Unit 7: Security

1. Ethics revisited

Learning objectives

1.5 Ethics
Candidates should be able to:
• show understanding of computer ethics, including copyright issues and plagiarism
• distinguish between free software, freeware and shareware
• show understanding of the ethical issues raised by the spread of electronic communication and computer systems, including hacking, cracking and production of malware

Introduction

Discussion & learning items

Exercises

Further practice & resources

2. Protecting data

Learning objectives

1.2.2 Security aspects

- show understanding of the security aspects of using the Internet and understand what methods are available to help minimise the risks
- show understanding of the Internet risks associated with malware, including viruses, spyware and hacking
- explain how anti-virus and other protection software helps to protect the user from security risk

1.4.2

- show understanding of how data are kept safe when stored and transmitted, including:
    - use of passwords, both entered at a keyboard and biometric
    - use of firewalls, both software and hardware, including proxy servers
    - use of security protocols such as Secure Socket Layer (SSL) and Transport Layer Security (TLS)
    - ~~use of symmetric encryption (plain text, cypher text and use of a key) showing understanding that increasing the length of a key increases the strength of the encryption~~

Introduction

with passwords, biometric, firewalls, encryption

Discussion & learning items

Exercises

Further practice & resources

3, 4. Programming & inspecting sockets

Learning objectives

- use of security protocols such as Secure Socket Layer (SSL) and Transport Layer Security (TLS)

Introduction

Discussion & learning items

Connecting using sockets in cleartext
Wireshark inspection of clear text
Connecting using sockets with SSL
Wireshark inspection

Exercises

Further practice & resources

5, 6. Symmetric encryption

Learning objectives

- use of symmetric encryption (plain text, cypher text and use of a key) showing understanding that increasing the length of a key increases the strength of the encryption

Introduction

Discussion & learning items

Exercises

symmetric encryption program

Further practice & resources

This example uses the pycryptodome library.

- To install on your local machine, pip install `pycryptodemo`
- To install on repl.it, install via the packages icon ----------> 

```
# Based on examples provided here
# https://pycryptodome.readthedocs.io/en/latest/src/examples.html
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

def generate_key():
    # Your key must be 16, 24 or 32 bytes long
    key = get_random_bytes(32)
    return key

def encrypt( message, key ):
    byte_array = message.encode("utf-8")
    # If we were given a string, convert it to a bytes array
    if type(key) == str:
        key = key.encode("utf-8")
    if len(key) not in [16,24,32]:
        print("Key must be 16, 24 or 32 bytes long")
        exit()
    cipher = AES.new(key, AES.MODE_EAX)
    ciphertext, tag = cipher.encrypt_and_digest(byte_array)
    return ciphertext, tag, cipher.nonce
```

```
def decrypt( encrypted, key, tag, nonce):
    key = key.encode("utf-8")
    cipher = AES.new(key, AES.MODE_EAX, nonce)
    byte_array = cipher.decrypt_and_verify(encrypted, tag)
    return byte_array.decode('utf-8')

def main():
    while True:
        task = input("(e)ncrypt, (d)ecrypt or (q)uit?")
        if task == "e": # Encrypt
            message = input("Your secret message: ")
            key = input("Your encryption key: ")
            ciphertext, tag, nonce = encrypt(message, key)
            print("You can safely send these...")
            print("Cipher text: "+ciphertext.hex())
            print("Tag:         "+tag.hex())
            print("Nonce:       "+nonce.hex())
        elif task == "d": # Decrypt
            ciphertext = input("Encrypted text (hex): ")
            tag = input("Your ciphertext tag (hex): ")
            nonce = input("Your ciphertext nonce (hex): ")
            key = input("Your encryption key (hex): ")
            # Convert from hex strings to byte array
            ciphertext = bytearray.fromhex(ciphertext)
            tag = bytearray.fromhex(tag)
            nonce = bytearray.fromhex(nonce)
            message = decrypt(ciphertext, key, tag, nonce)
            print(message)
        elif task == "q": # Quit
            return()

main()
```

Here is a message for you to decrypt. It has a 16 byte key that should be relatively easy for you to guess.

```
Cipher text:
4f7658764d3a538d3a8d8051187199345d9aa38bc6755d40861dc13d096859e1415ebbb69216d7e9608374
fd936b2d06785242413e04b4e1d50bdb32adec8230d6ea889707ab0ba19b20515336bc44c0f07ed54126c7
bb473670b4c0e64eefce56dbaa5ad4c8f07aecc4b6fa
Tag:         70927dcb435058b6a94a280cfbff4fff
Nonce:       16ea29729cda6a86c986ec3dad0b147a
```

7. Hashing algorithms compared

Learning objectives

- Hashing algorithms are a critical tool for ensuring the integrity of data transmitted

Introduction

Discussion & learning items

Discussion comparing the SHA* algorithms
https://crypto.stackexchange.com/a/68314

Exercises

md5
sha1
sha2
https://docs.python.org/3/library/hashlib.html

```
import hashlib

original = input("Message: ")

# Convert string to array of bytes
byte_array = original.encode('utf-8')

md5 = hashlib.md5(byte_array).digest()
sha1 = hashlib.sha1(byte_array).digest()
sha3_256 = hashlib.sha256(byte_array).digest()

print("md5 = "+md5.hex())
print("sha1 = "+sha1.hex())
print("sha3_256 = "+sha3_256.hex())
```

Further practice & resources

Hashing Algorithms and Security - Computerphile (Tom Scott)
https://www.youtube.com/watch?v=b4b8ktEV4Bg

Rainbow tables of hashes - just how vulnerable is md5 and sha1?
https://project-rainbowcrack.com/table.htm

SHA: Secure Hashing Algorithm - Computerphile ~10mins
Primer on "how to write a hashing algorithm"
https://www.youtube.com/watch?v=DMtFhACPnTY

8. Protecting projects from DoS, phishing, pharming

1.4.3

- show understanding of the need to keep online systems safe from attacks including denial of service attacks, phishing, pharming

Learning objectives

Introduction

Discussion & learning items

Exercises

Further practice & resources

9. Protecting data from corruption, human error, malicious action, malware

Learning objectives

1.4.1
Candidates should be able to:
• show understanding of the need to keep data safe from accidental damage, including corruption and human errors
• show understanding of the need to keep data safe from malicious actions, including unauthorised viewing, deleting, copying and corruption

1.4.4
Candidates should be able to:
• describe how the knowledge from 1.4.1, 1.4.2 and 1.4.3 can be applied to real-life scenarios including, for example, online banking, shopping

Introduction

Discussion & learning items

Exercises

Further practice & resources

How not to store passwords
https://www.youtube.com/watch?v=8ZtInClXe1Q

- Clear text
- Same encryption key for all passwords
- Hashing of the raw password
- Use random salts!

10. Quiz