"));if(c=="rgba(0, 0, 0, 0)")c="tra...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #7: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.90
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.flot.js:32
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
* produce a color rather than just crashing. */ >>>
(function($){$.color={};$.color.make=function(r,g,b,a){var
o={};o.r=r||0;o.g=g||0;o.b=b||0;o.a=a!=null?a:1;o.add=function(c,d){for(var
i=0;i<c.length;++i)o[c.charAt(i)]+=d;return
o.normalize()};o.scale=function(c,f){for(var
i=0;i<c.length;++i)o[c.charAt(i)]*=f;return o.normalize()};o.toString=functio
n(){if(o.a>=1){return"rgb("+[o.r,o.g,o.b].join(",")+")"}else{return"rgba("+[o
.r,o.g,o.b,o.a].join(",")+")"}};o.normalize=function(){function
clamp(min,value,max){return value<min?min:value>max?max:value}o.r=clamp(0,par
seInt(o.r),255);o.g=clamp(0,parseInt(o.g),255);o.b=clamp(0,parseInt(o.b),255)
;o.a=clamp(0,o.a,1);return o};o.clone=function(){return
$.color.make(o.r,o.b,o.g,o.a)};return
o.normalize()};$.color.extract=function(elem,css){var c;do{c=elem.css(css).to
LowerCase();if(c!=""&&c;!="transparent")break;elem=elem.parent()}while(elem.l
ength&&!$.nodeName(elem.get(0),"body"));if(c=="rgba(0, 0, 0, 0)")c="tra...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #8: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.90
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.flot.js:32
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
* produce a color rather than just crashing. */ >>>
(function($){$.color={};$.color.make=function(r,g,b,a){var
o={};o.r=r||0;o.g=g||0;o.b=b||0;o.a=a!=null?a:1;o.add=function(c,d){for(var
i=0;i<c.length;++i)o[c.charAt(i)]+=d;return
o.normalize()};o.scale=function(c,f){for(var
i=0;i<c.length;++i)o[c.charAt(i)]*=f;return o.normalize()};o.toString=functio
n(){if(o.a>=1){return"rgb("+[o.r,o.g,o.b].join(",")+")"}else{return"rgba("+[o
.r,o.g,o.b,o.a].join(",")+")"}};o.normalize=function(){function
clamp(min,value,max){return value<min?min:value>max?max:value}o.r=clamp(0,par
seInt(o.r),255);o.g=clamp(0,parseInt(o.g),255);o.b=clamp(0,parseInt(o.b),255)
;o.a=clamp(0,o.a,1);return o};o.clone=function(){return
$.color.make(o.r,o.b,o.g,o.a)};return
o.normalize()};$.color.extract=function(elem,css){var c;do{c=elem.css(css).to
LowerCase();if(c!=""&&c;!="transparent")break;elem=elem.parent()}while(elem.l
ength&&!$.nodeName(elem.get(0),"body"));if(c=="rgba(0, 0, 0, 0)")c="tra...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #9: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.90
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.flot.js:32
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
* produce a color rather than just crashing. */ >>>
(function($){$.color={};$.color.make=function(r,g,b,a){var
o={};o.r=r||0;o.g=g||0;o.b=b||0;o.a=a!=null?a:1;o.add=function(c,d){for(var
i=0;i<c.length;++i)o[c.charAt(i)]+=d;return
o.normalize()};o.scale=function(c,f){for(var
i=0;i<c.length;++i)o[c.charAt(i)]*=f;return o.normalize()};o.toString=functio
n(){if(o.a>=1){return"rgb("+[o.r,o.g,o.b].join(",")+")"}else{return"rgba("+[o
.r,o.g,o.b,o.a].join(",")+")"}};o.normalize=function(){function
clamp(min,value,max){return value<min?min:value>max?max:value}o.r=clamp(0,par
seInt(o.r),255);o.g=clamp(0,parseInt(o.g),255);o.b=clamp(0,parseInt(o.b),255)
;o.a=clamp(0,o.a,1);return o};o.clone=function(){return
$.color.make(o.r,o.b,o.g,o.a)};return
o.normalize()};$.color.extract=function(elem,css){var c;do{c=elem.css(css).to
LowerCase();if(c!=""&&c;!="transparent")break;elem=elem.parent()}while(elem.l
ength&&!$.nodeName(elem.get(0),"body"));if(c=="rgba(0, 0, 0, 0)")c="tra...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #10: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.90
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.flot.js:32
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
* produce a color rather than just crashing. */ >>>
(function($){$.color={};$.color.make=function(r,g,b,a){var
o={};o.r=r||0;o.g=g||0;o.b=b||0;o.a=a!=null?a:1;o.add=function(c,d){for(var
i=0;i<c.length;++i)o[c.charAt(i)]+=d;return
o.normalize()};o.scale=function(c,f){for(var
i=0;i<c.length;++i)o[c.charAt(i)]*=f;return o.normalize()};o.toString=functio
n(){if(o.a>=1){return"rgb("+[o.r,o.g,o.b].join(",")+")"}else{return"rgba("+[o
.r,o.g,o.b,o.a].join(",")+")"}};o.normalize=function(){function
clamp(min,value,max){return value<min?min:value>max?max:value}o.r=clamp(0,par
seInt(o.r),255);o.g=clamp(0,parseInt(o.g),255);o.b=clamp(0,parseInt(o.b),255)
;o.a=clamp(0,o.a,1);return o};o.clone=function(){return
$.color.make(o.r,o.b,o.g,o.a)};return
o.normalize()};$.color.extract=function(elem,css){var c;do{c=elem.css(css).to
LowerCase();if(c!=""&&c;!="transparent")break;elem=elem.parent()}while(elem.l
ength&&!$.nodeName(elem.get(0),"body"));if(c=="rgba(0, 0, 0, 0)")c="tra...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #11: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.sparkline
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
** Licensed under the New BSD License - see above site for details */ >>>
(function(a,b,c){(function(a){typeof define=="function"&&define.amd;?define([
"jquery"],a):jQuery&&!jQuery.fn.sparkline&&a;(jQuery)})(function(d){"use
strict";var e={},f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,A,B,C,D,E,F,G,H,I,
J,K,L=0;f=function(){return{common:{type:"line",lineColor:"#00f",fillColor:"#
cdf",defaultPixelsPerValue:3,width:"auto",height:"auto",composite:!1,tagValue
sAttribute:"values",tagOptionsPrefix:"spark",enableTagOptions:!1,enableHighli
ght:!0,highlightLighten:1.4,tooltipSkipNull:!0,tooltipPrefix:"",tooltipSuffix
:"",disableHiddenCheck:!1,numberFormatter:!1,numberDigitGroupCount:3,numberDi
gitGroupSep:",",numberDecimalMark:".",disableTooltips:!1,disableInteraction:!
1},line:{spotColor:"#f80",highlightSpotColor:"#5f5",highlightLineColor:"#f22"
,spotRadius:1.5,minSpotColor:"#f80",maxSpotColor:"#f80",lineWidth:1,normalRan
geMin:c,normalRangeMax:c,normalRangeColor:"#ccc",drawNormalOnTop:!1...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #12: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.sparkline
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
** Licensed under the New BSD License - see above site for details */ >>>
(function(a,b,c){(function(a){typeof define=="function"&&define.amd;?define([
"jquery"],a):jQuery&&!jQuery.fn.sparkline&&a;(jQuery)})(function(d){"use
strict";var e={},f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,A,B,C,D,E,F,G,H,I,
J,K,L=0;f=function(){return{common:{type:"line",lineColor:"#00f",fillColor:"#
cdf",defaultPixelsPerValue:3,width:"auto",height:"auto",composite:!1,tagValue
sAttribute:"values",tagOptionsPrefix:"spark",enableTagOptions:!1,enableHighli
ght:!0,highlightLighten:1.4,tooltipSkipNull:!0,tooltipPrefix:"",tooltipSuffix
:"",disableHiddenCheck:!1,numberFormatter:!1,numberDigitGroupCount:3,numberDi
gitGroupSep:",",numberDecimalMark:".",disableTooltips:!1,disableInteraction:!
1},line:{spotColor:"#f80",highlightSpotColor:"#5f5",highlightLineColor:"#f22"
,spotRadius:1.5,minSpotColor:"#f80",maxSpotColor:"#f80",lineWidth:1,normalRan
geMin:c,normalRangeMax:c,normalRangeColor:"#ccc",drawNormalOnTop:!1...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #13: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.sparkline
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
** Licensed under the New BSD License - see above site for details */ >>>
(function(a,b,c){(function(a){typeof define=="function"&&define.amd;?define([
"jquery"],a):jQuery&&!jQuery.fn.sparkline&&a;(jQuery)})(function(d){"use
strict";var e={},f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,A,B,C,D,E,F,G,H,I,
J,K,L=0;f=function(){return{common:{type:"line",lineColor:"#00f",fillColor:"#
cdf",defaultPixelsPerValue:3,width:"auto",height:"auto",composite:!1,tagValue
sAttribute:"values",tagOptionsPrefix:"spark",enableTagOptions:!1,enableHighli
ght:!0,highlightLighten:1.4,tooltipSkipNull:!0,tooltipPrefix:"",tooltipSuffix
:"",disableHiddenCheck:!1,numberFormatter:!1,numberDigitGroupCount:3,numberDi
gitGroupSep:",",numberDecimalMark:".",disableTooltips:!1,disableInteraction:!
1},line:{spotColor:"#f80",highlightSpotColor:"#5f5",highlightLineColor:"#f22"
,spotRadius:1.5,minSpotColor:"#f80",maxSpotColor:"#f80",lineWidth:1,normalRan
geMin:c,normalRangeMax:c,normalRangeColor:"#ccc",drawNormalOnTop:!1...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #14: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.sparkline
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
(function(a,b,c){(function(a){typeof define=="function"&&define.amd;?define([
"jquery"],a):jQuery&&!jQuery.fn.sparkline&&a;(jQuery)})(function(d){"use
strict";var e={},f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,A,B,C,D,E,F,G,H,I,
J,K,L=0;f=function(){return{common:{type:"line",lineColor:"#00f",fillColor:"#
cdf",defaultPixelsPerValue:3,width:"auto",height:"auto",composite:!1,tagValue
sAttribute:"values",tagOptionsPrefix:"spark",enableTagOptions:!1,enableHighli
ght:!0,highlightLighten:1.4,tooltipSkipNull:!0,tooltipPrefix:"",tooltipSuffix
:"",disableHiddenCheck:!1,numberFormatter:!1,numberDigitGroupCount:3,numberDi
gitGroupSep:",",numberDecimalMark:".",disableTooltips:!1,disableInteraction:!
1},line:{spotColor:"#f80",highlightSpotColor:"#5f5",highlightLineColor:"#f22"
,spotRadius:1.5,minSpotColor:"#f80",maxSpotColor:"#f80",lineWidth:1,normalRan
geMin:c,normalRangeMax:c,normalRangeColor:"#ccc",drawNormalOnTop:!1,chartRang
eMin:c,chartRangeMax:c,chartRangeMinX:c,chartRangeMaxX:c,tooltipFormat:ne...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #15: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.sparkline
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
(function(a,b,c){(function(a){typeof define=="function"&&define.amd;?define([
"jquery"],a):jQuery&&!jQuery.fn.sparkline&&a;(jQuery)})(function(d){"use
strict";var e={},f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,A,B,C,D,E,F,G,H,I,
J,K,L=0;f=function(){return{common:{type:"line",lineColor:"#00f",fillColor:"#
cdf",defaultPixelsPerValue:3,width:"auto",height:"auto",composite:!1,tagValue
sAttribute:"values",tagOptionsPrefix:"spark",enableTagOptions:!1,enableHighli
ght:!0,highlightLighten:1.4,tooltipSkipNull:!0,tooltipPrefix:"",tooltipSuffix
:"",disableHiddenCheck:!1,numberFormatter:!1,numberDigitGroupCount:3,numberDi
gitGroupSep:",",numberDecimalMark:".",disableTooltips:!1,disableInteraction:!
1},line:{spotColor:"#f80",highlightSpotColor:"#5f5",highlightLineColor:"#f22"
,spotRadius:1.5,minSpotColor:"#f80",maxSpotColor:"#f80",lineWidth:1,normalRan
geMin:c,normalRangeMax:c,normalRangeColor:"#ccc",drawNormalOnTop:!1,chartRang
eMin:c,chartRangeMax:c,chartRangeMinX:c,chartRangeMaxX:c,tooltipFormat:ne...
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #16: Code injection via eval

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via eval detected in JavaScript code

Code Evidence:

```
*/ (function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.documen
t,s=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.pus
h,d=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e
,n){return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.sour
ce,w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)
$/,k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=funct
ion(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("l
oad",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(
e,t,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0
)&&">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]
&&t;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){i
f(t=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.owne
rDocument||t:o,!0)),C.test(r[1])&&x.isPla...;
```

Proof of Concept:

Steps to Reproduce:
1. Locate the vulnerable eval() function in the source code
2. Identify user input that reaches the eval() function
3. Craft malicious JavaScript payload
4. Execute payload through the vulnerable input vector
5. Observe code execution in the application context

Impact: Remote code execution, full application compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #17: Code injection via eval

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:6
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via eval detected in JavaScript code

Code Evidence:

```
(function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.document,s
=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.push,d
=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e,n)
{return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,
w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)$/,
k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=function
(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("load
",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(e,t
,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0)&&
">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]&&t
;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){if(t
=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.ownerDo
cument||t:o,!0)),C.test(r[1])&&x.isPlainObjec...;
```

Proof of Concept:

Steps to Reproduce:
1. Locate the vulnerable eval() function in the source code
2. Identify user input that reaches the eval() function
3. Craft malicious JavaScript payload
4. Execute payload through the vulnerable input vector
5. Observe code execution in the application context

Impact: Remote code execution, full application compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #18: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:4
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
//@ sourceMappingURL=jquery.min.map */ >>> (function(e,undefined){var
t,n,r=typeof undefined,i=e.location,o=e.document,s=o.documentElement,a=e.jQue
ry,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.push,d=c.slice,g=c.indexOf,m=l.to
String,y=l.hasOwnProperty,v=f.trim,x=function(e,n){return new x.fn.init(e,n,t
)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,w=/\S+/g,T=/^(?:(<[\w\W]+>)
[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)$/,k=/^-ms-/,N=/-([\da-z])/gi,
E=function(e,t){return t.toUpperCase()},S=function(){o.removeEventListener("D
OMContentLoaded",S,!1),e.removeEventListener("load",S,!1),x.ready()};x.fn=x.p
rototype={jquery:f,constructor:x,init:function(e,t,n){var r,i;if(!e)return
this;if("string"==typeof e){if(r="<"===e.charAt(0)&&">"===e.charAt(e.length-1
)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]&&t;)return!t||t.jquery?(t||
n).find(e):this.constructor(t).find(e);if(r[1]){if(t=t instanceof
x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeTyp...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #19: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:4
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
//@ sourceMappingURL=jquery.min.map */ >>> (function(e,undefined){var
t,n,r=typeof undefined,i=e.location,o=e.document,s=o.documentElement,a=e.jQue
ry,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.push,d=c.slice,g=c.indexOf,m=l.to
String,y=l.hasOwnProperty,v=f.trim,x=function(e,n){return new x.fn.init(e,n,t
)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,w=/\S+/g,T=/^(?:(<[\w\W]+>)
[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)$/,k=/^-ms-/,N=/-([\da-z])/gi,
E=function(e,t){return t.toUpperCase()},S=function(){o.removeEventListener("D
OMContentLoaded",S,!1),e.removeEventListener("load",S,!1),x.ready()};x.fn=x.p
rototype={jquery:f,constructor:x,init:function(e,t,n){var r,i;if(!e)return
this;if("string"==typeof e){if(r="<"===e.charAt(0)&&">"===e.charAt(e.length-1
)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]&&t;)return!t||t.jquery?(t||
n).find(e):this.constructor(t).find(e);if(r[1]){if(t=t instanceof
x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeTyp...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #20: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:4
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
//@ sourceMappingURL=jquery.min.map */ >>> (function(e,undefined){var
t,n,r=typeof undefined,i=e.location,o=e.document,s=o.documentElement,a=e.jQue
ry,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.push,d=c.slice,g=c.indexOf,m=l.to
String,y=l.hasOwnProperty,v=f.trim,x=function(e,n){return new x.fn.init(e,n,t
)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,w=/\S+/g,T=/^(?:(<[\w\W]+>)
[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)$/,k=/^-ms-/,N=/-([\da-z])/gi,
E=function(e,t){return t.toUpperCase()},S=function(){o.removeEventListener("D
OMContentLoaded",S,!1),e.removeEventListener("load",S,!1),x.ready()};x.fn=x.p
rototype={jquery:f,constructor:x,init:function(e,t,n){var r,i;if(!e)return
this;if("string"==typeof e){if(r="<"===e.charAt(0)&&">"===e.charAt(e.length-1
)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]&&t;)return!t||t.jquery?(t||
n).find(e):this.constructor(t).find(e);if(r[1]){if(t=t instanceof
x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeTyp...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #21: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:4
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
//@ sourceMappingURL=jquery.min.map */ >>> (function(e,undefined){var
t,n,r=typeof undefined,i=e.location,o=e.document,s=o.documentElement,a=e.jQue
ry,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.push,d=c.slice,g=c.indexOf,m=l.to
String,y=l.hasOwnProperty,v=f.trim,x=function(e,n){return new x.fn.init(e,n,t
)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,w=/\S+/g,T=/^(?:(<[\w\W]+>)
[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)$/,k=/^-ms-/,N=/-([\da-z])/gi,
E=function(e,t){return t.toUpperCase()},S=function(){o.removeEventListener("D
OMContentLoaded",S,!1),e.removeEventListener("load",S,!1),x.ready()};x.fn=x.p
rototype={jquery:f,constructor:x,init:function(e,t,n){var r,i;if(!e)return
this;if("string"==typeof e){if(r="<"===e.charAt(0)&&">"===e.charAt(e.length-1
)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]&&t;)return!t||t.jquery?(t||
n).find(e):this.constructor(t).find(e);if(r[1]){if(t=t instanceof
x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeTyp...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #22: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:4
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
//@ sourceMappingURL=jquery.min.map */ >>> (function(e,undefined){var
t,n,r=typeof undefined,i=e.location,o=e.document,s=o.documentElement,a=e.jQue
ry,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.push,d=c.slice,g=c.indexOf,m=l.to
String,y=l.hasOwnProperty,v=f.trim,x=function(e,n){return new x.fn.init(e,n,t
)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,w=/\S+/g,T=/^(?:(<[\w\W]+>)
[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)$/,k=/^-ms-/,N=/-([\da-z])/gi,
E=function(e,t){return t.toUpperCase()},S=function(){o.removeEventListener("D
OMContentLoaded",S,!1),e.removeEventListener("load",S,!1),x.ready()};x.fn=x.p
rototype={jquery:f,constructor:x,init:function(e,t,n){var r,i;if(!e)return
this;if("string"==typeof e){if(r="<"===e.charAt(0)&&">"===e.charAt(e.length-1
)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]&&t;)return!t||t.jquery?(t||
n).find(e):this.constructor(t).find(e);if(r[1]){if(t=t instanceof
x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeTyp...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #23: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:4
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
//@ sourceMappingURL=jquery.min.map */ >>> (function(e,undefined){var
t,n,r=typeof undefined,i=e.location,o=e.document,s=o.documentElement,a=e.jQue
ry,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.push,d=c.slice,g=c.indexOf,m=l.to
String,y=l.hasOwnProperty,v=f.trim,x=function(e,n){return new x.fn.init(e,n,t
)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,w=/\S+/g,T=/^(?:(<[\w\W]+>)
[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)$/,k=/^-ms-/,N=/-([\da-z])/gi,
E=function(e,t){return t.toUpperCase()},S=function(){o.removeEventListener("D
OMContentLoaded",S,!1),e.removeEventListener("load",S,!1),x.ready()};x.fn=x.p
rototype={jquery:f,constructor:x,init:function(e,t,n){var r,i;if(!e)return
this;if("string"==typeof e){if(r="<"===e.charAt(0)&&">"===e.charAt(e.length-1
)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]&&t;)return!t||t.jquery?(t||
n).find(e):this.constructor(t).find(e);if(r[1]){if(t=t instanceof
x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeTyp...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #24: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
*/ (function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.documen
t,s=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.pus
h,d=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e
,n){return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.sour
ce,w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)
$/,k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=funct
ion(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("l
oad",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(
e,t,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0
)&&">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]
&&t;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){i
f(t=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.owne
rDocument||t:o,!0)),C.test(r[1])&&x.isPla...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #25: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
*/ (function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.documen
t,s=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.pus
h,d=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e
,n){return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.sour
ce,w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)
$/,k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=funct
ion(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("l
oad",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(
e,t,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0
)&&">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]
&&t;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){i
f(t=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.owne
rDocument||t:o,!0)),C.test(r[1])&&x.isPla...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #26: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
*/ (function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.documen
t,s=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.pus
h,d=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e
,n){return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.sour
ce,w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)
$/,k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=funct
ion(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("l
oad",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(
e,t,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0
)&&">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]
&&t;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){i
f(t=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.owne
rDocument||t:o,!0)),C.test(r[1])&&x.isPla...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #27: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
*/ (function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.documen
t,s=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.pus
h,d=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e
,n){return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.sour
ce,w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)
$/,k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=funct
ion(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("l
oad",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(
e,t,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0
)&&">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]
&&t;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){i
f(t=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.owne
rDocument||t:o,!0)),C.test(r[1])&&x.isPla...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #28: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
*/ (function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.documen
t,s=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.pus
h,d=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e
,n){return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.sour
ce,w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)
$/,k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=funct
ion(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("l
oad",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(
e,t,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0
)&&">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]
&&t;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){i
f(t=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.owne
rDocument||t:o,!0)),C.test(r[1])&&x.isPla...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #29: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
*/ (function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.documen
t,s=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.pus
h,d=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e
,n){return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.sour
ce,w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)
$/,k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=funct
ion(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("l
oad",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(
e,t,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0
)&&">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]
&&t;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){i
f(t=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.owne
rDocument||t:o,!0)),C.test(r[1])&&x.isPla...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #30: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
*/ (function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.documen
t,s=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.pus
h,d=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e
,n){return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.sour
ce,w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)
$/,k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=funct
ion(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("l
oad",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(
e,t,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0
)&&">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]
&&t;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){i
f(t=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.owne
rDocument||t:o,!0)),C.test(r[1])&&x.isPla...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #31: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:5
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
*/ (function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.documen
t,s=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.pus
h,d=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e
,n){return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.sour
ce,w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)
$/,k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=funct
ion(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("l
oad",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(
e,t,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0
)&&">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]
&&t;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){i
f(t=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.owne
rDocument||t:o,!0)),C.test(r[1])&&x.isPla...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #32: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:6
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
(function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.document,s
=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.push,d
=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e,n)
{return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,
w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)$/,
k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=function
(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("load
",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(e,t
,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0)&&
">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]&&t
;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){if(t
=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.ownerDo
cument||t:o,!0)),C.test(r[1])&&x.isPlainObjec...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #33: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:6
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
(function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.document,s
=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.push,d
=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e,n)
{return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,
w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)$/,
k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=function
(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("load
",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(e,t
,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0)&&
">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]&&t
;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){if(t
=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.ownerDo
cument||t:o,!0)),C.test(r[1])&&x.isPlainObjec...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #34: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:6
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
(function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.document,s
=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.push,d
=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e,n)
{return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,
w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)$/,
k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=function
(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("load
",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(e,t
,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0)&&
">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]&&t
;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){if(t
=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.ownerDo
cument||t:o,!0)),C.test(r[1])&&x.isPlainObjec...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #35: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery-2.0.0.min.js:6
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
(function(e,undefined){var t,n,r=typeof undefined,i=e.location,o=e.document,s
=o.documentElement,a=e.jQuery,u=e.$,l={},c=[],f="2.0.0",p=c.concat,h=c.push,d
=c.slice,g=c.indexOf,m=l.toString,y=l.hasOwnProperty,v=f.trim,x=function(e,n)
{return new x.fn.init(e,n,t)},b=/[+-]?(?:\d*\.|)\d+(?:[eE][+-]?\d+|)/.source,
w=/\S+/g,T=/^(?:(<[\w\W]+>)[^>]*|#([\w-]*))$/,C=/^<(\w+)\s*\/?>(?:<\/\1>|)$/,
k=/^-ms-/,N=/-([\da-z])/gi,E=function(e,t){return t.toUpperCase()},S=function
(){o.removeEventListener("DOMContentLoaded",S,!1),e.removeEventListener("load
",S,!1),x.ready()};x.fn=x.prototype={jquery:f,constructor:x,init:function(e,t
,n){var r,i;if(!e)return this;if("string"==typeof e){if(r="<"===e.charAt(0)&&
">"===e.charAt(e.length-1)&&e.length;>=3?[null,e,null]:T.exec(e),!r||!r[1]&&t
;)return!t||t.jquery?(t||n).find(e):this.constructor(t).find(e);if(r[1]){if(t
=t instanceof x?t[0]:t,x.merge(this,x.parseHTML(r[1],t&&t.nodeType;?t.ownerDo
cument||t:o,!0)),C.test(r[1])&&x.isPlainObjec...;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #36: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment.js:2200
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
var i, l, string = config._i, >>> match = extendedIsoRegex.exec(string) ||
basicIsoRegex.exec(string), allowTime, dateFormat, timeFormat, tzFormat;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #37: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment.js:2200
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
var i, l, string = config._i, >>> match = extendedIsoRegex.exec(string) ||
basicIsoRegex.exec(string), allowTime, dateFormat, timeFormat, tzFormat;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #38: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering

CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment.js:2207
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
for (i = 0, l = isoDates.length; i < l; i++) { >>> if
(isoDates[i][1].exec(match[1])) { dateFormat = isoDates[i][0]; allowTime =
isoDates[i][2] !== false;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #39: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment.js:2219
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
if (match[3]) { for (i = 0, l = isoTimes.length; i < l; i++) { >>> if
(isoTimes[i][1].exec(match[3])) { // match[2] should be 'T' or space
timeFormat = (match[2] || ' ') + isoTimes[i][0];
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #40: Code injection via exec

Property	Details
Severity	Critical

STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment.js:2235
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
} if (match[4]) { >>> if (tzRegex.exec(match[4])) { tzFormat = 'Z'; } else {
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #41: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment.js:2325
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
// date and time from ref 2822 format function configFromRFC2822(config) { >>>
var match = rfc2822.exec(preprocessRFC2822(config._i)); if (match) { var
parsedArray = extractFromRFC2822Strings(match[4], match[3], match[2],
match[5], match[6], match[7]);
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #42: Code injection via exec

Property	Details

Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment.js:2346
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
// date from iso format or fallback function configFromString(config) { >>>
var matched = aspNetJsonRegex.exec(config._i); if (matched !== null) {
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #43: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.80
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment.js:3031
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
duration.milliseconds = input; } >>> } else if (!!(match =
aspNetRegex.exec(input))) { sign = (match[1] === '-') ? -1 : 1; duration = {
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #44: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering

CWE ID	CWE-95
Confidence Score	0.80
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment.js:3041
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
ms : toInt(absRound(match[MILLISECOND] * 1000)) * sign // the millisecond
decimal point is included in the match }; >>> } else if (!!(match =
isoRegex.exec(input))) { sign = (match[1] === '-') ? -1 : 1; duration = {
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #45: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:2198
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
var i, l, string = config._i, >>> match = extendedIsoRegex.exec(string) ||
basicIsoRegex.exec(string), allowTime, dateFormat, timeFormat, tzFormat;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #46: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering

CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:2198
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
var i, l, string = config._i, >>> match = extendedIsoRegex.exec(string) ||
basicIsoRegex.exec(string), allowTime, dateFormat, timeFormat, tzFormat;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #47: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:2205
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
for (i = 0, l = isoDates.length; i < l; i++) { >>> if
(isoDates[i][1].exec(match[1])) { dateFormat = isoDates[i][0]; allowTime =
isoDates[i][2] !== false;
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #48: Code injection via exec

Property	Details
Severity	Critical

STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:2217
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
if (match[3]) { for (i = 0, l = isoTimes.length; i < l; i++) { >>> if
(isoTimes[i][1].exec(match[3])) { // match[2] should be 'T' or space
timeFormat = (match[2] || ' ') + isoTimes[i][0];
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #49: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:2233
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
} if (match[4]) { >>> if (tzRegex.exec(match[4])) { tzFormat = 'Z'; } else {
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #50: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering

CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:2323
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
// date and time from ref 2822 format function configFromRFC2822(config) { >>>
var match = rfc2822.exec(preprocessRFC2822(config._i)); if (match) { var
parsedArray = extractFromRFC2822Strings(match[4], match[3], match[2],
match[5], match[6], match[7]);
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #51: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:2344
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
// date from iso format or fallback function configFromString(config) { >>>
var matched = aspNetJsonRegex.exec(config._i); if (matched !== null) {
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #52: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering

CWE ID	CWE-95
Confidence Score	0.80
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:3029
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
duration.milliseconds = input; } >>> } else if (!!(match =
aspNetRegex.exec(input))) { sign = (match[1] === '-') ? -1 : 1; duration = {
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #53: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.80
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:3039
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
ms : toInt(absRound(match[MILLISECOND] * 1000)) * sign // the millisecond
decimal point is included in the match }; >>> } else if (!!(match =
isoRegex.exec(input))) { sign = (match[1] === '-') ? -1 : 1; duration = {
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #54: Code injection via exec

Property	Details
Severity	Critical

STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:6137
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
relativeTime : { future : function (output) { >>> var affix =
/■■■■■$/i.exec(output) ? '■■■' : /■■■$/i.exec(output) ? '■■■' : '■■■';
return output + affix; },
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #55: Code injection via exec

Property	Details
Severity	Critical
STRIDE Category	Tampering
CWE ID	CWE-95
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:6137
Attack Vector	Data modification, code injection, integrity violations

Description:

Code injection via exec detected in JavaScript code

Code Evidence:

```
relativeTime : { future : function (output) { >>> var affix =
/■■■■■$/i.exec(output) ? '■■■' : /■■■$/i.exec(output) ? '■■■' : '■■■';
return output + affix; },
```

Proof of Concept:

Steps to Reproduce:
1. Identify the exec() function call in the source code
2. Trace user input flow to the exec() function
3. Prepare malicious code payload
4. Submit payload through vulnerable input
5. Confirm code execution on the server

Impact: Server-side code execution, system compromise

Remediation:

Avoid dynamic code execution, use safe alternatives

Business Impact:

Data corruption, financial loss, operational disruption, legal liability

Finding #56: Insecure HTTP usage

Property	Details
Severity	Medium

STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/autoNumeric-min.js:10
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* Contributor by Sokolov Yura on 2010-11-07 * >>> * Copyright (c) 2011 Robert J. Knothe http://www.decorplanit.com/plugin/ * * The MIT License (http://www.opensource.org/licenses/mit-license.php)
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #57: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/autoNumeric-min.js:12
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* Copyright (c) 2011 Robert J. Knothe http://www.decorplanit.com/plugin/ * >>>
* The MIT License (http://www.opensource.org/licenses/mit-license.php) * *
Permission is hereby granted, free of charge, to any person
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #58: Insecure HTTP usage

Property	Details
Severity	Medium

STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/EditTable.js:7
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* * Licensed under the MIT license: >>> *
http://www.opensource.org/licenses/mit-license.php * * Project home:
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #59: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/EditTable.js:10
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* * Project home: >>> * http://www.appelsiini.net/projects/jeditable * * Based on editable by Dylan Verheul <dylan_at_dyve.net>:
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #60: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure

CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/EditTable.js:13
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* * Based on editable by Dylan Verheul <dylan_at_dyve.net>: >>> *
http://www.dyve.net/jquery/?editable * */
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #61: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/EditTable.js:152
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
/* figure out how wide and tall we are, saved width and height */ >>> /* are workaround for http://dev.jquery.com/ticket/2190 */ if (0 == $(self).width()) { //$(self).css('visibility', 'hidden');
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #62: Insecure HTTP usage

Property	Details
Severity	Medium

STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/curvedLines.js:2
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
// Creates an array of splines, one for each segment of the original curve.
Algorithm based on the wikipedia articles: // >>> // http://de.wikipedia.org/
w/index.php?title=Kubisch_Hermitescher_Spline&oldid;=130168003 and // http://
en.wikipedia.org/w/index.php?title=Monotone_cubic_interpolation&oldid;=622341
725 and the description of Fritsch-Carlson from // http://math.stackexchange.
com/questions/45218/implementation-of-monotone-cubic-interpolation
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #63: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/curvedLines.js:2
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
// // http://de.wikipedia.org/w/index.php?title=Kubisch_Hermitescher_Spline&oldid;=130168003 and >>> // http://en.wikipedia.org/w/index.php?title=Monotone_cubic_interpolation&oldid;=622341725 and the description of Fritsch-Carlson from // http://math.stackexchange.com/questions/45218/implementation-of-monotone-cubic-interpolation // for a detailed description see https://github.com/MichaelZinsmaier/CurvedLines/docu
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #64: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/curvedLines.js:2
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
// http://de.wikipedia.org/w/index.php?title=Kubisch_Hermitescher_Spline&oldi
d;=130168003 and // http://en.wikipedia.org/w/index.php?title=Monotone_cubic_
interpolation&oldid;=622341725 and the description of Fritsch-Carlson from
>>> // http://math.stackexchange.com/questions/45218/implementation-of-monoto
ne-cubic-interpolation // for a detailed description see
https://github.com/MichaelZinsmaier/CurvedLines/docu function
createHermiteSplines(datapoints, curvedLinesOptions, yPos) {
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #65: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.vmap.min
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* @author JQVMap <me@peterschmalfeldt.com> * @version 1.5.1 >>> * @link
http://jqvmap.com * @license
https://github.com/manifestinteractive/jqvmap/blob/master/LICENSE *
@builddate 2016/05/18
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #66: Insecure HTTP usage

Property	Details

Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.flot.resize
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
/* Inline dependency: * jQuery resize event - v1.1 - 3/14/2010 >>> *
http://benalman.com/projects/jquery-resize-plugin/ * * Copyright (c) 2010
"Cowboy" Ben Alman
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #67: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.80
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.flot.resize
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* Copyright (c) 2010 "Cowboy" Ben Alman * Dual licensed under the MIT and GPL
licenses. >>> * http://benalman.com/about/license/ */
(function($,e,t){"$:nomunge";var i=[],n=$.resize=$.extend($.resize,{}),a,r=fa
lse,s="setTimeout",u="resize",m=u+"-special-event",o="pendingDelay",l="active
Delay",f="throttleWindow";n[o]=200;n[l]=20;n[f]=true;$.event.special[u]={setu
p:function(){if(!n[f]&&this;[s]){return false}var e=$(this);i.push(this);e.da
ta(m,{w:e.width(),h:e.height()});if(i.length===1){a=t;h()}},teardown:function
(){if(!n[f]&&this;[s]){return false}var e=$(this);for(var t=i.length-1;t>=0;t
--){if(i[t]==this){i.splice(t,1);break}}e.removeData(m);if(!i.length){if(r){c
ancelAnimationFrame(a)}else{clearTimeout(a)}a=null}},add:function(e){if(!n[f]
&&this;[s]){return false}var i;function a(e,n,a){var r=$(this),s=r.data(m)||{
};s.w=n!==t?n:r.width();s.h=a!==t?a:r.height();i.apply(this,arguments)}if($.i
sFunction(e)){i=e;return a}else{i=e.handler;e.handler=a}};function h...
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #68: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.easypiech
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* * @license >>> * @author Robert Fleischmann <rendro87@gmail.com>
(http://robert-fleischmann.de) * @version 2.1.6 **/
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #69: Information disclosure in errors

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-209
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.flot.js:135
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Information disclosure in errors detected in JavaScript code

Code Evidence:

```
if (width <= 0 || height <= 0) { >>> throw new Error("Invalid dimensions for plot, width = " + width + ", height = " + height); }
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Implement proper error handling without information disclosure

Business Impact:

Minor data exposure, potential privacy concerns

Finding #70: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure

CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/img/browsers/index_files/jquery.sparkline
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
>>> /* jquery.sparkline 2.1.2 - http://omnipotent.net/jquery.sparkline/ **
Licensed under the New BSD License - see above site for details */
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #71: Information disclosure in errors

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-209
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery.smartWizard.js:466
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Information disclosure in errors detected in JavaScript code

Code Evidence:

```
return rv; } else { >>> $.error('Method ' + method + ' does not exist on
jQuery.smartWizard'); } }
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Implement proper error handling without information disclosure

Business Impact:

Minor data exposure, potential privacy concerns

Finding #72: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure

CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery.smartWizard.js:10
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* * Original URLs: >>> * http://www.techlaboratory.net *
http://tech-laboratory.blogspot.com */
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #73: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/smartwizard/js/jquery.smartWizard.js:11
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* Original URLs: * http://www.techlaboratory.net >>> *
http://tech-laboratory.blogspot.com */
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #74: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure

CWE ID	CWE-319
Confidence Score	0.60
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment.js:3122
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
if (period !== null && !isNaN(+period)) { deprecateSimple(name, 'moment().' +
name + '(period, number) is deprecated. Please use moment().' + name +
'(number, period). ' + >>> 'See
http://momentjs.com/guides/#/warnings/add-inverted-param/ for more info.');
tmp = val; val = period; period = tmp; }
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #75: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.60
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment.js:3914
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
proto.years = deprecate('years accessor is deprecated. Use year instead',
getSetYear); proto.zone = deprecate('moment().zone is deprecated, use
moment().utcOffset instead. http://momentjs.com/guides/#/warnings/zone/',
getSetZone); >>> proto.isDSTShifted = deprecate('isDSTShifted is deprecated.
See http://momentjs.com/guides/#/warnings/dst-shifted/ for more information',
isDaylightSavingTimeShifted); function createUnix (input) {
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #76: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.60
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:3120
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
if (period !== null && !isNaN(+period)) { deprecateSimple(name, 'moment().' +
name + '(period, number) is deprecated. Please use moment().' + name +
'(number, period). ' + >>> 'See
http://momentjs.com/guides/#/warnings/add-inverted-param/ for more info.');
tmp = val; val = period; period = tmp; }
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #77: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.60
File Location	hospitalportal-master/HospitalPortal/Content/js/moment/moment-with-locales.js:3912
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
proto.years = deprecate('years accessor is deprecated. Use year instead',
getSetYear); proto.zone = deprecate('moment().zone is deprecated, use
moment().utcOffset instead. http://momentjs.com/guides/#/warnings/zone/',
getSetZone); >>> proto.isDSTShifted = deprecate('isDSTShifted is deprecated.
See http://momentjs.com/guides/#/warnings/dst-shifted/ for more information',
isDaylightSavingTimeShifted); function createUnix (input) {
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #78: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.70
File Location	hospitalportal-master/HospitalPortal/Content/js/numeral/locales.min.js:5
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* locales : 2.0.6 * license : MIT >>> *
http://adamwdraper.github.com/Numeral-js/ */
!function(a,b){"function"==typeof
define&&define.amd;?define(["numeral"],b):b("object"==typeof module&&module.e
xports;?require("./numeral"):a.numeral)}(this,function(a){!function(){a.regis
ter("locale","bg",{delimiters:{thousands:" ",decimal:","},abbreviations:{thou
sand:"■■■",million:"■■■",billion:"■■■■",trillion:"■■■■"},ordinal:function
(a){return""},currency:{symbol:"■■"}})}(),function(){a.register("locale","chs
",{delimiters:{thousands:",",decimal:"."},abbreviations:{thousand:"■",million
:"■■",billion:"■■",trillion:"■"},ordinal:function(a){return"."},currency:{sy
mbol:"¥"}})}(),function(){a.register("locale","cs",{delimiters:{thousands:" "
,decimal:","},abbreviations:{thousand:"tis.",million:"mil.",billion:"b",trill
ion:"t"},ordinal:function(){return"."},currency:{symbol:"K■"}})}(),function()
{a.register("locale","da-dk",{delimiters:{thousands:".",decimal:","},abbrevia
tions:{thous...
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Finding #79: Information disclosure in errors

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-209
Confidence Score	0.60
File Location	hospitalportal-master/HospitalPortal/Content/js/numeral/numeral.min.js:8
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Information disclosure in errors detected in JavaScript code

Code Evidence:

```
* http://adamwdraper.github.com/Numeral-js/ */ >>>
!function(a,b){"function"==typeof
define&&define.amd;?define(b):"object"==typeof module&&module.exports;?module
.exports=b():a.numeral=b()}(this,function(){function
a(a,b){this._input=a,this._value=b}var b,c,d="2.0.6",e={},f={},g={currentLoca
le:"en",zeroFormat:null,nullFormat:null,defaultFormat:"0,0",scalePercentBy100
:!0},h={currentLocale:g.currentLocale,zeroFormat:g.zeroFormat,nullFormat:g.nu
llFormat,defaultFormat:g.defaultFormat,scalePercentBy100:g.scalePercentBy100}
;return b=function(d){var f,g,i,j;if(b.isNumeral(d))f=d.value();else
if(0===d||"undefined"==typeof d)f=0;else if(null===d||c.isNaN(d))f=null;else
if("string"==typeof d)if(h.zeroFormat&&d;===h.zeroFormat)f=0;else if(h.nullFo
rmat&&d;===h.nullFormat||!d.replace(/[^0-9]+/g,"").length)f=null;else{for(g
in e)if(j="function"==typeof e[g].regexps.unformat?e[g].regexps.unformat():e[
g].regexps.unformat,j&&d.match;(j)){i=e[g].unformat;break}i=i||b._.stringToNu
mber,f=i(d...
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Implement proper error handling without information disclosure

Business Impact:

Minor data exposure, potential privacy concerns

Finding #80: Information disclosure in errors

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-209
Confidence Score	0.60
File Location	hospitalportal-master/HospitalPortal/Content/js/numeral/numeral.min.js:8
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Information disclosure in errors detected in JavaScript code

Code Evidence:

```
* http://adamwdraper.github.com/Numeral-js/ */ >>>
!function(a,b){"function"==typeof
define&&define.amd;?define(b):"object"==typeof module&&module.exports;?module
.exports=b():a.numeral=b()}(this,function(){function
a(a,b){this._input=a,this._value=b}var b,c,d="2.0.6",e={},f={},g={currentLoca
le:"en",zeroFormat:null,nullFormat:null,defaultFormat:"0,0",scalePercentBy100
:!0},h={currentLocale:g.currentLocale,zeroFormat:g.zeroFormat,nullFormat:g.nu
llFormat,defaultFormat:g.defaultFormat,scalePercentBy100:g.scalePercentBy100}
;return b=function(d){var f,g,i,j;if(b.isNumeral(d))f=d.value();else
if(0===d||"undefined"==typeof d)f=0;else if(null===d||c.isNaN(d))f=null;else
if("string"==typeof d)if(h.zeroFormat&&d;===h.zeroFormat)f=0;else if(h.nullFo
rmat&&d;===h.nullFormat||!d.replace(/[^0-9]+/g,"").length)f=null;else{for(g
in e)if(j="function"==typeof e[g].regexps.unformat?e[g].regexps.unformat():e[
g].regexps.unformat,j&&d.match;(j)){i=e[g].unformat;break}i=i||b._.stringToNu
mber,f=i(d...
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Implement proper error handling without information disclosure

Business Impact:

Minor data exposure, potential privacy concerns

Finding #81: Information disclosure in errors

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-209
Confidence Score	0.60
File Location	hospitalportal-master/HospitalPortal/Content/js/numeral/numeral.min.js:8
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Information disclosure in errors detected in JavaScript code

Code Evidence:

```
* http://adamwdraper.github.com/Numeral-js/ */ >>>
!function(a,b){"function"==typeof
define&&define.amd;?define(b):"object"==typeof module&&module.exports;?module
.exports=b():a.numeral=b()}(this,function(){function
a(a,b){this._input=a,this._value=b}var b,c,d="2.0.6",e={},f={},g={currentLoca
le:"en",zeroFormat:null,nullFormat:null,defaultFormat:"0,0",scalePercentBy100
:!0},h={currentLocale:g.currentLocale,zeroFormat:g.zeroFormat,nullFormat:g.nu
llFormat,defaultFormat:g.defaultFormat,scalePercentBy100:g.scalePercentBy100}
;return b=function(d){var f,g,i,j;if(b.isNumeral(d))f=d.value();else
if(0===d||"undefined"==typeof d)f=0;else if(null===d||c.isNaN(d))f=null;else
if("string"==typeof d)if(h.zeroFormat&&d;===h.zeroFormat)f=0;else if(h.nullFo
rmat&&d;===h.nullFormat||!d.replace(/[^0-9]+/g,"").length)f=null;else{for(g
in e)if(j="function"==typeof e[g].regexps.unformat?e[g].regexps.unformat():e[
g].regexps.unformat,j&&d.match;(j)){i=e[g].unformat;break}i=i||b._.stringToNu
mber,f=i(d...
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Implement proper error handling without information disclosure

Business Impact:

Minor data exposure, potential privacy concerns

Finding #82: Insecure HTTP usage

Property	Details
Severity	Medium
STRIDE Category	Information_Disclosure
CWE ID	CWE-319
Confidence Score	0.60
File Location	hospitalportal-master/HospitalPortal/Content/js/numeral/numeral.min.js:6
Attack Vector	Data leakage, privacy violations, sensitive exposure

Description:

Insecure HTTP usage detected in JavaScript code

Code Evidence:

```
* author : Adam Draper * license : MIT >>> *
http://adamwdraper.github.com/Numeral-js/ */
!function(a,b){"function"==typeof
define&&define.amd;?define(b):"object"==typeof module&&module.exports;?module
.exports=b():a.numeral=b()}(this,function(){function
a(a,b){this._input=a,this._value=b}var b,c,d="2.0.6",e={},f={},g={currentLoca
le:"en",zeroFormat:null,nullFormat:null,defaultFormat:"0,0",scalePercentBy100
:!0},h={currentLocale:g.currentLocale,zeroFormat:g.zeroFormat,nullFormat:g.nu
llFormat,defaultFormat:g.defaultFormat,scalePercentBy100:g.scalePercentBy100}
;return b=function(d){var f,g,i,j;if(b.isNumeral(d))f=d.value();else
if(0===d||"undefined"==typeof d)f=0;else if(null===d||c.isNaN(d))f=null;else
if("string"==typeof d)if(h.zeroFormat&&d;===h.zeroFormat)f=0;else if(h.nullFo
rmat&&d;===h.nullFormat||!d.replace(/[^0-9]+/g,"").length)f=null;else{for(g
in e)if(j="function"==typeof e[g].regexps.unformat?e[g].regexps.unformat():e[
g].regexps.unformat,j&&d.match;(j)){i=...
```

Proof of Concept:

Steps to Reproduce:
1. Identify information exposure point
2. Analyze data access controls
3. Attempt unauthorized data access
4. Extract sensitive information
5. Verify information disclosure

Impact: Data breach, privacy violation

Remediation:

Use HTTPS/TLS for all communications

Business Impact:

Minor data exposure, potential privacy concerns

Remediation Summary

Remediation Priority Matrix

■ IMMEDIATE (0-7 days) - Critical Issues: 55
Critical vulnerabilities pose immediate risk to business operations and must be addressed urgently. Recommended actions: Emergency patches, temporary mitigations, incident response preparation.

■ HIGH PRIORITY (1-4 weeks) - High Severity: 0
High-severity issues should be addressed in the next sprint cycle. Recommended actions: Security patches, code reviews, testing validation.

■ MEDIUM PRIORITY (1-3 months) - Medium Severity: 27
Medium-severity issues can be addressed in regular development cycles. Recommended actions: Security improvements, best practice implementation, monitoring enhancement.

Implementation Guidelines:
• Establish security champion within development team
• Implement security testing in CI/CD pipeline
• Conduct regular security code reviews
• Provide security training for developers
• Monitor for new vulnerabilities and threat intelligence
• Regular penetration testing and security assessments