Cyber Security Assignment 2

Course: CS 503

Department of Computer Science & Engineering

Contents

1 Identity Theft in Cybercrime

1.1 What is Identity Theft?

Identity theft is a serious cybercrime where an attacker illegally obtains and uses another person's private and personal information for fraudulent purposes. The goal is often financial gain, but it can also be to impersonate the victim, commit other crimes under their name, or damage their reputation. The stolen information, known as Personally Identifiable Information (PII), can range from basic details like name and address to highly sensitive data like bank account numbers, credit card details, and social security numbers.

1.2 How Attackers Steal Identities Online

Attackers employ a variety of sophisticated techniques to steal identities. These methods often exploit human psychology, technical vulnerabilities, or a lack of security awareness.

1.2.1 Common Attack Vectors

Here are the most prevalent methods used by cybercriminals:

Table 1: Methods of Online Identity Theft

Method	Description
Phishing	Attackers send fraudulent emails or messages that appear to be from legitimate sources (e.g., banks, social media sites). These messages trick victims into revealing sensitive information like passwords and credit card numbers.
Malware	Malicious software, such as spyware, keyloggers, and trojans, is secretly installed on a victim's computer. This software can capture keystrokes, steal files, and transmit PII back to the attacker.
Data Breaches	Cybercriminals attack corporate or government databases to steal massive amounts of customer or user data. This stolen data is often sold on the dark web.
Social Engineering	This is the psychological manipulation of people into performing actions or divulging confidential information. Phishing is a form of social engineering, but it can also include pretexting (creating a fake scenario) or baiting.
Wi-Fi Eavesdropping	On unsecured public Wi-Fi networks, attackers can intercept data transmitted between a user's device and the internet. This is known as a "Man-in-the-Middle" (MitM) attack.
Pharming	A technical attack where a user is redirected to a fraudulent website even if they type the correct URL. This is done by poisoning the DNS server or compromising the user's host file.

1.2.2 Key Points and Prevention

- **Be Skeptical:** Always be cautious of unsolicited emails or messages asking for personal information.
- **Use Strong Passwords:** Create complex, unique passwords for different accounts and use a password manager.
- **Enable Two-Factor Authentication (2FA):** This adds an extra layer of security.
- **Secure Your Devices:** Keep your operating system, browser, and antivirus software up-to-date.

- **Avoid Public Wi-Fi for Sensitive Transactions:** If you must use public Wi-Fi, use a VPN (Virtual Private Network).
- **Monitor Your Accounts:** Regularly check your bank and credit card statements for any suspicious activity.

2 Web Jacking

2.1 What is Web Jacking?

Web Jacking is a type of cyber attack where a perpetrator hijacks a website and takes unauthorized control of it. The "jacking" can occur at different levels, from modifying the visual content of a webpage to redirecting the entire domain to a malicious server. The attacker's motive can be anything from activism (hacktivism), fraud, espionage, or simply causing disruption. Web jacking is a broad term that encompasses several specific attack types like DNS hijacking, BGP hijacking, and session hijacking.

2.2 Real-World Examples

2.2.1 Twitter DNS Hijacking (2009)

In December 2009, the popular social media platform Twitter was targeted. Attackers managed to gain access to the administrative credentials for Twitter's Domain Name System (DNS) records.

- **Method:** They modified the DNS records for twitter.com.
- **Impact:** When users tried to visit Twitter, they were redirected to a page displaying a message from a group calling themselves the "Iranian Cyber Army." The site was down for about an hour. This attack highlighted the vulnerability of a single point of failure (the DNS provider) for even the largest web services.

2.2.2 New York Times and others vs. Syrian Electronic Army (2013)

In August 2013, a hacktivist group known as the Syrian Electronic Army (SEA) launched a major attack against several high-profile media websites, including The New York Times, Twitter, and The Huffington Post.

- **Method:** The SEA used sophisticated spear-phishing techniques to gain credentials for Melbourne IT, the domain registrar for these companies. Once inside, they altered the DNS records.
- **Impact:** Visitors to nytimes.com were redirected to a site controlled by the SEA. The attack caused a major service outage for The New York Times that lasted for several hours. It was a powerful demonstration of how compromising a third-party registrar could bring down major online entities.

2.3 Advantages vs. Disadvantages for the Attacker

Table 2: Attacker's Perspective on Web Jacking

Advantages (for Attacker)	**Disadvantages (for Attacker)**
High impact and visibility.	Technically complex to execute successfully.
Can be used for widespread phishing or malware distribution.	Can be detected and mitigated quickly by security teams.
Damages the reputation and trust of the targeted brand.	Leaves digital footprints that can lead to identification.
Can cause significant financial loss to the victim.	Requires compromising a secure third-party like a domain registrar or DNS provider, which is difficult.

3 Password Strength Checker Program in Python

3.1 Program Overview

This Python program checks the strength of a user-provided password based on a set of defined criteria. It provides feedback on whether the password is "Weak", "Moderate", or "Strong".

3.1.1 Criteria for Strength

- **Length:** Minimum of 8 characters.
- **Complexity:** Must contain a mix of the following:
  - Uppercase letters (A-Z)
  - Lowercase letters (a-z)
  - Numbers (0-9)
  - Special characters (e.g., !, @, #, $, %)

3.1.2 Strength Levels

- **Strong:** Meets the length requirement and contains all four character types.
- **Moderate:** Meets the length requirement and contains three of the four character types.
- **Weak:** Does not meet the length requirement or contains two or fewer character types.

3.2 Python Code

```
import re

def check_password_strength(password):
    """
    Checks the strength of a password based on length and character types.
    Returns feedback: "Weak", "Moderate", or "Strong".
    """

    # --- Criteria flags ---
    length_ok = len(password) >= 8
    has_lower = re.search(r'[a-z]', password) is not None
    has_upper = re.search(r'[A-Z]', password) is not None
    has_digit = re.search(r'\d', password) is not None
    # Python's \W matches any non-alphanumeric character.
    has_special = re.search(r'\W', password) is not None

    # Calculate score based on criteria met
    # Length is the most important base criterion.
    if not length_ok:
        return "Weak (must be at least 8 characters long)"

    score = sum([has_lower, has_upper, has_digit, has_special])

    # --- Determine strength level ---
    if score == 4:
        return "Strong"
    elif score == 3:
        return "Moderate"
    else: # score is 2 or less
        return "Weak (include more character types)"

# --- Main part of the program ---
```

```python
if __name__ == "__main__":
    print("--- Password Strength Checker ---")
    print("A strong password should be at least 8 characters long and include uppercase, lowercase, numbers, and special characters.")

    # Get user input
    user_password = input("\nPlease enter a password to check: ")

    # Check strength and print feedback
    strength = check_password_strength(user_password)
    print(f"\nPassword Strength: {strength}")

    # Example Demonstrations
    print("\n--- Examples ---")
    passwords_to_test = ["password", "Password123", "P@ssword123!", "pass123"]
    for p in passwords_to_test:
        result = check_password_strength(p)
        print(f"'{p}': {result}")
```

Listing 1: Password Strength Checker in Python

4 Artificial Intelligence in Cybersecurity

4.1 How AI is Used to Improve Cybersecurity (Defense)

Artificial Intelligence (AI) and Machine Learning (ML) are transforming cybersecurity by enabling systems to predict, detect, and respond to threats faster and more effectively than traditional methods. AI's ability to analyze vast amounts of data and identify subtle patterns makes it a powerful tool for cyber defense.

Table 3: Defensive Applications of AI in Cybersecurity

Application	Description
Threat Detection	AI algorithms analyze network traffic, system logs, and user behavior to detect anomalies that may indicate a security breach. This is much faster and more comprehensive than manual analysis.
Spam Filtering	AI is used to analyze email content, sender reputation, and other factors to accurately identify and block spam and phishing attempts before they reach the user's inbox.
Vulnerability Mgmt.	AI systems can predict which vulnerabilities in a system are most likely to be exploited by attackers, helping security teams prioritize patching and remediation efforts.
Behavioral Analytics	User and Entity Behavior Analytics (UEBA) systems use AI to create a baseline of normal behavior for users and devices. Deviations from this baseline can trigger an alert for a potential insider threat or compromised account.
Automated Response	AI-powered Security Orchestration, Automation, and Response (SOAR) platforms can automatically take action against detected threats, such as quarantining an infected device or blocking a malicious IP address.

4.2 How Cybercriminals Use AI (Offense)

Unfortunately, just as AI can be used for defense, it can also be weaponized by attackers to create more sophisticated and evasive threats.

Table 4: Offensive Applications of AI by Cybercriminals

Application	Description
Advanced Phishing	AI can generate highly convincing and personalized spear-phishing emails automatically. It can learn a target's communication style from public data to create messages that are very difficult to distinguish from legitimate ones.
Automated Hacking	AI can automate the entire hacking process, from scanning for vulnerabilities to exploiting them and covering its tracks. This allows attackers to launch attacks at a scale and speed that is impossible for human hackers.
Evading Detection	Attackers use AI to create "polymorphic" malware that constantly changes its code to evade detection by traditional signature-based antivirus software.
Deepfakes	AI can be used to create realistic fake audio or video of individuals (e.g., a CEO authorizing a fraudulent wire transfer), making social engineering attacks much more believable and dangerous.
Cracking CAPTCHAs	Adversarial AI models can be trained to solve CAPTCHAs, the tests used to distinguish humans from bots, allowing attackers to automate the creation of fake accounts for spam or fraud.

4.3 Benefits and Risks of AI in Cyber Defense

Table 5: Benefits vs. Risks of AI in Cyber Defense

Benefits (Advantages)	**Risks (Disadvantages)**
Speed and Scale: AI can analyze data and identify threats far faster than humans.	**Adversarial AI:** Attackers can craft inputs to deceive AI models (e.g., slightly altering malware to bypass AI detection).
Predictive Analysis: AI can identify potential threats before they materialize based on global threat intelligence.	**False Positives/Negatives:** AI systems can be prone to errors, leading to legitimate activity being blocked or real threats being missed.
Automation: Frees up human analysts from repetitive tasks to focus on more complex strategic work.	**Complexity and Cost:** Implementing and maintaining sophisticated AI security systems can be expensive and requires specialized expertise.
Learning and Adaptation: ML models continuously learn from new data, improving their detection capabilities over time.	**Over-Reliance:** An excessive reliance on AI could lead to complacency and a decline in the skills of human security professionals.

5 Symmetric Encryption Program in Python

5.1 What is Symmetric Encryption?

Symmetric encryption is a type of encryption where a single key is used for both the encryption of plaintext and the decryption of ciphertext. The sender and the receiver must have the same secret key. This method is generally faster and less computationally intensive than asymmetric encryption. The main challenge is the secure distribution of the single key.

Table 6: Components of Symmetric Encryption

Component	Description
Plaintext	The original, readable message.
Key	A secret piece of information used to encrypt and decrypt data.
Encryption Algorithm	The set of rules (cipher) that transforms plaintext into ciphertext.
Ciphertext	The scrambled, unreadable message.
Decryption Algorithm	The process of turning the ciphertext back into plaintext using the key.

5.2 Python Program using `cryptography` library

This program uses the `Fernet` implementation from the popular `cryptography` library in Python. Fernet guarantees that a message encrypted using it cannot be manipulated or read without the key.

Prerequisite

You must first install the library:

```
pip install cryptography
```

5.2.1 Python Code

```
from cryptography.fernet import Fernet

def demonstrate_symmetric_encryption():
    """
    A simple demonstration of symmetric encryption using Fernet.
    1. Generates a key.
    2. Encrypts a message.
    3. Decrypts the message.
    """

    # 1. GENERATE A SECRET KEY
    # This key must be kept secret. In a real application, you would
    # save this key securely and share it only with the intended recipient.
    # The key must be a url-safe base64-encoded 32-byte key.
    key = Fernet.generate_key()

    # Create an instance of the Fernet cipher with the key
    cipher_suite = Fernet(key)

    print("--- Symmetric Encryption Demonstration ---")
    print(f"Generated Key (keep this secret!): {key.decode()}")
    print("-" * 30)

    # 2. ENCRYPT A MESSAGE
    # The message must be in bytes, so we encode the string.
    original_message = "This is a secret message that needs to be encrypted."
    message_bytes = original_message.encode('utf-8')

```

```
    print(f"Original Message: {original_message}")

    # Encrypt the message
    encrypted_message = cipher_suite.encrypt(message_bytes)

    print(f"Encrypted Message (Ciphertext): {encrypted_message.decode()}")
    print("-" * 30)

    # 3. DECRYPT THE MESSAGE
    # Now, we use the same key to decrypt the message.
    # If the wrong key were used, this would raise an InvalidToken error.
    try:
        decrypted_message_bytes = cipher_suite.decrypt(encrypted_message)
        decrypted_message = decrypted_message_bytes.decode('utf-8')

        print(f"Decrypted Message: {decrypted_message}")

        # Verify that the original and decrypted messages match
        assert original_message == decrypted_message
        print("\nSuccess: Original and decrypted messages are identical.")

    except Exception as e:
        print(f"Decryption failed: {e}")

# --- Run the demonstration ---
if __name__ == "__main__":
    demonstrate_symmetric_encryption()
```

Listing 2: Symmetric Encryption Demonstration in Python