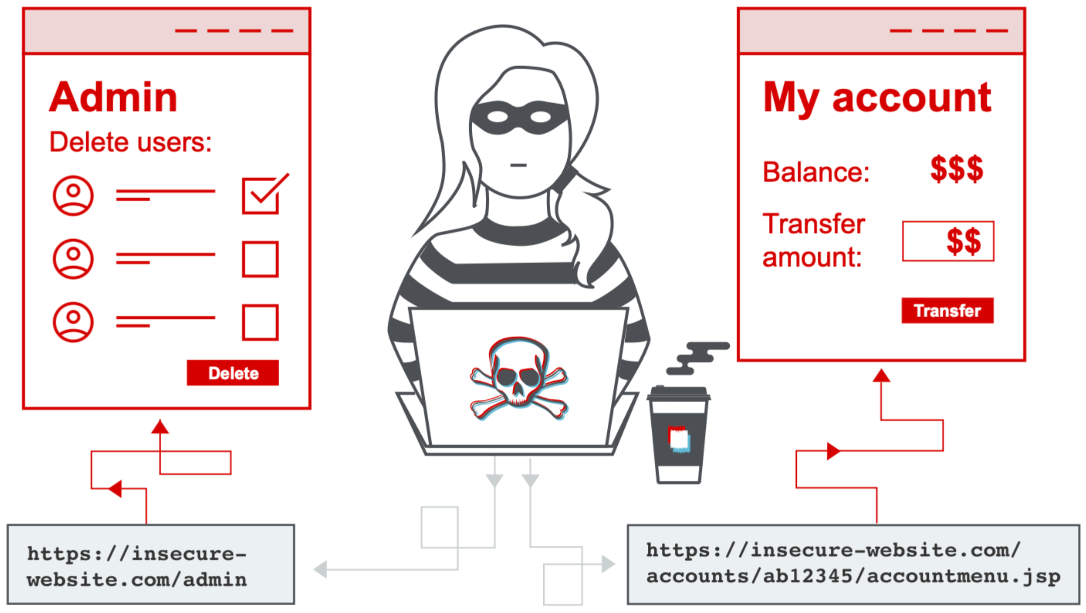
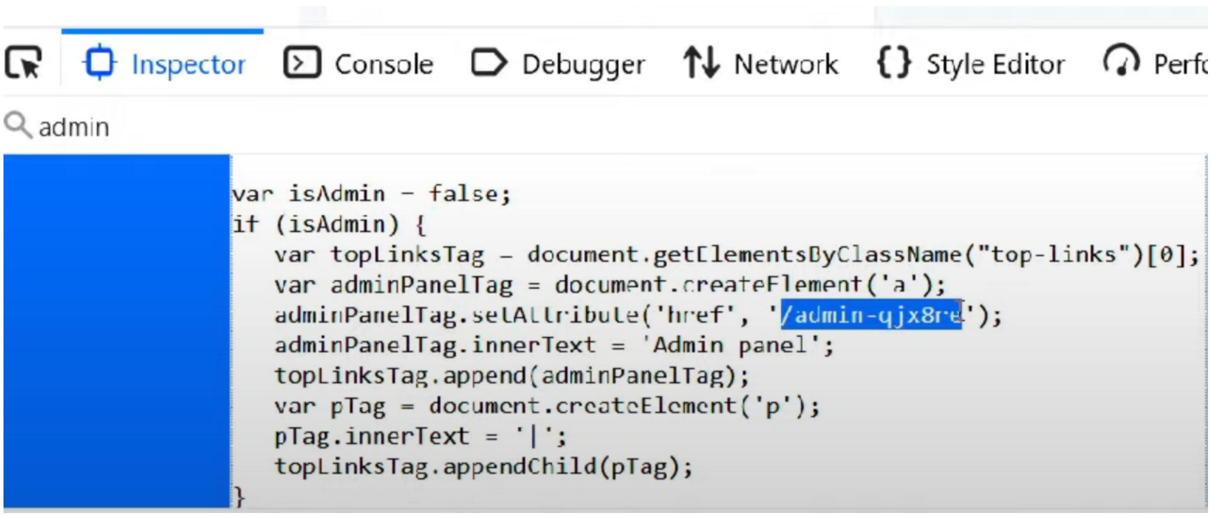
Siddharth Sharma
Assingement-1

Broken access controls: are a commonly encountered and often critical security vulnerability. Design and management of access controls is a complex and dynamic problem that applies business, organizational, and legal constraints to a technical implementation. Access control design decisions have to be made by humans, not technology, and the potential for errors is high.

From a user perspective, access controls can be divided into the following categories:

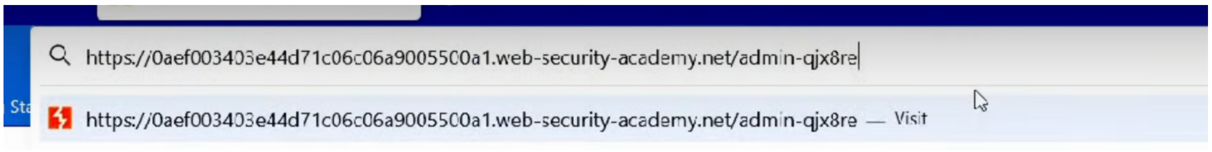
- Vertical access controls
- Horizontal access controls
- Context-dependent access controls


Admin
Delete users:
Delete
https://insecure-website.com/admin
My account
Balance: $$$
Transfer amount: $$
Transfer
https://insecure-website.com/accounts/ab12345/accountmenu.jsp



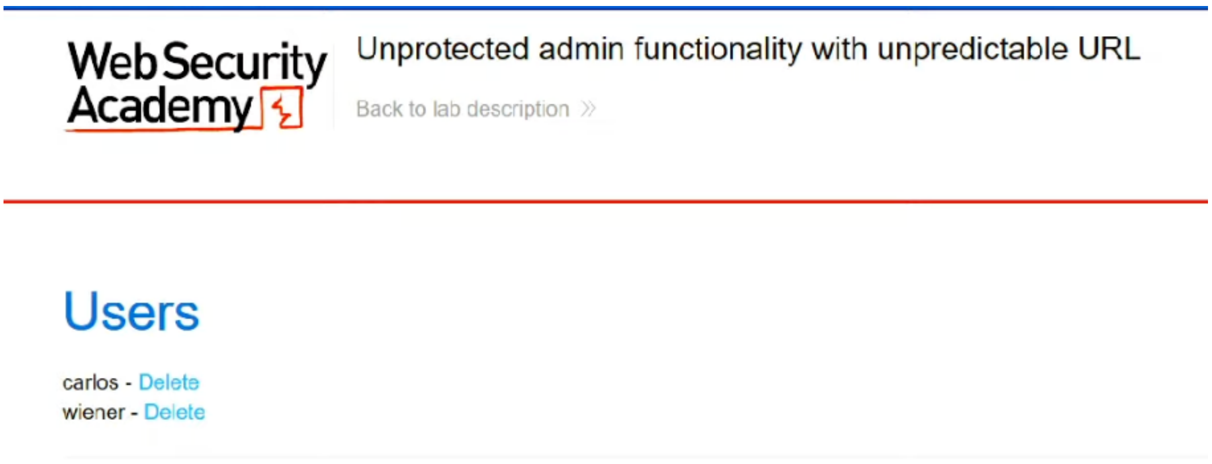
admin

```
var isAdmin = false;
if (isAdmin) {
   var topLinksTag = document.getElementsByClassName("top-links")[0];
   var adminPanelTag = document.createElement('a');
   adminPanelTag.setAttribute('href', '/admin-qjx8re');
   adminPanelTag.innerText = 'Admin panel';
   topLinksTag.append(adminPanelTag);
   var pTag = document.createElement('p');
   pTag.innerText = '|';
   topLinksTag.appendChild(pTag);
}
```



https://0aef003403e44d71c06c06a9005500a1.web-security-academy.net/admin-qjx8re

https://0aef003403e44d71c06c06a9005500a1.web-security-academy.net/admin-qjx8re — Visit



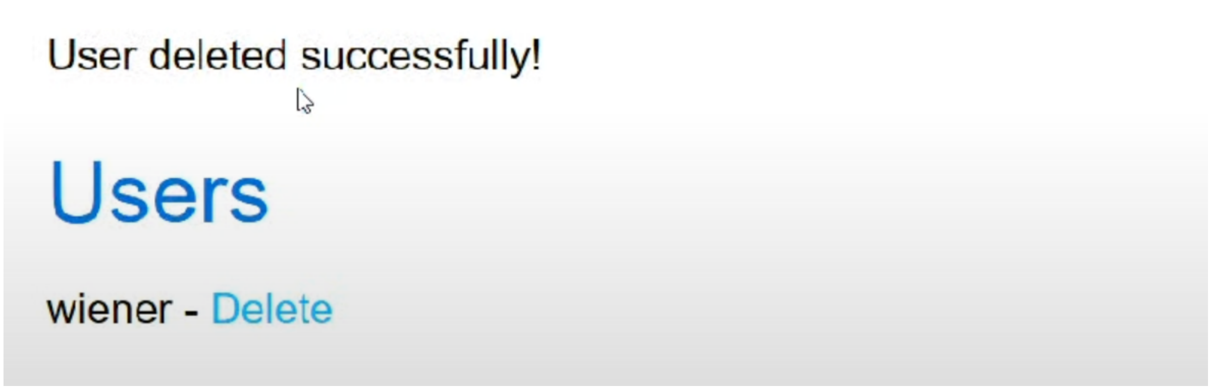
Unprotected admin functionality with unpredictable URL

Back to lab description »

Users

carlos - Delete
wiener - Delete

Congratulations, you solved the lab!


User deleted successfully!

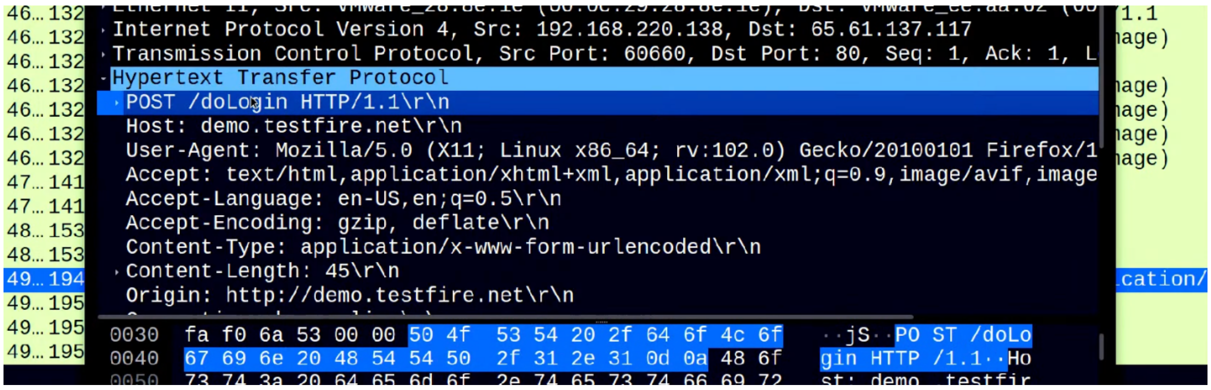
Users

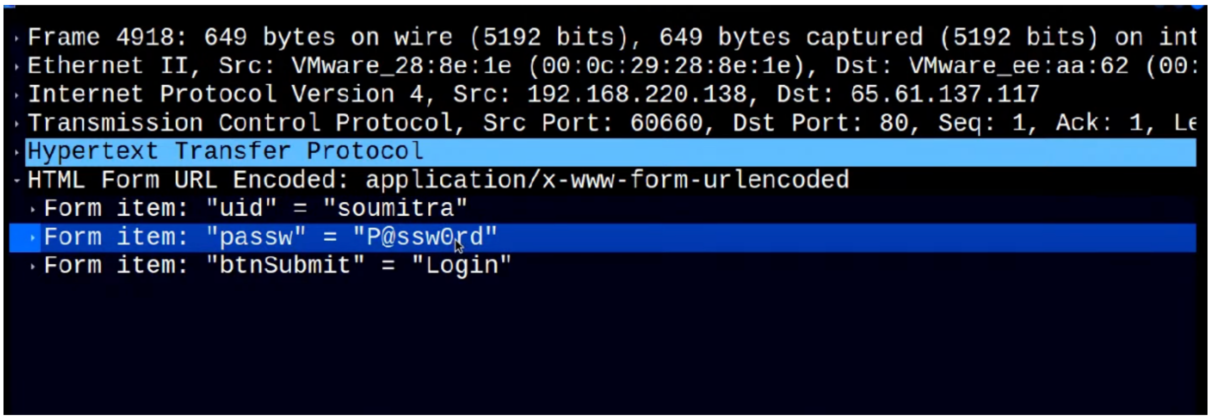
wiener - Delete

Cryptographic Failure Vulnerability:

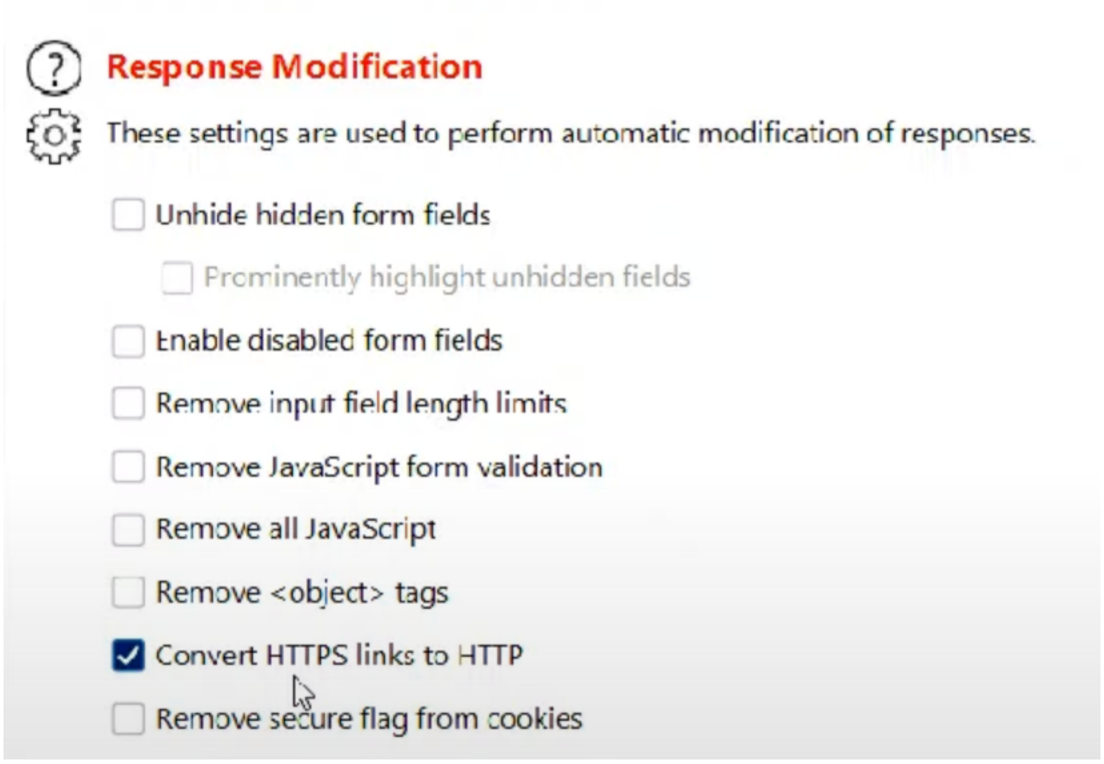
Less than 4 years ago, a very small (<10 employees) marketing and data aggregation firm called Exactis accidentally exposed its database that contained around 340 million individual records. Be it experience, negligence, or ignorance, the people in charge had put the database on a publicly accessible server. What that means is that anyone (anyone who knew where to look, that is) could access this data.

The exposed records included names, phone numbers, emails, and other sensitive data of **millions of US citizens**. And because this information was intended for highly targeted marketing purposes, it was much more detailed and personal than what people usually expose in an everyday data breach.


```
46… 132  Internet Protocol Version 4, Src: 192.168.220.138, Dst: 65.61.137.117
46… 132  Transmission Control Protocol, Src Port: 60660, Dst Port: 80, Seq: 1, Ack: 1, L
46… 132  Hypertext Transfer Protocol
46… 132    POST /doLogin HTTP/1.1\r\n
46… 132    Host: demo.testfire.net\r\n
46… 132    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/1
47… 141    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
47… 141    Accept-Language: en-US,en;q=0.5\r\n
48… 153    Accept-Encoding: gzip, deflate\r\n
48… 153    Content-Type: application/x-www-form-urlencoded\r\n
49… 194    Content-Length: 45\r\n
49… 195    Origin: http://demo.testfire.net\r\n
49… 195
49… 195  0030  fa f0 6a 53 00 00 50 4f  53 54 20 2f 64 6f 4c 6f   ··jS··PO ST /doLo
         0040  67 69 6e 20 48 54 54 50  2f 31 2e 31 0d 0a 48 6f   gin HTTP /1.1··Ho
         0050  73 74 3a 20 64 65 6d 6f  2e 74 65 73 74 66 69 72   st: demo .testfir
```



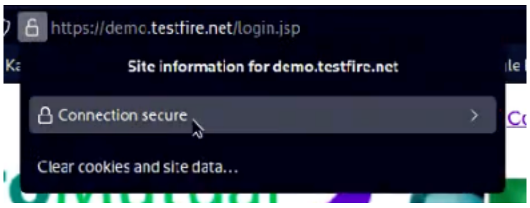
```
› Frame 4918: 649 bytes on wire (5192 bits), 649 bytes captured (5192 bits) on int
› Ethernet II, Src: VMware_28:8e:1e (00:0c:29:28:8e:1e), Dst: VMware_ee:aa:62 (00:
› Internet Protocol Version 4, Src: 192.168.220.138, Dst: 65.61.137.117
› Transmission Control Protocol, Src Port: 60660, Dst Port: 80, Seq: 1, Ack: 1, Le
› Hypertext Transfer Protocol
⌄ HTML Form URL Encoded: application/x-www-form-urlencoded
  › Form item: "uid" = "soumitra"
  › Form item: "passw" = "P@ssw0rd"
  › Form item: "btnSubmit" = "Login"
```



Response Modification

These settings are used to perform automatic modification of responses.

- [ ] Unhide hidden form fields
  - [ ] Prominently highlight unhidden fields
- [ ] Enable disabled form fields
- [ ] Remove input field length limits
- [ ] Remove JavaScript form validation
- [ ] Remove all JavaScript
- [ ] Remove <object> tags
- [✓] Convert HTTPS links to HTTP
- [ ] Remove secure flag from cookies



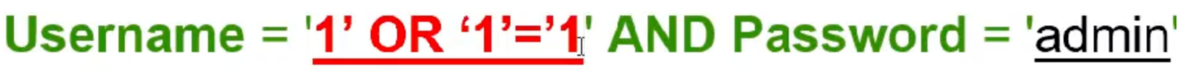
https://demo.testfire.net/login.jsp

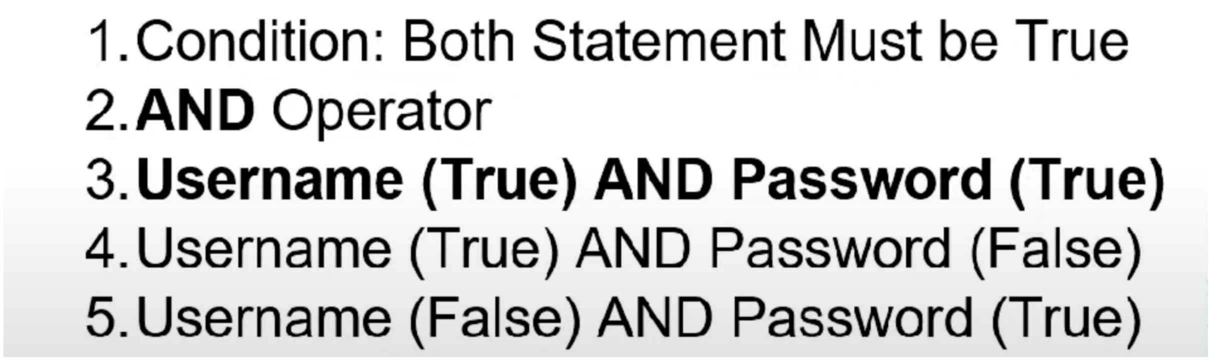
Site information for demo.testfire.net

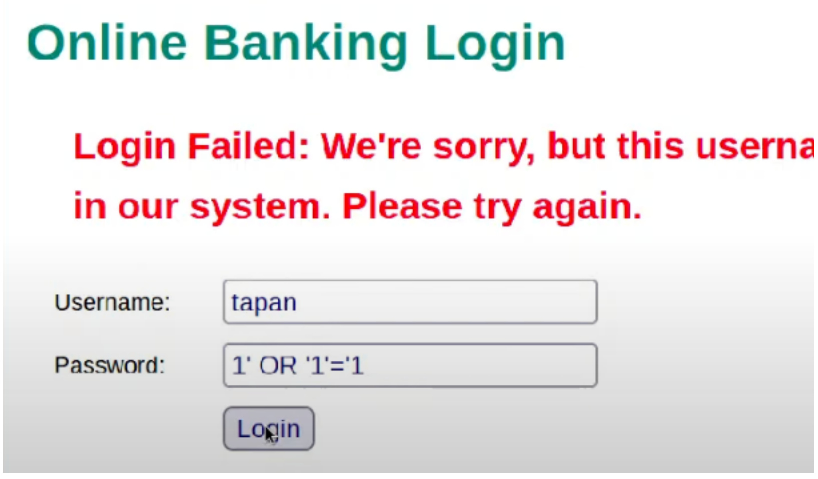
Connection secure

Clear cookies and site data...

Injection :

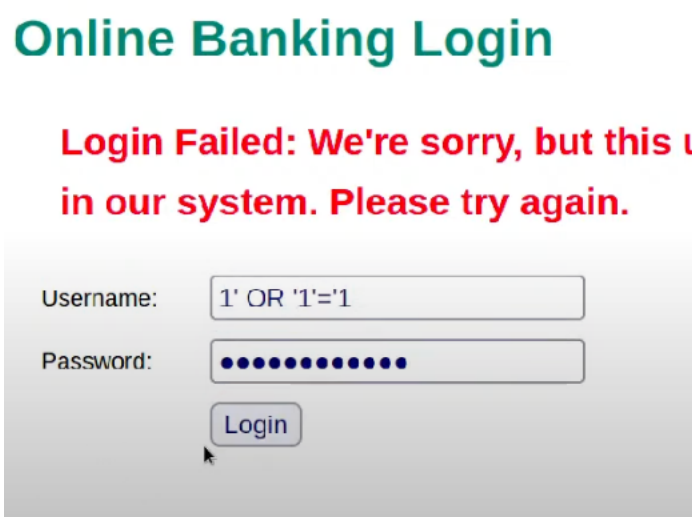

Username = '1' OR '1'='1' AND Password = 'admin'


1. Condition: Both Statement Must be True
2. **AND** Operator
3. **Username (True) AND Password (True)**
4. Username (True) AND Password (False)
5. Username (False) AND Password (True)


Online Banking Login

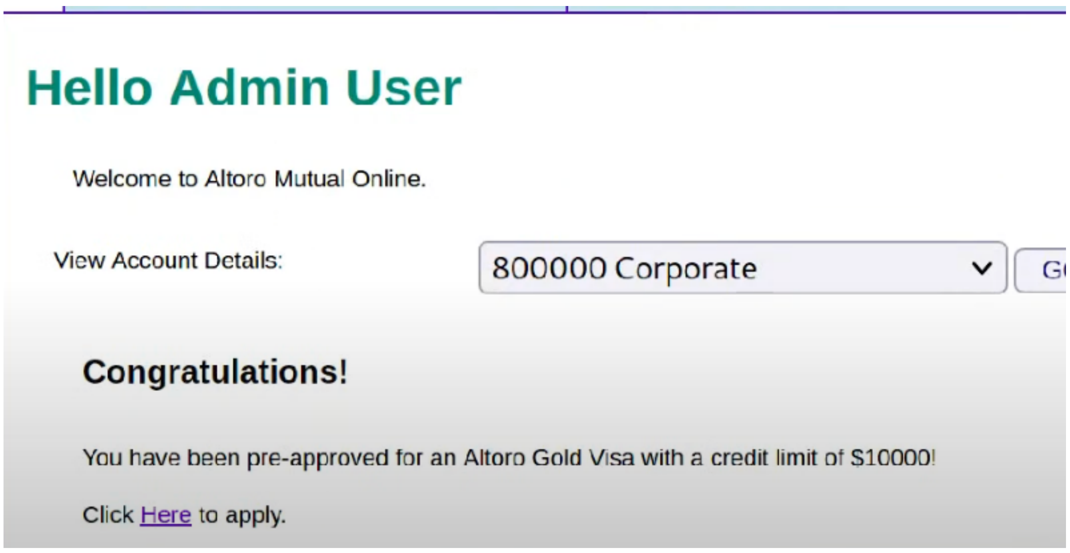
Login Failed: We're sorry, but this userna
in our system. Please try again.

Username:	tapan
Password:	1' OR '1'='1


Online Banking Login

Login Failed: We're sorry, but this u
in our system. Please try again.

Username:	1' OR '1'='1
Password:	●●●●●●●●●●●●



Hello Admin User

Welcome to Altoro Mutual Online.

View Account Details: 800000 Corporate

Congratulations!

You have been pre-approved for an Altoro Gold Visa with a credit limit of $10000!

Click Here to apply.

Insecure Design:

What is Insecure Design?

Insecure design encompasses various risks that arise from **ignoring design** and **architectural best practices**, starting from the planning phase before actual implementation. A quick point to note here is that an insecure design differs from an insecure implementation, and a near-perfect implementation cannot prevent defects arising from an insecure design. While the Insecure design flaw is a new entrant to the **OWASP top 10**, it ranks [number four on the 2021 list]() since mitigating risks at the design phase is considered fundamental toward '**Shift Left**' security practices.

Vulnerabilities?

Insecure design vulnerabilities arise when developers, QA, and/or security teams **fail to anticipate** and **evaluate threats** during the code design phase. These vulnerabilities are also a consequence of the non-adherence of **security best practices** while designing an application. As the threat landscape evolves, mitigating design vulnerabilities requires consistent threat modeling to prevent known attack methods. Without a secure design, it is difficult to detect and remediate architectural flaws such as:

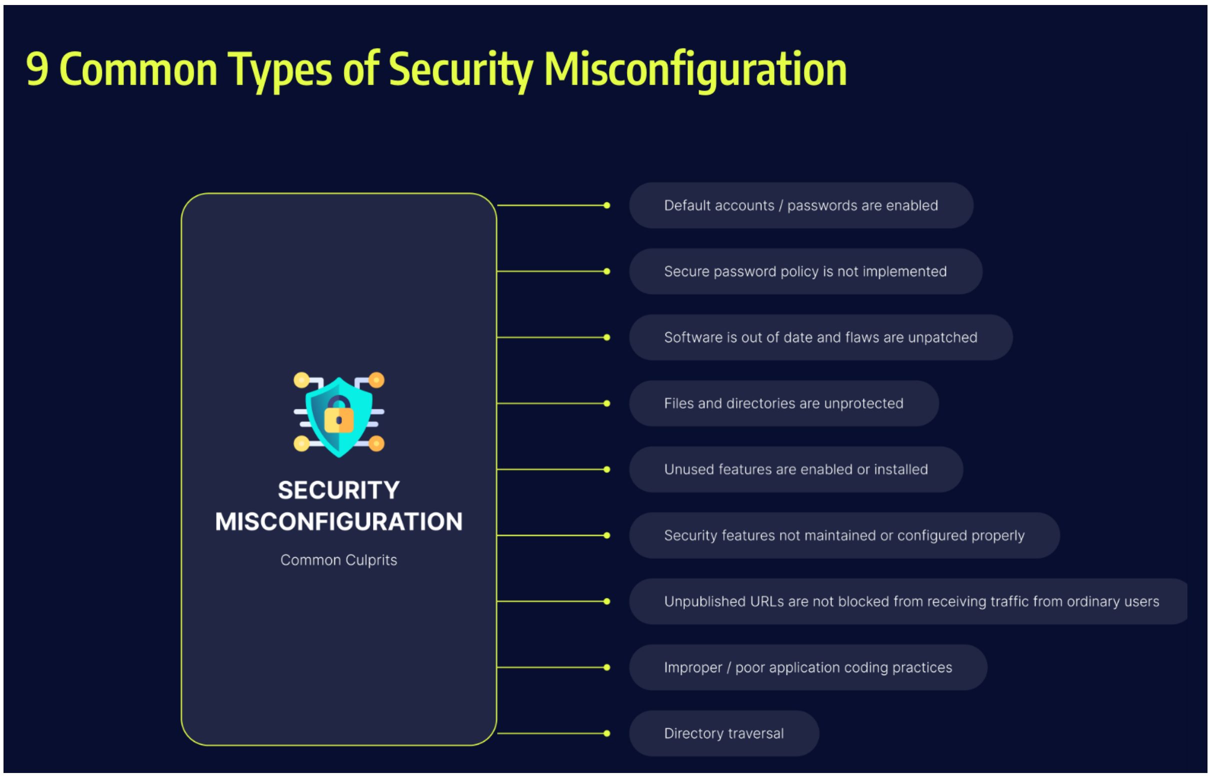
```
try { openDbConnection(); } //print exception message that
includes exception message and configuration file location catch
(Exception $e) { echo 'Caught exception: ', $e->getMessage(),
'\n'; echo 'Check credentials in config file at: ',
$Mysql_config_location, '\n'; }
```

```
SELECT * FROM vehicles WHERE type = ‘SUV’ AND registered=1
```

```
SELECT * FROM vehicles WHERE type= ‘SUV’--’ AND registered = 1
```

The — characters indicate the start of a comment in SQL, so the server interprets the rest of the query as a comment. By commenting out ***registered= 1*** part of the query, the attacker forces the server to return details of all vehicles in the SUV category, even the **unregistered ones**.

Security Misconfiguration:


9 Common Types of Security Misconfiguration
SECURITY MISCONFIGURATION
Common Culprits
Default accounts / passwords are enabled
Secure password policy is not implemented
Software is out of date and flaws are unpatched
Files and directories are unprotected
Unused features are enabled or installed
Security features not maintained or configured properly
Unpublished URLs are not blocked from receiving traffic from ordinary users
Improper / poor application coding practices
Directory traversal

Here are a few real life attacks that caused damage to major organizations, as a result of security misconfigurations:

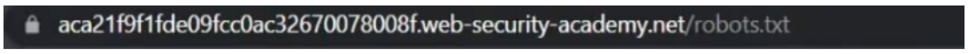
- NASA authorization misconfiguration attack – NASA because vulnerable to a misconfiguration in Atlassian JIRA. An authorization misconfiguration in Global Permissions enabled exposure of sensitive data to attackers.
- Amazon S3 – many organizations experienced data breaches as a result of unsecured storage buckets on Amazon's popular S3 storage service. For example, the US Army Intelligence and Security Command inadvertently stored sensitive database files, some of them marked top secret, in S3 without proper authentication.

Here are some efficient ways to minimize security misconfiguration:

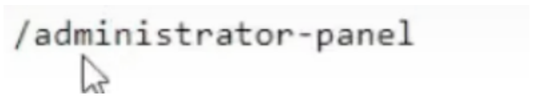
- Establish a hardening process that is repeatable, so that it's fast and simple to deploy correctly configured new environments. The production, development, and QA environments must all be configured in the same way, but with distinct passwords used in every environment. Automate this process to easily establish a secure environment.

• Install patches and software updates regularly and in a timely way in every environment. You can also patch a golden image and deploy the image into your environment.

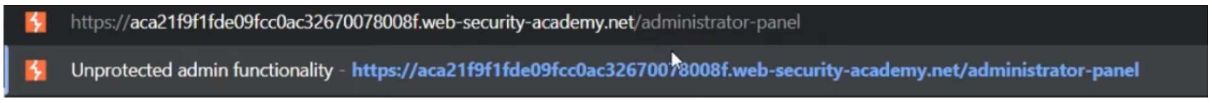
• Develop an application architecture that offers effective and secure separation of elements.



aca21f9f1fde09fcc0ac32670078008f.web-security-academy.net/robots.txt

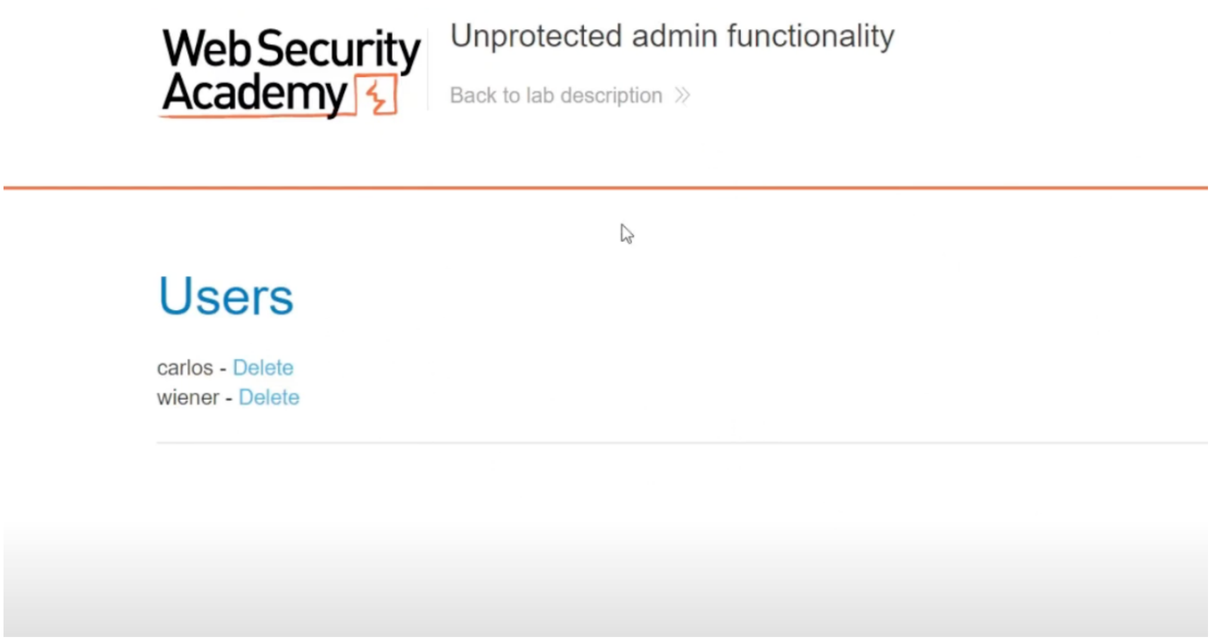


```
/administrator-panel
```



https://aca21f9f1fde09fcc0ac32670078008f.web-security-academy.net/administrator-panel

Unprotected admin functionality - https://aca21f9f1fde09fcc0ac32670078008f.web-security-academy.net/administrator-panel

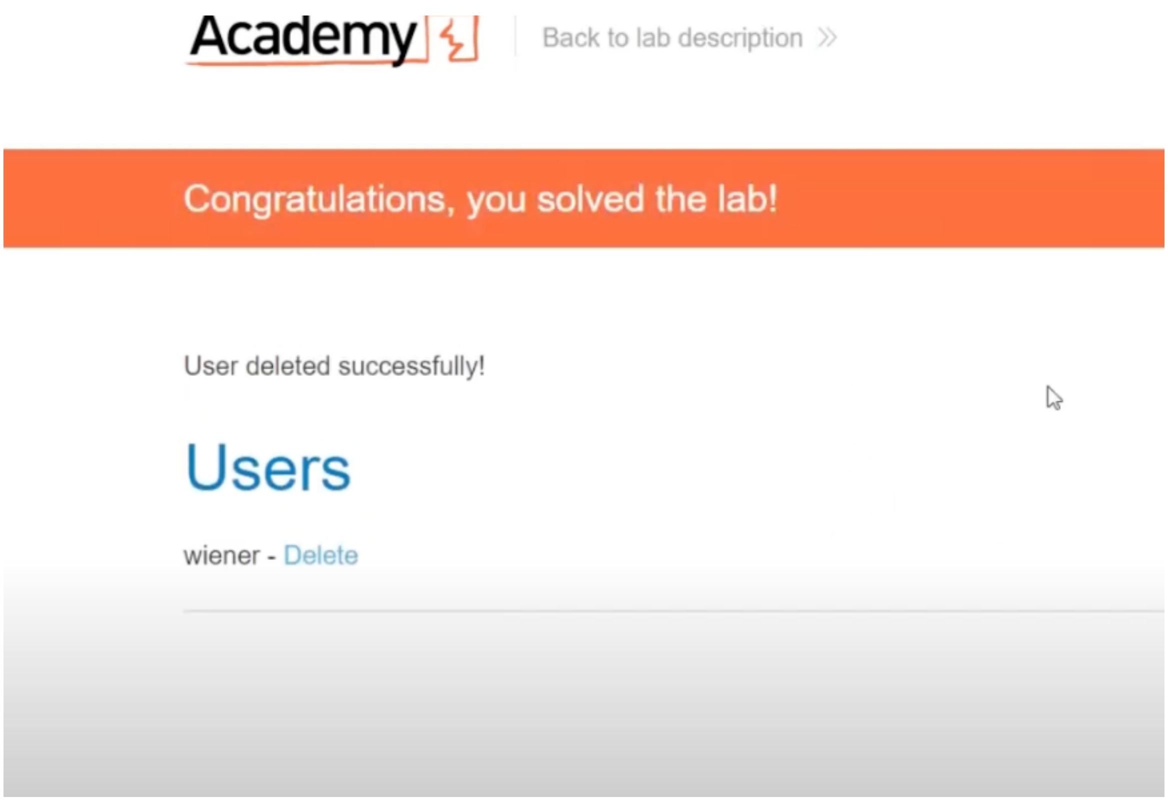


Unprotected admin functionality

Back to lab description »

Users

carlos - Delete
wiener - Delete


Academy

Back to lab description »

Congratulations, you solved the lab!

User deleted successfully!

Users

wiener - Delete