Assignment : Develop, Exploit, and Patch a Vulnerable SCADA Interfaces

Pick one task and register it to the given Google sheet, sheet can be found below. You need 3 people.

Learning outcomes:

- **Design and implement a basic SCADA web interface**
- **Identify and intentionally introduce security vulnerabilities** in SCADA interfaces, demonstrating an understanding of how insecure coding practices manifest in real-world OT systems.
- **Analyze the attack surface of SCADA web applications**, including authentication mechanisms, data acquisition components, control endpoints, and third-party integrations.
- **Exploit common web and OT-specific vulnerabilities**
- **Demonstrate the operational impact of vulnerabilities** by showing how successful exploitation can affect process integrity, availability, and safety.
- **Apply penetration-testing tools and techniques** (e.g., Burp Suite, sqlmap, custom scripts) to systematically discover, validate, and exploit vulnerabilities in SCADA interfaces.
- **Assess the limitations of naive or incorrect security controls**, such as blacklist-based filtering, manual input escaping, or trust in third-party data sources.
- **Implement secure coding countermeasures** to remediate identified vulnerabilities, including proper input validation, parameterized queries, authentication hardening, CSRF protection, and secure file handling.
- **Patch and refactor vulnerable code** while preserving system functionality and operational requirements typical of OT environments.
- **Evaluate the effectiveness of applied security fixes** through re-testing and verification, demonstrating that vulnerabilities are no longer exploitable.
- **Document vulnerabilities, exploitation steps, and remediation actions** using a report.

Rules :

Each task can be only taken twice !
Your team must consist of 3 people.
Built a SCADA web application based on Scenario
Both vulnerable and patched versions are needed!
Report is needed explain your vulnerable parts and also patches
In your report, explanations must show the vulnerable parts also patched parts as well as how exploitation has been done. Pentesting tools must be used to demonstrate the vulnerabilities and patched versions.
Each SQL database should be populated with at least 100 records.
Also include population script and (if possible) database.
No Forced vulnerabilities: Make sure that vulnerability you are putting is logical, for instance OS command injection should not be introduced in the login page.

Each group must develop a monitoring system for both vulnerable and patched systems:
The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.
Each vulnerability must be on a separate page

What is needed in the submission:

1) Report
2) Vulnerable source codes and requirements
3) Patched version source codes and requirements
4) Dockerized version of vulnerable and patched version
5) Video explaining your application vulnerable version
    a) How these vulnerabilities are exploited
    b) How these vulnerabilities are patched
    c) How these vulnerabilities are tested (Which tools and demonstration is needed!)

Do not forget there will be a demonstration session (zoom meeting) and all group members must be present!

..
Task 1 :

Scenario: **City Water Reservoir Level Management** System: Municipal water reservoir SCADA Interface shows: Reservoir level (m^3) Inflow/outflow rates Pump status (ON/OFF) Valve positions (%) Logs generated: Level changes every 5 minutes Pump start/stop events Emergency overflow alerts Capabilities: Start/stop intake pumps Set max/min water levels Enable automatic overflow protection
Vulnerabilities : **LDAP injection (2 vulnerabilities)** -> After authentication while gathering usernames from IDAP server (1st vulnerability) LDAP server must be populated with usernames, surname, and department there should be at least 100 random records. Random record generation algorithm must be also provided in the submission, explanation must be included in the report and video -> Another LDAP injection (2nd another vulnerability different input place- 2 insufficient protection mechanism at the same time The developer tried to add security to protect against LDAP vulnerability but fails and the

attacker can still abuse the server.
Partial Escaping of Special Characters must be one of the methods
SQL İnjection:
SQL injection due to Blacklist-Based Keyword Filtering (At least 35 very relevant keywords including SELECT, UNION, OR)
Explain each keywords why relevant
Find a real attack!
Authentication system without CAPTCHA and rate limiting
Must be vulnerable to dictionary attacks

Patched version : Please also create a separated patched version and make sure to explain all the patches. Patching should involve LDAPS. LDAPS (Lightweight Directory Access Protocol over SSL/TLS) is the secure version of LDAP, used to authenticate users, authorize access, and query directory services

Monitoring system:

The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.

Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions.
Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 2 :

Scenario:
Water Distribution Pump Station Control
System: District pump station
Interface shows:
Pump RPM
Line pressure (bar)
Power consumption (kW)
Logs generated:
Pressure fluctuation logs
Pump fault logs
Power spike alerts
Capabilities:
Change pump speed
Switch between manual/automatic modes
Acknowledge fault alarms

Vulnerabilities : XXE injection (2 vulnerabilities)
1: XXE in SVG Uploads (XML External Entity (XXE) vulnerabilities hidden inside SVG image uploads)
-A commonly overlooked real-world attack vector. Students will analyze and exploit insecure SVG handling.

-XXE in SVG Uploads refers to a class of vulnerabilities where malicious SVG files exploit XML External Entity (XXE) processing during file upload, parsing, or rendering.
-Your vulnerable application will accept .png, .jpg, and .svg.
Server validates uploaded images, generates thumbnails server-side and stores metadata in a database. The development team believes SVG files are safe because: “They are just images and browsers already support them.”
Your task is to prove otherwise.

2: Classic XXE to steal information from the files
This will be a separate vulnerability in another page

Admin Authentication system without CAPTCHA and rate limiting
Must be vulnerable to dictionary attacks

SQL injection vulnerability
User Authentication system vulnerable WAF With Generic SQLi Rules
Web Application Firewall (WAF) with generic SQLi rules is a security control that attempts to detect and block SQL injection attacks using **predefined, pattern-based signatures** rather than application-specific logic or semantic query understanding.

Patched version : Please also create a separated patched version and make sure to explain all the patches.

Monitoring system:

The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.

Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions.
Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 3:

Scenario: Drinking Water Quality Monitoring
System: Water treatment plant
Interface shows:
pH level
Turbidity (NTU)
Chlorine concentration (ppm)
Logs generated:
Sensor readings every minute
Out-of-range alerts
Calibration history
Capabilities:
Trigger manual sampling
Adjust chlorine injection rate

Disable faulty sensors

Vulnerabilities :
Server-Side Request Forgery (SSRF) (4 vulnerabilities)
1:SSRF via PDF Generation Engines (first vulnerability)
HTML-to-PDF engines fetch external CSS, fonts, or images.
Vulnerability happens during document rendering, not request handling.
2nd & 3rd: Server-Side Request Forgery (SSRF) (Can be both used to query internal server and online resources, could be used for port scan) (different pages different functions)
4: Classic CWE-434 Unrestricted Upload of File with Dangerous Type with lack of protection
SQL injection vulnerability:
SQL injection Due to Inconsistent Input Sanitization (Same input place)
-Only some parameters sanitized (GET but not POST, or vice versa)
-SQLi exists but only under specific request types

Admin Authentication system without CAPTCHA and rate limiting
Must be vulnerable to dictionary attacks

Also include IDOR vulnerability!

Patched version : Please also create a separated patched version and make sure to explain all the patches.

Monitoring system:

The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.

Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions.
Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 4 :

Scenario: Chlorination Injection Control Panel
System: Chlorine dosing subsystem
Interface shows:
- Chlorine tank level
- Injection rate
- Safety interlock status

Logs generated:
- Injection changes
- Tank refill events
- Safety lock triggers

Capabilities:
- Increase/decrease dosing

- Enable emergency shutdown
- Switch to backup tank

Vulnerabilities :
3 vulnerabilities on different pages:
1: Missing CSRF on a POST form
Cross-Site Request Forgery (CSRF) is an attack where:
A victim is already authenticated to a target application (via cookies, session, NTLM, etc.)
The attacker tricks the victim's browser into sending a state-changing request
The request is accepted as legitimate because authentication credentials are automatically attached
The application exposes a POST endpoint that performs a state-changing action
The request does not include a CSRF protection mechanism
The server does not verify that the request originated from:
The same site
A legitimate user interaction
POST alone does NOT protect against CSRF

2: Classic Server-side Request Forgery via Template Injection in Reporting Engines
Injection into report generators (PDF/CSV/HTML) used for OT analytics or compliance reporting.
Reports should be gathered from online sources and capable of causing a malfunction.
malfunction should be visible for demonstration purposes

3: File Path Injection (Directory Traversal)
Injecting paths into log export, backup, or firmware-restore functionality.
SQL Injection:
SQL injection Due to manual escaping Instead of Parameterization
This vulnerability occurs when developers try to "sanitize" user input by escaping characters manually (e.g., replacing ' with \') instead of using parameterized queries (prepared statements).
It is a classic secure-coding anti-pattern and still very common in real systems.

Patched version : Please also create a separated patched version and make sure to explain all the patches.

Monitoring system:

The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.

Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions.
Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 5 :

Scenario: Remote Valve Management System
System: Field valve network
Interface shows:
Valve open/close percentage
Last command timestamp
Communication status
Logs generated:
Command execution logs
Failed valve response logs
Communication timeout logs
Capabilities:
Open/close valves
Schedule valve operations
Force re-synchronization

Vulnerabilities : CWE-434 Unrestricted Upload of File with Dangerous Type
->1st Classic CWE-434 with lack of protection
->2nd CWE- 434 with lack of protection (2 insufficient protection mechanism at the same time)
There must be 2 protection mechanisms which can be bypassed by the attacks
One of this must be size of the file
->3rd File scanning pipeline bypass via encryption
Server scans only plaintext; attacker uploads encrypted blob that's decrypted later.
Student should be able demonstrate all this

SQL injection :
SQL injection Conditional Escaping Based on User Role (Same input area)
-Admin input not escaped, user input is
-SQLi only exploitable with higher privileges
-Missed during low-privilege testing

Patched version : Please also create a separated patched version and make sure to explain all the patches.

Monitoring system:

The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.

Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions.
Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 6 :

Scenario: Water Demand Forecast Dashboard
System: Consumption analytics
Interface shows:
Hourly demand graph
Predicted peak usage
Historical comparison
Logs generated:
Prediction updates
Model confidence scores
Manual overrides
Capabilities:
Trigger pump pre-activation
Adjust safety margins
Export demand reports

Vulnerabilities : LDAP injection
-> After authentication while gathering usernames from IDAP server (1st vulnerability)
LDAP server must be populated with usernames, surname, and department
there should be at least 100 random records.
Random record generation algorithm must be also provided in the submission, explanation must be included in the report and video
-> Another LDAP injection (2nd another vulnerability different input place- 2 insufficient protection mechanism at the same time
The developer tried to add security to protect against LDAP vulnerability but fails and the attacker can still abuse the server.
Partial Escaping of Special Characters must be one of the methods
SQL injection:
SQL injection due to Input Validation Only on Client Side
-Direct HTTP requests bypass validation completely, manual testing may miss this if relying on browser behavior
Authentication system without CAPTCHA and rate limiting
Must be vulnerable to dictionary attacks

Patched version : Please also create a separated patched version and make sure to explain all the patches. Patching should involve LDAPS. LDAPS (Lightweight Directory Access Protocol over SSL/TLS) is the secure version of LDAP, used to authenticate users, authorize access, and query directory services

Monitoring system:

The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.

Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions.
Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 7 :

Scenario: SCADA Alarm Management Console System: Central alarm handling Interface shows: Active alarms Alarm severity Time since triggered Logs generated: Alarm creation Acknowledgement events Escalation actions Capabilities: Acknowledge alarms Silence alarms temporarily Escalate to supervisor
Vulnerabilities : **Server-Side Request Forgery (SSRF)** -4 different functions -2 of them for internal scans (create OT services, make sure that your application can find them using this vulnerability) -2 of them for external scans **Authentication system without CAPTCHA and rate limiting** Must be vulnerable to dictionary attacks **SQL injection** SQL injection triggered Only After State Change -Injection works only after specific workflow steps -Requires correct order of actions -Automated scanners often miss this
Patched version : Please also create a separated patched version and make sure to explain all the patches.
Monitoring system: The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.
Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions. Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 8 :

Scenario:

Vulnerabilities :
XXE injection (4 vulnerabilities)
1: XXE & Remote code execution vulnerability
The server parses the XML with an unsafe XML parser → XXE can read local files or hit internal endpoints.
The attacker uses that to obtain secrets used by the app in the same workflow (DB creds, service tokens, signing keys, CI deploy token).
With those secrets, the same request flow (or immediate next call) hits an admin endpoint / deploy hook / template engine / job runner → RCE.

2: Classic XXE to cause DOS attack
This will be a separate vulnerability in another page

3: Classic XXE steal information from the files
This will be a separate vulnerability in another page

4: Merge Server-Side Request Forgery (SSRF) with XXE
An OT monitoring platform collects sensor status reports from field gateways and generates health dashboards.
Gateways upload XML files that describe:
Device metadata
External schema references
Optional enrichment data fetched from URLs
The backend:
Parses XML using a vulnerable XML parser
Resolves external entities
Fetches remote resources referenced in XML

SQL injection:
SQL injection due to rate limiting and CAPTCHA issues (Both has to be implemented for the same input field)
No proper rate limiting
İnsecure CAPTCHA is needed!

Patched version : Please also create a separated patched version and make sure to explain all the patches.

Monitoring system:

The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.

Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions.
Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 9 :

Scenario: **SCADA Alarm Management Console**
System: Central alarm handling
Interface shows:
Active alarms
Alarm severity
Time since triggered
Logs generated:
Alarm creation
Acknowledgement events
Escalation actions
Capabilities:
Acknowledge alarms
Silence alarms temporarily
Escalate to supervisor

Vulnerabilities :
3 vulnerabilities on different pages:
1: Missing CSRF on a POST form
Cross-Site Request Forgery (CSRF) is an attack where:
A victim is already authenticated to a target application (via cookies, session, NTLM, etc.)
The attacker tricks the victim's browser into sending a state-changing request
The request is accepted as legitimate because authentication credentials are automatically attached
The application exposes a POST endpoint that performs a state-changing action
The request does not include a CSRF protection mechanism
The server does not verify that the request originated from:
The same site
A legitimate user interaction
POST alone does NOT protect against CSRF

2: Server-side Request Forgery via Template Injection in Reporting Engines
Injection into report generators (PDF/CSV/HTML) used for OT analytics or compliance reporting.
Reports should be gathered from online sources and capable of causing a malfunction.
malfunction should be visible for demonstration purposes

3: File Path Injection (Directory Traversal)
Injecting paths into log export, backup, or firmware-restore functionality.

SQL injection:
SQL injection Only Works With Specific Encodings
This vulnerability appears when SQL injection is possible only after a particular character encoding, decoding order, or charset mismatch, meaning the attack fails with normal ASCII input but succeeds when a specific encoding is used (e.g., URL-encoding, Unicode, double-encoding, multibyte charsets).

-UTF-16, UTF-7, or multibyte encodings -Filters break ASCII payloads -Rare but dangerous
Patched version : Please also create a separated patched version and make sure to explain all the patches.
Monitoring system: The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.
Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions. Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 10 :

Scenario: SCADA Maintenance Mode Interface System: Maintenance operations Interface shows: Devices in maintenance Lockout/tagout status Technician assignments Logs generated: Maintenance start/end Device isolation logs Technician access logs Capabilities: Place device in maintenance Release maintenance lock Assign technicians
Vulnerabilities : 3 vulnerabilities on different pages: 1:CWE-1395 Dependency on Vulnerable Third-Party Component The product has a dependency on a third-party component that contains one or more known vulnerabilities. Many products are large enough or complex enough that part of their functionality uses libraries, modules, or other intellectual property developed by third parties who are not the product creator. For example, even an entire operating system might be from a third-party supplier in some hardware products. Whether open or closed source, these components may contain publicly known vulnerabilities that could be exploited by adversaries to compromise the product. Reference: https://cwe.mitre.org/data/definitions/1395.html 2: CWE-1329: Reliance on Component That is Not Updateable The product contains a component that cannot be updated or patched in order to remove

vulnerabilities or significant bugs. No cryptography related vulnerabilities allowed here. Explain why it is not Updateable.
If the component is discovered to contain a vulnerability or critical bug, but the issue cannot be fixed using an update or patch, then the product's owner will not be able to protect against the issue. The only option might be replacement of the product, which could be too financially or operationally expensive for the product owner. As a result, the inability to patch or update can leave the product open to attacker exploitation or critical operation failures. This weakness can be especially difficult to manage when using ROM, firmware, or similar components that traditionally have had limited or no update capabilities.
Finding that component is very important!
3: Classic CWE-434 Unrestricted Upload of File with Dangerous Type vulnerability with lack of protection

SQL injection:
SQL injection due to trust to third party compromised API for data gathering
Both API and main vulnerable server has to be implemented
This vulnerability occurs when an application blindly trusts data received from an external API (partner service, SaaS, sensor feed, threat-intel API, ERP, CRM, OT data collector, etc.) and uses that data to build SQL queries dynamically, assuming it is *safe because it comes from a "trusted" source*.

If the third-party API is compromised, malicious, or spoofed, it becomes an indirect SQL injection vector.

Patched version : Please also create a separated patched version and make sure to explain all the patches.

Monitoring system:

The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.

Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions.
Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 11 :

Scenario: SCADA Maintenance Mode Interface
System: Maintenance operations
Interface shows:
Devices in maintenance
Lockout/tagout status
Technician assignments
Logs generated:

Maintenance start/end
Device isolation logs
Technician access logs
Capabilities:
Place device in maintenance
Release maintenance lock
Assign technicians

Vulnerabilities :
3 vulnerabilities on different pages:
1:CWE-1395 Dependency on Vulnerable Third-Party Component
The product has a dependency on a third-party component that contains one or more known vulnerabilities.
Many products are large enough or complex enough that part of their functionality uses libraries, modules, or other intellectual property developed by third parties who are not the product creator. For example, even an entire operating system might be from a third-party supplier in some hardware products. Whether open or closed source, these components may contain publicly known vulnerabilities that could be exploited by adversaries to compromise the product.
Reference: https://cwe.mitre.org/data/definitions/1395.html
2: CWE-1329: Reliance on Component That is Not Updateable
The product contains a component that cannot be updated or patched in order to remove vulnerabilities or significant bugs. No cryptography related vulnerabilities allowed here. Explain why it is not Updateable.
If the component is discovered to contain a vulnerability or critical bug, but the issue cannot be fixed using an update or patch, then the product's owner will not be able to protect against the issue. The only option might be replacement of the product, which could be too financially or operationally expensive for the product owner. As a result, the inability to patch or update can leave the product open to attacker exploitation or critical operation failures. This weakness can be especially difficult to manage when using ROM, firmware, or similar components that traditionally have had limited or no update capabilities.
Finding that component is very important!
3: Classic CWE-434 Unrestricted Upload of File with Dangerous Type vulnerability with lack of protection to log system

SQL injection:

SQL injection due to Blacklist-Based Keyword Filtering (At least 25 very relevant keywords including SELECT, UNION, OR)
*This vulnerability occurs when developers attempt to prevent SQL injection by blocking or removing specific SQL keywords (a *blacklist*), instead of using parameterized queries.
Even when many keywords are filtered, SQL injection remains possible because SQL is flexible, context-dependent, and DB-specific.

Patched version : Please also create a separated patched version and make sure to explain all the patches.

Monitoring system:

The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.

Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions.
Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 12 :

Scenario: SCADA Maintenance Mode Interface
System: Maintenance operations
Interface shows:
Devices in maintenance
Lockout/tagout status
Technician assignments
Logs generated:
Maintenance start/end
Device isolation logs
Technician access logs
Capabilities:
Place device in maintenance
Release maintenance lock
Assign technicians

Vulnerabilities :
Vulnerable OT Remote Diagnostics Portal (7 vulnerabilities)
A) IDOR: Diagnostic reports downloadable by guessing IDs.
B) Deserialization bug: Diagnostic result objects deserialized; corrupted payload crashes rendering engine.
C) File overwrite: Uploaded diagnostic scripts overwrite existing ones.
D)Unsafe temp files: Temporary PDF reports are created with predictable names and reused across users.
E) Classic CWE-434 Unrestricted Upload of File with Dangerous Type with lack of protection
F) Classic Server-Side Request Forgery (SSRF) vulnerability with remote access
G)Classic XXE injection to steal data

SQL injection:
Critical SQL Injection Vulnerability in Django (CVE-2025-64459)
Scenario 1: Authentication Bypass
Application accepts login credentials via dictionary expansion. Attacker adds _connector=OR&is_superuser=True to gain admin access without valid credentials.
Scenario 2: Data Exfiltration

Document access API filters by owner. Attacker adds _connector=OR&confidential=True to access all confidential documents regardless of ownership.
Patched version : Please also create a separated patched version and make sure to explain all the patches.
Monitoring system: The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.
Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions. Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Task 13 :

Scenario:
Vulnerabilities : **Vulnerable OT Remote Diagnostics Portal (7 vulnerabilities)** A) IDOR: Diagnostic reports downloadable by guessing IDs. B) Deserialization bug: Diagnostic result objects deserialized; corrupted payload crashes rendering engine. C) File overwrite: Uploaded diagnostic scripts overwrite existing ones. D)Unsafe temp files: Temporary PDF reports are created with predictable names and reused across users. E)Classic CWE-434 Unrestricted Upload of File with Dangerous Type with lack of protection F)Classic Server-Side Request Forgery (SSRF) vulnerability with remote access (Different function) G)Classic Server-Side Request Forgery (SSRF) vulnerability with internal access (Different function) **SQL injection:** Critical SQL Injection Vulnerability in Django (CVE-2025-64459) Scenario 1: Authentication Bypass Application accepts login credentials via dictionary expansion. Attacker adds _connector=OR&is_superuser=True to gain admin access without valid credentials. Scenario 2: Data Exfiltration Document access API filters by owner. Attacker adds _connector=OR&confidential=True to access all confidential documents regardless of ownership.
Patched version : Please also create a separated patched version and make sure to explain all the patches.
Monitoring system:

The system should be able monitor each vulnerability, abuse,attack and access point. This page should be able to display correct details of the request / attacks and present classification of the attacks.
Testing : Pentesting tools must be used to demonstrate the vulnerabilities and patched versions. Some examples: Burp Suite, OWASP ZAP, SQLmap, XSStrike, Postman, Nuclei, Nikto

Please register your task and group members here:

Use your Sabancı University Account please, otherwise you won't be able to see!

2025 CS 437 Assignment 1 Groups 2025

USE Sabancı University Account to get the link