Solutions

Exercises

Exercise 3.1 (Authentication Bypass)

#	Username	Password	Created SQL Query	Query Result
1	`horst`	`n0Rd4kAD3m!E`	`SELECT id FROM users WHERE name = 'horst' AND password = 'n0Rd4kAD3m!E'`	`42`
2	`'`	`qwertz`	`SELECT id FROM users WHERE name = ''' AND password = 'qwertz'`	`Error`
3	`'--`	`abc123`	`SELECT id FROM users WHERE name = ''-- AND password = 'abc123'`	`nothing`

#	Username	Password	Created SQL Query	Query Result
4	`horst'--`	`qwertz`	`SELECT id FROM users WHERE name = 'horst'-- AND password = 'qwertz'`	`42`
5	`admin'--`	`<anything>`	`SELECT id FROM users WHERE name = 'admin'`	`1`
6	`' OR 1=1--`	`<anything>`	`SELECT id FROM users`	`1`, `2`, …

Exercise 6.1 (Info. Classification)

Practice	Public	Internal	Confidential	Secret
Publish on Internet	✓	✗	✗	✗
Publish on Intranet	✓	✓	✗	✗
Print on 🖨️	✓	✓	✓ if picked up immediately	✓ on personal or otherwise secured printer

Practice	Public	Internal	Confidential	Secret
Share with third parties	✓	✓ with NDA	✓ with NDA + permission	✓ with NDA + permission
Copy to USB key	✓	✓	✓ with encryption + permission	✓ with encryption + permission

⚠️ *Many organizations do not allow the use of USB keys **in general**. This kind of restriction would obviously **overrule** any of the above "Copy to USB" assessments with* ❌.

Exercise 6.2 (Data Lifecycle Phases)

Phase	Internal	Confidential	Secret
Permanent storage	🔴 Access Control (against external access)	🔴 Access Control ⭕Access logs, Encryption	🔴 Access Control, Access logs, Encryption
Transfer (internal network)	No restrictions	⭕ Encryption (e.g. TLS)	🔴 Encryption (e.g. TLS) ⭕/🔴 End-to-end encryption (e.g. PGP, Signal)

Phase	Internal	Confidential	Secret
Transfer (public network)	⭕ Encryption (e.g. VPN)	⭕ Encryption (e.g. VPN, TLS)	🔴 Encryption (e.g. VPN, TLS) ⭕/🔴 End-to-end encryption (e.g. PGP, Signal)
Disposal	No restrictions	🔴 Shredding, secure deletion, data wipe	🔴 Shredding, secure deletion, data wipe ⭕/🔴 Destroy medium physically (🔨, 🔥)

ℹ️ *For "Public" data no restrictions for any lifecycle phases apply.*

Exercise 8.2 (ArrayList Deserialization)

```
/**
 * The maximum size of array to allocate.
 * Some VMs reserve some header words in an array.
 * Attempts to allocate larger arrays may result in
 * OutOfMemoryError: Requested array size exceeds VM limit
 */
private static final int MAX_ARRAY_SIZE = Integer.MAX_VALUE - 8;
```

💥 *Whenever an* `OutOfMemoryError` *occurs, the affected JVM crashes.*

Exercise 8.3 (HashSet Deserialization)

💥 *With its members recursively linked to each other, when deserializing* `root`*, the JVM will begin creating a recursive object graph. It will never complete, and consume CPU indefinitely.*

🔍 *If you view this as a PDF, zoom in as much as possible on the above code snippet to get an idea what is going on. You might want to look at the original Markdown file to actually be able to read something.*

Exercise 9.1 (OWASP Benchmark)


OWASP Benchmark Guide