

Report

Perimeter Testing Report

Presented to: Example Company

Prepared by: CommSec

Date: XX/X/20XX

Our business is to safeguard yours.



Contents

- 1 Document Details 3
- 2 Overview 4
 - 2.1 Project Description 4
 - 2.2 Statement of Responsibility 4
 - 2.3 Circulation 5
 - 2.4 Authorised Personnel 5
 - 2.5 Overall Results 5
 - 2.6 Risk Classification 6
- 3 Assignment 8
 - 3.1 Scope 8
- 4 Findings 9
 - 4.1 Overview 9
 - 4.2 Table of Findings 9
- 5 Vulnerability Scan Results 10
 - 5.1 Medium Risk Findings 10
 - 5.1.1 Cross-origin resource sharing: arbitrary origin trusted 10
 - 5.1.2 HSTS Missing From HTTPS Server (RFC 6797) 12
 - 5.1.3 IIS Tilde Enumeration 13
 - 5.1.4 Vulnerable version of the library 'angularjs' found 15
 - 5.1.5 Vulnerable version of the library 'moment.js' found 18
 - 5.2 Low Risk Findings 19
 - 5.2.1 Detailed Error Messages Revealed 19
 - 5.2.2 Software Version Numbers Revealed 23
 - 5.2.3 Strict transport security not enforced 24
 - 5.2.4 Cacheable HTTPS response 26
 - 5.2.5 Cross-origin resource sharing 29
 - 5.2.6 Frameable response (potential Clickjacking) 31
- 6 APPENDICES 34
 - 6.1 About Us 34

Introduction

CommSec is an Irish company specialising in delivering managed IT security services and solutions to our clients in Ireland and abroad. We have a focused, motivated team of people who are experienced and skilled across the IT security landscape. CommSec believe in delivering a high level of service and empower our customers to operate more effectively.

CommSec are 100% focused on delivering cybersecurity services. We believe that this focus and expertise delivers the best possible advice to our clients and that we can provide all of your cyber & information security requirements from a dedicated specialist.

1 Document Details

Information classification	Confidential
Customer	ExampleCompany
Document title	CommSec Perimeter Scan Report
Purpose	To present the findings from Perimeter conducted by CommSec for ExampleCompany
Target audience	IT Management Team ExampleCompany
Author/owner	CommSec
Contact	Darragh Reynolds, CommSec
Status	Draft
Version control	1.0
Version publication date	XX/XX/XXXX

2 Overview

2.1 Project Description

CommSec have been engaged by ExampleCompany for the Perimeter continuous vulnerability scanning service.

The issues outlined in this document are based on several security tests conducted on 1/9/2026.

2.2 Statement of Responsibility

The confidentiality, integrity, and availability of data held by ExampleCompany are of paramount importance. The reputational impact of a security breach can significantly harm ExampleCompany's public image and may lead to regulatory sanctions or other legal consequences.

Perimeter Scan employs advanced automated vulnerability scanning techniques, including network checks, to identify a broad spectrum of potential issues such as web application vulnerabilities, server misconfigurations, outdated software versions, and more.

The scanning engine receives regular updates to incorporate the latest CVEs (Common Vulnerabilities and Exposures) as new vulnerabilities are dynamically released.

Professional security analysts manually validate Perimeter's comprehensive automated results, confirming exploitable technical vulnerabilities present within ExampleCompany's environment at the time of assessment. It should be noted that this assessment considers only known vulnerabilities existing during testing.

2.3 Circulation

As the information held in this document could reasonably be expected to cause serious damage to the information systems infrastructure of ExampleCompany, the circulation of this document and any subsequent documents will be strictly reserved to authorised staff within ExampleCompany and CommSec.

2.4 Authorised Personnel

Personnel authorised to access this document are.

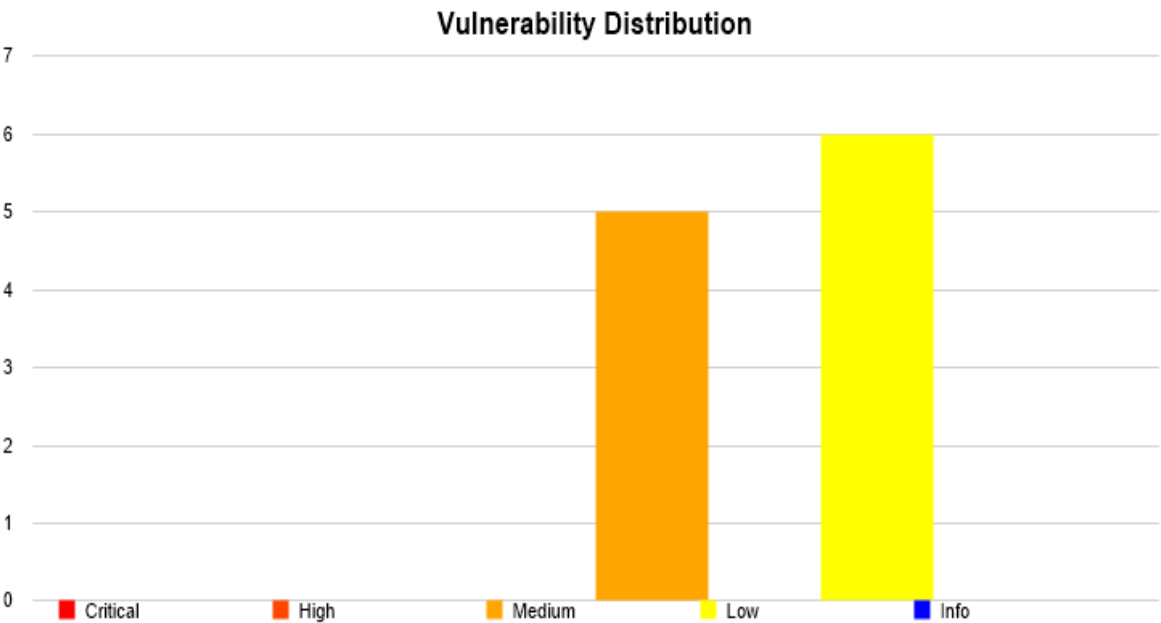
Client Name – ExampleCompany

Additional personnel can access this document upon authorisation by ExampleCompany.

2.5 Overall Results

The identified vulnerabilities were summarised as follows:

- 0 Critical
- 0 High
- 5 Medium
- 6 Low
- 0 Informational





2.6 Risk Classification

Any vulnerabilities discovered by the various tools employed during the assessment were reviewed by CommSec and classified based on their potential impact. The classification of the risk is as follows.

Level	Characteristics	Impact examples
Critical	• Easy to trigger the vulnerability • No extensive knowledge required • Multiple tools and scripts available which trigger the vulnerability • May be triggered without manual actions • Can be triggered without requiring authentication.	Availability Disrupting the service of one or more important systems which: • impacts multiple users; • this will result in loss of data; • noticeable by the outer world. Confidentiality Information is leaked which: • provides access towards other systems (user account information); • is confidential information of multiple users; • is confidential corporate information which may harm the business Integrity Alteration of business critical or private data.
High	• Easy to exploit • No extensive knowledge required • Multiple tools and scripts available which trigger the vulnerability • May be triggered without manual actions	Availability Disrupting the service of one or more important systems which • impacts a small subset of users; • not noticeable by the outer world. Confidentiality Information is leaked which: • provides an insight in the setup of the environment; • is specific information that may be interpreted by someone with the necessary skills; • is non-private data. Integrity Alteration of non-critical data is possible.
Medium	• Only possible when several requirements are met. • Less likely to occur due to the need of special skills. • There is little or no impact towards the business. • Can only be used in combination with other vulnerabilities. • Cannot be automated.	Availability Can the hosted service be disrupted? Confidentiality Can confidential data be obtained? Integrity Is manipulation of data possible?



Low	• Theoretical vulnerability • Cannot be automated.	Availability This impacts old non-used legacy systems. Confidentiality Can confidential data be obtained? Integrity Is manipulation of data possible?
------------	--	--



3. Assignment

The Perimeter Scan was focused on identifying vulnerabilities which can be used by a malicious attacker to access, disrupt, manipulate and/or affect the availability of sensitive data or resources.

2.7 Scope

The scope of the Perimeter Scan was limited to the details given below.

System Name / IP Addresses	https://exampleurl.ie, exampleurl.ie:443
Description	ExampleCompany
With credentials	Yes
Test Dates	17/9/2026



3 Findings

3.1 Overview

During the Perimeter Scan the following vulnerabilities were identified. The table below gives an overview of the vulnerabilities and the risk classification.

	Critical	High	Medium	Low	Informational
External Assessment	0	0	5	6	0

3.2 Table of Findings

Reference	Issue	Risk	State
5.1.1	Cross-origin resource sharing: arbitrary origin trusted	Medium	Open
5.1.2	HSTS Missing From HTTPS Server (RFC 6797)	Medium	Open
5.1.3	IIS Tilde Enumeration	Medium	Open
5.1.4	Vulnerable version of the library 'angularjs' found	Medium	Open
5.1.5	Vulnerable version of the library 'moment.js' found	Medium	Open
5.2.1	Detailed Error Messages Revealed	Low	Open
5.2.2	Software Version Numbers Revealed	Low	Open
5.2.3	Strict transport security not enforced	Low	Open
5.2.4	Cacheable HTTPS response	Low	Open
5.2.5	Cross-origin resource sharing	Low	Open
5.2.6	Frameable response (potential Clickjacking)	Low	Open

4 Vulnerability Scan Results

4.1 Medium Risk Findings

4.1.1 Cross-origin resource sharing: arbitrary origin trusted

Status	Open
Risk Rating	Medium
Affected Host	https://exampleurl.ie

Description

Issue Detail:

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request that allows access from any domain.

The application allowed access from the requested origin

https://fklbjhvmpyzi.com

Since the Vary: Origin header was not present in the response, reverse proxies and intermediate servers may cache it. This may enable an attacker to carry out cache poisoning attacks.

Background:

An HTML5 cross-origin resource sharing (CORS) policy controls whether and how content running on other domains can perform two-way interaction with the domain that publishes the policy. The policy is fine-grained and can apply access controls per-request based on the URL and other features of the request.

Trusting arbitrary origins effectively disables the same-origin policy, allowing two-way interaction by third-party web sites. Unless the response consists only of unprotected public content, this policy is likely to present a security risk.

If the site specifies the header Access-Control-Allow-Credentials: true, third-party sites may be able to carry out privileged actions and retrieve sensitive information. Even if it does not, attackers may be able to bypass any IP-based access controls by proxying through users' browsers.

References:

- Web Security Academy: Cross-origin resource sharing (CORS)
- Exploiting CORS Misconfigurations

Remediation

Rather than using a wildcard or programmatically verifying supplied origins, use a whitelist of trusted domains.

Evidence

HTTP Request:

```
GET /api/labels/short HTTP/2
Host: exampleurl.ie
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Origin: https://fk1bjhvmpyzi.com
```

HTTP Response:

```
HTTP/2 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/10.0
Access-Control-Allow-Origin: https://fk1bjhvmpyzi.com
Access-Control-Allow-Credentials: true

{""}

```

References

[Web Security Academy: Cross-origin resource sharing \(CORS\)](#)

[Exploiting CORS Misconfigurations](#)



4.1.2 HSTS Missing From HTTPS Server (RFC 6797)

Status	Open
Risk Rating	Medium
CVSS Score	6.5
Affected Host	exampleurl.ie:443

Description

The remote web server is not enforcing HSTS, as defined by RFC 6797. HSTS is an optional response header that can be configured on the server to instruct the browser to only communicate via HTTPS. The lack of HSTS allows downgrade attacks, SSL-stripping man-in-the-middle attacks and weakens cookie-hijacking protections.

Impact

The remote web server is not enforcing HSTS, as defined by RFC 6797.

Remediation

Configure the remote web server to use HSTS.

Evidence

```
HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Tue, 20 Sep 2022 16:41:26 GMT
Accept-Ranges: bytes
ETag: "cfb3d3d3fcdd81:0"
Date: Fri, 28 Aug 2026 18:05:34 GMT
Content-Length: 46810
```

The remote HTTPS server does not send the HTTP "Strict-Transport-Security" header.

References

<https://tools.ietf.org/html/rfc6797>



4.1.3 IIS Tilde Enumeration

Status	Open
Risk Rating	Medium
Affected Host	https://exampleurl.ie

Description

Microsoft Internet Information Server (IIS) suffers from a vulnerability which allows the detection of short names of files and directories which have an equivalent in the 8.3 version of the file naming scheme. By crafting specific requests containing the tilde '~' character, an attacker could leverage this vulnerability to find files or directories that are normally not visible and gain access to sensitive information.

Given the underlying filesystem calls generated by the remote server, the attacker could also attempt a denial of service on the target application.

Impact

The IIS Tilde Enumeration vulnerability poses significant risks to organisations using Microsoft Internet Information Server (IIS). By exploiting this vulnerability, an attacker can identify hidden files and directories on the server that utilise the 8.3 filename format, providing insights into the server's filesystem structure. This information disclosure can lead to several potential consequences:

- 1. Information Disclosure:** An attacker could enumerate sensitive files (such as backup files or configuration settings) that are not intended for public access. This could lead to the exposure of credentials, proprietary source code, or personal data, resulting in data breaches or misappropriation of sensitive information.
- 2. Denial of Service Attacks:** Given that the attacker can exploit IIS's filesystem calls through crafted requests, there's a risk of causing a denial of service (DoS) on the affected application. By issuing repeated requests or requesting the enumeration of numerous short names, the server may experience high load, leading to performance degradation or even crashes.
- 3. Exploitation of Further Vulnerabilities:** Knowledge of previously concealed file names and paths may assist attackers in crafting more sophisticated attacks, including targeted exploitation of vulnerabilities associated with web applications or services running on the affected IIS instance.
- 4. Compliance and Legal Risks:** If sensitive information is disclosed due to exploitation of this vulnerability, organisations may face compliance violations, leading to potential legal ramifications and financial penalties, especially under data protection regulations such as GDPR or CCPA.
- 5. Reputation Damage:** The loss of sensitive data can lead to significant reputational harm as clients and customers may lose trust in an organisation's ability to protect their information.

Remediation

To remediate the IIS Tilde Enumeration vulnerability, organisations should undertake the following steps:



1. *Disable 8.3 Filename Generation*: On the Windows filesystem, disable the creation of 8.3 filenames where possible. This can be done using the following PowerShell command:

Powershell fsutil behaviour set disable8dot3 1

2. *Apply Security Patches*: Always ensure that IIS is updated to the latest version, as updates may include security patches addressing known vulnerabilities. Regularly monitor Microsoft's security bulletins for updates relevant to IIS.

3. *Web Server Configuration*:

- Review and tighten access controls on directories, ensuring that sensitive files are not accessible via the web server.

- Implement rules to block requests that contain '~' character or other potentially malicious patterns in the URL, using URL filtering through the web server or application firewall (WAF).

- Utilise request filtering in IIS by adding rules that block enumeration attempts.

4. *Logging and Monitoring*: Enhance logging to capture attempts at file enumeration and implement alerting mechanisms to detect unusual activities. This can help identify potential exploitation attempts in real-time.

5. *Security Audits*: Conduct regular security assessments and vulnerability scans on your IIS web server to identify and mitigate other potential vulnerabilities, ensuring comprehensive coverage of security hygiene.

6. *Educate Development Teams*: Train web developers on secure coding practices to ensure that applications do not expose sensitive information through poor configurations or lack of validation.

7. *Backup Sensitive Data*: Regular backups should be taken for critical data and configurations. Store these backups in secure locations to prevent exposure.

References:

- Microsoft IIS Documentation: [IIS Security](<https://docs.microsoft.com/en-us/iis/get-started/introduction-to-iis/iis-security>)

- NIST National Vulnerability Database: [Vulnerability Details](<https://nvd.nist.gov/>)

- OWASP: [Security Misconfiguration](https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html)

Following these steps will significantly reduce the risk associated with the IIS Tilde Enumeration vulnerability and enhance the overall security posture of the web server environment.

Evidence

HTTP Request:

```
OPTIONS /%2A%7E1%2A%2F%7E1%2F.rem HTTP/2
Host: exampleurl.ie
Cache-Control: max-age=0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/146.0.0.0 Safari/537.36
```

HTTP Response:

HTTP/2 404 Not Found
Server: Microsoft-IIS/10.0
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Date: Mon, 31 Aug 2026 10:01:40 GMT

4.1.4 Vulnerable version of the library 'angularjs' found

Status	Open
Risk Rating	Medium
Affected Host	https://exampleurl.ie

Description

Issue Detail:

The library angularjs version 1.5.11 has known security issues.

Affected versions

The vulnerability is affecting all versions prior 1.6.0 (between * and 1.6.0)

The vulnerability might be affecting a feature of the library that the website is not using. If the vulnerable feature is not used, this alert can be considered false positive.

Impact

The presence of the vulnerable AngularJS library version 1.5.11 increases the risk of various security threats to applications that rely on it. AngularJS versions before 1.6.0 have known vulnerabilities, including potential Cross-Site Scripting (XSS) attacks, which could lead to unauthorised access to sensitive user data, session hijacking, and the execution of malicious scripts within the context of the user’s session. The exploitation of these vulnerabilities can compromise user accounts, leading to data theft, data corruption, or the potential for the injection of malware into the systems accessing the affected applications.

Additionally, the use of an outdated library may put organisations at risk of compliance violations, especially if personal or sensitive data is being processed under regulations that require up-to-date security measures (e.g., GDPR, HIPAA). Failing to address this vulnerability could result in not only data breaches but also potential penalties from regulatory bodies, loss of customer trust, and reputational damage.

Affected systems include any web application or website utilising AngularJS version 1.5.11 or any version prior to 1.6.0; this encompasses both frontend and backend frameworks that might leverage AngularJS for data binding, templating, or other features. Furthermore, third-party integrations or libraries that depend on this specific version can also inherit these vulnerabilities.

Remediation



To effectively remediate the identified vulnerability associated with AngularJS version 1.5.11, follow these comprehensive steps:

1. Upgrade AngularJS:

- Immediately upgrade AngularJS to the latest stable version (at least 1.6.0 or later, with preference for 1.8.x as the latest version) to eliminate known vulnerabilities.
- Test the application thoroughly after upgrading to ensure that no functionalities break due to the upgrade.

2. Conduct Security Assessments:

- After the upgrade, conduct a comprehensive security assessment such as code reviews and penetration testing to identify any other potential vulnerabilities in the application.

3. Monitor Dependencies:

- Implement a regular dependency management process. Identify and audit third-party libraries; consider using tools like npm audit, Snyk, or OWASP Dependency-Check to monitor for known vulnerabilities.

4. Utilise Security Best Practices:

- Follow best practices as outlined by the OWASP Top Ten and AngularJS security guidelines. This includes input validation, output sanitisation, and strict content security policies (CSP) to mitigate potential attacks.
- Ensure proper configuration of CORS (Cross-Origin Resource Sharing) and authentication mechanisms (such as OAuth2) if applicable.

5. Educate Development Teams:

- Conduct training sessions for developers on secure coding practices and up-to-date security measures applicable to popular libraries and frameworks like AngularJS.

6. Implement Application Monitoring:

- Deploy Application Performance Management (APM) and logging tools (like Sentry, Loggly, or ELK stack) to monitor application behaviour for unusual patterns that may indicate a security breach.

By following these steps, you will significantly reduce the risk of exploitation from the known vulnerabilities in AngularJS and enhance your application's overall security posture.

For additional information on best practices and guidelines, refer to the official AngularJS documentation (<https://docs.angularjs.org/guide/security>) and the OWASP Foundation [Top Ten Project](<https://owasp.org/www-project-top-ten/>).

Evidence

HTTP Request:

```
GET /js/vendor/angular-animate/angular-animate.min.js HTTP/2
Host: exampleurl.ie
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/146.0.0.0 Safari/537.36
```



HTTP Response:

```
HTTP/2 200 OK
Content-Type: application/javascript

/*
  AngularJS v1.5.11
  (c) 2010-2017 Google, Inc. http://angularjs.org
  License: MIT
*/
```



4.1.5 Vulnerable version of the library 'moment.js' found

Status	Open
Risk Rating	Medium
Affected Host	https://exampleurl.ie:443

Description

Issue Detail:

The library moment.js version 2.13.0 has known security issues.

For more information, visit these websites:

- <https://security.snyk.io/vuln/npm:moment:20161019>

Affected versions

The vulnerability is affecting all versions prior 2.15.2 (between * and 2.15.2)

The vulnerability might be affecting a feature of the library that the website is not using. If the vulnerable feature is not used, this alert can be considered false positive.

Evidence

HTTP Request:

```
GET /js/vendor/moment/min/moment.min.js HTTP/2
Host: exampleurl.ie
```

HTTP Response:

```
HTTP/2 200 OK
Content-Type: application/javascript
Date: Sun, 15 Feb 2026 10:28:42 GMT

/*! moment.js
/*! version : 2.13.0
/*! authors : Tim Wood, Iskren Chernev, Moment.js contributors
/*! license : MIT
/*! momentjs.com
```



4.2 Low Risk Findings

4.2.1 Detailed Error Messages Revealed

Status	Open
Risk Rating	Low
Affected Host	https://exampleurl.ie:443

Description

The application displays detailed error messages when unhandled ASP.Net exceptions occur.

Detailed technical error messages can allow an adversary to gain information about the application and database that could be used to conduct further attacks. The following expressions were matched in the HTTP response:

- `([A-Za-z]{1,32}\.)+[A-Za-z]{0,32}\((([A-Za-z0-9]+\s+[A-Za-z0-9]+[, \s]*)*\)\s+\{1\}d+`
- `[A-Za-z\.]+\((([A-Za-z0-9, ]+)?\)\s+[0-9]+`
- `\bat ([a-zA-Z0-9_]*\.)*([a-zA-Z0-9_]+)([a-zA-Z0-9, \[\]\&\;]*)\)`
- `System.([A-Za-z]{1,32}\.)*[A-Za-z]{0,32}Exception:`

Impact

The disclosure of detailed error messages in an application can severely compromise the security posture of the system by exposing sensitive information that an attacker could exploit. When an application, particularly one developed in ASP.Net, reveals technical stack traces or debug information, it provides attackers with insights into the application's architecture, database schema, and underlying technology.

- *Information Leakage:* Attackers can gain knowledge about the internal workings of the application, including functions, variables, and database queries that could be used as a step in a larger attack strategy.

- *SQL Injection and Other Attacks:* With knowledge of the database structure, attackers might craft targeted SQL injection attacks or other exploits that leverage the exposed information.

- *Denial of Service (DoS):* Attackers could deliberately trigger exceptions to glean more information or disrupt service, ultimately affecting availability.

- *Reputation Damage:* A successful attack due to poor error handling can lead to data breaches or system compromises, negatively impacting user trust and the organisation's reputation.

If attackers successfully exploit these vulnerabilities, they may gain unauthorised access to sensitive data, manipulate database content, or launch further attacks against the application or network. This may lead to compliance violations, financial loss, and extended recovery efforts.



Remediation

To effectively remediate the issue of detailed error messages being revealed, the following actionable steps and best practices should be implemented:

1. *Modify Error Handling Configurations.*

- Update the web.config file of the ASP.Net application to ensure that detailed error messages are not displayed in a production environment. Change the `customErrors` attribute to "On" or "RemoteOnly":

2. *Error Logging.*

- Instead of displaying errors to end users, configure the application to log errors internally. Use logging frameworks like Log4Net, Serilog, or built-in ASP.Net logging capabilities to log full error details. Ensure that logs are stored securely and have appropriate permissions.

3. *User-friendly Error Pages.*

- Create user-friendly error pages that inform users that an error has occurred without revealing any technical details. These pages should guide users on how to proceed without exposing sensitive information.

4. *Audit and Review.*

- Conduct regular audits of the application's error handling practices and review any custom exception handlers for compliance with security best practices. Utilise code review sessions to catch potential vulnerabilities in the error handling logic.

5. *Implement Security Headers.*

- Consider adding security headers like `X-Content-Type-Options: nosniff`, `X-XSS-Protection: 1; mode=block`, and `X-Frame-Options: DENY` to further mitigate the risk of browser-based attacks that may exploit error messages.

6. *Education and Training.*

- Educate developers and team members about secure coding practices, including error handling and the importance of not exposing stack traces in production environments.

References:

- OWASP Secure Coding Practices - Quick Reference Guide:

<https://owasp.org/www/publications/secure-coding-practices-quick-reference-guide>

- Microsoft ASP.NET Security Best Practices: <https://docs.microsoft.com/en-us/aspnet/security/anti-request-forgery>

- NIST SP 800-53 Security and Privacy Controls: <https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final>



Evidence

HTTP Request:

```
GET
/view/%3cblq%20xmlns%3axi%3d%22http%3a/www.w3.org/2001/XInclude%22%3e%3cxi%3ainclude%20href%3d%22http%3a//9z9esytlvy89462jhc32nhifo6uai667uymc91.oastify.com/foo%22/%3e%3c/blq%3e?accountNumber=
HTTP/2
Host: exampleurl.ie
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/144.0.0.0 Safari/537.36
```

HTTP Response:

```
HTTP/2 400 Bad Request
Cache-Control: private
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html>
  <head>
    <title>A potentially dangerous Request.Path value was detected from the client
(&lt;).</title>
    <meta name="viewport" content="width=device-width" />
    <style>
      body {font-family:"Verdana";font-weight:normal;font-size: .7em;color:black;}
      p {font-family:"Verdana";font-weight:normal;color:black;margin-top: -5px}
      b {font-family:"Verdana";font-weight:bold;color:black;margin-top: -5px}
      H1 { font-family:"Verdana";font-weight:normal;font-size:18pt;color:red }
      H2 { font-family:"Verdana";font-weight:normal;font-size:14pt;color:maroon }
      pre {font-family:"Consolas","Lucida Console",Monospace;font-
size:11pt;margin:0;padding:0.5em;line-height:14pt}
      .marker {font-weight: bold; color: black;text-decoration: none;}
      .version {color: gray;}
      .error {margin-bottom: 10px;}
      .expandable { text-decoration:underline; font-weight:bold; color:navy; cursor:hand; }
      @media screen and (max-width: 639px) {
        pre { width: 440px; overflow: auto; white-space: pre-wrap; word-wrap: break-word; }
      }
      @media screen and (max-width: 479px) {
        pre { width: 280px; }
      }
    </style>
  </head>

  <body bgcolor="white">

    <span><H1>Server Error in '/' Application.<hr width=100% size=1 color=silver></H1>

    <h2> <i>A potentially dangerous Request.Path value was detected from the client
(&lt;).</i> </h2></span>
```



 Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

 Exception Details: System.Web.HttpException: A potentially dangerous Request.Path value was detected from the client (<).

Source Error:

<code></code>

An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below.</code>

</td> </tr>

Stack Trace:

<code><pre></code>

[HttpException (0x80004005): A potentially dangerous Request.Path value was detected from the client (<).]

System.Web.HttpRequest.ValidateInputIfRequiredByConfig() +11988056
System.Web.PipelineStepManager.ValidateHelper(HttpContext context) +52
</pre></code>

</td> </tr>

<hr width=100% size=1 color=silver>

Version Information: Microsoft .NET Framework Version:4.0.30319; ASP.NET Version:4.7.4144.0

</body>
</html>

```
<!--
[HttpException]: A potentially dangerous Request.Path value was detected from the client (&lt;).
    at System.Web.HttpRequest.ValidateInputIfRequiredByConfig()
    at System.Web.HttpApplication.PipelineStepManager.ValidateHelper(HttpContext context)
--><!--
This error page might contain sensitive information because ASP.NET is configured to show verbose
error messages using &lt;customErrors mode="Off"/&gt;. Consider using &lt;customErrors
mode="On"/&gt; or &lt;customErrors mode="RemoteOnly"/&gt; in production environments.-->
```

4.2.2 Software Version Numbers Revealed

Status	Open
Risk Rating	Low
Affected Host	https://exampleurl.ie:443

Description

Issue Detail:

The server software versions used by the application are revealed by the web server.

Displaying version information of software information could allow an attacker to determine which vulnerabilities are present in the software, particularly if an outdated software version is in use with published vulnerabilities.

The following software appears to be in use:

- Microsoft IIS: 10.0
- ASP.Net: 4.0.30319

Evidence

The server software versions used by the application are revealed by the web server. Displaying version information of software information could allow an attacker to determine which vulnerabilities are present in the software, particularly if an outdated software version is in use with published vulnerabilities.

HTTP Request:

```
GET /apiHTTP/1.1
Host: exampleurl.ie
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/144.0.0.0 Safari/537.36
```

HTTP Response:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
```



```
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
```

```
{" "}
```

4.2.3 Strict transport security not enforced

Status	Open
Risk Rating	Low
Affected Host	https://exampleurl.ie:443

Description

Issue Detail:

This issue was found in multiple locations under the reported path.

Background:

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Remediation

The application should instruct web browsers to only access the application using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value 'max-age=expireTime', where expireTime is the time in seconds that browsers should remember that the site should only be accessed using HTTPS. Consider adding the 'includeSubDomains' flag if appropriate.

Note that because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header, and will therefore still be vulnerable to SSL stripping attacks. To mitigate this risk, you can optionally add the 'preload' flag to the HSTS header, and submit the domain for review by browser vendors.

Evidence

This issue was found in multiple locations under the reported path.

HTTP Request:

```
GET /api HTTP/2
Host: exampleurl.ie
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/144.0.0.0 Safari/537.36
```

HTTP Response:

```
HTTP/2 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Server: Microsoft-IIS/10.0
Date: Sun, 15 Feb 2026 10:34:22 GMT
Content-Length: 890

{" "}
```

References

[HTTP Strict Transport Security](#)

[sslstrip](#)

[HSTS Preload Form](#)



4.2.4 Cacheable HTTPS response

Status	Open
Risk Rating	Low
Affected Host	https://exampleurl.ie:443

Description

Issue Detail:

This issue was found in multiple locations under the reported path.

Background:

Unless directed otherwise, browsers may store a local cached copy of content received from web servers. Some browsers, including Internet Explorer, cache content accessed via HTTPS. If sensitive information in application responses is stored in the local cache, then this may be retrieved by other users who have access to the same computer at a future time.

Impact

No impact assessment provided.

Remediation

Applications should return caching directives instructing browsers not to store local copies of any sensitive data. Often, this can be achieved by configuring the web server to prevent caching for relevant paths within the web root. Alternatively, most web development platforms allow you to control the server's caching directives from within individual scripts. Ideally, the web server should return the following HTTP headers in all responses containing sensitive content:

- Cache-control: no-store
- Pragma: no-cache

Evidence

This issue was found in multiple locations under the reported path.

HTTP Request:

```
GET /view/ HTTP/2
Host: exampleurl.ie
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/144.0.0.0 Safari/537.36
```

HTTP Response:

```
HTTP/2 200 OK
```

Content-Type: text/html

Last-Modified: Tue, 20 Sep 2022 16:48:57 GMT

```
<div class="page-title-wrapper page-login-title-wrapper" >
  <div class="mob_title only_mob"><span>ONLINE RENTS ACCOUNT</span>welcome</div>
</div>

<div
  ng-if="vm.isBrowserPlatform()" class="logo_mob_show login-logo-center" href="#/statement">
  
</div>
<div class="login_wrap reset_wrap clearfix">

  <div class="message_wrap error_message" ng-show="notice.isVisible">{{notice.text}}</div>
  <div class="message_wrap" data-ng-show="notice.isVisibleSuccess">{{notice.text}}</div>

  <div class="login_page">
    <div class="login_page_wrap" cg-busy="[vm.resetPasswordPromise]">
      <div class="inner_title btn_login">Reset Password</div>
      <div class="login_wrap_cell ">
        <form name="formReset" novalidate >
          <div class="login_cell reset_input">
            <!-- <div class="input_cell clearfix">
              <label for="accountNumber">Customer Number</label>
              <div class="input_wrap">
                <input id="accountNumber" name="accountNumber" type="text" ng-
model="vm.user.accountNumber" required/>
                <div
                  class="error"
                  data-ng-messages="formReset.accountNumber.$dirty &&
formReset.accountNumber.$error">
                  <div data-ng-messages-include="view/line/messages.html"></div>
                </div>
              </div> -->
              <div class="input_cell input_cell_floating clearfix">
                <input
                  style="border: 1px solid #dce0e2; height: 35px !important;"
                  id="accountNumber"
                  name="accountNumber"
                  type="text"
                  ng-model="vm.user.accountNumber"
                  required
                />
                <label style="padding-left: 7px !important; margin-top: 5px !important;"
for="accountNumber">Customer number</label>
                <div class="login_ico_note" ng-click="vm.toggleLoginTooltip('accountNumber')">
                  <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0
24 24">
                    <g fill="none" fill-rule="evenodd">
                      <path fill="#7F7F7F" d="M11.1 17.4h1.8v-1.8h-1.8v1.8zM12 3c-4.968 0-9
4.032-9 9s4.032 9 9 9-4.032 9-9-9zM0 16.2c-3.969 0-7.2-3.231-7.2-7.2 0-3.969 3.231-
7.2 7.2-7.2 3.969 0 7.2 3.231 7.2 7.2 0 3.969-3.231 7.2-7.2 7.2zM0-12.6a3.599 3.599 0 0 0-3.6
3.6h1.8c0-.99.81-1.8 1.8-1.8s1.8.81 1.8 1.8c0 1.8-2.7 1.575-2.7 4.5h1.8c0-2.025 2.7-2.25 2.7-4.5
0-1.989-1.611-3.6-3.6-3.6z"/>
                    </g>
                  </svg>
                </div>
                <div class="login_note_tooltip" ng-if="vm.isShowTooltip('accountNumber')">
                  <div class="p_close" ng-click="vm.toggleLoginTooltip('accountNumber')">
```

```

        <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20"
viewBox="0 0 24 24">
            <g fill="none" fill-rule="evenodd">
                <path d="M0 0h24v24H0z"></path>
                <path fill="#000" d="M19 6.41L17.59 5 12 10.59 6.41 5 5 6.41
10.59 12 5 17.59 6.41 19 12 13.41 17.59 19 17.59 13.41 12z"></path>
            </g>
        </svg>
    </div>
    {{:: CONTENTS.TOOLTIP.CUSTOMER_NUMBER}}
</div>
<div class="input_wrap">
    <div
        class="error"
        data-ng-messages="formLogin.accountNumber.$touched &&
formLogin.accountNumber.$dirty && formLogin.accountNumber.$error"
    >
        <div data-ng-messages-include="view/line/messages.html"></div>
    </div>
</div>
<div class="captcha-wrapper"
    vc-recaptcha
    on-success="vm.setResponseCaptha(response)"
    key="vm.publicKeyCaptcha"
    ng-if="vm.isBrowserPlatform()" ></div>
</div>
<div class="btn_wrap login_btn_wrap">
    <button
        type="submit"
        class="btn_1"
        ng-click="vm.resetPassword()"
        ng-disabled="formReset.$invalid">
        {{:: CONTENTS.BUTTON.RESET}}
    </button>
</div>
<div data-ng-click="vm.goToLoginPage()" class="back-container">
    <div class="svg-icon-back">
        <?xml version="1.0" encoding="iso-8859-1"?>
        <!-- Generator: Adobe Illustrator 19.0.0, SVG Export Plug-In . SVG Version:
6.00 Build 0) -->
        <svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
viewBox="0 0 476.213 476.213" style="enable-background:new 0 0 476.213
476.213;" xml:space="preserve">
... [truncated 18 more lines]

```

References

[Web Security Academy: Information disclosure](#)



4.2.5 Cross-origin resource sharing

Status	Open
Risk Rating	Low
Affected Host	https://exampleurl.ie:443

Description

Issue Detail:

The application implements an HTML5 cross-origin resource sharing (CORS) policy for this request.

Since the Vary: Origin header was not present in the response, reverse proxies and intermediate servers may cache it. This may enable an attacker to carry out cache poisoning attacks.

Background:

An HTML5 cross-origin resource sharing (CORS) policy controls whether and how content running on other domains can perform two-way interaction with the domain that publishes the policy. The policy is fine-grained and can apply access controls per-request based on the URL and other features of the request.

If another domain is allowed by the policy, then that domain can potentially attack users of the application. If a user is logged in to the application, and visits a domain allowed by the policy, then any malicious content running on that domain can potentially retrieve content from the application, and sometimes carry out actions within the security context of the logged in user.

Even if an allowed domain is not overtly malicious in itself, security vulnerabilities within that domain could potentially be leveraged by an attacker to exploit the trust relationship and attack the application that allows access. CORS policies on pages containing sensitive information should be reviewed to determine whether it is appropriate for the application to trust both the intentions and security posture of any domains granted access.

References:

- Web Security Academy: Cross-origin resource sharing (CORS)
- Exploiting CORS Misconfigurations

Remediation

Any inappropriate domains should be removed from the CORS policy.

Evidence

HTTP Request:

```
GET /api/labels HTTP/2
Host: exampleurl.ie
Cache-Control: max-age=0
Sec-CH-UA: "Chromium";v="144", "Not;A=Brand";v="24", "Google Chrome";v="144"
Sec-CH-UA-Mobile: ?0
Sec-CH-UA-Platform: "Windows"
```

```
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/144.0.0.0 Safari/537.36
```

HTTP Response:

```
HTTP/2 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Access-Control-Allow-Origin: https://exampleurl.ie
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: authorization,content-security-policy,content-type
Date: Mon, 31 Aug 2026 12:46:24 GMT
Content-Length: 890
```

```
{" "}
```

References

[Web Security Academy: Cross-origin resource sharing \(CORS\)](#)

[Exploiting CORS Misconfigurations](#)



4.2.6 Frameable response (potential Clickjacking)

Status	Open
Risk Rating	Low
Affected Host	https://exampleurl.ie:443

Description

Issue Detail:

The response contains an X-Frame-Options header with an unexpected value. Browsers will ignore this header and allow the response to be framed.

Background:

If a page fails to set an appropriate X-Frame-Options or Content-Security-Policy HTTP header, it might be possible for a page controlled by an attacker to load it within an iframe. This may enable a clickjacking attack, in which the attacker's page overlays the target application's interface with a different interface provided by the attacker. By inducing victim users to perform actions such as mouse clicks and keystrokes, the attacker can cause them to unwittingly carry out actions within the application that is being targeted. This technique allows the attacker to circumvent defenses against cross-site request forgery, and may result in unauthorised actions.

Note that some applications attempt to prevent these attacks from within the HTML page itself, using "framebusting" code. However, this type of defense is normally ineffective and can usually be circumvented by a skilled attacker.

You should determine whether any functions accessible within frameable pages can be used by application users to perform any sensitive actions within the application.

Remediation:

The X-Frame-Options header should only have one of the expected values: DENY or SAMEORIGIN.

References:

- Web Security Academy: Clickjacking
- X-Frame-Options

Remediation

To effectively prevent framing attacks, the application should return a response header with the name X-Frame-Options and the value DENY to prevent framing altogether, or the value SAMEORIGIN to allow framing only by pages on the same origin as the response itself. Note that the SAMEORIGIN header can be partially bypassed if the application itself can be made to frame untrusted websites.

Evidence

HTTP Request:

```
GET / HTTP/2
Host: exampleurl.ie
Cache-Control: max-age=0
Sec-CH-UA: "Chromium";v="146", "Not;A=Brand";v="24", "Google Chrome";v="146"
Sec-CH-UA-Mobile: ?0
Sec-CH-UA-Platform: "Windows"
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/146.0.0.0 Safari/537.36
```

HTTP Response:

```
HTTP/2 200 OK
Content-Type: text/html
Last-Modified: Tue, 20 Sep 2022 16:41:26 GMT
Accept-Ranges: bytes
Content-Length: 46810

<!DOCTYPE html>
<!--[if lt IE 7]>
<html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>
<html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>
<html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js" ng-app="app">
  <!--<![endif]-->
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <!--      <meta-->

    <!--      <meta-->
    <!--      http-equiv="Content-Security-Policy"-->
    <!--      content="default-src 'self'"

"
  />
  <meta
    name="description"
    content="iOS position:fixed demo #3 - judder free"
  />
  <meta
    name="viewport"
    content="initial-scale=1, maximum-scale=1, user-scalable=no, width=device-width,
height=device-height, viewport-fit=cover"
  />
  <meta name="format-detection" content="telephone=no" />

  <link rel="shortcut icon" href="img/logo_new_1.png" />
  <link
    rel="stylesheet"
    type="text/css"
```



```
    href="js/vendor/angular-loading-bar/build/loading-bar.min.css"
  />
<link rel="stylesheet" type="text/css" href="css/fontawesome/css/all.css" />
<link
  rel="stylesheet"
  type="text/css"
  href="js/vendor/angular-busy/angular-busy.css"
/>
<link
  rel="stylesheet"
  type="text/css"
  href="js/vendor/jquery-form-styler/jquery.formstyler.css"
/>
<link
  rel="stylesheet"
  type="text/css"
  href="js/vendor/angular-ui-switch/angular-ui-switch.min.css"
/>

<link rel="stylesheet" href="css/main.css" />
<link rel="stylesheet" href="css/modal.css" />
<link rel="stylesheet" href="css/main_custom.css" />

</head>

<body
  class="{{controllerName}} {{wrapMob}} {{androidWrap}}"
  ng-class="{ 'invoices': controllerName == 'InvoiceController', 'page_desktop': pageDeskTop}"
  id="body"
  resize
>
  <!-- nav for invoices -->
  <div
    class="nav_invoices"
    data-ng-class="{ 'header_show_all': controllerName !== 'login' }"
    ng-if="controllerName === 'InvoiceController' && controllerName !== 'login'"
  >
    <div class="clearfix nav_invoices_wrap">
      <button class="btn_1" ng-click="prePaymentBack()">
        {{:: BUTTON.CANCEL}}
      </button>
      <button
        class="btn_1"
... [truncated 679 more lines]
```

References

[Web Security Academy: Clickjacking](#)

[X-Frame-Options](#)

5 APPENDICES

5.1 About Us

Based in Dublin, Ireland, CommSec is a dedicated cyber security managed service provider. Our core values of honesty and integrity drive our commitment to delivering measurable value and building long-term partnerships with clients across various sectors, including government, logistics, finance, manufacturing, and software development.

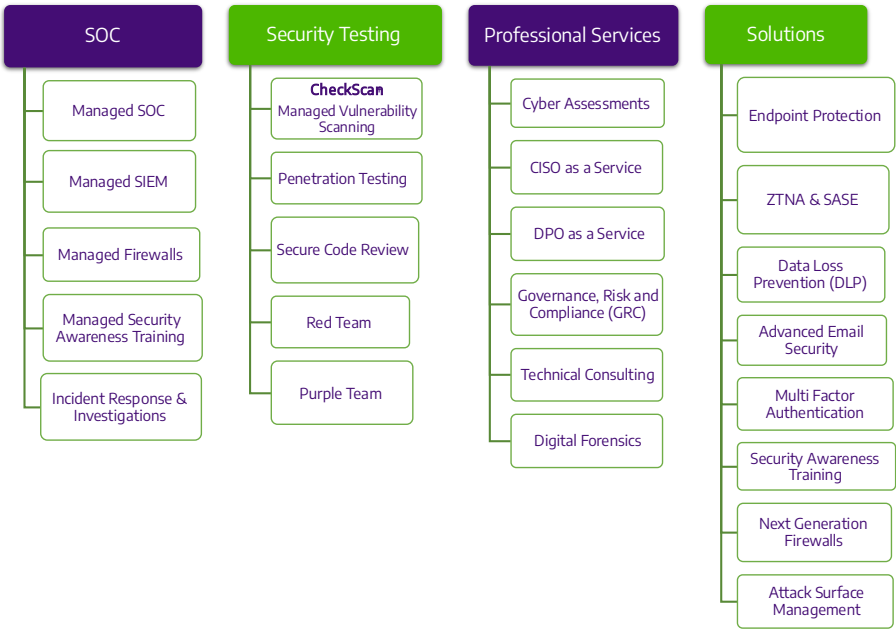
With over a decade of experience, we understand the complexities of the ever-evolving threat landscape. We pride ourselves on providing tailored solutions and maintaining the highest confidentiality for client data, as evidenced by our ISO/IEC 27001 accreditation and Cyber Essentials certification.

Our local 24/7 Security Operations Centre (SOC) ensures unparalleled support and responsiveness. Partnering with industry leaders like CheckPoint, Palo Alto, Fortinet, and Sophos, we offer market-leading solutions to address each client's specific needs. We strive to be the trusted advisor that organisations turn to for all their cyber security requirements.

CommSec Cyber Security Offerings

We provide end-to-end cyber security services from planning and implementation to support, advice, and remediation for our customers.

Our core offerings are summarised under the following 4 Pillars:





CommSec Cyber Security

Suite B109, The LINC, TUD,
Blanchardstown, Dublin D15 VPT3

t. 353 1 536 7320
e. info@commsec.ie
www.commsec.ie

