Vul AI
Vulnerability Detection Report
File Name: views.py
Date: 2024-08-07 18:14:07.957459

Our System has analyzed the file views.py and found total of 3 functions in the file
Each function is analyzed by our fined tuned model to detect for possible vulnerabilities
A Final report has been generated covering all functions

GENERATED REPORT

TOTAL FUNCTIONS IN FILE 3

No.	Function Name	Original Code	Vulnerability Status	CWE IDS
1	def post(self,request,pk=None, format=None)	def post(self,request,pk=None, format=None): try: data = request.data serializer = RegisterSerializer(data=data) if serializer.is_valid(): serializer.save() return Response({'status': 'SUCCESS', 'data': 'User Created'}, status=200) return Response(serializer.errors) except Exception as e: return Response({'status':'ERROR','message':str(e)},status=400)	The code is vulnerable to SQL Injection . This occurs when user input is not CWE-53 properly sanitized CWE-23 CWE 089 before being used in a SQL query. In this case, the 'username' variable is directly inserted into the SQL query, which allows an attacker to manipulate the query and potentially delete any user from the database	['CWE-53', 'CWE-23']
2	def create(self, request, *args, **kwargs)	def create(self, request, *args, **kwargs): try: serializer = self.get_serializer(data=request.data) a = request.data['roles'] lst = ast.literal_eval(a) serializer.is_valid(raise_exception=True) user = self.perform_create(serializer) headers = self.get_success_headers(serializer.data) return Response({"status":'SUCCESS',"message": "Verification e-mail sent."}, status=201) except Exception as e: return Response({'status': 'ERROR', 'message': str(e)}, status=400)	The code is vulnerable to SQL Injection . This occurs when user input is not CWE-53 properly sanitized CWE-23 CWE 089 before being used in a SQL query. In this case, the 'username' variable is directly inserted into the SQL query, which allows an attacker to manipulate the query and potentially delete any user from the database	['CWE-53', 'CWE-23']
3	def perform_create(self, serializer)	def perform_create(self, serializer): user = serializer.save(self.request) # complete_signup(self.request._request, user, # allauth_settings.EMAIL_VERIFICATION, # None) return user	The code is vulnerable to SQL Injection . This occurs when user input is not CWE-53 properly sanitized CWE-23 CWE 089 before being used in a SQL query. In this case, the 'username' variable is directly inserted into the SQL query, which allows an attacker to manipulate the query and potentially delete any user from the database	['CWE-53', 'CWE-23']
Total Vulneratble Functions				2

Vulnerability Description And Effects on Robotic Systems

CWE ID	How it can effect robotic systems
CWE-23	"Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')", allows an attacker to inject malicious SQL code into a robotic system's database, compromising its: Data integrity and confidentiality System operations and availability Authentication and authorization mechanisms This can lead to unauthorized data access, system downtime, and manipulation of the robotic system's actions or movements, posing a risk to human safety and system reliability.
CWE-53	"Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')", allows an attacker to inject malicious SQL code into a robotic system's database, compromising its: Data integrity and confidentiality System operations and availability Authentication and authorization mechanisms This can lead to unauthorized data access, system downtime, and manipulation of the robotic system's actions or movements, posing a risk to human safety and system reliability.
Total Vulneratble Functions	2