

## Vulnerability test on websites

1. Broken Access Control

Unprotected admin panel

The screenshot shows a web browser with the address bar displaying `0a17002c044b1157812e3ee800d9003e.web-security-academy.net/robots.txt`. Below the address bar, the text `User-agent: *` and `Disallow: /administrator-panel` is visible. The main content area shows the Web Security Academy logo and the title "Unprotected admin functionality". A green button labeled "LAB" and a status "Not solved" are present. Below this, there is a section titled "Users" with a list of users: "wiener - Delete" and "carlos - Delete". At the bottom, a message "User deleted successfully!" is displayed, followed by another "Users" section showing only "wiener - Delete".

Web Security Academy

Unprotected admin functionality

LAB Not solved

Back to lab description >>

Home | My account

Users

wiener - Delete  
carlos - Delete

User deleted successfully!

Users

wiener - Delete

Once the administrator link is found, there is no access control which makes the application vulnerable.

## 2. Cryptographic failure

### Information disclosure in error messages

← → ↻ 0a7b005d04d9e29c8006c14100bf005f.web-security-academy.net/product?productId=1

**WebSecurity Academy** Information disclosure in error messages  
[Submit solution](#) [Back to lab description >>](#)

---

Lightbulb Moments  
★★★★☆ \$81.57



← → ↻ 0a7b005d04d9e29c8006c14100bf005f.web-security-academy.net/product?productId=test

```
Internal Server Error: java.lang.NumberFormatException: For input string: "test"
    at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
    at java.base/java.lang.Integer.parseInt(Integer.java:668)
    at java.base/java.lang.Integer.parseInt(Integer.java:786)
    at lab.t.x.r.z.N(Unknown Source)
    at lab.k.i.u.a.O(Unknown Source)
    at lab.k.i.s.i.b.B(Unknown Source)
    at lab.k.i.s.k.lambda$handleSubRequest$0(Unknown Source)
    at c.z.i.a.lambda$null$3(Unknown Source)
    at c.z.i.a.x(Unknown Source)
    at c.z.i.a.lambda$uncheckedFunction$4(Unknown Source)
    at java.base/java.util.Optional.map(Optional.java:260)
    at lab.k.i.s.k.N(Unknown Source)
    at lab.server.o.g.w.c(Unknown Source)
    at lab.k.i.n.T(Unknown Source)
    at lab.k.i.n.c(Unknown Source)
    at lab.server.o.g.j.v.A(Unknown Source)
    at lab.server.o.g.j.f.lambda$handle$0(Unknown Source)
    at lab.t.u.n.y.c(Unknown Source)
    at lab.server.o.g.j.f.B(Unknown Source)
    at lab.server.o.g.c.x(Unknown Source)
    at c.z.i.a.lambda$null$3(Unknown Source)
    at c.z.i.a.x(Unknown Source)
    at c.z.i.a.lambda$uncheckedFunction$4(Unknown Source)
    at lab.server.z.l.O(Unknown Source)
    at lab.server.o.g.c.i(Unknown Source)
    at lab.server.o.f.q.l(Unknown Source)
    at lab.server.o.d.o(Unknown Source)
    at lab.server.o.v.o(Unknown Source)
    at lab.server.z_.P(Unknown Source)
    at lab.server.z_.f(Unknown Source)
    at lab.r.k.lambda$consume$0(Unknown Source)
    at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1136)
    at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635)
    at java.base/java.lang.Thread.run(Thread.java:833)
```

Apache Struts 2 2.3.31

Version number of the framework is visible in the error message and is not encrypted.

### 3. Injection

Email: [xxx@xxx.xxx](#)

Password: xxx') OR 1 = 1 — ]

```
SELECT * FROM users WHERE email = 'xxx@xxx.xxx' OR 1 = 1 LIMIT 1 — ' ]
AND password = md5('1234');
```

- **xxx@xxx.xxx** ends with a single quote which completes the string quote
- **OR 1 = 1 LIMIT 1** is a condition that will always be true and limits the returned results to only one record.
- **— ' AND ...** is a SQL comment that eliminates the password part.

The screenshot shows two browser windows. The top window is the 'Login | Personal Contacts Manager v1.0' page. The email field contains 'xxx@xxx.xxx' and the password field contains a long string of dots, representing the injected payload. The 'Submit' button is highlighted. The bottom window is the 'Dashboard | Personal Contacts Manager v1.0' page. It features a table with 9 contacts. The first contact has ID 1, First Name 'mynams', Last Name 'jenefry', Mobile No '9898989898', and Email 'admin@gmail.com'. The other contacts have IDs 67840 through 67847. The table includes columns for ID, First Name, Last Name, Mobile No, Email, and Actions (with an 'Edit' link). A 'Total Records Count: 9' is shown at the bottom of the table.

Dashboard | Personal Contacts Manager v1.0

Add New Contact Log Out

ID	First Name	Last Name	Mobile No	Email	Actions
1	mynams	jenefry	9898989898	admin@gmail.com	
67840	http://124.29.206.118:9999/trasur/	maiden	05465488498	djahhlsd2@gmail.com	<a href="#">Edit</a>
67841		wgr	0565	dcaskj@gmail.com	<a href="#">Edit</a>
67842	david	david	02167357	grush346@gmail.com	<a href="#">Edit</a>
67843	david	david	02167357	grush346@gmail.com	<a href="#">Edit</a>
67844	damian	dami	0217218721	grush346@gmail.com	<a href="#">Edit</a>
67845	imam	uhuy	762873617831	grush346@gmail.com	<a href="#">Edit</a>
67846	dima	Антоненко	+79516240323	tatarnikovafialka@mail.ru	<a href="#">Edit</a>
67847	dima	Антоненко	+79516240323	tatarnikovafialka@mail.ru	<a href="#">Edit</a>

Total Records Count: 9

## 4. Insecure Design

```
1 [
2   {
3     "name": "Electrician Ellie",
4     "role": "grunt"
5   },
6   {
7     "name": "Plumber Polly",
8     "role": "grunt"
9   },
10  {
11    "name": "Developer Friend",
12    "role": "developer"
13  },
14  {
15    "name": "You",
16    "role": "castle_leader"
17  }
18 ]
```

```
1 [
2   {
3     "name": "Electrician Ellie",
4     "role": "grunt"
5   },
6   {
7     "name": "Plumber Polly",
8     "role": "grunt"
9   },
10  {
11    "name": "Developer Friend",
12    "role": "developer"
13  },
14  {
15    "name": "Sir Snake Oil",
16    "role": "developer"
17  }
18 ]
```

```
import json

f = open('permissions.json')
employee_roles = json.load(f)

def remove_employee(employee_id, requesting_person):
    # Fetch role from permissions.json where name equals $requesting_person
    role = list(filter(lambda x:x["name"]==requesting_person, employee_roles))[0]["role"]

    # Only allow the roles 'developer' and 'castle_leader' to delete employees
    if role == "developer" or role == "castle_leader":
        print(requesting_person, employee_id)
        cursor.execute("DELETE FROM employees WHERE ID = ?", [employee_id])
```

In the above case, the newly entered developer friend can delete the leader and increase his privilege because of the error in the design.

Here, the privilege of the can be controlled by:

```
[
  {
    "name": "Electrician Ellie",
    "role": ["receive_order", "submit_timecard"]
  },
  {
    "name": "Plumber Polly",
    "role": ["receive_order", "submit_timecard"]
  },
  {
    "name": "Developer Friend",
```

```

    "role": ["receive_order", "submit_timecard", "change_permissions", "edit_code",
"add_employee", "remove_employee", ...]
  },
  {
    "name": "Sir Snake Oil",
    "role": ["receive_order", "submit_timecard", "edit_code", ...]
  }
  {
    "name": "You",
    "role": [ ... ]
  }
]

```

## 5. Security Misconfiguration

**First**, Launch Webgoat and navigate to insecure configuration section



We can try out as many options as we can think of. All we need to find the URL of config file and we know that the developers follow kind of naming convention for config files. It can be anything that is listed below. It is usually done by BRUTE force technique.

- web.config
- config
- appname.config
- conf

