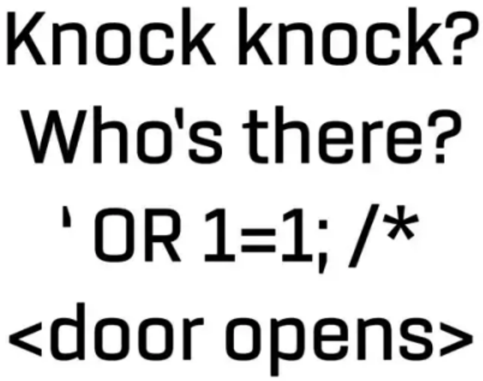
Hardik Solanki Follow
Dec 16, 2020 · 4 min read · Listen

Authentication Admin Panel Bypass-Which leads to full admin access control [CVE-2020–35276]

In this section, I will explain you, how i was able to bypass the admin login panel and from which it leads to full admin access control.

> *CVE ID:* CVE-2020–35276



Knock knock?
Who's there?
' OR 1=1; /*
<door opens>

> *What is SQL injection (SQLi)?*

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

In some situations, an attacker can escalate an SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack.

> *SQL Injection Type :*

1. **Error-based:** This type of SQL injection relies on the `error messages` being thrown by the database server, which might provide us some useful information regarding the database structure.
2. **Union-based:** This technique uses the SQL `UNION` operator to combine the results of two `SELECT` queries and return a single table. It allows an attacker to extract information from other tables by appending the results to the original query made to the database.
3. **Blind Injection:** This happens when the application is vulnerable to `SQL Injection` but the results of the `SQL query` are not returned in the `HTTP response` . In this case, we query the database for any true/false statement and see the changes for both true and false conditions. It is of two types:
4. **Content-based:** In this technique, the database server is queried with any conditional statement and the `response` from the server is analyzed for any difference while sending a `true` condition and a `false` condition.
5. **Time-based:** This technique relies on injecting an SQL query that makes the database wait for a specific time based on the specified condition. The time taken by the server to send back a response determines if the query is true/false.
6. **Out-of-band injection**(uncommon): This is not a very common type of `SQL Injection` as it depends on the features being enabled on the database server. It relies on the database server's capability to make a web request like `HTTP` , `DNS` , and `ftp` to send data to the attacker.

> *while doing my pentest research, I ended upon a project management software called EgavilanMedia.*

Product Details:

Product: EGM Address Book CPanel

Vendor: Egavilanmedia

Vendor URL: http://egavilanmedia.com

Component URL: http://demo.egavilanmedia.com/Address%20Book/login.php

Bug: SQLi- Authentication Admin Panel Bypass

Exploitable: Yes

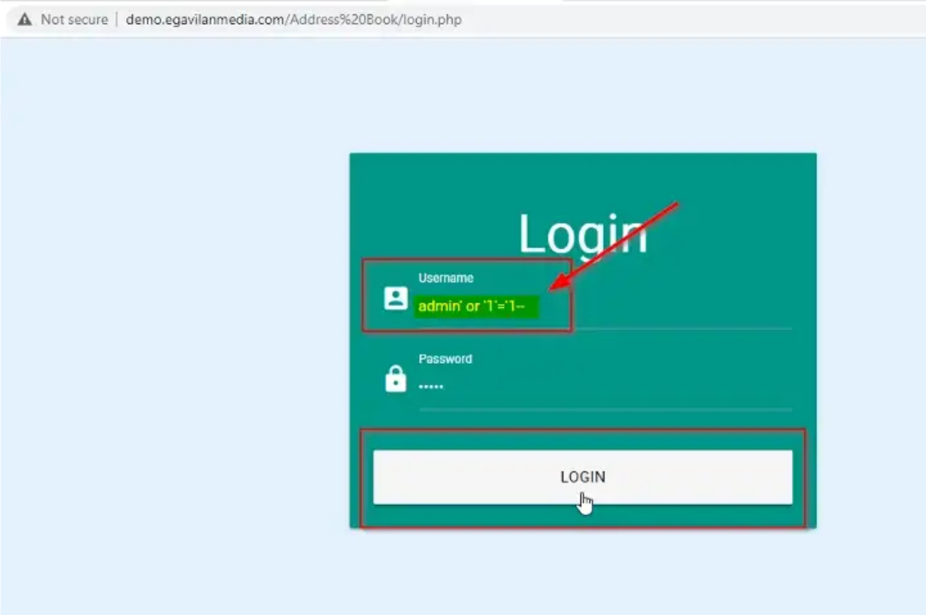
Impact of the vulnerability:

Attacker can Bypass admin Login panel from SQLi and get Full Admin access and attacker can add or remove any user.

Detailed Steps:

1. Open admin login page using the following URL: http://demo.egavilanmedia.com/Address%20Book/login.php

2. Now put Payload “**admin’ or ‘1’=’1**-- “ in the Username field and any random password. Then, click the Login button.



Not secure | demo.egavilanmedia.com/Address%20Book/login.php

Login

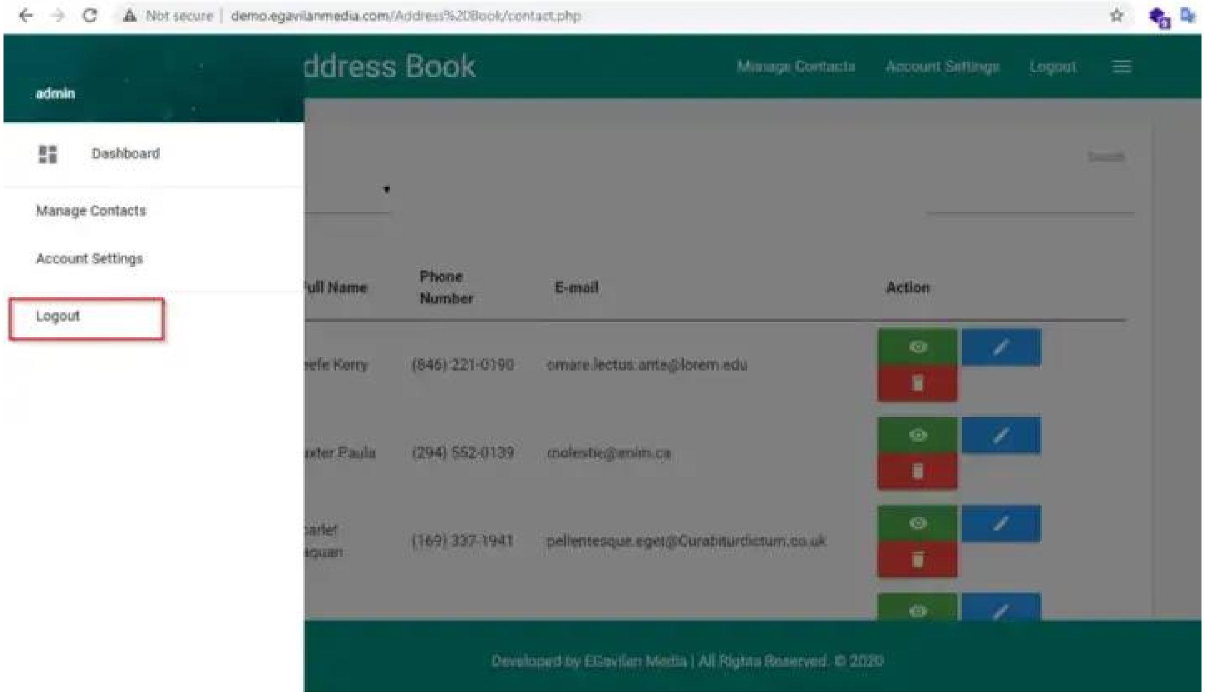
Username

admin' or '1'='1--

Password

......

3. Server accepted our payload and we are successfully able to bypass the panel without any credentials. And also be able to add or remove any users.



Not secure | demo.egavilanmedia.com/Address%20Book/contact.php

ull Name	Phone Number	E-mail	Action
eefe Kerry	(846) 221-0190	ornare.lectus.ante@lorem.edu	
xter Paula	(294) 552-0139	molestie@enim.ca	
arlet iquan	(169) 337-1941	pellentesque.eget@Curabiturdictum.co.uk	

Developed by EGavilan Media | All Rights Reserved. © 2020

> **How to protect your code from SQL Injection?**

1. Never construct a query directly with the user's input. Instead, use **Parameterized Statements**. They make sure that the inputs passed into SQL queries are treated safely.
2. It's always good the **sanitize** the user input. Also, proper **input validation** should be done for example, a name can't be digits or a phone number can't be alphabets. However, this can be bypassed at times.
3. Use a **safe driver** to interact with your SQL Database. They automatically prevent against all SQL Injection attacks. For example, SQLAlchemy for python.

> *Cheers!*
>
> *Happy Hunting!!!!!!!!!!!!*

Get the Medium app