IA 321: Information and Communication Systems Security

Instructor:
Dr. Jabhera Matogoro
Office# B13 – CIVE Administration Block
E-Mail: jabhera.matogoro@udom.ac.tz
Phone: +255 784 423 615

Course Aim

- The course will discuss the importance of information and Communication technology to any modern organization
- It also emphasizes the importance of developing secure information systems
- The course will cover diverse issues on security management.

Course Objective

- The main objectives of this course is to introduce to students the importance and techniques for ensuring security to information and communication technology (ICT) systems.
- The programming language for this course is JAVA.

Learning Outcomes

- Upon completion of this course, students should:
  1. Explain the fundamental concepts, principles, and mechanisms in ICT security
  2. Analyze the basic cryptographic concepts commonly used techniques and protocols
  3. Classify of common ICT vulnerabilities and techniques to address these vulnerabilities
  4. Provide background for advanced topics in ICT security and prepare students to work in information system security.

Learning Outcomes

- Upon completion of this course, students should:
  1. Explain the fundamental concepts, principles, and mechanisms in ICT security
  2. Analyze the basic cryptographic concepts commonly used techniques and protocols
  3. Classify of common ICT vulnerabilities and techniques to address these vulnerabilities
  4. Provide background for advanced topics in ICT security and prepare students to work in information system security.

Introduction to Java Cryptography

- Assignment#1: Write the History of Java computer programming language – when did it started, who inverted it, what was the main purpose, what was the first application of java, what is the difference between java and other programming language?, how java transformed the current Internet? Consumer devices? What is the relationship of java with oracle?

Java Cryptography

- Cryptography is the study of secure communications techniques that allow only the sender and intended recipient of a message to view its contents.
- Cryptography provides three services that are crucial in secure programming.
- These include a cryptographic cipher that protects the secrecy of your data; cryptographic certificates, which prove identity (authentication); and digital signatures, which ensure your data has not been damaged or tampered with.

Java Cryptography

- This course covers cryptographic programming in Java.
- Java 1.1 and Java 1.2 provided extensive support for cryptography with an elegant architecture, the Java Cryptography Architecture (JCA).
- Another set of classes, the Java Cryptography Extension (JCE), provides additional cryptographic functionality.
- The course covers the JCA and the JCE from top to bottom, describing the use of the cryptographic classes as well as their innards.

Secure Systems

- Computer applications enable people to do work.
- Applications are parts of a larger system (a business, usually) that also involves people, fax machines, white boards, credit cards, paper forms, and anything else that makes the whole system run.
- Secure systems make it hard for people to do things they are not supposed to do.

Secure Systems

- For example, a bank is designed as a secure system.
- You shouldn't be able to withdraw money from someone else's account, whether you try at the teller window, or by using the bank machine, or by telephone.
- Of course, you could bribe the teller or disassemble the bank machine, but these things are usually not worth the cost.
- Secure systems are designed so that the cost of breaking any component of the system outweighs the rewards.

Secure Systems

- Cost is usually measured in money, time, and risk, both legal and personal.
- The benefits of breaking systems are generally control, money, or information that can be sold for money.
- The security of the system should be proportional to the resources it protects; it should be a lot harder to break into a brokerage than a magazine subscription list, for example.
- The term "secure systems" is a little misleading; it implies that systems are either secure or insecure. In truth, there is no absolute security.

What can you do with cryptography

- Secure network communications
- Secure hard disk
- Secure email

Platform Security

- One of the things that makes Java so interesting is the security features that are built in to the platform itself.
- Java was designed to enable small programs, applets, to be downloaded and run without danger.
- Applets are nifty, but without the right precautions they would be very dangerous.
- Java's bytecode verifier, ClassLoader, and SecurityManager work in tandem to safely execute downloaded classes.

Platform Security…

- The Java Development Kit (JDK™) 1.2 includes some interesting security enhancements, including the concepts of protection domains, permissions, and policies.
- The security that the Java platform provides comes "for free" to application developers.
- Application-level security, however, needs to be developed into the application.
- This course is about programming application-level security through the use of cryptography.

Astute Inequalities

- At the 1997 JavaOne conference, the Java Security Architect, Li Gong, gave a presentation on Java security.
- One of his slides is particularly useful for understanding Java security and cryptography.
- It contains a list of five inequalities, to which I've added explanations.

Astute Inequalities

- Security != cryptography
- Correct security model != bug-free implementation
- Testing != formal verification
- Component security != overall system security
- Java security != applet containment

Message Digest

```
import java.io.*;
import java.security.*;
import sun.misc.*;
public class Masher {
 public static void main(String[] args) throws Exception {
 // Check arguments.
 if (args.length != 1) {
 System.out.println("Usage: Masher filename");
 return;
 }
 // Obtain a message digest object.
 MessageDigest md = MessageDigest.getInstance("MD5");
 // Calculate the digest for the given file.
 FileInputStream in = new FileInputStream(args[0]);
 byte[] buffer = new byte[8192];
 int length;
 while ((length = in.read(buffer)) != -1)
 md.update(buffer, 0, length);
 byte[] raw = md.digest();
 // Print out the digest in base64.
 BASE64Encoder encoder = new BASE64Encoder();
 String base64 = encoder.encode(raw);
 System.out.println(base64);
 }
}
```

- To use this program, just compile it and give it a file to digest.
- Here, I use the source code, Masher.java, as the file:

C:\ java Masher Masher.java
nfEOH/5M+yDLaxaJ+XpJ5Q==

Message Digest…

- A message digest is sometimes called a cryptographic hash.
- It's an example of a one-way function , which means that although you can calculate a message digest, given some data, you can't figure out what data produced a given message digest.
- Let's say that your friend, Josephine, wants to send you a file.
- She's afraid that your mutual enemy, Edith, will modify the file before it gets to you.

Message Digest…

- If Josephine sends the original file and the message digest, you can check the validity of the file by calculating your own message digest and comparing it to the one Josephine sent you.
- If Edith changes the file at all, your calculated message digest will be different and you'll know there's something awry.
- Of course, there's a way around this: Edith changes the file, calculates a new message digest for the changed file, and sends the whole thing to you.
- You have no way of knowing whether Edith has changed the file or not. Digital signatures extend message digests to solve this problem;

SecretWriting

- The next example uses classes that are found only in the Java Cryptography Extension (JCE).
- The JCE contains cryptographic software whose export is limited by the U.S. government.
- If you live outside the United States or Canada, it is not legal to download this software.
- The SecretWriting program encrypts and decrypts text. Here is a sample session: C:\ java SecretWriting -e Hello, world!
- The -e option encrypts data, and the -d option decrypts it.
- A cipher is used to do this work.

SecretWriting

- The cipher uses a key.
- Different keys will produce different results.
- SecretWriting stores its key in a file called SecretKey.ser.
- The first time you run the program, SecretWriting generates a key and stores it in the file.
- Subsequently, the key is loaded from the file.
- If you remove the file, SecretWriting will create a new key.
- Note that you must use the same key to encrypt and decrypt data.
- This is a property of a symmetric cipher.

Java Cryptography: Concepts and Architecture

- At the application programming level, there are many options for making a program secure.
- Correctly used, cryptography provides these standard security features:
  - Confidentiality assures you that data cannot be viewed by unauthorized people.
  - Integrity assures you that data has not been changed without your knowledge.
  - Authentication assures you that people you deal with are not imposters.
- Java cryptography software comes in two pieces, Java Development Kit (JDK) itself, which includes cryptographic classes for authentication and Java Cryptography Extension (JCE), includes so-called "strong cryptography.“

Random Numbers

- Random numbers are important for cryptography.
- Computers are not very good at producing truly random data.
- Instead, they rely on a pseudo-random number generator (PRNG).
- A cryptographically strong PRNG, seeded with truly random values, is a PRNG that does a good job of spewing out unpredictable data.
- But if the PRNG is not cryptographically strong, or if the seed data is not random, the security of your application can be compromised.

Assignment#2

- Develop a simple Java application that store students information in the database. These information may includes; first name, middle name, surname, registration number, student courses for first and second semester.
- The developed java application should accept username and password for each student and store them in a database.
- The password must be encrypted using MD5 algorithm.
- Make sure all concepts covered in chapter one to chapter four are included in your developed java application.