[High Possibility] SQL Injection

SQL Injection occurs when data input for example by a user is interpreted as a SQL command rather than normal data by the backend database. This is an extremely common vulnerability and its successful exploitation can have critical implications. Even though Netsparker believes that there is a SQL Injection in here it **could not confirm** it. There can be numerous reasons for Netsparker not being able to confirm this. We strongly recommend investigating the issue manually to ensure that it is an SQL Injection and that it needs to be addressed. You can also consider sending the details of this issue to us, in order that we can address this issue for the next time and give you a more precise result.

Summary

Severity :	Critical
Confirmation :	**Confirmed**
Detection Accuracy :	
Vulnerable URL :	http://jornalismointerativo.com.br/game/google-maps/mapa.php?id=%27
Vulnerability Classifications:	PCI 6.5.2 OWASP A1 CAPEC-66 CWE-89 98
Parameter Name:	id
Parameter Type:	Querystring
Attack Pattern:	%27

Impact

Depending on the backend database, database connection settings and the operating system, an attacker can mount one or more of the following type of attacks successfully:

- Reading, Updating and Deleting arbitrary data from the database
- Executing commands on the underlying operating system
- Reading, Updating and Deleting arbitrary tables from the database

Actions to Take

1. See the remedy for solution.
2. If you are not using a database access layer (DAL) within the architecture consider its benefits and implement if appropriate. As a minimum the use of s DAL will help centralize the issue and its resolution. You can also use an ORM (*object relational mapping*). Most ORM systems use parameterized queries and this can solve many if not all SQL Injection based problems.
3. Locate all of the dynamically generated SQL queries and convert them to parameterised queries. (*If you decide to use a DAL/ORM, change all legacy code to use these new libraries*)
4. Monitor and review weblogs and application logs in order to uncover active or previous exploitation attempts.

Remedy

A very robust method for mitigating the threat of SQL Injection based vulnerabilities is to use parameterized queries (*prepared statements*). Almost all modern languages provide built in libraries for this. Wherever possible do not create dynamic SQL queries or SQL queries with string concatenation.

Required Skills for Successful Exploitation

There are numerous freely available tools to test for SQL Injection vulnerabilities. This is a complex area with many dependencies, however it should be noted that the numerous resources available in this area have raised both attacker awareness of the issues and their ability to discover and leverage them. SQL Injection is one of the most common web application vulnerabilities.

Password Transmitted Over HTTP

Table of Content
Netsparker identified that password data is sent over HTTP.

Summary

Severity :	Important
Confirmation :	**Confirmed**
Detection Accuracy :	
Vulnerable URL :	http://jornalismointerativo.com.br/noticia.php?sessao=17&noticia=29
Vulnerability Classifications:	PCI 6.5.9 OWASP A9 CWE-311 319
Form target action:	index.php?login=1

Impact

If an attacker can intercept network traffic he/she can steal users credentials.

Actions to Take

1. See the remedy for solution.
2. Move all of your critical forms and pages to HTTPS and do not serve them over HTTP.

Remedy

All sensitive data should be transferred over HTTPS rather than HTTP. Forms should be served over HTTPS. All aspects of the application that accept user input starting from the login process should only be served over HTTPS.

Cross-site Scripting

XSS (Cross-site Scripting) 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.

XSS targets the users of the application instead of the server. Although this is a limitation, since it allows attackers to hijack other users' session, an attacker might attack an administrator to gain full control over the application.

Summary

Severity :	Important
Confirmation :	**Confirmed**
Detection Accuracy :	
Vulnerable URL :	http://jornalismointerativo.com.br/noticia.php?sessao=" stYle="x:expre/**/ssion(alert(9)) &noticia=29&info=FIFA
Vulnerability Classifications:	PCI 6.5.1 OWASP A2 CAPEC-19 CWE-79 79
Parameter Name:	sessao
Parameter Type:	Querystring
Attack Pattern:	" stYle="x:expre/**/ssion(alert(9))

Impact

There are many different attacks that can be leveraged through the use of XSS, including:

- Hi-jacking users' active session
- Changing the look of the page within the victims browser.
- Mounting a successful phishing attack.
- Intercept data and perform man-in-the-middle attacks.

Remedy

The issue occurs because the browser interprets the input as active HTML, Javascript or VbScript. To avoid this, all input and output from the application should be filtered. Output should be filtered according to the output format and location. Typically the output location is HTML. Where the output is HTML ensure that all active content is removed prior to its presentation to the server.

Prior to sanitizing user input, ensure you have a pre-defined list of both expected and acceptable characters with which you populate a white-list. This list needs only be defined once and should be used to sanitize and validate all subsequent input.

There are a number of pre-defined, well structured white-list libraries available for many different environments, good examples of these include, OWASP Reform and Microsoft Anti Cross-site Scripting libraries are good examples.

Remedy References

- [ASP.NET] - Microsoft Anti-XSS Library
- OWASP XSS Prevention Cheat Sheet

External References

- XSS Cheat Sheet
- OWASP - Cross-site Scripting
- XSS Shell
- XSS Tunnelling Paper

Permanent Cross-site Scripting

Netsparker **confirmed** this vulnerability by analyzing the execution of injected JavaScript.

Permanent XSS (Cross-site Scripting) allows an attacker to execute dynamic scripts (*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 and to steal the user's credentials. This happens because the input entered by the user has been interpreted by HTML/Javascript/VbScript within the browser.

Permanent means that the attack will be stored in the back-end system. In normal XSS attacks an attack needs to e-mail the victim but in a permanent XSS an attacker can just execute the attack and wait for users to see the affected page. As soon as someone visits the page, the attacker's stored payload will get executed.

XSS targets the users of the application instead of the server. Although this is a limitation, since it only allows attackers to hijack other users' session the attacker might attack an administrator to gain full control over the application.

Summary

Severity :	Important
Confirmation :	**Confirmed**
Detection Accuracy :	
Vulnerable URL :	http://jornalismointerativo.com.br/noticia.php?sessao=17&noticia=29
Vulnerability Classifications:	PCI 6.5.1 OWASP A2 CAPEC-19 CWE-79 79
Injection URL:	http://jornalismointerativo.com.br/noticia.php?sessao=17&noticia=29
Parameter Name:	comentario
Parameter Type:	Post
Attack Pattern:	"+(select 1 and row(1,1)>(select count(*),concat(CONCAT(CHAR(95),CHAR(33),CHAR(64),CHAR(52),CHAR(100),CHAR(105),CHAR(108),CHAR(101),CHAR(109),CHAR(109),CHAR(97)),0 from (select 1 union select 2)a group by x limit 1))+"

Impact

Permanent XSS is a dangerous issue that has many exploitation vectors, some of which includes:

- User session sensitive information such as cookies can be stolen.
- XSS can enable client-side worms which could modify, delete or steal other users' data within the application.
- The website can be redirected to a new location, defaced or used as a phishing site.

Remedy

The issue occurs because the browser interprets the input as active HTML, Javascript or VbScript. To avoid this, all input and output from the application should be filtered. Output should be filtered according to the output format and location. Typically the output location is HTML. Where the output is HTML ensure that all active content is removed prior to its presentation to the server.

Prior to sanitizing user input, ensure you have a pre-defined list of both expected and acceptable characters with which you populate a white-list. This list needs only be defined once and should be used to sanitize and validate all subsequent input.

There are a number of pre-defined, well structured white-list libraries available for many different environments, good examples of these include, OWASP Reform and Microsoft Anti Cross-site Scripting libraries are good examples.

Remedy References

- [ASP.NET] - Microsoft Anti-XSS Library
- OWASP XSS Prevention Cheat Sheet

External References

- XSS Cheat Sheet
- OWASP - Cross-site Scripting
- XSS Shell
- XSS Tunnelling Paper

Cookie Not Marked As HttpOnly

Cookie was not marked as HTTPOnly. HTTPOnly cookies can not be read by client-side scripts therefore marking a cookie as HTTPOnly can provide an additional layer of protection against Cross-site Scripting attacks..

Summary

Severity :	Low
Confirmation :	**Confirmed**
Detection Accuracy :	
Vulnerable URL :	http://jornalismointerativo.com.br/
Vulnerability Classifications:	OWASP A6 CWE-16
Identified Cookie:	PORTAL_VC_REPORTER

Impact

During a Cross-site Scripting attack an attacker might easily access cookies and hijack the victim's session.

Actions to Take

1. See the remedy for solution
2. Consider marking all of the cookies used by the application as HTTPOnly (*After these changes javascript code will not able to read cookies.*

Remedy

Mark the cookie as HTTPOnly. This will be an extra layer of defence against XSS. However this is not a silver bullet and will not protect the system against Cross-site Scripting attacks. An attacker can use a tool such as XSS Tunnel to bypass HTTPOnly protection.

External References

- OWASP HTTPOnly Cookies
- MSDN - ASP.NET HTTPOnly Cookies

Auto Complete Enabled

"Auto Complete" was enabled in one or more of the form fields. These were either "password" fields or important fields such as "Credit Card".

Summary

Severity :	Low
Confirmation :	**Confirmed**
Detection Accuracy :	
Vulnerable URL :	http://jornalismointerativo.com.br/noticia.php?sessao=17&noticia=29
Vulnerability Classifications:	-
Identified Field Name:	senha

Impact

Data entered in these fields will be cached by the browser. An attacker who can access the victim's browser could steal this information. This is especially important if the application is commonly used in shared computers such as cyber cafes or airport terminals.

Remedy

Add the attribute autocomplete="off" to the form tag or to individual "input" fields.

Actions to Take

1. See the remedy for the solution.
2. Find all instances of inputs which store private data and disable autocomplete. Fields which contain data such as "Credit Card" or "CCV" type data should not be cached. You can allow the application to cache usernames and remember passwords, however, in most cases this is not recommended.
3. Re-scan the application after addressing the identified issues to ensure that all of the fixes have been applied properly.

Required Skills for Successful Exploitation

Dumping all data from a browser can be fairly easy and there exist a number of automated tools to undertake this. Where the attacker cannot dump the data, he/she could still browse the recently visited websites and activate the auto-complete feature to see previously entered values.

External References

- Using AutoComplete in HTML Forms

Programming Error Message

Netsparker identified a programming error message.

Summary

Severity :	Low
Confirmation :	**Confirmed**
Detection Accuracy :	
Vulnerable URL :	http://jornalismointerativo.com.br/game/google-maps/index.php?erro=Enviado com sucesso
Vulnerability Classifications:	PCI 6.5.6 OWASP A6 CAPEC-118 CWE-200 209
Identified Error Message:	• <b>Warning</b>: Missing argument 2 for req(), called in /home2/jornalis/public_html/game/google-maps/cls_promocao.php on line 224 and defined in <b>/home2/jornalis/public_html/_classes/classe_geral.php</b> on line <b>3</b> • <b>Warning</b>: Missing argument 2 for req(), called in /home2/jornalis/public_html/game/google-maps/cls_promocao.php on line 227 and defined in <b>/home2/jornalis/public_html/_classes/classe_geral.php</b> on line <b>3</b>

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. type 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.

Database Error Message

Netsparker identified a database error message.

Summary

Severity :	Low
Confirmation :	**Confirmed**
Detection Accuracy :	
Vulnerable URL :	http://jornalismointerativo.com.br/admin_editar.php
Vulnerability Classifications:	PCI 6.5.6 OWASP A6 CAPEC-118 CWE-200 209

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. In rare conditions this may be a clue for an SQL Injection vulnerability. Most of the time Netsparker will detect and report that problem separately.

Remedy

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

File Upload Functionality Identified

This page allows users to upload files to the web server. Upload forms are generally dangerous unless they are coded with a great deal of care. This issue is **reported for information only**. If there is any other vulnerability identified regarding this resource Netsparker will report it as a separate issue.

Summary

Severity :	Information
Confirmation :	**Confirmed**
Detection Accuracy :	
Vulnerable URL :	http://jornalismointerativo.com.br/index.php?sessao=1
Vulnerability Classifications:	-
Form Name:	chamada_foto

E-mail Address Disclosure

Netsparker found e-mail addresses on the web site.

Summary

Severity :	Information
Confirmation :	**Confirmed**
Detection Accuracy :	
Vulnerable URL :	http://jornalismointerativo.com.br/
Vulnerability Classifications:	-
Found E-mails:	paulohinkel@gmail.com

Impact

E-mail addresses discovered within the application can be used by both spam email engines and also brute force tools. Furthermore valid email addresses may lead to social engineering attacks .

Remedy

Use generic email addresses such as contact@ or info@ for general communications, remove user/people specific e-mail addresses from the web site, should this be required use submission forms for this purpose.

External References

- Wikipedia - E-Mail Spam

Directory Listing (Apache)

Table of Content

The web server responded with a list of files located in the target directory.

Summary

Severity :	Information
Confirmation :	**Confirmed**
Detection Accuracy :	
Vulnerable URL :	http://jornalismointerativo.com.br/_includes/styles/
Vulnerability Classifications:	PCI 6.5.6 OWASP A6 CAPEC-127 CWE-548

Impact

An attacker can see the files located in the directory and could potentially access files which disclose sensitive information.

Actions to Take

1. See the remedy for solution.
2. Configure the web server to disallow directory listing requests.
3. This can also be caused the web server products that don't have latest security patches. Ensure that all of the patches have been applied.

Remedy

Change your httpd.conf file. A secure configuration for the requested directory should be similar to the following one:

```
<Directory /{YOUR DIRECTORY}>   Options FollowSymLinks   </Directory>
```

Remove the *Indexes* option from configuration. Do not forget to remove *MultiViews* as well.

External References

- WASC - Directory Indexing
- Apache Directory Listing Vulnerability