ETHICAL HACKING INTERNSHIP

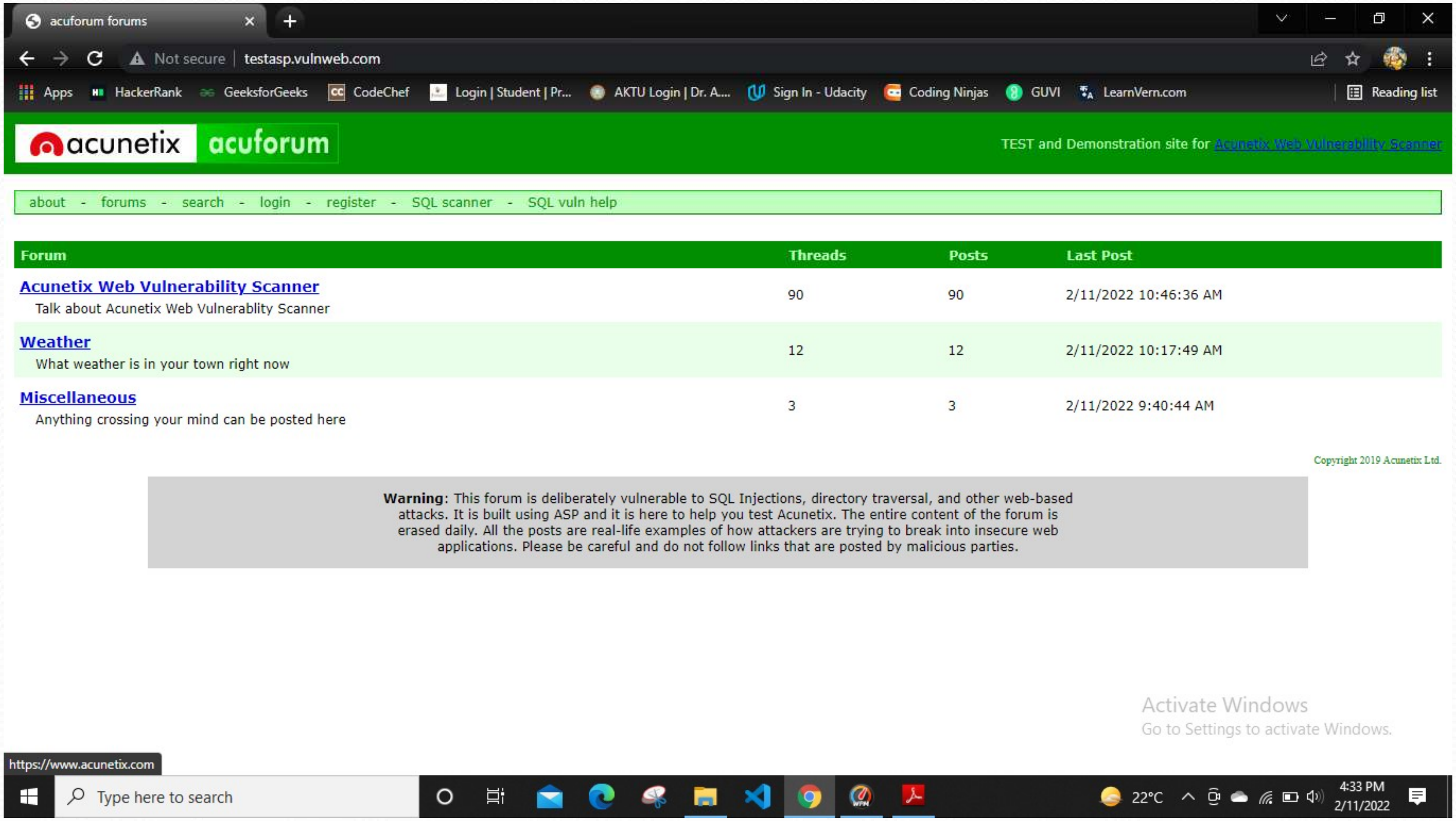
Project

ETHICAL HACKING INTERNSHIP

Submitted By :-

Govind Gupta

ETHICAL HACKING INTERNSHIP

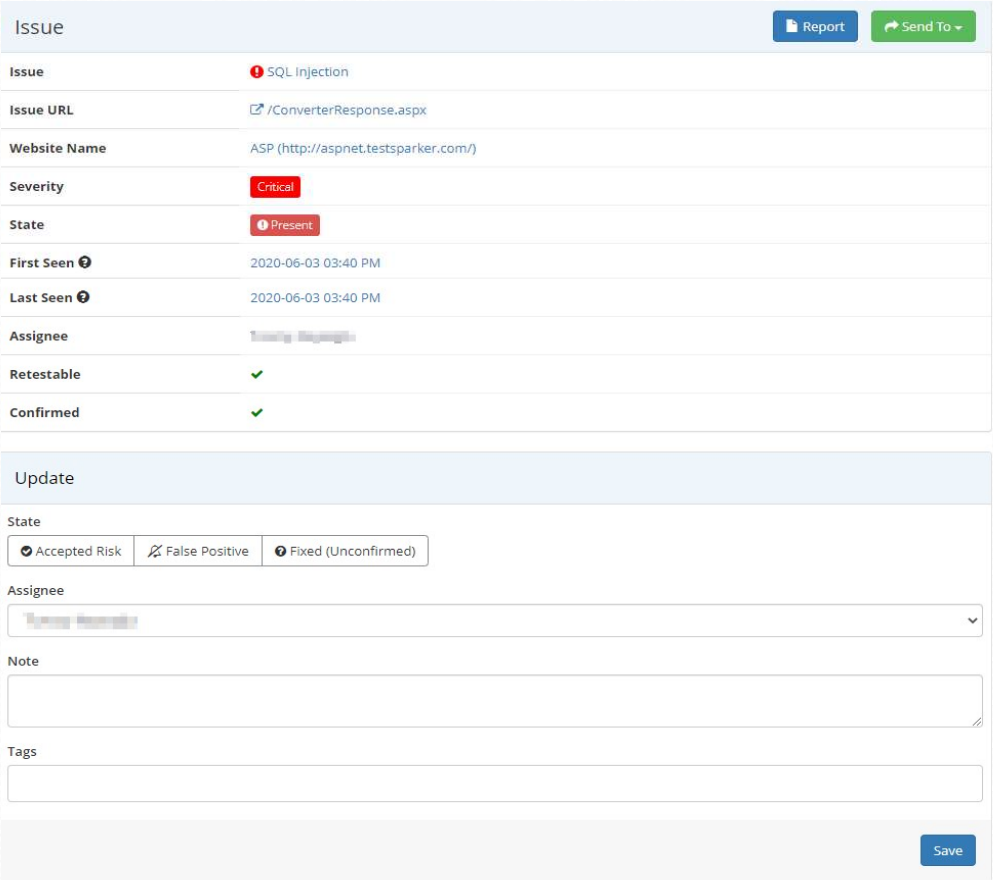


TEST and Demonstration site for Acunetix Web Vulnerablity Scanner

Forum	Threads	Posts	Last Post
Acunetix Web Vulnerability Scanner Talk about Acunetix Web Vulnerablity Scanner	90	90	2/11/2022 10:46:36 AM
Weather What weather is in your town right now	12	12	2/11/2022 10:17:49 AM
Miscellaneous Anything crossing your mind can be posted here	3	3	2/11/2022 9:40:44 AM

Warning: This forum is deliberately vulnerable to SQL Injections, directory traversal, and other web-based attacks. It is built using ASP and it is here to help you test Acunetix. The entire content of the forum is erased daily. All the posts are real-life examples of how attackers are trying to break into insecure web applications. Please be careful and do not follow links that are posted by malicious parties.

ETHICAL HACKING INTERNSHIP



Issue

Issue	SQL Injection
Issue URL	/ConverterResponse.aspx
Website Name	ASP (http://aspnet.testsparker.com/)
Severity	Critical
State	Present
First Seen	2020-06-03 03:40 PM
Last Seen	2020-06-03 03:40 PM
Assignee	[redacted]
Retestable	✓
Confirmed	✓

Update

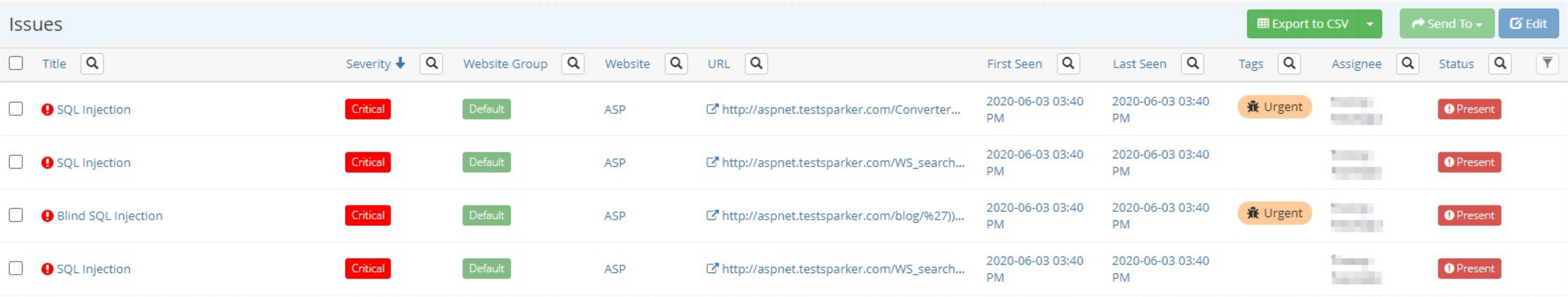
State

Assignee: [redacted]

Note

Tags

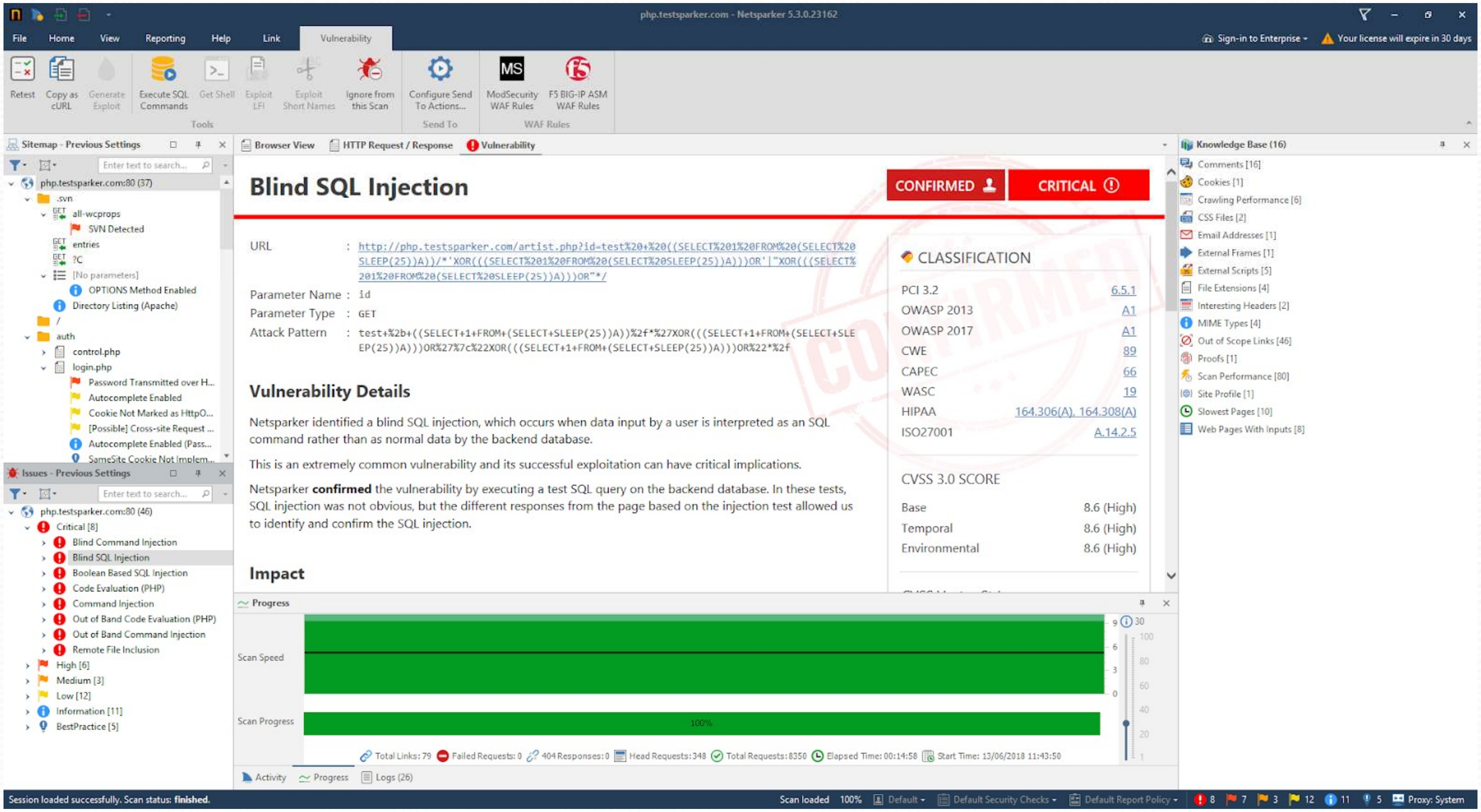
ETHICAL HACKING INTERNSHIP



Issues

□	Title	Severity	Website Group	Website	URL	First Seen	Last Seen	Tags	Assignee	Status
□	SQL Injection	Critical	Default	ASP	http://aspnet.testsparker.com/Converter...	2020-06-03 03:40 PM	2020-06-03 03:40 PM	Urgent	[redacted]	Present
□	SQL Injection	Critical	Default	ASP	http://aspnet.testsparker.com/WS_search...	2020-06-03 03:40 PM	2020-06-03 03:40 PM		[redacted]	Present
□	Blind SQL Injection	Critical	Default	ASP	http://aspnet.testsparker.com/blog/%27))...	2020-06-03 03:40 PM	2020-06-03 03:40 PM	Urgent	[redacted]	Present
□	SQL Injection	Critical	Default	ASP	http://aspnet.testsparker.com/WS_search...	2020-06-03 03:40 PM	2020-06-03 03:40 PM		[redacted]	Present

ETHICAL HACKING INTERNSHIP



Sitemap:

- php.testsparker.com:80 (37)
  - .svn
    - all-wcprops
      - SVN Detected
    - entries
    - ?C
    - [No parameters]
      - OPTIONS Method Enabled
    - Directory Listing (Apache)
  - /
  - auth
    - control.php
    - login.php
      - Password Transmitted over H...
      - Autocomplete Enabled
      - Cookie Not Marked as HttpO...
      - [Possible] Cross-site Request ...
      - Autocomplete Enabled (Pass...
      - SameSite Cookie Not Implem...

Issues:

- php.testsparker.com:80 (46)
  - Critical [8]
    - Blind Command Injection
    - Blind SQL Injection
    - Boolean Based SQL Injection
    - Code Evaluation (PHP)
    - Command Injection
    - Out of Band Code Evaluation (PHP)
    - Out of Band Command Injection
    - Remote File Inclusion
  - High [6]
  - Medium [3]
  - Low [12]
  - Information [11]
  - BestPractice [5]

Blind SQL Injection

CONFIRMED CRITICAL

URL : http://php.testsparker.com/artist.php?id=test%20+%20((SELECT%201%20FROM%20(SELECT%20SLEEP(25))A))/*'XOR(((SELECT%201%20FROM%20(SELECT%20SLEEP(25))A)))OR'|"XOR(((SELECT%201%20FROM%20(SELECT%20SLEEP(25))A)))OR"*/

Parameter Name : id

Parameter Type : GET

Attack Pattern : test+%2b+((SELECT+1+FROM+(SELECT+SLEEP(25))A))%2f*%27XOR(((SELECT+1+FROM+(SELECT+SLEEP(25))A)))OR%27%7c%22XOR(((SELECT+1+FROM+(SELECT+SLEEP(25))A)))OR%22*%2f

Vulnerability Details

Netsparker identified a blind SQL injection, which occurs when data input by a user is interpreted as an SQL command rather than as normal data by the backend database.

This is an extremely common vulnerability and its successful exploitation can have critical implications.

Netsparker **confirmed** the vulnerability by executing a test SQL query on the backend database. In these tests, SQL injection was not obvious, but the different responses from the page based on the injection test allowed us to identify and confirm the SQL injection.

Impact

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	8.6 (High)
Temporal	8.6 (High)
Environmental	8.6 (High)

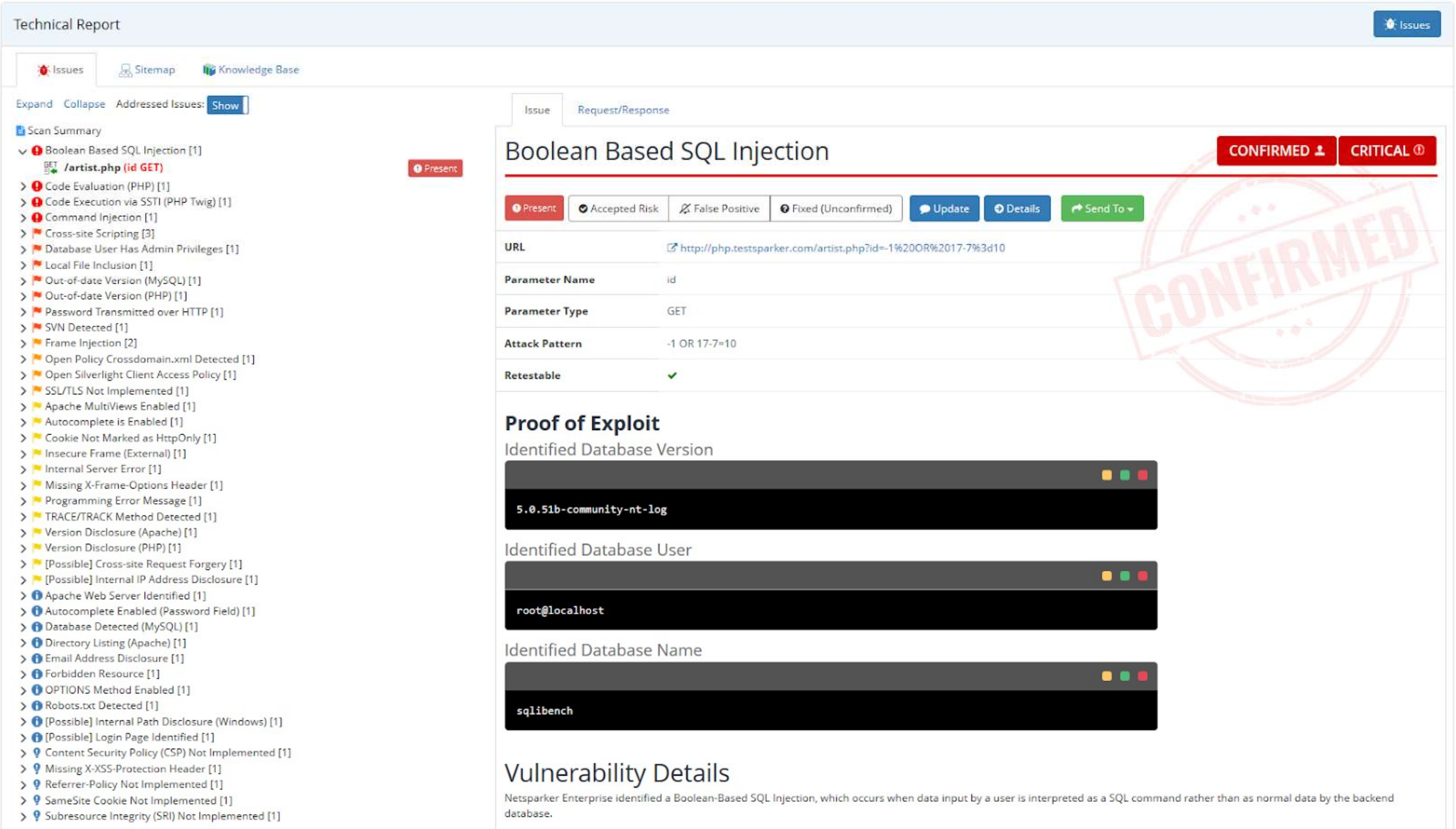
Knowledge Base (16):

- Comments [16]
- Cookies [1]
- Crawling Performance [6]
- CSS Files [2]
- Email Addresses [1]
- External Frames [1]
- External Scripts [5]
- File Extensions [4]
- Interesting Headers [2]
- MIME Types [4]
- Out of Scope Links [46]
- Proofs [1]
- Scan Performance [80]
- Site Profile [1]
- Slowest Pages [10]
- Web Pages With Inputs [8]

```
Total Links: 79   Failed Requests: 0   404 Responses: 0   Head Requests: 348   Total Requests: 8350   Elapsed Time: 00:14:58   Start Time: 13/06/2018 11:43:50
```

```
Session loaded successfully. Scan status: finished.     Scan loaded  100%  Default  Default Security Checks  Default Report Policy  8  7  3  12  11  5  Proxy: System
```

ETHICAL HACKING INTERNSHIP



Scan Summary

- Boolean Based SQL Injection [1]
  - /artist.php (id GET) Present
- Code Evaluation (PHP) [1]
- Code Execution via SSTI (PHP Twig) [1]
- Command Injection [1]
- Cross-site Scripting [3]
- Database User Has Admin Privileges [1]
- Local File Inclusion [1]
- Out-of-date Version (MySQL) [1]
- Out-of-date Version (PHP) [1]
- Password Transmitted over HTTP [1]
- SVN Detected [1]
- Frame Injection [2]
- Open Policy Crossdomain.xml Detected [1]
- Open Silverlight Client Access Policy [1]
- SSL/TLS Not Implemented [1]
- Apache MultiViews Enabled [1]
- Autocomplete is Enabled [1]
- Cookie Not Marked as HttpOnly [1]
- Insecure Frame (External) [1]
- Internal Server Error [1]
- Missing X-Frame-Options Header [1]
- Programming Error Message [1]
- TRACE/TRACK Method Detected [1]
- Version Disclosure (Apache) [1]
- Version Disclosure (PHP) [1]
- [Possible] Cross-site Request Forgery [1]
- [Possible] Internal IP Address Disclosure [1]
- Apache Web Server Identified [1]
- Autocomplete Enabled (Password Field) [1]
- Database Detected (MySQL) [1]
- Directory Listing (Apache) [1]
- Email Address Disclosure [1]
- Forbidden Resource [1]
- OPTIONS Method Enabled [1]
- Robots.txt Detected [1]
- [Possible] Internal Path Disclosure (Windows) [1]
- [Possible] Login Page Identified [1]
- Content Security Policy (CSP) Not Implemented [1]
- Missing X-XSS-Protection Header [1]
- Referrer-Policy Not Implemented [1]
- SameSite Cookie Not Implemented [1]
- Subresource Integrity (SRI) Not Implemented [1]

Boolean Based SQL Injection

CONFIRMED CRITICAL

URL	http://php.testsparker.com/artist.php?id=-1%20OR%2017-7%3d10
Parameter Name	id
Parameter Type	GET
Attack Pattern	-1 OR 17-7=10
Retestable	✓

CONFIRMED

Proof of Exploit

Identified Database Version

```
5.0.51b-community-nt-log
```

Identified Database User

```
root@localhost
```

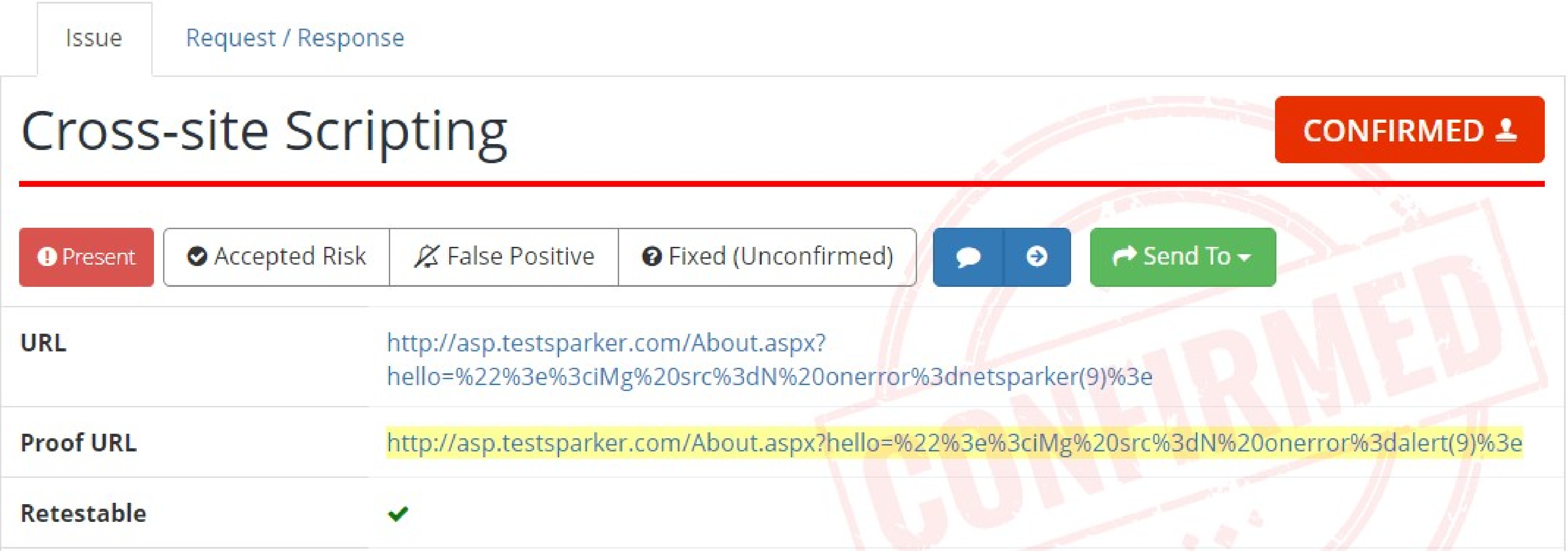
Identified Database Name

```
sqlibench
```

Vulnerability Details

Netsparker Enterprise identified a Boolean-Based SQL Injection, which occurs when data input by a user is interpreted as a SQL command rather than as normal data by the backend database.

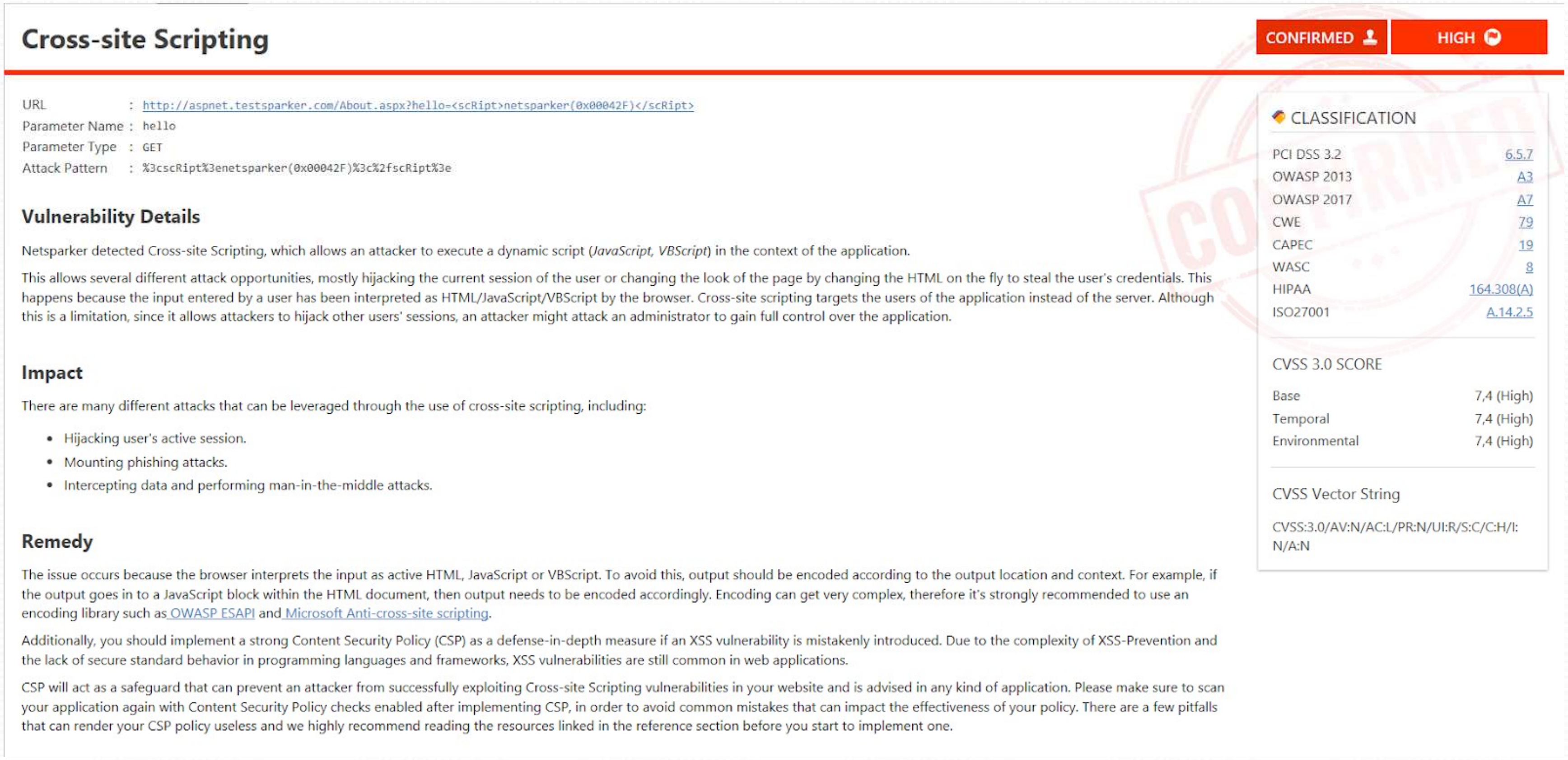
ETHICAL HACKING INTERNSHIP



Cross-site Scripting

URL	http://asp.testsparker.com/About.aspx? hello=%22%3e%3ciMg%20src%3dN%20onerror%3dnetsparker(9)%3e
Proof URL	http://asp.testsparker.com/About.aspx?hello=%22%3e%3ciMg%20src%3dN%20onerror%3dalert(9)%3e
Retestable	✓

ETHICAL HACKING INTERNSHIP



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

ETHICAL HACKING INTERNSHIP

END

.. Thank you ..