Vulnerability Analysis

*--Prepared by:- Samarth Dhol*

*--@InternshipSudio.com*

1)Vulnerability Report: Boolean-Based SQL Injection

Severity: Critical

**Summary:**

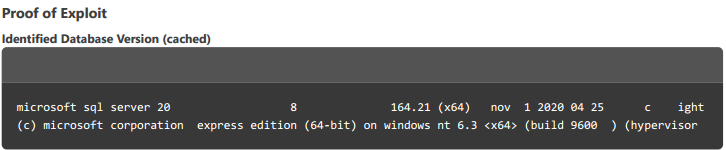
* Netsparker has detected a critical Boolean-Based SQL Injection vulnerability in the target website.
* This type of vulnerability arises when user input is treated as a SQL command by the back-end database, rather than as regular data.
* Successful exploitation of this vulnerability can have severe consequences, potentially allowing unauthorized access to, manipulation of, or execution of commands on the back-end database.

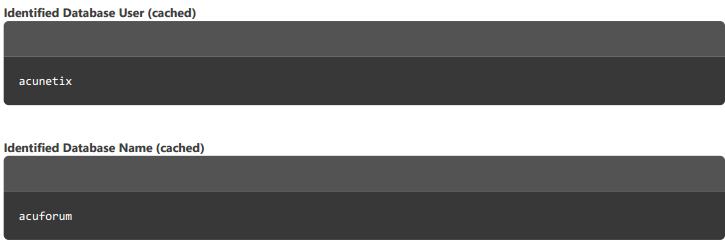
**Impact:**

* Depending on the back-end database, connection settings, and the operating system, an attacker could successfully execute the following types of attacks:

1. Reading, updating, and deleting arbitrary data or tables from the database.
2. Executing commands on the underlying operating system.

**Proof of Exploit:**





**Request:**



**Response:**



**Vulnerability Details:**

Affected URL:

`http://testasp.vulnweb.com/Login.asp?RetURL=%2FDefault.asp%3F`

Method: POST

Parameters:

-`tfUPass`: -1' OR 1=1 OR 'ns'='ns'

- `RetURL`: %2FDefault.asp%3F

- `tfUName`: Smith

**Actions to Take:**

1. Implement the recommended solutions provided in the 'Remedy' section.
2. Consider adopting a Database Access Layer (DAL) or Object-Relational Mapping (ORM) system to centralize and mitigate the issue.
3. Identify and convert all dynamically generated SQL queries to parameterized queries.
4. Review weblogs and application logs to identify any previous undetected attacks on the affected resource.

**Remedy:**

1. To safeguard your code against SQL injections, it is recommended to use parameterized queries.
2. Avoid creating dynamic SQL queries or queries with string concatenation whenever possible.

**Steps to Reproduce:**

1. Navigate to the login page: `http://testasp.vulnweb.com/Login.asp?RetURL=%2FDefault.asp%3F`
2. Input the following values in the respective fields:
3. Username (`tfUName`): Smith
4. Password (`tfUPass`): -1' OR 1=1 OR 'ns'='ns'
5. Submit the form.
6. Observe the page's response and analyze if there are any unexpected behaviors or data displayed.
7. Local File Inclusion

**Severity:** High

**Summary:**

* Netsparker detected an LFI vulnerability, wherein a file from the target system is injected into the attacked server page. This vulnerability was confirmed by reading files from the target web server.
* This report provides a comprehensive analysis of identified vulnerabilities in the target web application as discovered by Netsparker.
* The primary focus is on a confirmed Local File Inclusion (LFI) vulnerability, its potential impact, and recommended remediation steps.

**Vulnerability Details:**

Confirmed Instances: 1

Method: GET

Parameter:

id:-0 or 17-7=10

Affected URL:

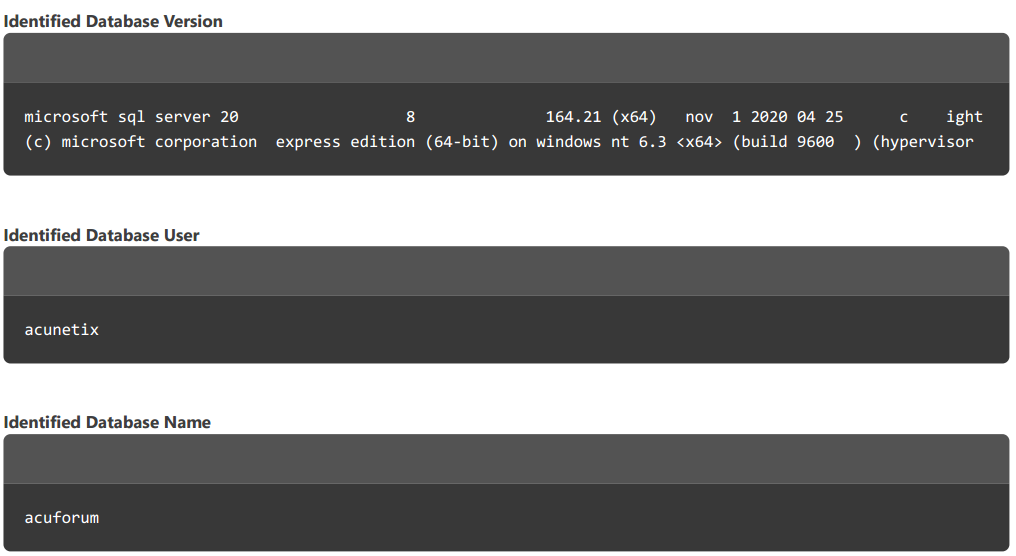
[http://testasp.vulnweb.com/Templatize.asp?item=%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2fwindows%2fwin.ini](http://testasp.vulnweb.com/Templatize.asp?item=%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2fwindows%2fwin.ini)

**Impact:**

* The severity of the impact depends on the exploitation and the read permissions of the web server user. Potential attacks include:

1. - Gathering usernames via an "/etc/passwd" file.
2. - Harvesting information from log files, such as "/apache/logs/error.log" or "/apache/logs/access.log."
3. - Remote execution of commands, possibly in conjunction with other attack vectors like file upload vulnerability or log injection.

**Proof of Exploit:**



**Request:**



**Response:**



**Steps to Reproduce:**

1. Access the vulnerable URL: [http://testasp.vulnweb.com/Templatize.asp?item=%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2fwindows%2fwin.ini](http://testasp.vulnweb.com/Templatize.asp?item=%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2f..%2fwindows%2fwin.ini)

2. Observe the proof of exploit confirming the presence of the vulnerability.

3. Note the details in the "Request" and "Response" sections for further analysis.

**Remedy:**

* Avoid permitting direct appending of file paths. Prefer hard-coded paths or select paths from a limited, hard-coded list using an index variable.
* If dynamic path concatenation is necessary, restrict accepted characters to "a-Z0-9" and disallow characters like ".." or "/" or "%00" (null byte) and other unexpected characters.
* Limit API inclusion to directories and directories below it to prevent directory traversal attacks.

1. Password Transmitted over HTTP

**Severity:** High

**Summary:**

This report outlines the findings from a security assessment conducted on the website [http://testasp.vulnweb.com](http://testasp.vulnweb.com) using Netsparker. The assessment revealed a critical vulnerability related to the transmission of passwords over HTTP, posing a significant risk to user credentials.

**Vulnerability Details:**

Password Transmitted over HTTP

Status: Confirmed

Instances: 1

Affected URL:[http://testasp.vulnweb.com/Register.asp](http://testasp.vulnweb.com/Register.asp)

HTTP Method: GET

Form Target Action: [http://testasp.vulnweb.com/Register.asp](http://testasp.vulnweb.com/Register.asp)

Form Name: frmRegister

**Request:**



**Response:**



**Impact:**

If an attacker can intercept network traffic, they can steal users' credentials, compromising the security and privacy of the affected users.

**Steps to Reproduce:**

1. Open a web browser and navigate to [http://testasp.vulnweb.com/Register.asp](http://testasp.vulnweb.com/Register.asp).
2. Inspect the network traffic using browser developer tools or a proxy tool.
3. Observe the transmission of password data over HTTP in the request/response.

**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.
* Move all critical forms and pages to HTTPS and do not serve them over HTTP.