Week 1 Task: Introduction to Network Security and Basic Security Practices

Intern: Abdul Wadood Talha

Task 1: Understanding Cybersecurity Fundamentals

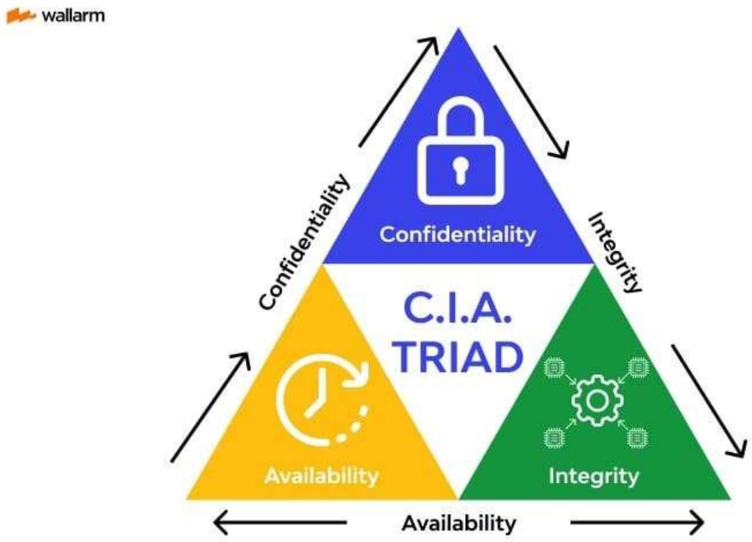
The CIA Triad: Confidentiality, Integrity, and Availability

The CIA Triad is the foundation of cybersecurity, guiding how we protect information systems and secondly, it is the only thing on which we have to work on installing mechanisms to preserve Triad.

1. **Confidentiality**
   Confidentiality ensures that information is only accessible to authorized individuals. Techniques like encryption, access controls, and secure passwords protect sensitive data from unauthorized access. For example, encrypting emails ensures that only the intended recipient can read them.

2. **Integrity**
   Integrity ensures data accuracy and trustworthiness by preventing unauthorized modifications. Tools like checksums, digital signatures, and version control systems detect and prevent tampering. For instance, banks rely on integrity measures to ensure financial transactions are accurate and unaltered.

3. **Availability**
   Availability ensures that information and systems are accessible when needed. Measures like redundant systems, regular backups, and denial-of-service (DoS) attack protection ensure uptime and reliability. For example, e-commerce sites use redundant servers to stay online during high traffic or attacks.



Common Types of Cyber Attacks

Cyber attacks exploit vulnerabilities to compromise systems. Key attack types include:

1. **Phishing**
   Phishing tricks users into revealing sensitive information, like passwords or credit card numbers, through fake emails or websites. A common example is a fraudulent email pretending to be from a bank, urging users to "verify" their accounts.
2. **Malware**
   Malware (malicious software) includes viruses, worms, ransomware, and spyware. It can damage systems, steal data, or lock users out until a ransom is paid. For example, ransomware like WannaCry encrypts files and demands payment for decryption.
3. **Denial-of-Service (DoS) and Distributed DoS (DDoS)**
   DoS attacks flood a system with traffic, overwhelming it and causing downtime. DDoS attacks amplify this by using multiple compromised systems. For instance, attackers may use botnets to bring down websites.

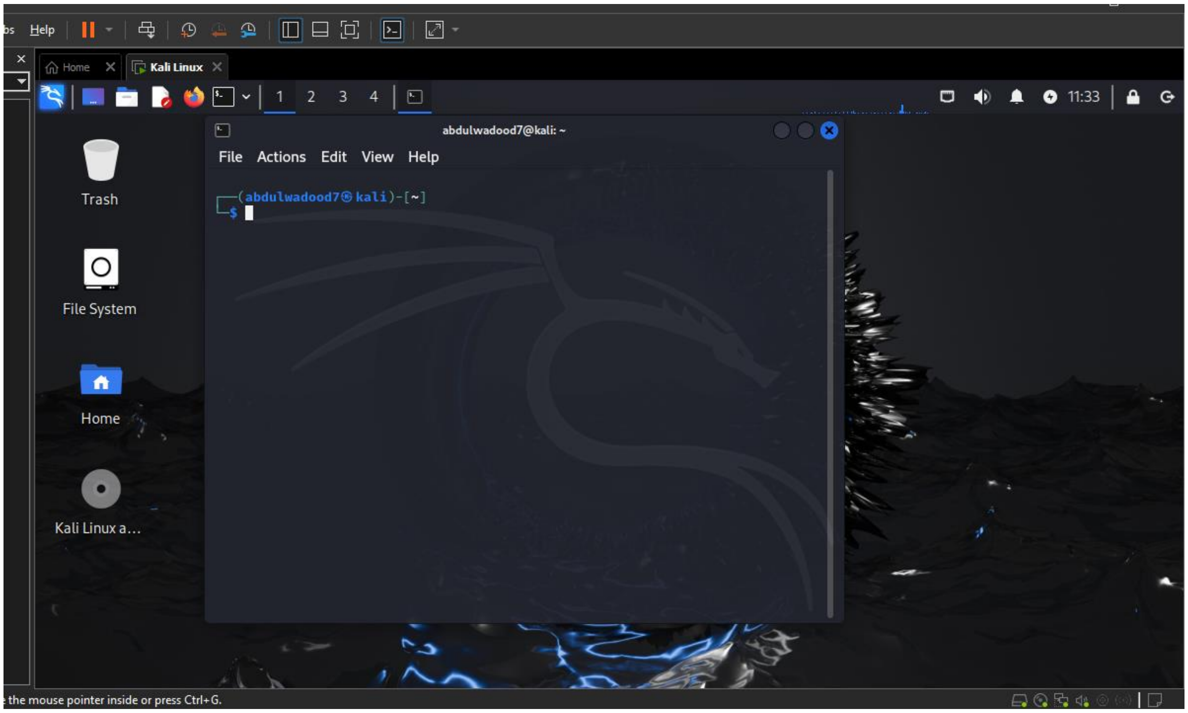
Importance of Network Security, Data Protection, and User Authentication

1. **Network Security**
   Protecting networks involves firewalls, intrusion detection systems (IDS), and secure Wi-Fi configurations. Network security prevents unauthorized access and ensures safe communication. For example, organizations use firewalls to block malicious traffic.

2. **Data Protection**
   Data protection safeguards sensitive information from breaches and leaks. Encryption, secure backups, and compliance with privacy laws like GDPR are essential. For instance, hospitals use encryption to protect patient records.

3. **User Authentication**
   Authentication verifies a user's identity, ensuring only authorized access. Strong authentication methods include multi-factor authentication (MFA), where users provide two or more verification factors (e.g., password + fingerprint). For example, banks often use MFA for online banking.

Task 2: Setting Up a Secure Virtual Environment



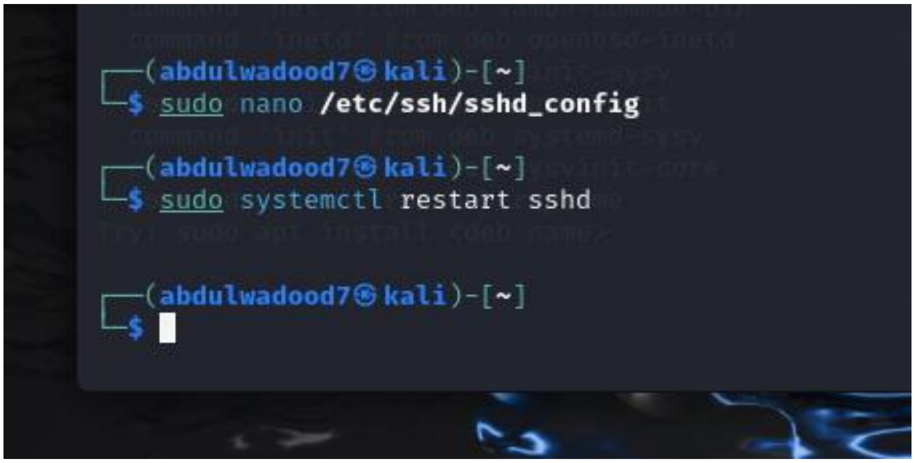
abdulwadood7@kali: ~

```
┌──(abdulwadood7㉿kali)-[~]
└─$ 
```

Configuring essential security settings:

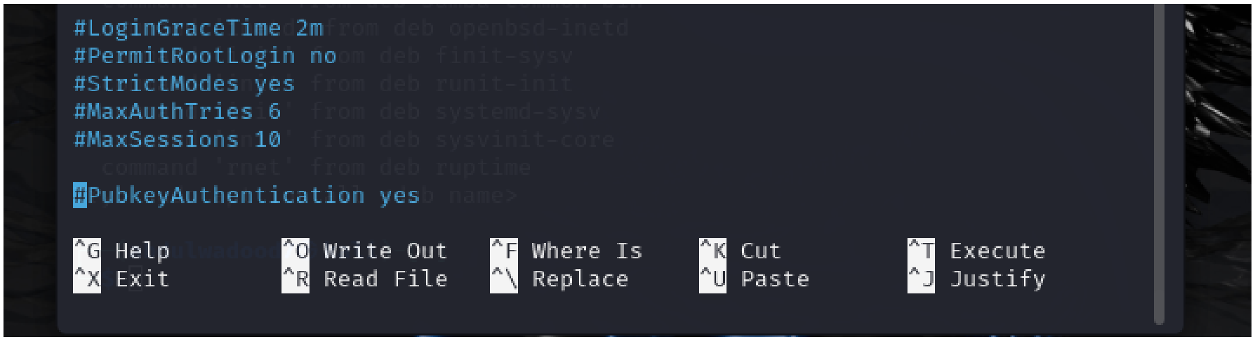
- **Disable root login for SSH**

Running these the first command took to a file where I had to disable root login to no!



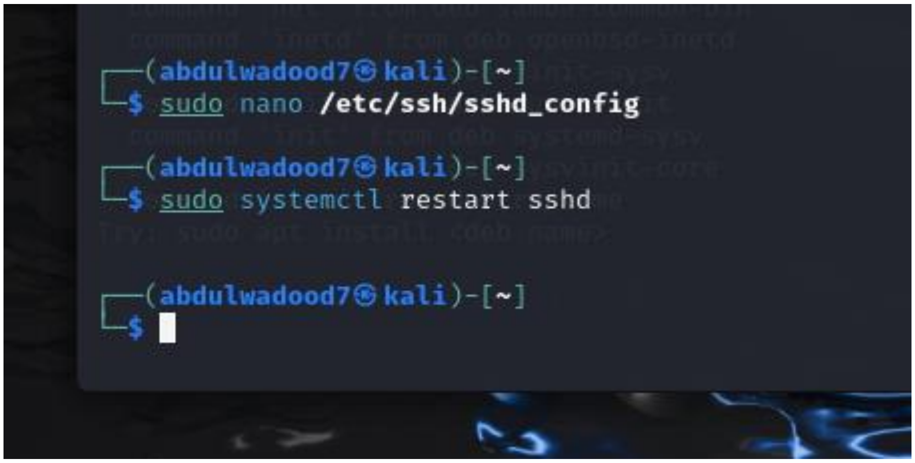
```
┌──(abdulwadood7㉿kali)-[~]
└─$ sudo nano /etc/ssh/sshd_config

┌──(abdulwadood7㉿kali)-[~]
└─$ sudo systemctl restart sshd

┌──(abdulwadood7㉿kali)-[~]
└─$ 
```

That's what I did it was on yes at default and I turned it to no.



```
#LoginGraceTime 2m
#PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

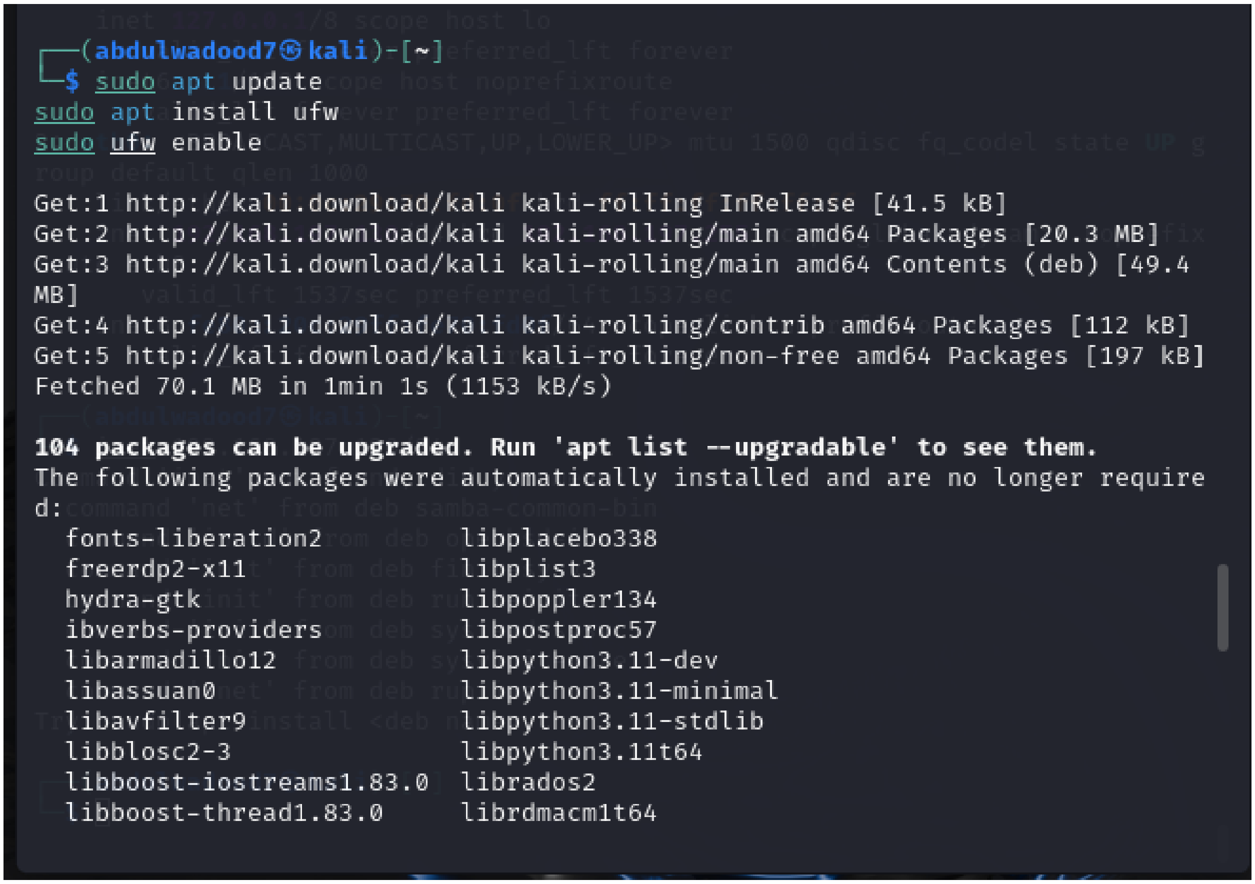
Lastly the restart command , so now root login is disabled.



```
┌──(abdulwadood7㉿kali)-[~]
└─$ sudo nano /etc/ssh/sshd_config

┌──(abdulwadood7㉿kali)-[~]
└─$ sudo systemctl restart sshd

┌──(abdulwadood7㉿kali)-[~]
└─$ 
```

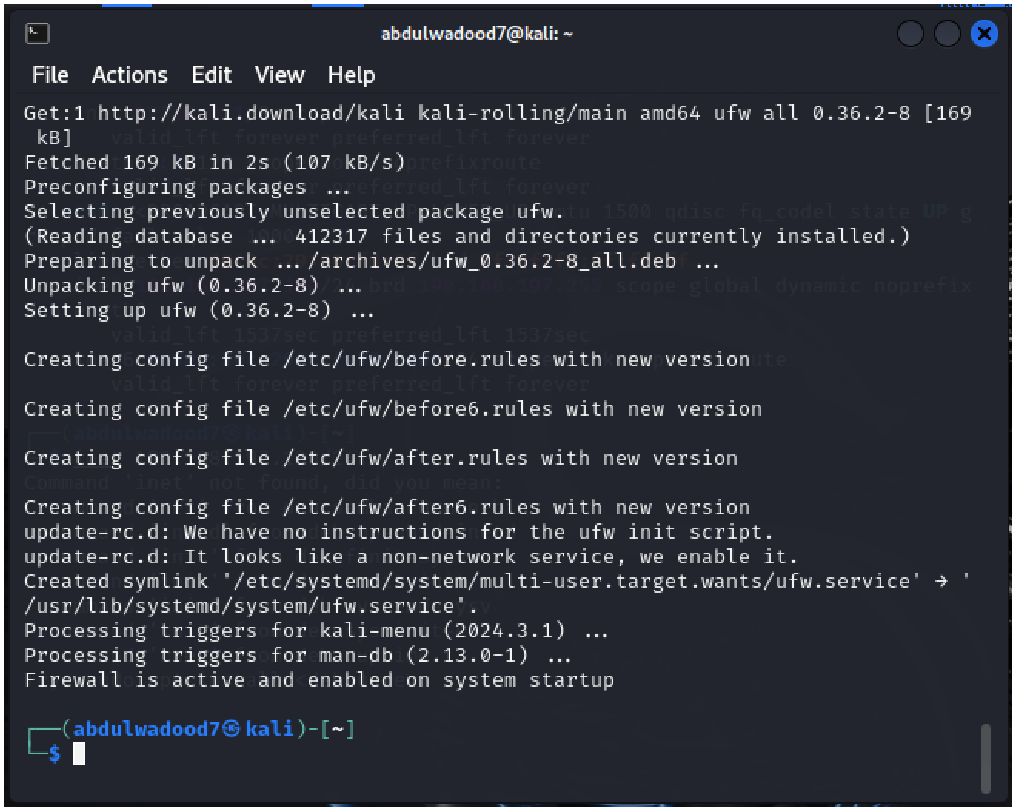
Configuring firewall settings using ufw (Uncomplicated Firewall)

Running normal update command then following install ufw command and finally enabling firewall.



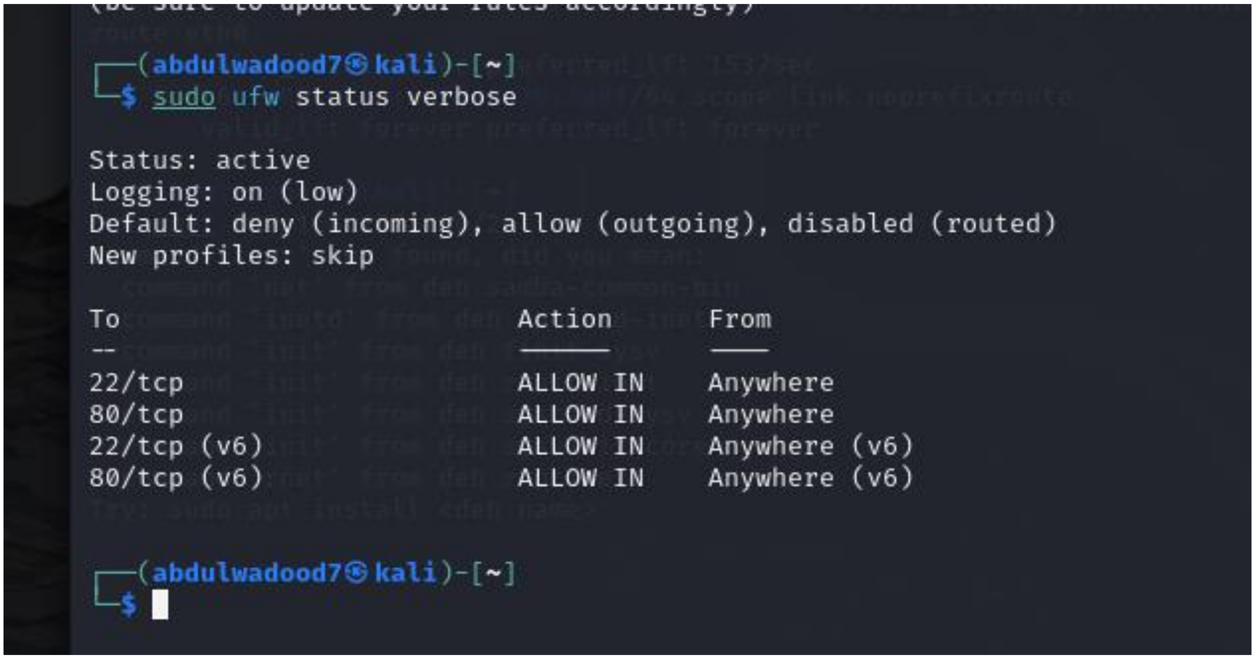
```
┌──(abdulwadood7㉿kali)-[~]
└─$ sudo apt update
sudo apt install ufw
sudo ufw enable
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [20.3 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [49.4
MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [112 kB]
Get:5 http://kali.download/kali kali-rolling/non-free amd64 Packages [197 kB]
Fetched 70.1 MB in 1min 1s (1153 kB/s)

104 packages can be upgraded. Run 'apt list --upgradable' to see them.
The following packages were automatically installed and are no longer require
d:
  fonts-liberation2          libplacebo338
  freerdp2-x11               libplist3
  hydra-gtk                  libpoppler134
  ibverbs-providers          libpostproc57
  libarmadillo12             libpython3.11-dev
  libassuan0                 libpython3.11-minimal
  libavfilter9               libpython3.11-stdlib
  libblosc2-3                libpython3.11t64
  libboost-iostreams1.83.0   librados2
  libboost-thread1.83.0      librdmacm1t64
```

As we can see The Firewall is active!



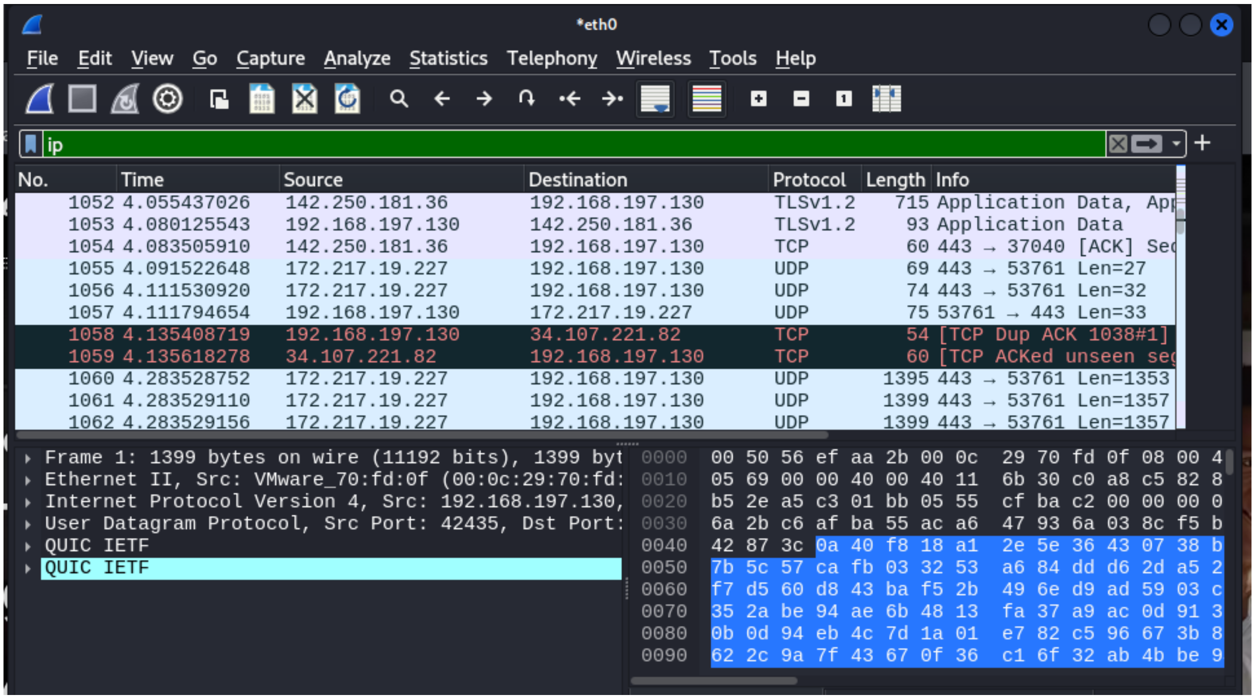
```
Get:1 http://kali.download/kali kali-rolling/main amd64 ufw all 0.36.2-8 [169
 kB]
Fetched 169 kB in 2s (107 kB/s)
Preconfiguring packages ...
Selecting previously unselected package ufw.
(Reading database ... 412317 files and directories currently installed.)
Preparing to unpack .../archives/ufw_0.36.2-8_all.deb ...
Unpacking ufw (0.36.2-8) ...
Setting up ufw (0.36.2-8) ...

Creating config file /etc/ufw/before.rules with new version

Creating config file /etc/ufw/before6.rules with new version

Creating config file /etc/ufw/after.rules with new version

Creating config file /etc/ufw/after6.rules with new version
update-rc.d: We have no instructions for the ufw init script.
update-rc.d: It looks like a non-network service, we enable it.
Created symlink '/etc/systemd/system/multi-user.target.wants/ufw.service' → '
/usr/lib/systemd/system/ufw.service'.
Processing triggers for kali-menu (2024.3.1) ...
Processing triggers for man-db (2.13.0-1) ...
Firewall is active and enabled on system startup

┌──(abdulwadood7㉿kali)-[~]
└─$ 
```

Now lastly checking status of firewall, as you can see I can enter my rules to allow and deny what I need.



```
(be sure to update your rules accordingly)

┌──(abdulwadood7㉿kali)-[~]
└─$ sudo ufw status verbose

Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
80/tcp                     ALLOW IN    Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)
80/tcp (v6)                ALLOW IN    Anywhere (v6)

┌──(abdulwadood7㉿kali)-[~]
└─$ 
```

Task 3: Network Security Basics: Packet Analysis



```
ip
```

```
No.    Time          Source            Destination       Protocol  Length Info
  1052 4.055437026   142.250.181.36    192.168.197.130   TLSv1.2     715 Application Data, App
  1053 4.080125543   192.168.197.130   142.250.181.36    TLSv1.2      93 Application Data
  1054 4.083505910   142.250.181.36    192.168.197.130   TCP          60 443 → 37040 [ACK] Seq
  1055 4.091522648   172.217.19.227    192.168.197.130   UDP          69 443 → 53761 Len=27
  1056 4.111530920   172.217.19.227    192.168.197.130   UDP          74 443 → 53761 Len=32
  1057 4.111794654   192.168.197.130   172.217.19.227    UDP          75 53761 → 443 Len=33
  1058 4.135408719   192.168.197.130   34.107.221.82     TCP          54 [TCP Dup ACK 1038#1]
  1059 4.135618278   34.107.221.82     192.168.197.130   TCP          60 [TCP ACKed unseen seg
  1060 4.283528752   172.217.19.227    192.168.197.130   UDP        1395 443 → 53761 Len=1353
  1061 4.283529110   172.217.19.227    192.168.197.130   UDP        1399 443 → 53761 Len=1357
  1062 4.283529156   172.217.19.227    192.168.197.130   UDP        1399 443 → 53761 Len=1357
```

```
▸ Frame 1: 1399 bytes on wire (11192 bits), 1399 byt
▸ Ethernet II, Src: VMware_70:fd:0f (00:0c:29:70:fd:
▸ Internet Protocol Version 4, Src: 192.168.197.130,
▸ User Datagram Protocol, Src Port: 42435, Dst Port:
▸ QUIC IETF
▸ QUIC IETF
```

```
0000  00 50 56 ef aa 2b 00 0c  29 70 fd 0f 08 00 4
0010  05 69 00 00 40 00 40 11  6b 30 c0 a8 c5 82 8
0020  b5 2e a5 c3 01 bb 05 55  cf ba c2 00 00 00 0
0030  6a 2b c6 af ba 55 ac a6  47 93 6a 03 8c f5 b
0040  42 87 3c 0a 40 f8 18 a1  2e 5e 36 43 07 38 b
0050  7b 5c 57 ca fb 03 32 53  a6 84 dd d6 2d a5 2
0060  f7 d5 60 d8 43 ba f5 2b  49 6e d9 ad 59 03 c
0070  35 2a be 94 ae 6b 48 13  fa 37 a9 ac 0d 91 3
0080  0b 0d 94 eb 4c 7d 1a 01  e7 82 c5 96 67 3b 8
0090  62 2c 9a 7f 43 67 0f 36  c1 6f 32 ab 4b be 9
```

IP Addresses:

- The source IP addresses observed are 142.250.181.36, 172.217.19.227, and 34.107.221.82.
- The destination IP address is consistently 192.168.197.130.

Protocols:

- The primary protocols observed are TLSV1.2, TCP, and UDP.

Ports:

- The source ports range from 715 to 75553761.
- The destination ports include 443, 30440, 53761, and others.

My data appears to show various network connections, between client devices and a central server or service at the 192.168.197.130 IP address on Ali Baba website. The mix of TCP, UDP, and TLS protocols suggests a variety of application-level network traffic, potentially including web browsing, file transfers, and other network activities. However, without additional context, I cannot make any firm conclusions about the specific nature of the network traffic.

Task 4: Password Security and Hashing

The process of hashing a sample password using the SHA-256 algorithm and applying a salt to the password before hashing.

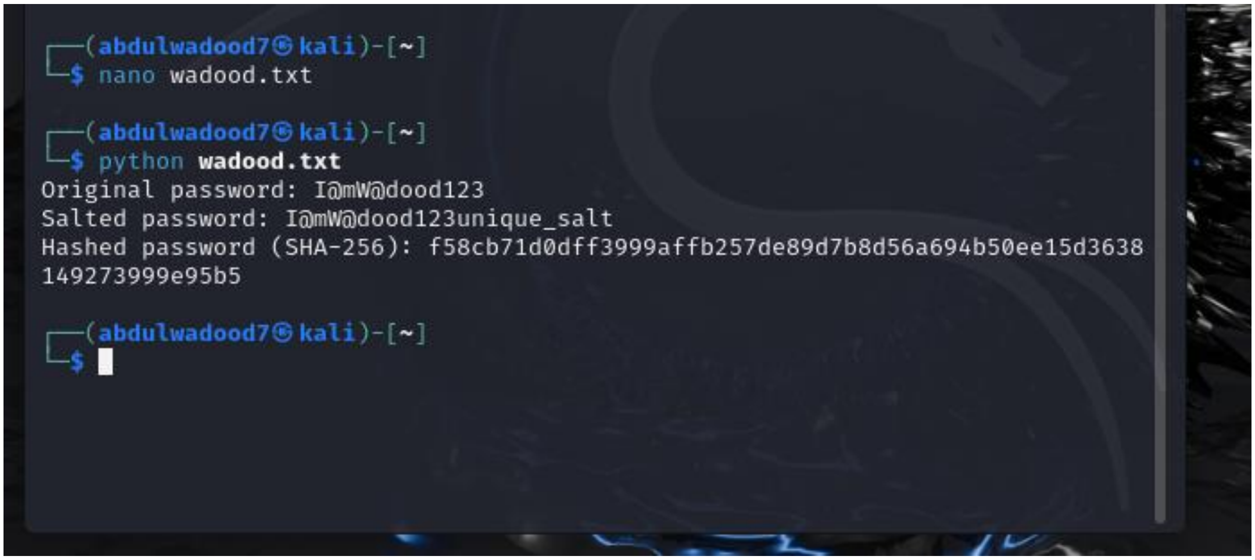
Script:

```
import hashlib

# Sample password
password = "I@mW@dood123"

# Add a salt
salt = "unique_salt"
salted_password = password + salt

# Hash the salted password using SHA-256
sha256_hash = hashlib.sha256(salted_password.encode()).hexdigest()

print("Original password:", password)
print("Salted password:", salted_password)
print("Hashed password (SHA-256):", sha256_hash)
```

We can modify the password and salt variables to experiment with different values and observe the changes in the hashed output.

Output:

The output shows the original password, the salted password, and the resulting hashed value.



```
┌──(abdulwadood7㉿kali)-[~]
└─$ nano wadood.txt

┌──(abdulwadood7㉿kali)-[~]
└─$ python wadood.txt
Original password: I@mW@dood123
Salted password: I@mW@dood123unique_salt
Hashed password (SHA-256): f58cb71d0dff3999affb257de89d7b8d56a694b50ee15d3638
149273999e95b5

┌──(abdulwadood7㉿kali)-[~]
└─$ 
```

Hashing:

Hashing is the process of transforming a piece of data (such as a password) into a fixed-length, unique string of characters called a hash value or hash.

Salting:

Salting is the process of adding a unique, random string (the "salt") to the password before hashing it.

Hashing and salting are essential in cybersecurity for the following reasons:

1. **Password Security**: Storing passwords in plain text is a major security risk. If the database containing the passwords is breached, all the passwords can be easily accessed and misused. Hashing the passwords using a strong algorithm like SHA-256 makes it much harder for attackers to recover the original passwords.
2. **Resistance to Rainbow Table Attacks**: Rainbow tables are precomputed tables of hashed values that can be used to quickly reverse-engineer hashed passwords. Adding a salt to the password before hashing makes each password unique, preventing the use of rainbow tables to crack the hashes.

3. **Increased Complexity**: When a salt is added to the password before hashing, the resulting hash becomes much more complex and unpredictable. This makes it significantly harder for attackers to crack the hashes using brute force or other attack methods.
4. **Slower Cracking**: Hashing algorithms like SHA-256 are designed to be computationally intensive, which slows down the process of cracking hashed passwords. This makes it more difficult for attackers to quickly guess or reverse-engineer the original passwords.
5. **Unique Hashes**: Even if two users have the same password, the addition of a unique salt ensures that their hashed passwords are different. This prevents attackers from exploiting shared or common passwords.

Hashing and salting are essential security practices that protect user passwords and make it much harder for attackers to gain unauthorized access to sensitive information.

Task 5: Basic Threat Identification

Common Security Threats in Web Applications:

1. Injection Attacks

Summary: Injection attacks occur when an attacker sends untrusted data to an interpreter, causing unintended commands to be executed. This can happen through SQL, NoSQL, OS, and LDAP injection.

Example: In a SQL injection attack, an attacker might input "' OR '1'='1" into a login form. If the application does not properly sanitize inputs, this could allow the attacker to bypass authentication and gain unauthorized access to the database.

2. Broken Authentication

Summary: Broken authentication refers to vulnerabilities that allow attackers to compromise user accounts. This can occur through poorly implemented authentication mechanisms, session management flaws, or predictable login credentials.

Example: An attacker could exploit a forgotten password feature that sends a password reset link without proper validation. By guessing or manipulating email addresses, the attacker can reset passwords for other users, gaining unauthorized access to their accounts.

3. Sensitive Data Exposure

Summary: Sensitive data exposure occurs when applications do not adequately protect sensitive information like passwords, credit card numbers, or personal data. This can lead to data breaches and identity theft.

Example: If a web application stores passwords in plain text instead of using hashing and salting, an attacker who gains access to the database can easily read and misuse these credentials.

4. Cross-Site Scripting (XSS)

Summary: XSS attacks occur when an application includes untrusted data in a web page without proper validation or escaping. This allows attackers to inject scripts into web pages viewed by other users.

Example: An attacker might input a malicious script in a comment section of a blog. When users view the page, the script executes in their browsers, potentially stealing cookies or redirecting them to malicious sites.

5. Security Misconfiguration

Summary: Security misconfiguration refers to improper security settings in an application, server, or database, often due to default settings or incomplete setups. This can expose vulnerabilities that attackers can exploit.

Example: If an application uses default credentials for its database or web server, an attacker can easily log in and manipulate data. Additionally, enabling unnecessary services can create potential entry points for attacks.

Conclusion:

Understanding these common security threats is crucial for web application developers and security professionals. Implementing best practices from resources like the OWASP Top 10 can significantly mitigate these risks and enhance the overall security of web applications.