invicti
AppSec with Zero Noise

Support

Search in support

Support Categories

EXPLANATIONS

Vulnerability Severity Levels

THIS DOCUMENT IS FOR:
Invicti Standard, Invicti Enterprise On-Premises, Invicti Enterprise On-Demand

Feedback

Invicti scans for a wide variety of **vulnerabilities** in websites, web applications and web services. Invicti's automation makes it easy to scan websites and prioritise the findings, helping you decide which ones to tackle first, based on defining acceptable risks from a corporate point of view.

Each vulnerability has a different impact:

- Some detected vulnerabilities need to be addressed urgently, because they cause the application to be compromised or damaged by attackers (Critical, High), while others are less of a priority (Low). For example, an **SQL injection** vulnerability should definitely be prioritized over an Internal IP address disclosure.
- Some highlighted findings are simply notes that give information that is relevant to the target application's infrastructure. Others, such as Best Practice or Information Alerts, help website owners implement additional security measures.

What Are Vulnerability Severities?

To help you better decide which vulnerabilities should be fixed first, Invicti categorizes them using risk scores in its scans and reports. There are four vulnerability levels:

- Critical (❗)
- High (🚩)
- Medium (🚩)
- Low (🚩)

There are two additional types of alerts: Best Practice (💡) and Information Alerts (ℹ).

For further information, see our Web Application Vulnerabilities Index.

Critical Severity Web Vulnerabilities

This section explains how we define and identify web vulnerabilities of Critical severity (❗).

Feedback

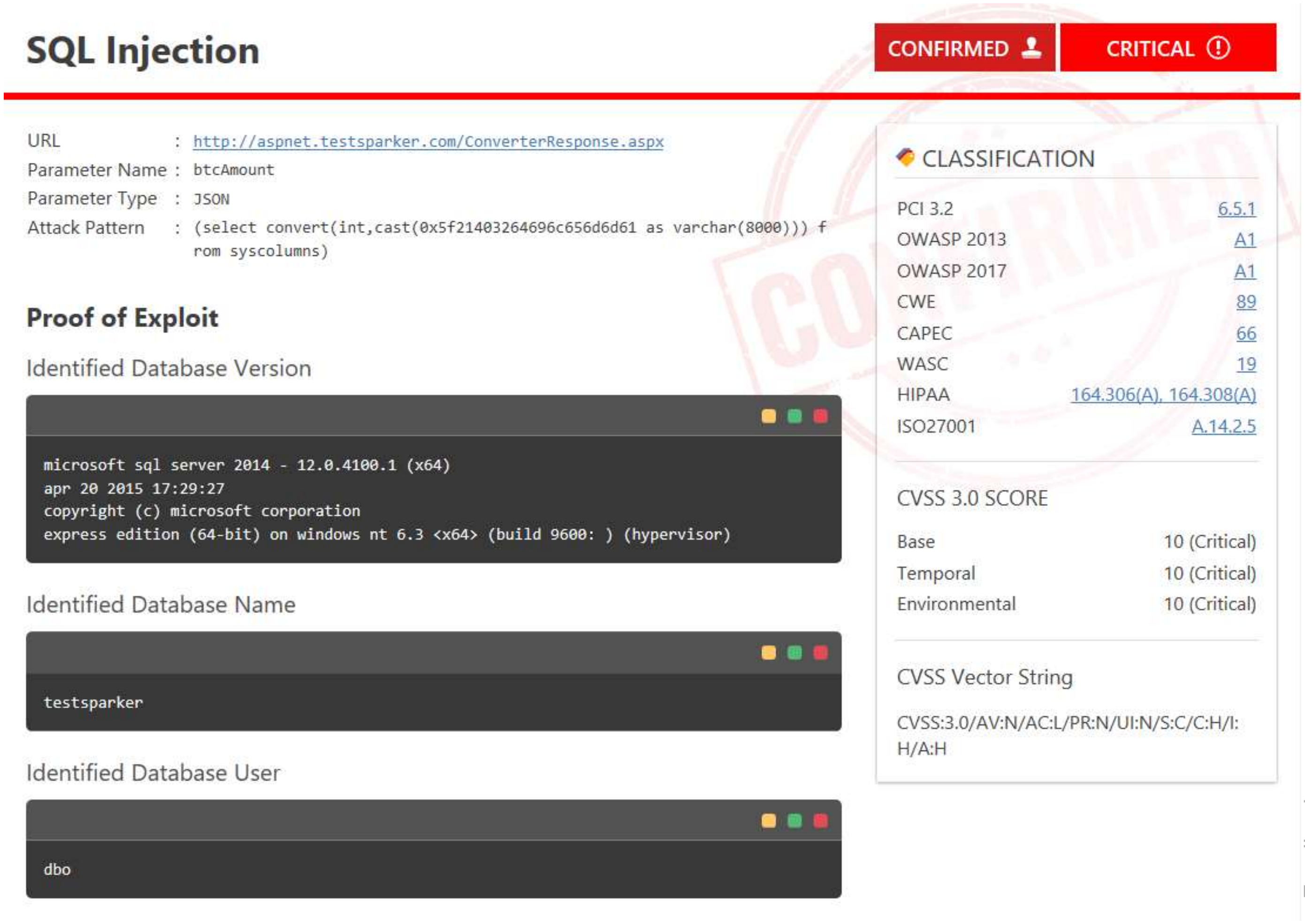
The issues marked as Critical Severity can allow attackers to execute code on the web application or application server, or access sensitive data.

Impacts of Critical Severity Web Vulnerabilities

- Examples include SQL Injection, Remote Code Execution and Command Injections. In exploiting this type of vulnerability, attackers could carry out a range of malicious acts that could, for example, affect an web application's availability, or put its confidentiality and security at risk.
- In addition, it is the existence and prevalence of automated exploitation tools that make fixing these types of issues urgent.

Critical Severity Example

This is what a report of a Critical severity vulnerability looks like in Invicti.



SQL Injection

CONFIRMED CRITICAL

URL	: http://aspnet.testsparker.com/ConverterResponse.aspx
Parameter Name	: btcAmount
Parameter Type	: JSON
Attack Pattern	: (select convert(int,cast(0x5f21403264696c656d6d61 as varchar(8000))) from syscolumns)

Proof of Exploit

Identified Database Version

```
microsoft sql server 2014 - 12.0.4100.1 (x64)
apr 20 2015 17:29:27
copyright (c) microsoft corporation
express edition (64-bit) on windows nt 6.3 <x64> (build 9600: ) (hypervisor)
```

Identified Database Name

```
testsparker
```

Identified Database User

```
dbo
```

CLASSIFICATION

PCI 3.2	6.5.1
OWASP 2013	A1
OWASP 2017	A1
CWE	89
CAPEC	66
WASC	19
HIPAA	164.306(A), 164.308(A)
ISO27001	A.14.2.5

CVSS 3.0 SCORE

Base	10 (Critical)
Temporal	10 (Critical)
Environmental	10 (Critical)

CVSS Vector String

CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

Feedback

Suggested Action for Critical Severity Vulnerabilities

A Critical severity vulnerability means that your website is at risk of being hacked at any time. We recommend that you make it your highest priority to fix these vulnerabilities *immediately*.

High Severity Web Vulnerabilities

This section explains how we define and identify web vulnerabilities of High severity (🚩).

The issues marked as High Severity can allow malicious attackers to access application resources and data. This can allow an attacker to steal session information or sensitive data from the application or server.

The difference between a Critical and High Severity is that with a High Severity vulnerability, a malicious attacker cannot execute code or a command on the application or server.

Impacts of High Severity Vulnerabilities

. In the case of a detected XSS vulnerability, an attacker could: Examples include XSS, XML External Entity Injection and LFI.

- Execute script code in the user's browser
- Steal the user's cookies

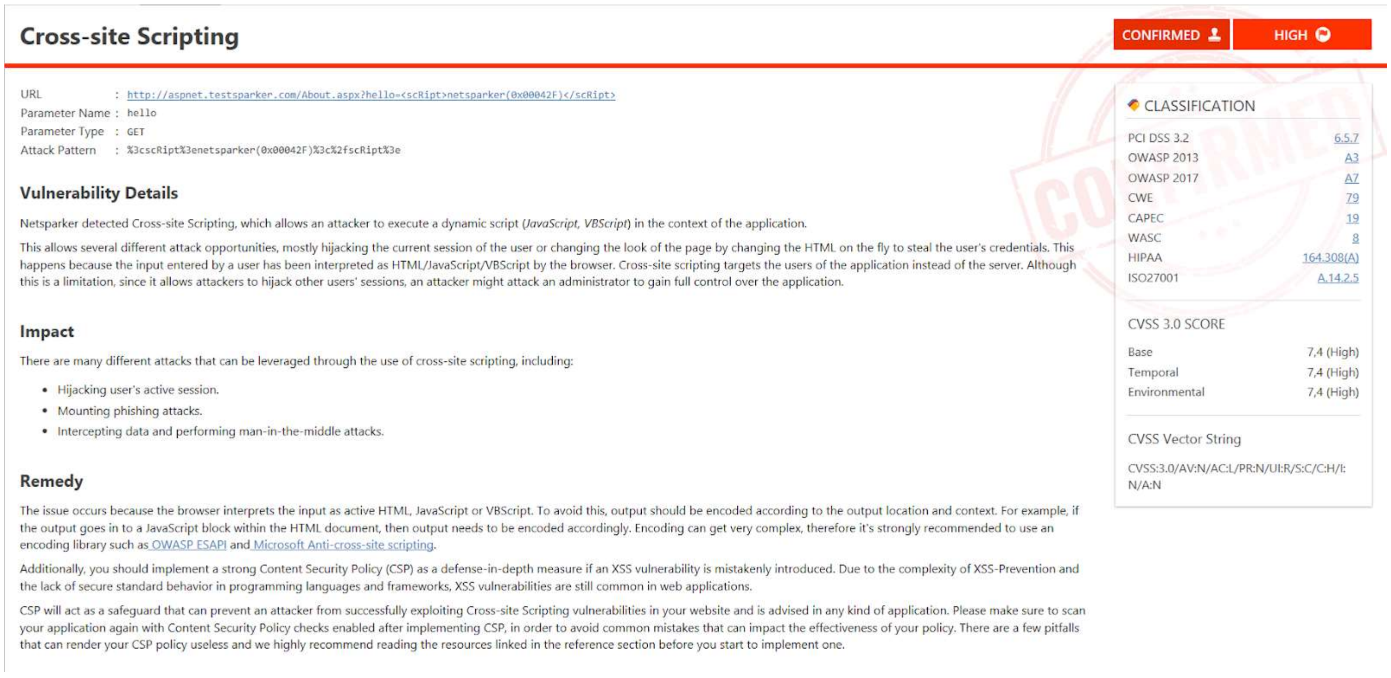
In the case of a detected XXE vulnerability, an attacker could:

- Read sensitive data in the server
- Make requests to internal or external resources

. Attackers conducting this type of attack have some technical skills, but many tools make the exploitation process automated.

High Severity Example

This is what a report of a High severity vulnerability looks like in Invicti.



Cross-site Scripting

URL	: http://aspnet.testsparker.com/About.aspx?hello=<scRipt>netsparker(0x00042F)</scRipt>
Parameter Name	: hello
Parameter Type	: GET
Attack Pattern	: %3cscRipt%3enetsparker(0x00042F)%3c%2fscRipt%3e

Vulnerability Details

Netsparker detected Cross-site Scripting, which allows an attacker to execute a dynamic script (*JavaScript, VBScript*) in the context of the application.

This allows several different attack opportunities, mostly hijacking the current session of the user or changing the look of the page by changing the HTML on the fly to steal the user's credentials. This happens because the input entered by a user has been interpreted as HTML/JavaScript/VBScript by the browser. Cross-site scripting targets the users of the application instead of the server. Although this is a limitation, since it allows attackers to hijack other users' sessions, an attacker might attack an administrator to gain full control over the application.

Impact

There are many different attacks that can be leveraged through the use of cross-site scripting, including:

- Hijacking user's active session.
- Mounting phishing attacks.
- Intercepting data and performing man-in-the-middle attacks.

Remedy

The issue occurs because the browser interprets the input as active HTML, JavaScript or VBScript. To avoid this, output should be encoded according to the output location and context. For example, if the output goes in to a JavaScript block within the HTML document, then output needs to be encoded accordingly. Encoding can get very complex, therefore it's strongly recommended to use an encoding library such as OWASP ESAPI and Microsoft Anti-cross-site scripting.

Additionally, you should implement a strong Content Security Policy (CSP) as a defense-in-depth measure if an XSS vulnerability is mistakenly introduced. Due to the complexity of XSS-Prevention and the lack of secure standard behavior in programming languages and frameworks, XSS vulnerabilities are still common in web applications.

CSP will act as a safeguard that can prevent an attacker from successfully exploiting Cross-site Scripting vulnerabilities in your website and is advised in any kind of application. Please make sure to scan your application again with Content Security Policy checks enabled after implementing CSP, in order to avoid common mistakes that can impact the effectiveness of your policy. There are a few pitfalls that can render your CSP policy useless and we highly recommend reading the resources linked in the reference section before you start to implement one.

CLASSIFICATION

PCI DSS 3.2	6.5.7
OWASP 2013	A3
OWASP 2017	A7
CWE	79
CAPEC	19
WASC	8
HIPAA	164.308(A)
ISO27001	A.14.2.5

CVSS 3.0 SCORE

Base	7,4 (High)
Temporal	7,4 (High)
Environmental	7,4 (High)

CVSS Vector String

CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:N/A:N

Suggested Action for High Severity Vulnerabilities

A High severity vulnerability means that your website can be hacked and can lead hackers to find other vulnerabilities which have a bigger impact. We recommend that you fix these types of vulnerabilities *immediately*.

Medium Severity Web Vulnerabilities

This section explains how we define and identify vulnerabilities of Medium severity (🚩).

The issues marked as Medium Severity usually arise because of errors and deficiencies in the application configuration. By exploiting these security issues, malicious attackers can access sensitive information on the application or server.

In comparison to Critical and High Severity issues, the impact is relatively limited.

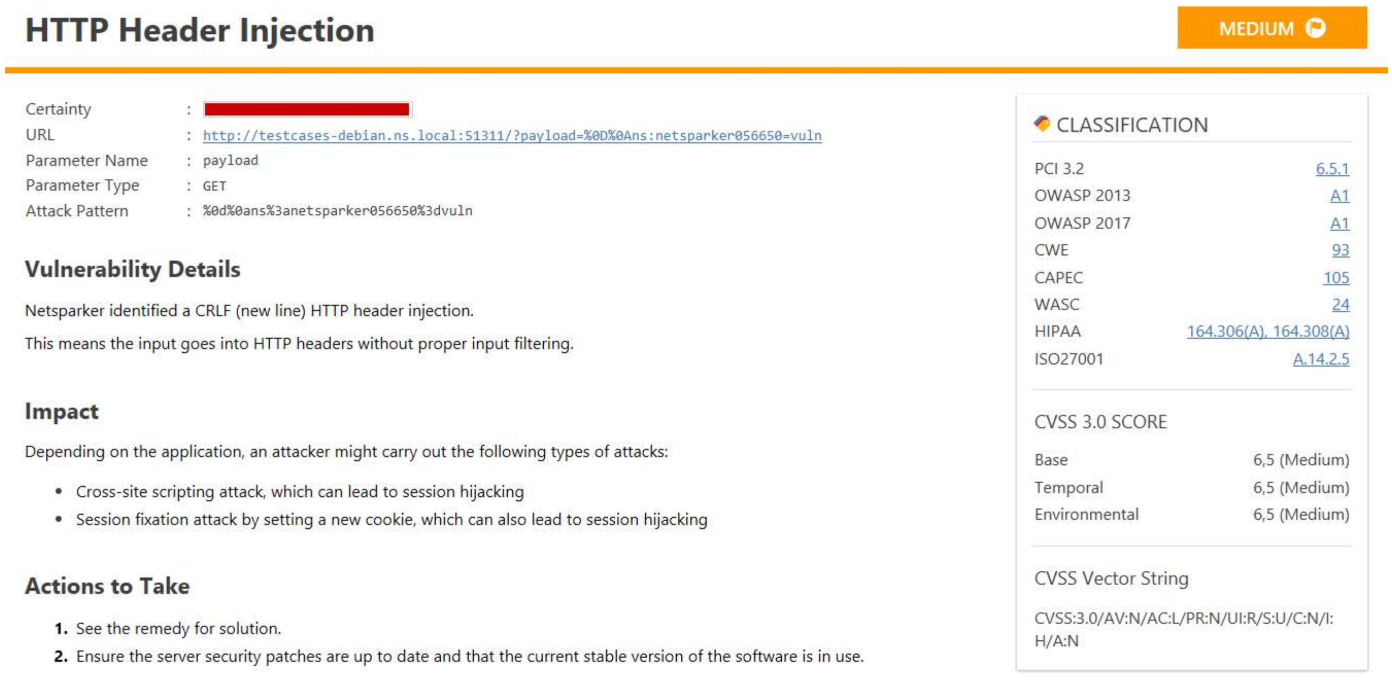
Feedback

Impacts of Medium Severity Vulnerabilities

- Attackers conducting this type of attack require more skill than those exploiting Critical and High Severities.
- Exploitation of these types of vulnerabilities can depend on the existence of some special conditions. For example, in the case of SSL/TLS certificate issues, or misconfiguration of TLS, an attacker has to be in an appropriate location to be able to eavesdrop on the connection of the victim.

Medium Severity Example

This is what a report of a Medium severity vulnerability looks like in Invicti.



HTTP Header Injection MEDIUM

Certainty	: 🟥
URL	: http://testcases-debian.ns.local:51311/?payload=%0D%0Ans:netsparker056650=vuln
Parameter Name	: payload
Parameter Type	: GET
Attack Pattern	: %0d%0ans%3anetsparker056650%3dvuln

Vulnerability Details

Netsparker identified a CRLF (new line) HTTP header injection.

This means the input goes into HTTP headers without proper input filtering.

Impact

Depending on the application, an attacker might carry out the following types of attacks:

- Cross-site scripting attack, which can lead to session hijacking
- Session fixation attack by setting a new cookie, which can also lead to session hijacking

Actions to Take

1. See the remedy for solution.
2. Ensure the server security patches are up to date and that the current stable version of the software is in use.

CLASSIFICATION

PCI 3.2	6.5.1
OWASP 2013	A1
OWASP 2017	A1
CWE	93
CAPEC	105
WASC	24
HIPAA	164.306(A), 164.308(A)
ISO27001	A.14.2.5

CVSS 3.0 SCORE

Base	6,5 (Medium)
Temporal	6,5 (Medium)
Environmental	6,5 (Medium)

CVSS Vector String

CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:N

Suggested Action for Medium Severity Vulnerabilities

Even though special conditions are required to exploit Medium Severity issues and they don't directly affect the application or system (in contrast to Critical and High Severities), in order to keep your web application secure and comply with the regulations, they should still be fixed.

Low Severity Web Vulnerabilities

This section explains how we define and identify web vulnerabilities of Low severity ( ).

The issues marked as Low Severity include information leakage, configuration errors and a lack of some security measures. They can be combined with other issues of a higher severity level, and can be used in conjunction with social engineering (manipulating people into following certain actions or revealing crucial information), to cause a more severe impact on the target.

In comparison to Critical, High and Medium Severity issues, these findings have limited effect.

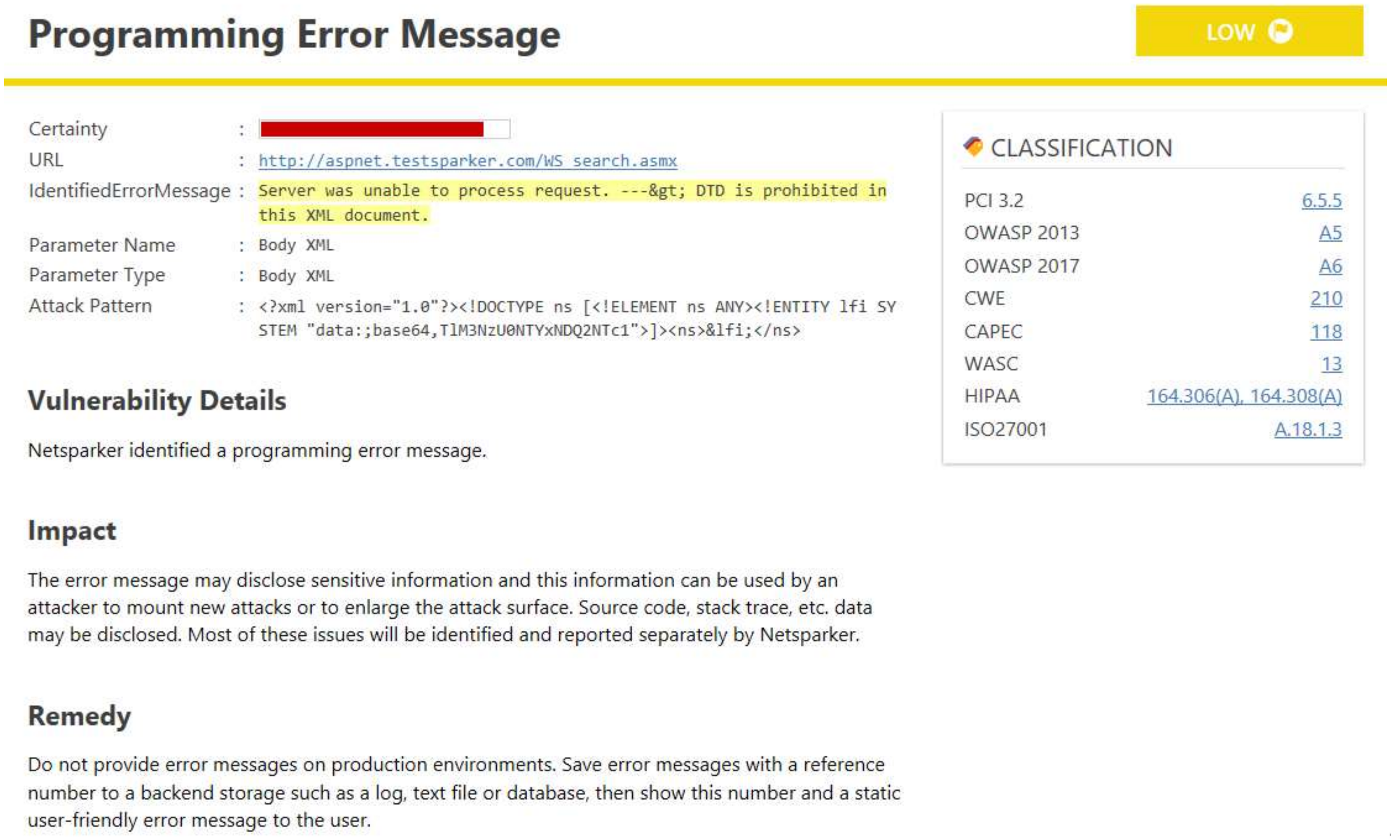
Impacts of Low Severity Vulnerabilities

- When a website reveals the version number of an application, an attacker can carry out Vulnerability Mapping by looking at the vulnerability database to see if an issue exists in that version of the application and then exploiting it.
- Invicti reports Username Disclosure vulnerabilities when related to Windows or Linux operating systems or RDBMS. Though they are flagged as low level vulnerabilities by themselves, an attacker could use this information to find a way to access the target application's operating system or database system.
- In the case of application configuration errors and deficiencies such as an X-Frame-Options header (XFO) – which controls whether a website is loaded by itself, another site or neither – Invicti reports a missing XFO if the scanned web application does not set, or mistakenly sets, the XFO header. An attacker could exploit these configuration errors by convincing an authorized user to click on a malicious link or button (a potentially state-changing operation), that could result in the deletion or records or uncover hidden resources.

Feedback

Low Severity Example

This is what a report of a Low severity vulnerability looks like in Invicti.



Programming Error Message

LOW

Certainty	:
URL	: http://aspnet.testsparker.com/WS_search.asmx
IdentifiedErrorMessage	: Server was unable to process request. ---&gt; DTD is prohibited in this XML document.
Parameter Name	: Body XML
Parameter Type	: Body XML
Attack Pattern	: <?xml version="1.0"?><!DOCTYPE ns [<!ELEMENT ns ANY><!ENTITY lfi SYSTEM "data:;base64,TlM3NzU0NTYxNDQ2NTc1">]><ns>&lfi;</ns>

CLASSIFICATION

PCI 3.2	6.5.5
OWASP 2013	A5
OWASP 2017	A6
CWE	210
CAPEC	118
WASC	13
HIPAA	164.306(A), 164.308(A)
ISO27001	A.18.1.3

Vulnerability Details

Netsparker identified a programming error message.

Impact

The error message may disclose sensitive information and this information can be used by an attacker to mount new attacks or to enlarge the attack surface. Source code, stack trace, etc. data may be disclosed. Most of these issues will be identified and reported separately by Netsparker.

Remedy

Do not provide error messages on production environments. Save error messages with a reference number to a backend storage such as a log, text file or database, then show this number and a static user-friendly error message to the user.

Feedback

Suggested Action for Low Severity Vulnerabilities

A decision on whether to fix these issues should be determined by assessing the context in the application, and by considering the business impacts.

Best Practice

This section explains how we define and identify issues that are marked as Best Practice (💡).

Making web applications secure involves more than taking rapid action on detected vulnerabilities. Browser vendors offer various features that make being proactive easier than ever. These preemptive standards and recommendations help software developers make web applications that are secure by design.

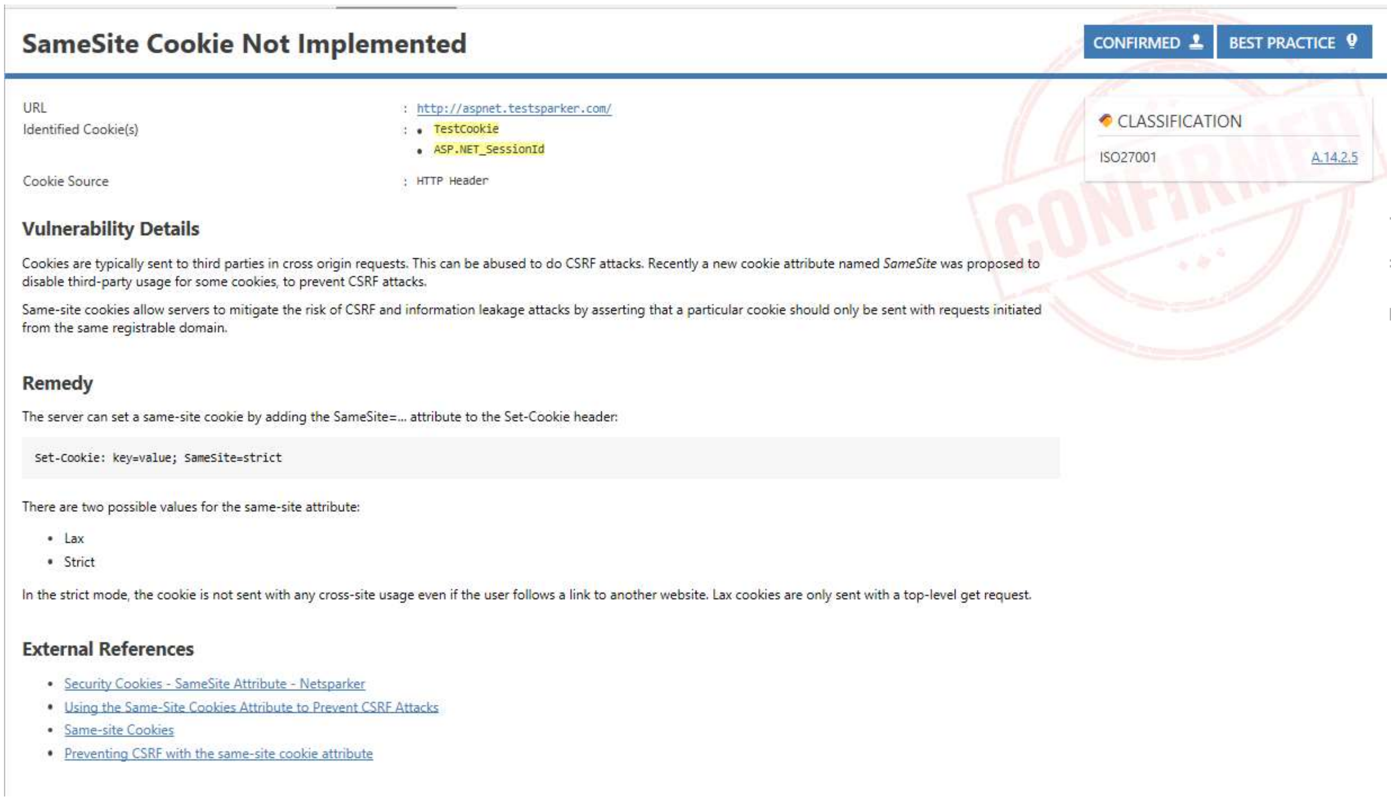
Impacts of Best Practice Issues

- The issues marked as Best Practice are recommendations, not vulnerabilities. Examples include **Content Security Policy**, **Referrer-Policy**, **Expect-CT**, Subresource Integrity – security implementations that are provided by browser vendors. Think of these recommendations as an extra security layer, defence in depth, to help continually contribute to the security of your web applications – proactively.

- Netspaker makes suggestions around these standards and reports errors and issues in the implementation of them.

Best Practice Example

This is what a report of a Best Practice issue looks like in Invicti.



SameSite Cookie Not Implemented

URL : http://aspnet.testsparker.com/
Identified Cookie(s) :
- TestCookie
- ASP.NET_SessionId

Cookie Source : HTTP Header

CLASSIFICATION
ISO27001 A.14.2.5

Vulnerability Details

Cookies are typically sent to third parties in cross origin requests. This can be abused to do CSRF attacks. Recently a new cookie attribute named *SameSite* was proposed to disable third-party usage for some cookies, to prevent CSRF attacks.

Same-site cookies allow servers to mitigate the risk of CSRF and information leakage attacks by asserting that a particular cookie should only be sent with requests initiated from the same registrable domain.

Remedy

The server can set a same-site cookie by adding the SameSite=... attribute to the Set-Cookie header:

```
Set-Cookie: key=value; SameSite=strict
```

There are two possible values for the same-site attribute:

- Lax
- Strict

In the strict mode, the cookie is not sent with any cross-site usage even if the user follows a link to another website. Lax cookies are only sent with a top-level get request.

External References

- Security Cookies - SameSite Attribute - Netsparker
- Using the Same-Site Cookies Attribute to Prevent CSRF Attacks
- Same-site Cookies
- Preventing CSRF with the same-site cookie attribute

Feedback

Suggested Action for Best Practice Issues

Invicti recommends that these Best Practice suggestions are implemented in order to make your web application secure.

Information Alerts

This section explains how we define and identify issues that are marked as Information Alerts (ℹ).

The findings reported are mostly for informing you about the target's ingredients and infrastructure. They help you to understand the application's technology stack and dependencies well.

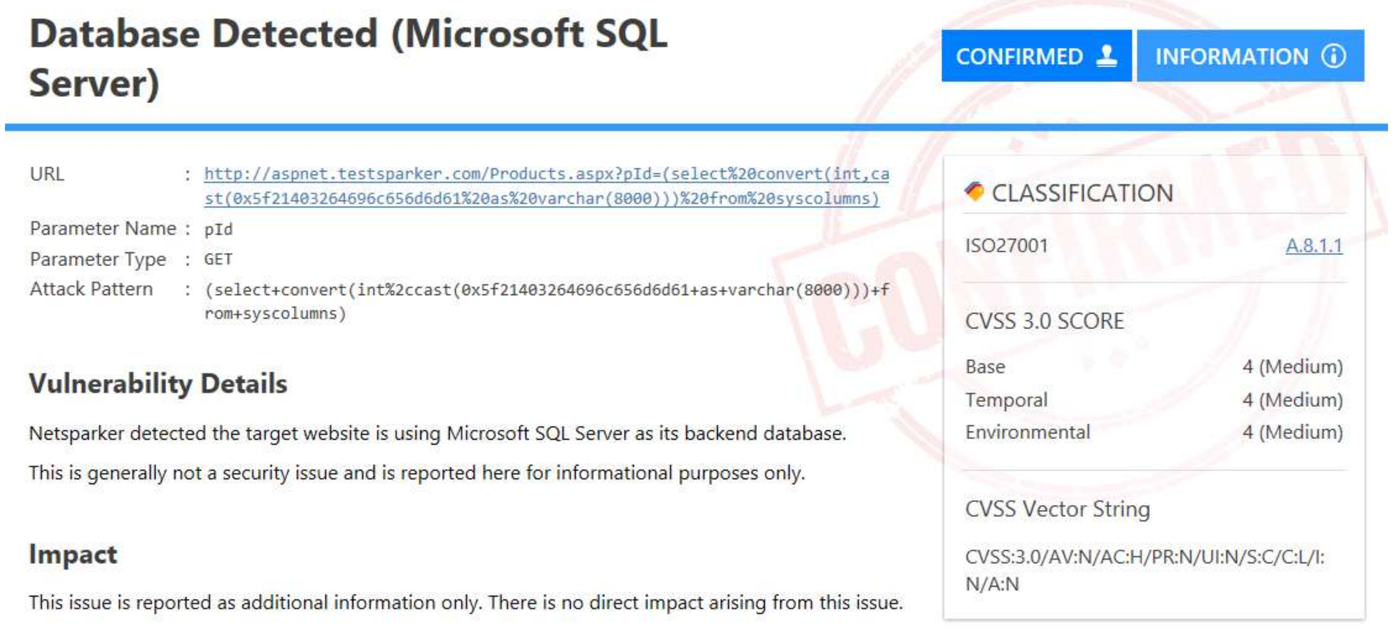
Impacts of Information Alerts

- The issues highlighted in these alerts can help attackers understand the target more and therefore tailor their attack better, eliminate other possibilities and conduct vulnerability mapping.
- For example, revealing that a website uses a certain IIS version does not seem that important at first sight. However, it means that the OS of the target web application is a Windows OS, for example. So, an attacker can eliminate attack possibilities regarding other operating systems. In addition, vendors who use IIS tend to prefer application infrastructures offered by Microsoft. An attacker could reasonably assume that the target application was developed using either ASP or .NET technologies. This can further help them eliminate other attack possibilities regarding other application infrastructure and save time.
- In the case of vulnerability mapping, if the target uses older versions of IIS that have known security issues, this can allow a target machine to be compromised by an attacker. For instance CVE-2017-7269 was an issue in IIS 6.0 and exploited since 2016. It allows remote attackers to execute arbitrary code in the target. (Please note that in the case of an out-of-date component, and an associated vulnerability, this would be reported at a higher level than Information Alert.

Feedback

Information Alert Example

This is what a report of an Information Alert issue looks like in Invicti.



Database Detected (Microsoft SQL Server)

CONFIRMED INFORMATION

URL	: http://aspnet.testsparker.com/Products.aspx?pId=(select%20convert(int,cast(0x5f21403264696c656d6d61%20as%20varchar(8000)))%20from%20syscolumns)
Parameter Name	: pId
Parameter Type	: GET
Attack Pattern	: (select+convert(int%2ccast(0x5f21403264696c656d6d61+as+varchar(8000)))+from+syscolumns)

Vulnerability Details

Netsparker detected the target website is using Microsoft SQL Server as its backend database.

This is generally not a security issue and is reported here for informational purposes only.

Impact

This issue is reported as additional information only. There is no direct impact arising from this issue.

CLASSIFICATION

ISO27001	A.8.1.1

CVSS 3.0 SCORE

Base	4 (Medium)
Temporal	4 (Medium)
Environmental	4 (Medium)

CVSS Vector String

CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:C/C:L/I:N/A:N

Suggested Action for Information Alerts

Most of the time, there is no need to take any action for Information level findings. This is why Invicti Enterprise marks them as Accepted Risk.

However, it is recommended that you manually review Information Alert level findings and modify the application to avoid revealing details that give hints or information regarding the application itself.

Feedback

Tweet

Share

Share

Email

TOP ARTICLES

What is Invicti?

Overview of Scan Policies

Scheduling Scans

Managing Integrations

Built-In Reports

Invicti Help Center

Our Support team is ready to provide you with technical help.

Go to Help Center ▶

This will redirect you to the ticketing system.

Feedback

invicti

Invicti Security Corp 1000 N Lamar Blvd Suite 300 Austin, TX 78703, US

RESOURCES

Features

Integrations

Plans

Case Studies

Advisories

Invicti Learn

USE CASES

Penetration Testing Software

Website Security Scanner

Ethical Hacking Software

Web Vulnerability Scanner

Comparisons

Online Application Scanner

WEB SECURITY

The Problem with False Positives

Why Pay for Web Scanners

SQL Injection Cheat Sheet

Getting Started with Web Security

Vulnerability Index

Using Content Security Policy to Secure Web Applications

COMPANY

About Us

Contact Us

Support

Careers

Resources

Partners

Feedback

Legal

Privacy Policy

California Privacy Rights

Terms of Use

Accessibility

Sitemap