INTERNET FUNDAMENTALS

SECTION-D (NOTES)

SEM-4TH

Software Complexity

The term complexity stands for state of events or things, which have multiple interconnected links and highly complicated structures. In software programming, as the design of software is realized, the number of elements and their interconnections gradually emerge to be huge, which becomes too difficult to understand at once.

Software design complexity is difficult to assess without using complexity metrics and measures. Let us see three important software complexity measures.

Software complexity is a natural byproduct of the functional complexity that the code is attempting to enable. With multiple system interfaces and complex requirements, the complexity of software systems sometimes grows beyond control, rendering applications and portfolios overly costly to maintain and risky to enhance. Left unchecked, software complexity can run rampant in delivered projects, leaving behind bloated, cumbersome applications.

The software engineering discipline has established some common measures of software complexity. Perhaps the most common measure is the McCabe essential complexity metric. This is also sometimes called cyclomatic complexity. It is a measure of the depth and quantity of routines in a piece of code.

Using cyclomatic complexity measured by itself, however, can produce the wrong results. A module can be complex, but have few interactions with outside modules. A module can also be relatively simple, but highly coupled to many other modules, actually raising the overall complexity of the codebase beyond measure. In the first case, complexity metrics will look bad, while in the second the complexity metrics will look good – but the result will be deceptive. Thus, it is important to also measure the coupling and cohesion of the modules in the codebase to get a true system-level software complexity measure.

Cyclomatic Complexity Measures

Every program encompasses statements to execute in order to perform some task and other decision-making statements that decide, what statements need to be executed. These decision-making constructs change the flow of the program.

If we compare two programs of same size, the one with more decision-making statements will be more complex as the control of program jumps frequently.

Process to make flow control graph:

- Break program in smaller blocks, delimited by decision-making constructs.
- Create nodes representing each of these nodes.
- Connect nodes as follows:
    - If control can branch from block i to block j

      Draw an arc

    - From exit node to entry node

Function Point

It is widely used to measure the size of software. Function Point concentrates on functionality provided by the system. Features and functionality of the system are used to measure the software complexity.Function point counts on five parameters, named as External Input, External Output, Logical Internal Files, External Interface Files, and External Inquiry. To consider the complexity of software each parameter is further categorized as simple, average or complex.

External Input

Every unique input to the system, from outside, is considered as external input. Uniqueness of input is measured, as no two inputs should have same formats. These inputs can either be data or control parameters.

- **Simple** - if input count is low and affects less internal files
- **Complex** - if input count is high and affects more internal files
- **Average** - in-between simple and complex.

External Output

All output types provided by the system are counted in this category. Output is considered unique if their output format and/or processing are unique.

- **Simple** - if output count is low
- **Complex** - if output count is high
- **Average** - in between simple and complex.

Logical Internal Files

Every software system maintains internal files in order to maintain its functional information and to function properly. These files hold logical data of the system. This logical data may contain both functional data and control data.

- **Simple** - if number of record types are low
- **Complex** - if number of record types are high
- **Average** - in between simple and complex.

External Interface Files

Software system may need to share its files with some external software or it may need to pass the file for processing or as parameter to some function. All these files are counted as external interface files.

- **Simple** - if number of record types in shared file are low
- **Complex** - if number of record types in shared file are high
- **Average** - in between simple and complex.

Encryption

In cryptography, **encryption** is the process of encoding messages or information in such a way that only authorized parties can access it. Encryption does not of itself prevent interference, but denies the intelligible content to a would-be interceptor. In an encryption scheme, the intended information or message, referred to as plaintext, is encrypted using an encryption algorithm, generating ciphertext that can only be read if decrypted. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is in principle possible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, large computational resources and skill are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients but not to unauthorized users.

Types

Symmetric key / Private key

In symmetric-key schemes, the encryption and decryption keys are the same. Communicating parties must have the same key before they can achieve secure communication.

Public key

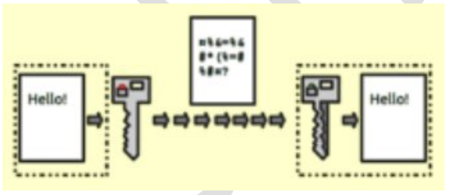

Illustration of how encryption is used within servers Public key encryption.

In public-key encryption schemes, the encryption key is published for anyone to use and encrypt messages. However, only the receiving party has access to the decryption key that enables messages to be read. Public-key encryption was first described in a secret document in 1973 before then all encryption schemes were symmetric-key (also called private-key).

A publicly available public key encryption application called Pretty Good Privacy (PGP) was written in 1991 by Phil Zimmermann, and distributed free of charge with source code; it was purchased by Symantec in 2010 and is regularly updated

Public key cryptography, or **asymmetric cryptography**, is any cryptographic system that uses pairs of keys: *public keys* which may be disseminated widely, and *private keys* which are known only to the owner. This accomplishes two functions: authentication, which is when the public key is used to verify that a holder of the paired private key sent the message, and encryption, whereby only the holder of the paired private key can decrypt the message encrypted with the public key.

In a public key encryption system, any person can encrypt a message using the public key of the receiver, but such a message can be decrypted only with the receiver's private key. For this to work it must be computationally easy for a user to generate a public and private key-pair to be used for encryption and decryption. The strength of a public key cryptography system relies on the degree of difficulty (computational impracticality) for a properly generated private key to be determined from its corresponding public key. Security then depends only on keeping the private key private, and the public key may be published without compromising security

Two of the best-known uses of public key cryptography are:

- *Public key encryption*, in which a message is encrypted with a recipient's public key. The message cannot be decrypted by anyone who does not possess the matching private key, who is thus presumed to be the owner of that key and the person associated with the public key. This is used in an attempt to ensure confidentiality.
- *Digital signatures*, in which a message is signed with the sender's private key and can be verified by anyone who has access to the sender's public key. This verification proves that the sender had access to the private key, and therefore is likely to be the person associated with the public key. This also ensures that the message has not been tampered with, as a signature is mathematically bound to the message it originally was made with, and verification will fail for practically any other message, no matter how similar to the original message.

SECURE WEB DOCUMENT:

Generally, secure websites use encryption and authentication standards to protect the confidentiality of web transactions.

- Currently, the most commonly used protocol for web security is TLS, or Transport Layer Security. This technology is still commonly referred to as SSL, or Secure Sockets Layer, a predecessor to TLS. In addition to providing security for HTTP (web hypertext) transactions, TLS works with other TCP/IP standards such as IMAP mail and LDAP directory access. For a security standard such as TLS/SSL to work, your browser and the web server must both be configured to use it.
- When you connect to a website using TLS, your browser asks the server to authenticate itself, or confirm its identity. The authentication process uses cryptography to verify that

a trusted independent third party, or certificate authority, such as Comodo, Thawte, or VeriSign, has registered and identified the server. TLS can also authenticate connecting users or their computers.

- In addition, TLS encrypts the data that you send, and incorporates a mechanism for detecting any alteration in transit, so that eavesdropping on or tampering with web traffic is almost impossible. This is essential for safely transmitting highly confidential information such as credit card numbers.
- Nearly all current browsers are set up by default to accept SSL certificates from most established certificate authorities, and to notify you when you are entering or leaving secure sites, including secure areas of comprehensive sites.

Digital signature

A **digital signature** is a mathematical scheme for demonstrating the authenticity of digital messages or documents. A valid digital signature gives a recipient reason to believe that the message was created by a known sender (authentication), that the sender cannot deny having sent the message (non-repudiation), and that the message was not altered in transit (integrity).

Digital signatures are a standard element of most cryptographic protocol suites, and are commonly used for software distribution, financial transactions, contract management software, and in other cases where it is important to detect forgery or tampering.

The digital equivalent of a handwritten signature or stamped seal, but offering far more inherent security, a digital signature is intended to solve the problem of tampering and impersonation in digital communications. Digital signatures can provide the added assurances of evidence to origin, identity and status of an electronic document, transaction or message, as well as acknowledging informed consent by the signer.

In many countries, including the United States, digital signatures have the same legal significance as the more traditional forms of signed documents. The United States Government Printing Office publishes electronic versions of the budget, public and private laws, and congressional bills with digital signatures.

A digital signature scheme typically consists of three algorithms;

- A *key generation* algorithm that selects a *private key* uniformly at random from a set of possible private keys. The algorithm outputs the private key and a corresponding *public key*.
- A *signing* algorithm that, given a message and a private key, produces a signature.
- A *signature verifying* algorithm that, given the message, public key and signature, either accepts or rejects the message's claim to authenticity.

Two main properties are required. First, the authenticity of a signature generated from a fixed message and fixed private key can be verified by using the corresponding public key. Secondly, it should be computationally infeasible to generate a valid signature for a party without knowing that party's private key. A digital signature is an authentication mechanism that enables the creator of the message to attach a code that acts as a signature. The Digital Signature Algorithm (DSA), developed by the National Institute of Standards and Technology, is one of many examples of a signing algorithm

How digital signatures work

Digital signatures are based on public key cryptography, also known as asymmetric cryptography. Using a public key algorithm such as RSA, one can generate two keys that are mathematically linked: one private and one public. To create a digital signature, signing software (such as an email program) creates a one-way hash of the electronic data to be signed. The private key is then used to encrypt the hash. The encrypted hash -- along with other information, such as the hashing algorithm -- is the digital signature. The reason for encrypting the hash instead of the entire message or document is that a hash function can convert an arbitrary input into a fixed length value, which is usually much shorter. This saves time since hashing is much faster than signing.

Firewall

In computing, a **firewall** is a network security system that monitors and controls the incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted, secure internal network and another outside network, such as the Internet, that is assumed not to be secure or trusted. Firewalls are often categorized as either *network firewalls* or *host-based firewalls*. Network firewalls filter traffic between two or more networks; they are either software appliances running on general purpose hardware, or hardware-based firewall computer appliances. Host-based firewalls provide a layer of software on one host that controls network traffic in and out of that single machine.[3][4] Firewall appliances may also offer other functionality to the internal network they protect, such as acting as a DHCP[5][6] or VPN[7][8][9][10] server for that network.

A firewall is a network security system designed to prevent unauthorized access to or from a private network. Firewalls can be implemented in both hardware and software, or a combination of both. Network firewalls are frequently used to prevent unauthorized Internet users from accessing private networks connected to the Internet, especially *intranets*. All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.

Hardware and Software Firewalls

Firewalls can be either hardware or software but the ideal configuration will consist of both. In addition to limiting access to your computer and network, a firewall is also useful for allowing remote access to a private network through secure authentication certificates and logins.

Hardware firewalls can be purchased as a stand-alone product but are also typically found in broadband routers, and should be considered an important part of your system and network set-up. Most hardware firewalls will have a minimum of four network ports to connect other computers, but for larger networks, business networking firewall solutions are available.

Software firewalls are installed on your computer (like any software) and you can customize it; allowing you some control over its function and protection features. A software firewall will protect your computer from outside attempts to control or gain access your computer.