Here are the Python and Java scripts for each task:

---

1. Analyze and visualize IP address allocation and subnetting

(Python | Scapy & Matplotlib)

```
import scapy.all as scapy
import matplotlib.pyplot as plt
from collections import Counter

def scan_network(network):
    arp_request = scapy.ARP(pdst=network)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_request_broadcast = broadcast / arp_request
    answered = scapy.srp(arp_request_broadcast, timeout=2, verbose=False)[0]

    ip_addresses = [entry[1].psrc for entry in answered]
    return ip_addresses

def visualize_ip_distribution(ip_addresses):
    ip_counts = Counter(ip_addresses)
    plt.bar(ip_counts.keys(), ip_counts.values())
    plt.xticks(rotation=45)
    plt.xlabel("IP Address")
    plt.ylabel("Occurrences")
    plt.title("IP Address Allocation")
    plt.show()

network_range = "192.168.1.1/24"
ips = scan_network(network_range)
visualize_ip_distribution(ips)
```

---

2. Modify Linux network configuration files

(Python | /etc/hosts, /etc/resolv.conf, /etc/network/interfaces)

```
def modify_hosts():
    with open("/etc/hosts", "a") as file:
        file.write("\n127.0.0.2 custom.local\n")
```

```
def modify_resolv():
    with open("/etc/resolv.conf", "a") as file:
        file.write("\nnameserver 8.8.8.8\n")

def modify_interfaces():
    with open("/etc/network/interfaces", "a") as file:
        file.write("\nauto eth0\niface eth0 inet dhcp\n")

modify_hosts()
modify_resolv()
modify_interfaces()
```

---

3. Capture and analyze active TCP/IP daemons

(Python | psutil & netstat)

```
import psutil
import os

def active_daemons():
    for conn in psutil.net_connections(kind='inet'):
        if conn.status == 'LISTEN':
            print(f"PID: {conn.pid}, Address: {conn.laddr}")

os.system("netstat -tulnp")
active_daemons()
```

---

4. Simple network daemon to log incoming connections

(Python | socket)

```
import socket

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("0.0.0.0", 8080))
server.listen(5)

while True:
    client, addr = server.accept()
    print(f"Connection from {addr}")
    with open("connections.log", "a") as log:
        log.write(f"New connection from {addr}\n")
```

---

5. Scan and list open ports on a target machine

(Python | socket & nmap)

```
import socket
import nmap

target = "127.0.0.1"
nm = nmap.PortScanner()
nm.scan(target, "1-1024")

for port in nm[target]['tcp']:
    print(f"Port {port} is open")
```

---

6. Extract and display IP settings (Java)

```
import java.net.*;

public class NetworkInfo {
    public static void main(String[] args) throws Exception {
        for (NetworkInterface netIf : NetworkInterface.getNetworkInterfaces()) {
            System.out.println("Interface: " + netIf.getName());
            for (InetAddress addr : netIf.getInetAddresses()) {
                System.out.println("IP Address: " + addr.getHostAddress());
            }
        }
    }
}
```

---

7. Log all incoming and outgoing network connections

(Python | psutil)

```
import psutil

def log_connections():
    with open("network_log.txt", "a") as log:
        for conn in psutil.net_connections():
            log.write(f"{conn.laddr} -> {conn.raddr} [{conn.status}]\n")

log_connections()
```

---

8. Monitor unauthorized changes to network config files

(Python | /etc/network/interfaces, /etc/resolv.conf)

```
import hashlib
import time

files = ["/etc/network/interfaces", "/etc/resolv.conf"]
hashes = {f: hashlib.md5(open(f, 'rb').read()).hexdigest() for f in files}

while True:
    time.sleep(10)
    for f in files:
        new_hash = hashlib.md5(open(f, 'rb').read()).hexdigest()
        if new_hash != hashes[f]:
            print(f"WARNING: {f} has been modified!")
            hashes[f] = new_hash
```

---

9. Encrypt and decrypt files before FTP transfer

(Python | PyCryptodome & ftplib)

```
from Crypto.Cipher import AES
import os

def encrypt_file(filename, key):
    cipher = AES.new(key, AES.MODE_EAX)
    data = open(filename, "rb").read()
    ciphertext, tag = cipher.encrypt_and_digest(data)
    open(filename + ".enc", "wb").write(cipher.nonce + tag + ciphertext)

key = os.urandom(16)
encrypt_file("test.txt", key)
```

---

10. SSH brute-force attack detection

(Python | Paramiko)

```
import paramiko
import socket

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("0.0.0.0", 22))
server.listen(5)

while True:
    client, addr = server.accept()
    print(f"Failed SSH login attempt from {addr}")
    with open("ssh_brute_force.log", "a") as log:
        log.write(f"Failed SSH login from {addr}\n")
```

Here are the Python scripts for each of the requested tasks:

---

8. HMAC-Based Authentication System

(Python | HMAC Module)

```
import hmac

import hashlib



secret_key = b"my_secret_key"



def generate_hmac(message):

    return hmac.new(secret_key, message.encode(), hashlib.sha256).hexdigest()



def verify_hmac(message, hmac_to_check):

    return hmac.compare_digest(generate_hmac(message), hmac_to_check)
```

```
msg = "Authenticate this message"
hmac_code = generate_hmac(msg)
print("Generated HMAC:", hmac_code)

print("Verification:", verify_hmac(msg, hmac_code))  # Should return True
```

---

19. Digital Signatures using PyCryptodome

(Python | PyCryptodome - RSA Signature)

```
from Crypto.PublicKey import RSA
from Crypto.Signature import pkcs1_15
from Crypto.Hash import SHA256

# Generate key pair
key = RSA.generate(2048)
public_key = key.publickey()

message = b"Secure message"
hash_obj = SHA256.new(message)

# Sign the message
signature = pkcs1_15.new(key).sign(hash_obj)
print("Digital Signature:", signature.hex())
```

```
# Verify signature
try:
    pkcs1_15.new(public_key).verify(hash_obj, signature)
    print("Signature verified!")
except (ValueError, TypeError):
    print("Verification failed!")
```

---

20. Challenge-Response Authentication (OTP)

(Python | OTP using secrets module)

```
import secrets


def generate_challenge():
    return secrets.token_hex(8)  # 16-character challenge


def generate_response(challenge, secret_key):
    return hmac.new(secret_key.encode(), challenge.encode(), hashlib.sha256).hexdigest()


challenge = generate_challenge()
secret_key = "shared_secret"
response = generate_response(challenge, secret_key)


print("Challenge:", challenge)
print("Response:", response)
```

21. Two-Factor Authentication (2FA) using pyotp

(Python | pyotp - TOTP-based 2FA)

```
import pyotp

# Generate a base32 secret key for the user
secret = pyotp.random_base32()
totp = pyotp.TOTP(secret)

print("Your OTP Secret Key:", secret)
print("Current OTP:", totp.now())

# Verify OTP
user_input = input("Enter OTP: ")
if totp.verify(user_input):
    print("Authentication successful!")
else:
    print("Invalid OTP!")
```

22. Compare SHA-256 and bcrypt Hashing Performance

(Python | hashlib & bcrypt)

```
import hashlib
import bcrypt
import time

password = b"securepassword"

# SHA-256 hashing
start = time.time()
sha256_hash = hashlib.sha256(password).hexdigest()
end = time.time()
print("SHA-256 Hash:", sha256_hash)
print("SHA-256 Time:", end - start)

# bcrypt hashing
start = time.time()
bcrypt_hash = bcrypt.hashpw(password, bcrypt.gensalt())
end = time.time()
print("bcrypt Hash:", bcrypt_hash)
print("bcrypt Time:", end - start)
```

---

23. AES-GCM Encryption & Authentication

(Python | PyCryptodome - AES-GCM)

```
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

key = get_random_bytes(16)  # 128-bit key
nonce = get_random_bytes(12)  # 96-bit nonce
cipher = AES.new(key, AES.MODE_GCM, nonce=nonce)

message = b"Confidential data"
ciphertext, tag = cipher.encrypt_and_digest(message)

# Decrypt and verify
decipher = AES.new(key, AES.MODE_GCM, nonce=nonce)
decrypted_message = decipher.decrypt_and_verify(ciphertext, tag)
print("Decrypted:", decrypted_message.decode())
```

---

24. Simulating a Ransomware Attack (Encrypt & Decrypt Files)

(Python | PyCryptodome - AES Encryption)

```
import os
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

key = b"thisisaverysecretkey123"  # Must be 16, 24, or 32 bytes
iv = b"thisisiv12345678"  # 16-byte IV
```

```
def encrypt_file(file_path):
    with open(file_path, "rb") as f:
        data = pad(f.read(), AES.block_size)

    cipher = AES.new(key, AES.MODE_CBC, iv)
    encrypted_data = cipher.encrypt(data)

    with open(file_path + ".enc", "wb") as f:
        f.write(encrypted_data)
    os.remove(file_path)

def decrypt_file(file_path):
    with open(file_path, "rb") as f:
        encrypted_data = f.read()

    cipher = AES.new(key, AES.MODE_CBC, iv)
    decrypted_data = unpad(cipher.decrypt(encrypted_data), AES.block_size)

    with open(file_path.replace(".enc", ""), "wb") as f:
        f.write(decrypted_data)
    os.remove(file_path)

# Encrypt all files in a folder
folder = "test_folder"
```

```
for file in os.listdir(folder):
    encrypt_file(os.path.join(folder, file))

# Decrypt files back
for file in os.listdir(folder):
    if file.endswith(".enc"):
        decrypt_file(os.path.join(folder, file))
```

---

25. Python-based Firewall Rule Tester

(Python | socket - Sending Test Packets)

```
import socket

def test_port(ip, port):
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.settimeout(1)  # Timeout for response
    result = sock.connect_ex((ip, port))
    if result == 0:
        print(f"Port {port} is OPEN")
    else:
        print(f"Port {port} is BLOCKED")
    sock.close()

# Example: Test firewall rules on ports
```

```
target_ip = "192.168.1.1"
test_ports = [22, 80, 443, 8080]

for port in test_ports:
    test_port(target_ip, port)
```

---

7. Compare Hash Functions (MD5, SHA-256, SHA-512)

(Python | File Integrity Verification)

```
import hashlib

def hash_file(file_path, algo):
    hasher = hashlib.new(algo)
    with open(file_path, "rb") as f:
        hasher.update(f.read())
    return hasher.hexdigest()

file = "test.txt"
print("MD5:    ", hash_file(file, "md5"))
print("SHA-256:", hash_file(file, "sha256"))
print("SHA-512:", hash_file(file, "sha512"))
```

---

1. Secure Login System using Java Security Manager

(Java | Security Manager Policies)

```
import java.io.FilePermission;
import java.security.Permission;

public class SecureLogin {
    public static void main(String[] args) {
        System.setSecurityManager(new SecurityManager() {
            @Override
            public void checkPermission(Permission perm) {
                if (perm instanceof FilePermission) {
                    throw new SecurityException("File access is restricted!");
                }
            }
        });

        System.out.println("Secure login system initialized.");
    }
}
```

2. Java HTTPS Server with SSL Encryption

(Java | SSL Libraries for Secure Communication)

```
import com.sun.net.httpserver.*;
import javax.net.ssl.*;
import java.io.*;
import java.net.InetSocketAddress;
import java.security.KeyStore;

public class SecureHttpServer {
    public static void main(String[] args) throws Exception {
        HttpsServer server = HttpsServer.create(new InetSocketAddress(8443), 0);
        SSLContext sslContext = SSLContext.getInstance("TLS");

        KeyStore ks = KeyStore.getInstance("JKS");
        ks.load(new FileInputStream("keystore.jks"), "password".toCharArray());

        KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509");
        kmf.init(ks, "password".toCharArray());

        sslContext.init(kmf.getKeyManagers(), null, null);
        server.setHttpsConfigurator(new HttpsConfigurator(sslContext));

        server.createContext("/", exchange -> {
            String response = "Secure Connection Established!";
            exchange.sendResponseHeaders(200, response.length());
            exchange.getResponseBody().write(response.getBytes());
            exchange.close();
        });

        server.start();
    }
}
```

---

3. AES Encryption for Secure Storage

(Java | AES for Data Security)

```
import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class AESEncryption {
    public static void main(String[] args) throws Exception {
        String key = "1234567890123456", data = "Sensitive Data";
        Cipher cipher = Cipher.getInstance("AES");
        SecretKeySpec secretKey = new SecretKeySpec(key.getBytes(), "AES");

        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        String encrypted = Base64.getEncoder().encodeToString(cipher.doFinal(data.getBytes()));
        System.out.println("Encrypted: " + encrypted);
        cipher.init(Cipher.DECRYPT_MODE, secretKey);

        String decrypted = new String(cipher.doFinal(Base64.getDecoder().decode(encrypted)));

        System.out.println("Decrypted: " + decrypted);

    }

}
```

---

4. SHA-256 File Integrity Check

(Java | SHA-256 Hashing for File Verification)

```
import java.io.*;

import java.security.*;


public class FileIntegrity {

    public static void main(String[] args) throws Exception {

        MessageDigest md = MessageDigest.getInstance("SHA-256");

        FileInputStream fis = new FileInputStream("test.txt");

        byte[] data = fis.readAllBytes();

        fis.close();


        byte[] hash = md.digest(data);

        System.out.println("SHA-256 Hash: " + bytesToHex(hash));

    }


    private static String bytesToHex(byte[] hash) {
        StringBuilder sb = new StringBuilder();
        for (byte b : hash) sb.append(String.format("%02x", b));
        return sb.toString();
    }
}
```

---

5. RSA Encryption & Decryption

(Java | RSA Secure Communication)

```
import javax.crypto.Cipher;
import java.security.*;

public class RSAEncryption {
    public static void main(String[] args) throws Exception {
        KeyPair keyPair = KeyPairGenerator.getInstance("RSA").generateKeyPair();
        Cipher cipher = Cipher.getInstance("RSA");

        cipher.init(Cipher.ENCRYPT_MODE, keyPair.getPublic());
        byte[] encrypted = cipher.doFinal("SecretMessage".getBytes());
        System.out.println("Encrypted: " + new String(encrypted));

        cipher.init(Cipher.DECRYPT_MODE, keyPair.getPrivate());
        System.out.println("Decrypted: " + new String(cipher.doFinal(encrypted)));
    }
}
```

---

6. Secure Authentication with Hashed Passwords (Salting)

(Java | Hashing Passwords for Secure Login)

```
import java.security.MessageDigest;
import java.security.SecureRandom;
import java.util.Base64;

public class SecureAuthentication {
    public static void main(String[] args) throws Exception {
        String password = "mypassword";
        byte[] salt = new byte[16];
        new SecureRandom().nextBytes(salt);

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        md.update(salt);
        byte[] hash = md.digest(password.getBytes());

        System.out.println("Salt: " + Base64.getEncoder().encodeToString(salt));
        System.out.println("Hashed Password: " + Base64.getEncoder().encodeToString(hash));
    }
}
```

Here are concise implementations for your requested tasks in **Python, JavaScript, and HTML**:

Substitution Techniques (Encryption & Decryption)

1. Caesar Cipher (Python)

```
def caesar_cipher(text, shift, decrypt=False):
    if decrypt: shift = -shift
    return ''.join(chr((ord(c) - 65 + shift) % 26 + 65) if c.isupper() else c for c in text)

message = "HELLO"
cipher_text = caesar_cipher(message, 3)
print("Encrypted:", cipher_text)
print("Decrypted:", caesar_cipher(cipher_text, 3, True))
```

---

2. Playfair Cipher (Python)

```
from itertools import product

def create_playfair_matrix(key):
    key = "".join(dict.fromkeys(key.upper().replace('J', 'I') +
"ABCDEFGHIKLMNOPQRSTUVWXYZ"))
    return [list(key[i:i+5]) for i in range(0, 25, 5)]

def encrypt_playfair(text, key):
    matrix = create_playfair_matrix(key)
    pairs = [(text[i], text[i+1] if i+1 < len(text) else 'X') for i in range(0, len(text), 2)]
    return ''.join(find_playfair_pair(matrix, a, b) for a, b in pairs)

def find_playfair_pair(matrix, a, b):
    pos = {matrix[r][c]: (r, c) for r, c in product(range(5), repeat=2)}
    r1, c1, r2, c2 = *pos[a], *pos[b]
    return matrix[r1][(c1+1)%5] + matrix[r2][(c2+1)%5] if r1 == r2 else \
        matrix[(r1+1)%5][c1] + matrix[(r2+1)%5][c2] if c1 == c2 else \
        matrix[r1][c2] + matrix[r2][c1]

print(encrypt_playfair("HELLO", "KEY"))
```

---

3. Hill Cipher (Python)

```
import numpy as np

def hill_cipher(text, key_matrix):
    text_vector = np.array([ord(c) - 65 for c in text]).reshape(-1, 2)
    result = (np.dot(text_vector, key_matrix) % 26) + 65
    return ''.join(chr(int(c)) for row in result for c in row)

key = np.array([[3, 2], [5, 7]])
message = "HI"
cipher_text = hill_cipher(message, key)
print("Encrypted:", cipher_text)
```

---

4. Vigenère Cipher (Python)

```
def vigenere_cipher(text, key, decrypt=False):
    key = (key * (len(text) // len(key) + 1)).upper()[:len(text)]
    shift = (-1 if decrypt else 1)
    return ''.join(chr((ord(t) + shift * (ord(k) - 65)) % 26 + 65) if t.isupper() else t for t, k in zip(text, key))

print("Encrypted:", vigenere_cipher("HELLO", "KEY"))
print("Decrypted:", vigenere_cipher(vigenere_cipher("HELLO", "KEY"), "KEY", True))
```

Transposition Techniques (Encryption & Decryption)

5. Rail Fence Cipher (Python)

```
def rail_fence(text, rails):
    fence = [[] for _ in range(rails)]
    direction, row = 1, 0
    for char in text:
        fence[row].append(char)
        row += direction
        if row == 0 or row == rails-1:
            direction *= -1
    return ''.join(sum(fence, []))
```

```
print("Encrypted:", rail_fence("HELLOWORLD", 3))
```

6. Row & Column Transformation (Python)

```
import numpy as np

def row_column_transposition(text, cols):
    rows = int(np.ceil(len(text) / cols))
    matrix = np.array(list(text) + [' '] * (rows * cols - len(text))).reshape(rows, cols)
    return ''.join(matrix[:, i].tolist() for i in range(cols))

print("Encrypted:", row_column_transposition("HELLOWORLD", 3))
```

Practical Applications

7. Data Encryption Standard (DES) (Python)

```
from Crypto.Cipher import DES

key = b"8charkey"
cipher = DES.new(key, DES.MODE_ECB)
message = b"HELLO123"
cipher_text = cipher.encrypt(message)
print("Encrypted:", cipher_text.hex())
```

8. AES Encryption (Python)

```
from Crypto.Cipher import AES
import os

key = os.urandom(16)
cipher = AES.new(key, AES.MODE_EAX)
ciphertext, tag = cipher.encrypt_and_digest(b"Sensitive Data")
print("Encrypted:", ciphertext.hex())
```

9. RSA Algorithm (HTML & JavaScript)

```
<!DOCTYPE html>
<html>
<head><script
src="https://cdnjs.cloudflare.com/ajax/libs/jsencrypt/3.0.0/jsencrypt.min.js"></script></head>
<body>
<script>
   let rsa = new JSEncrypt();
   let publicKey = rsa.getPublicKey();
   let privateKey = rsa.getPrivateKey();

   rsa.setPublicKey(publicKey);
   let encrypted = rsa.encrypt("Secure Data");
   console.log("Encrypted:", encrypted);

   rsa.setPrivateKey(privateKey);
   console.log("Decrypted:", rsa.decrypt(encrypted));
</script>
</body>
</html>
```

10. Diffie-Hellman Key Exchange (Python)

```
import random

p, g = 23, 5
a, b = random.randint(1, p), random.randint(1, p)
A, B = pow(g, a, p), pow(g, b, p)
shared_secret_a, shared_secret_b = pow(B, a, p), pow(A, b, p)

print("Shared Secret:", shared_secret_a == shared_secret_b)
```

11. SHA-1 Message Digest (Python)

```
import hashlib

message = "Hello"
digest = hashlib.sha1(message.encode()).hexdigest()
print("SHA-1 Digest:", digest)
```

12. Digital Signature Standard (Python)

```
from Crypto.PublicKey import RSA
from Crypto.Signature import pkcs1_15
from Crypto.Hash import SHA256

key = RSA.generate(2048)
message = b"Verify This"
hash_obj = SHA256.new(message)
signature = pkcs1_15.new(key).sign(hash_obj)

print("Signature:", signature.hex())
```

---

13. Intrusion Detection System (Snort)

```
sudo apt-get install snort
sudo snort -A console -i eth0 -c /etc/snort/snort.conf
```

---

14. Vulnerability Assessment (N-Stalker)

1. **Download & Install**: N-Stalker
2. **Scan Web Apps**: Run a vulnerability scan on target URLs.
