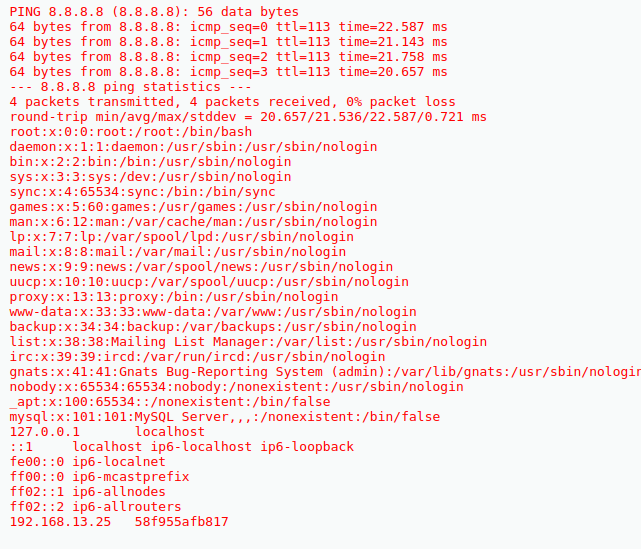
Homework 15 Submission File

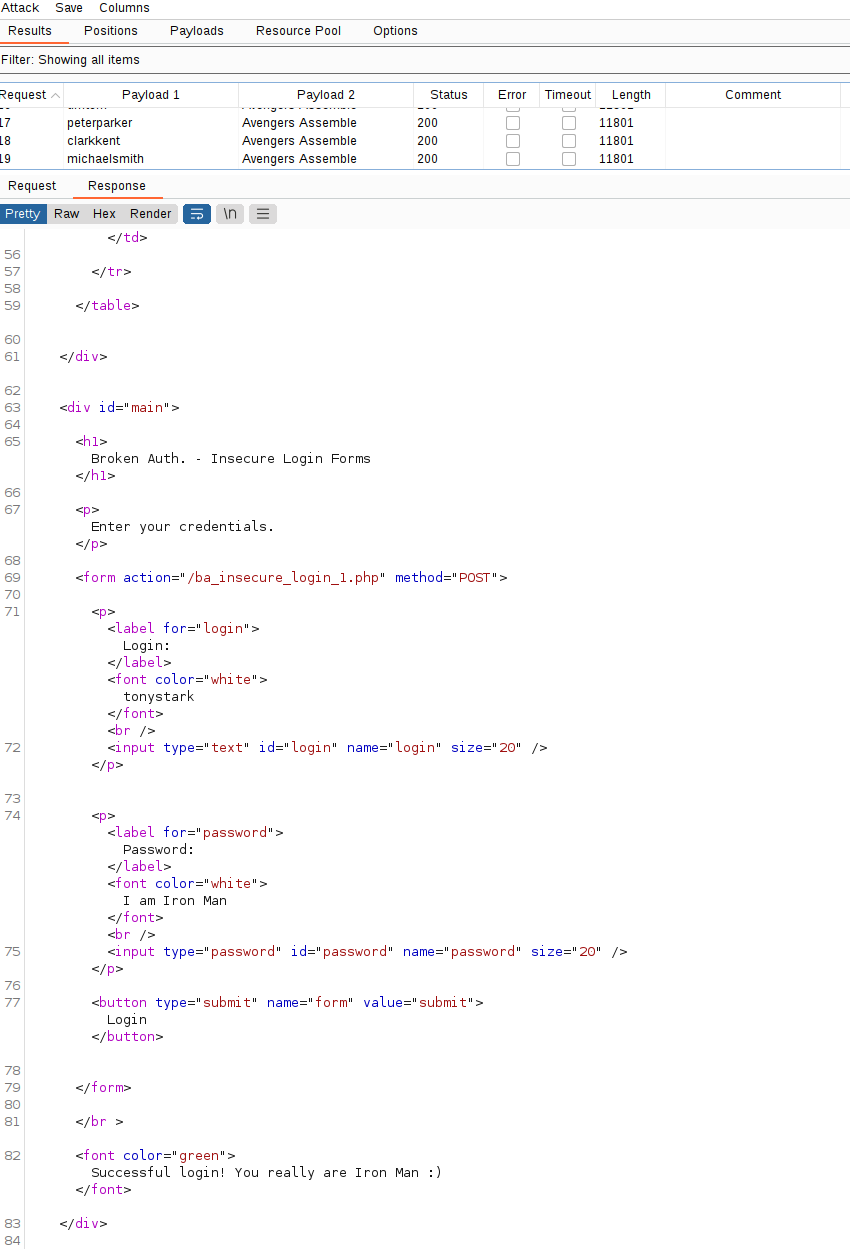
Web Vulnerabilities and Hardening

**Web Application 1 - Your Wish is My Command Injection**



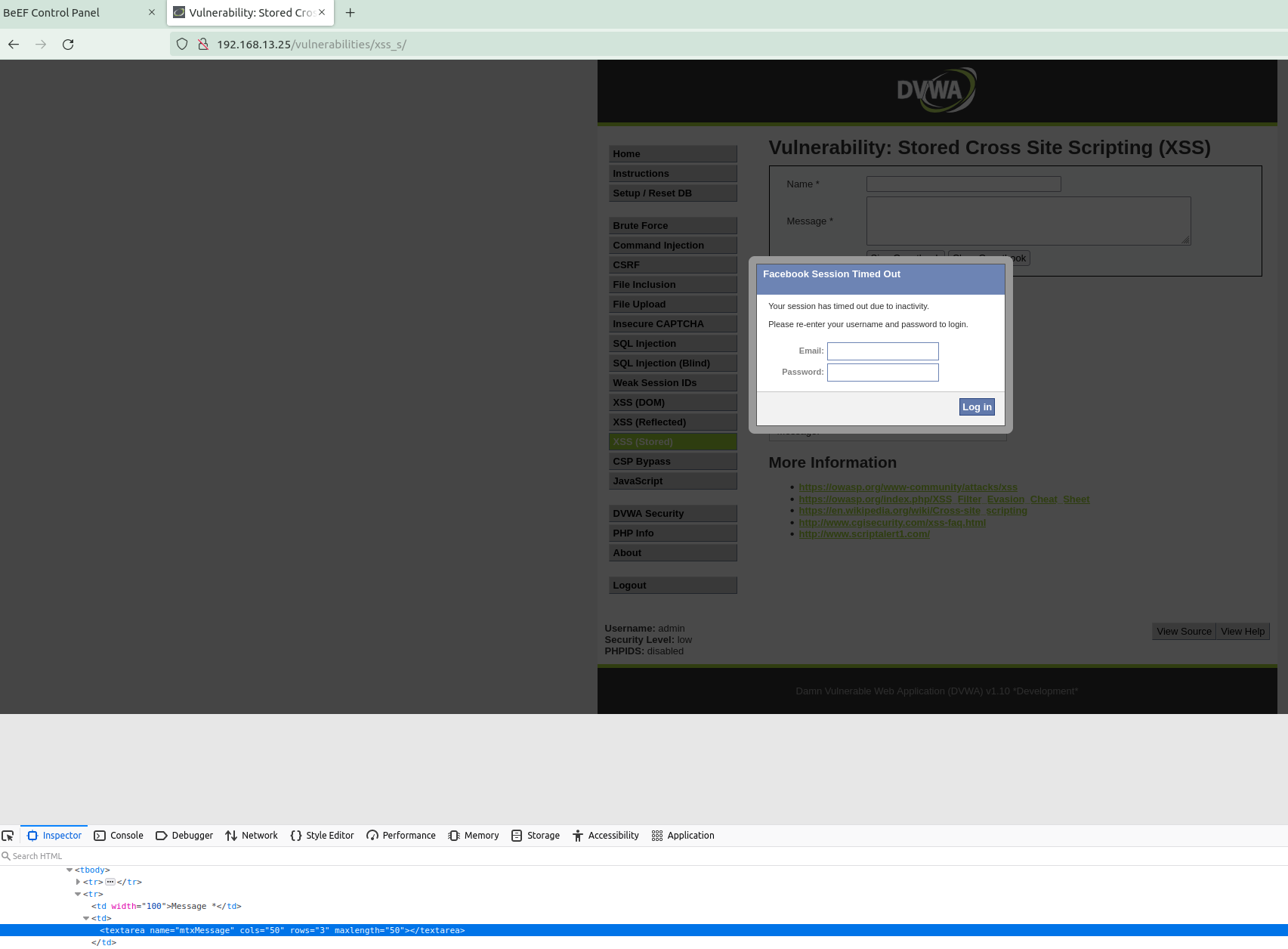
As the above screenshot indicates, the Command Injection vulnerability can allow unintended access to sensitive files such as `/etc/passwd` and `/etc/hosts`. This can be mitigated by using an API instead of the OS itself. If that cannot be achieved, limiting character input and setting a selection of which characters are legal (e.g. only numbers and periods for an IP address) can help defend against this type of attack.

**Web Application 2 - A Brute Force to Be Reckoned With**



As the above screenshot indicates, a brute force attack was successful in achieving a logon as an administrator. The most common mitigation strategy for this attack is to lock-out the account after a certain number of failed login attempts. However, this can lead to problems with DoS attacks intentionally locking users out of their accounts. A more nuanced solution would be to insist on another authentication factor after a certain number of failed login attempts, such as answering a secret question.

**Web Application 3 - Where’s the BeEF?**



As the above screenshot indicates, it can be a simple matter to exploit a website with either Stored XSS or Reflected XSS attacks. A strong first line of defense against these attacks is context-sensitive server side output encoding. Input validation, data sanitization, and proper data encoding (HTML, attributes, Javascript, CSS, URL, etc) can all help mitigate these attacks.