Assignment : Terminal Commands

1. mkdir -p backup && cp *.txt backup/

2. nano amina.txt

```
3. cd /usr/local/bin && ls -la
```

4. `find . -type f -name '*.tmp' -exec rm -i {} \;`
   -

Assignment on RRC and HTTP

How the HTTP Protocol May Be Breached

The HTTP protocol, being a stateless and plaintext protocol, has several inherent vulnerabilities that can be exploited if proper security measures are not in place. Here are some of the primary ways in which HTTP-based communications can be breached:

- **Lack of Encryption:**
  Since HTTP does not encrypt data, any information transmitted between the client and the server (such as login credentials, session cookies, or form data) can be intercepted by attackers using packet sniffing tools. This is especially dangerous on unsecured networks (e.g., public Wi-Fi).
- **Man-in-the-Middle (MITM) Attacks:**
  Attackers can position themselves between the client and the server. By intercepting and potentially modifying requests and responses, they can steal sensitive information or inject malicious code. For example, an attacker might alter the content of a page or redirect the user to a phishing site.
- **Session Hijacking:**
  HTTP relies on cookies to maintain session state. If these cookies are transmitted over an unsecured connection, an attacker can capture them and use them to impersonate the user. This can lead to unauthorized access to user accounts.
- **HTTP Header Manipulation:**
  Since HTTP headers are not encrypted, they can be manipulated. Attackers might inject malicious content or modify the request parameters to bypass security controls implemented on the server, such as altering cache settings or misrepresenting the request origin.
- **Injection Attacks:**
  Web applications that use HTTP to receive user input can be vulnerable to various injection attacks if input validation is weak. This includes SQL injection, command injection, and Cross-Site Scripting (XSS). In each case, an attacker crafts a malicious request that exploits the application's failure to properly sanitize input, potentially gaining unauthorized access or executing arbitrary code.

- **Cross-Site Request Forgery (CSRF):**
  An attacker can trick a user into performing unintended actions on a web application where they're authenticated. By exploiting the trust that a site has in a user's browser, the attacker can forge a request that performs actions on behalf of the user without their consent.
- **Replay Attacks:**
  In a replay attack, an attacker captures a valid HTTP request and resends it, often to duplicate transactions or gain unauthorized access. Since HTTP does not include built-in mechanisms for ensuring request uniqueness or timeliness, without additional safeguards (like nonces or timestamps), the protocol can be susceptible to such replayed requests.

Mitigation Measures:
To reduce these vulnerabilities, it's highly recommended to use HTTPS (HTTP Secure), which leverages TLS/SSL to encrypt data in transit. Additionally, employing proper input validation, secure cookie practices (like setting the HttpOnly and Secure flags), and using anti-CSRF tokens can help protect against these common attack vectors.