Safe-handling input data across boundaries

- Stored procedures, parameterized queries for SQL Injection
- Validation at each boundary / input point
- Whitelist/blacklist
- Sanitization
- Challenges of expressive languages on the web

Snippets of Doom: An Exercise

```
def path = System.console().readLine 'Enter file path:'
if (path.startsWith("/safe_dir/"))
{
    File f = new File(path);
    f.delete()
}
```

```
// Get username from parameters
String username = request.getParameter("username");
// Create a statement from database connection
Statement statement = connection.createStatement();
// Create unsafe query by concatenating user defined data with query string
String query = "SELECT secret FROM Users WHERE (username = '" + username + "' AND NOT role = 'admin')";
// ... OR ...
// Insecurely format the query string using user defined data
String query = String.format("SELECT secret FROM Users WHERE (username = '%s' AND NOT role = 'admin')",
username);
// Execute query and return the results
ResultSet result = statement.executeQuery(query);
```

```
@RestController
public class XSSController {

    @GetMapping("/hello")
    ResponseEntity<String> hello(@RequestParam(value = "name", defaultValue = "World") String name) {
        return new ResponseEntity<>("Hello World!" + name, HttpStatus.OK);
    }

}
```

https://github.com/snoopysecurity/Vulnerable-Code-Snippets