Cryptography

Week 6

**Cryptography** – use of mathematical algorithms to transform info into a form where it’s not readable by unauthorized individuals.

2 principles of cryptography:

**Encryption –** converts information from plaintext form into cipher text

**Decryption** – converts cipher text into plaintext

Encryption algorithms have 2 inputs: plaint text message and encryption key.

The encryption key transforms plaintext into cipher text.

Decryption does the same thing, but turns cipher text into plain text

2 ways to categorize are symmetric or asymmetric.

**Symmetric –** encryption and decryption use the same key

**Asymmetric -** encryption and decryption use different keys from the same pair

The number of symmetric keys required for a group to communicate is n(n-1)/2 . Where n is the number of people in the group. As the group increases in size, the number of keys grows astronomically. Asymmetric keys solve this problem.

Asymmetric cryptography solves the number of key issue by having key pairs. 1 key is the public key that they can freely distribute to communication partners and a private key that they keep secret. In asymmetric cryptography, anything encrypted with one key from the pair can be decrypted with the other key from the same pair.

Asymmetric cryptography is slower than symmetric keys but it requires far fewer keys.

The 5 goals of cryptography are:

**Confidentiality:** No unauthorized access

**Integrity:** no unauthorized changes

**Authentication**: proof of identity claims

**Obfuscation**: hiding sensitive data

**Non-repudiation:** verification of origin (utilized with digital signatures). Non repudiation is only possible with asymmetric cryptography.

Cryptography involves a tradeoff between security strength and resource consumption.

**Code**: A system that substitutes one word or phrase for another intended to provide secrecy or efficiency.

**Cipher:** systems that use mathematical algorithms to encrypt and decrypt messages.

Ciphers have 2 different ways of processing a message:

1. **Stream ciphers** – operate on one character or bit of a message at a time
2. **Block ciphers** – operate on large segments of the message at the same time.

Ciphers perform operations using 2 things:

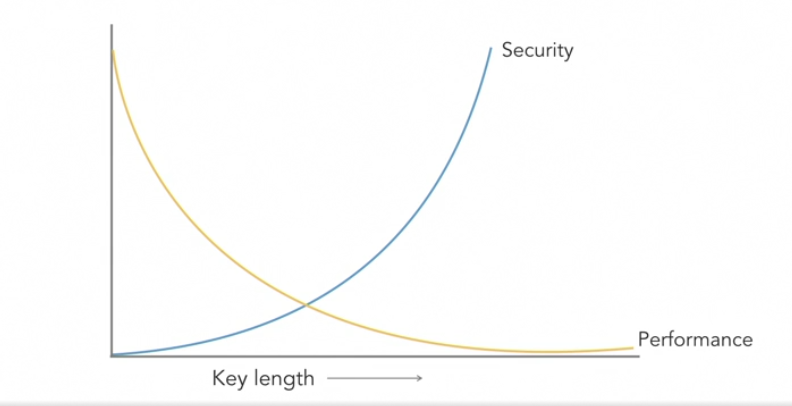
1. **Substitution ciphers**: change the characters in a message
2. **Transposition ciphers**: change the positions of the characters in a message.

To attain **confusion** every bit of the cipher text must depend upon more than one bit of the encryption key.

**Diffusion:** changing a single bit of the plaintext should change about 50% of the cipher text bits.

**Obfuscation**: Uses cryptography to hide source code from users (run the code but not view it)

When choosing an encryption algorithm you need to balance security and performance.



As algorithms age, they often become insecure. The lifecycle of an algorithm is:

1. Imitation
2. Development and acquisition
3. Implementation of assessment
4. Operations and maintenance
5. Sunset

Symmetric cryptography through history

* Data Encryption standard(DES)
* 3DES
* AES, Blowfish, Twofish: AES is still used today, it uses a combination of transposing and transforming. It is the successor to DES.
* RC4
* Cipher modes: describes how an algorithm encrypts and decrypts data. Some examples are: Electronic codebook mode, Cipher block chaining mode, counter mode
* Steganography : hides data in larger files like images (hiding information in plain sight)

Asymmetric cryptography through history:

* **Rivest, Shamir, Adelman (RSA)**: slow but still used today after 40 years. Involves selecting 2 very large prime numbers, the keys are generated. The user is responsible for keeping the private key secure. RSA is used to exchange symmetric keys. Key length is between 1,024 and 4,096.
* **PGP** and GnuPG – open source. Combines both symmetric and asymmetric cryptography. The process involves generating a random symmetric key, the random symmetric key is encrypts the message, and then e encrypts the random key with the recipient’s public key. The sender than transmits the message to the recipient. The decryption process involves the decrypting the random key using their private key, Producing the random key generated by the sender, which is then used to decrypt the message.
* **Elliptic curve and quantum cryptography**

For symmetric keys, sender and receiver must find a way to give each other the shared key securely. To exchange a secure key, they can use out of bound options. This can involve face to face, mailing or calling. Another way to do it I using In-band key exchange such as using the diffie hellman algorithm.

The diffie hellman algorithm involves providing symmetric key exchange capability.

The diffie hellman algorithm works by:

1. Both parties communicating a common value. i.e p=13 and g=7
2. Then each party member selects a secret number. i.e a=5 and b=8
3. Using the algorithm A = g^a mod p -> A=7^5 mod 13, person a sends the value of A to person b (A = 11)
4. Using the algorithm B = g^b mod p -> B = 7^8 mod 13, person b sends the value of B to person A (B = 3)
5. Person A then computes a shared secret number S with S=B^a mod p -> S=3^5 mod 13 (S=9)
6. Peron B then computes a shared secret number S with S=A^b mod p -> S =11^8 mod 13 (S=9)

Requirements for key exchange:

* 2 parties must be confident that they are really communicating with each other and that neither one is an imposter
* The 2 parties must be confident that nobody is eavesdropping on the key exchange. (Diffie Helman solves this)

3 ways to ensure that one of the parties that are communicating is not an imposter (these are all trust models):

1. Personal knowledge – cumbersome and difficult
2. Web of trust (WOT) – takes advantage of participants digitally signing the public keys of people they know personally. If this web of trust is large enough, it will make it easy to communicate. The draw backs are: Decentralized, High barrier to entry requires technical knowledge.
3. Public key infrastructure (PKI) – introduces the process of trusted certificate authorities instead of peer review. Certificate authorities (CAs) are trusted 3rd party organizations that verify the identity of individuals or organizations and then issue digital certification containing both identity information and a copy of the subject’s public key.

**Hash functions** – one way function that transforms a variable length input into a unique fixed length output. (Message digest is the same as hash functions)

Hash function characteristics:

* One way functions can’t be reserved
* The output will always be the same length regardless of input.
* No two inputs to a hash function should produce the same output.

Hash functions may fail if:

* They are reversible
* Not **collision resistant** (two inputs can have the same output)

The hash functions that were used:

* MD5(not secure)
* SHA-1(not secure)
* SHA-2( still used but same insecurities as sha1)
* SHA-3(secure)
* RIPEMD(160,256, 320 are secure)

One of the uses of hash functions are hash based message authentication code (HMAC). This combines symmetric cryptography and hashing to provide authentication and integrity.

Hash functions are used with asymmetric cryptography for digital signatures and digital certificates.

Digital signatures ensure:

* The owner of the public key is the person who signed the message (authentication)
* Message was not altered when signed(integrity)
* The recipient can prove these facts to a third party(non repudiation)

Digital signatures depend on:

* Collision resistant hash function
* Asymmetric cryptography

We use private key to create digital signatures.

Digital signing does not provide confidentiality as anyone can read the message.

**Brute force attack** – the attacker guesses repeatedly to get the key. The attacker only needs an example of cipher text. Brute force attacks are bad against keys with large lengths. Brute force is good for algorithms that may be vulnerable.

**Key space –** The set of all possible encryption keys usable with an algorithm.

**Knowledge based attacks:**

* Frequency analysis – pattern recognition with the cipher text
* The birthday attack – involves attackers finding two inputs with the same hash values

**Eavesdropping attacks:**

* Relies on a compromised communications path

Eavesdrop attacks can happen over the internet with a man in the middle attack where the attacker intercepts information from the user to the server.

To prevent replay attacks, information should include unique characteristics like a token or a time stamp.

**Implementation attacks** involve cryptographic systems containing implementation flaws that render them vulnerable to attack. Examples of implementation attacks are:

* Fault injection attacks – uses externally forced errors
* Side-channel attack – measures encryption footprints
* Timing attack – measure encryption time

Week 7

To prioritize what we want to do security scans on, we can look at 3 things:

1. Impact: What is the highest data classification handled in the system
2. Likelihood – what is the network exposure? What services are exposed
3. What impact does the system impact have on the business?

Penetration testing places security professionals in the roles of attackers. The goal is to test security controls. The penetration should have rules of engagements. There are 3 levels of understanding in penetration tests.

* White box: Attackers have full access to information before they begin the test
* Grey box: attackers have limited information before they begin the test
* Black box: attackers have no information about the target before they begin the test.

The penetration testing involves going back and forth between discovery and attack phases. The attack phase involves gaining access, escalating privileges, system browsing, and install additional tool (backdoor). Attackers can go back and forth between phases.

**Pivoting** is used by attackers after exploiting a vulnerability in a system, attackers use that system as a base from which to target other systems on the same local network

**Persistence** is used after exploiting vulnerability in a system; attackers install tools on that system to allow future access-even if the initial vulnerability is corrected.

**Breach and attack simulation (BAS):** Platforms that automate penetration testing by injecting threat indicators onto systems and networks.

The **window of vulnerability** begins with the initial discovery of an issue and ends with a successful deployment of a patch.

After finding a vulnerability, the individual must share the vulnerability with the vendor first, providing them with a deadline after which the vulnerability will be publicly announced.

**Bug bounties** are open security testing for a reward.

Cyber security exercises involve a red team, that try to undermine the technique, a blue team that will try to secure the systems from the attack, and a white team that will monitor the whole thing.

Code reviews use peer analysis to assess code.

The most formal code review process is the Fagan inspection that follows a 6 step process. The six steps are: planning, overview, preparation, meeting, and rework, follow up.

Code test uses technology to inspect software. The 2 types of tests are:

1. Static test: use automated techniques to analyze code for errors and security flaws without executing it.
2. Dynamic tests: Execute code to verify that it is functioning correctly and doesn’t have security flaws.

**Fuzzing :** Software testing technique that feeds software many different inputs to cause an unpredictable state or unauthorized access.

Use ZAP to try fuzzing

Standardized interfaces allow for different develops to share code to make the development process easier. These interfaces are called Application programming interfaces (APIs).

APIs must be tested carefully.

Misuse case testing involves testing unexpected usages of the software. These can include

* Unexpected input
* Missing input
* Injection attacks
* Unavailable funds

Week 8

**End point applications** are applications that are self contained on a device.

**Web Applications** are client/server over the web

The classic approach to software design is the waterfall approach. The waterfall approach is very rigid and doesn’t allow going back and forwarding with steps. The spiral system is very similar but the process is more iterative.

The approach most commonly used today is the agile approach.

**Maturity models** provide standard benchmarks to gauge where developers are at. The capability mature model integration (CMMI) assesses an organization’s process maturity. The 5 levels of CMMI are:

1. Initial – getting started with mismanaged time budgets
2. Managed – configuration management, project planning, requirements management
3. Defined – formal, documented management. Decisions analysis, integrated project management, product integration, risk management, verification
4. Quantitatively managed – use numbers to assess their performance
5. Optimizing - continuous process improvement with feedback being used to go improve constantly.

The IDEAL model also has 5 steps:

1. Initiating
2. Diagnosing
3. Establishing
4. Action
5. Learning

Change management has 3 elements:

1. Request control – allows customers to request , manage, evaluate for inbound request from customers
2. Change control – grants permissions for developers to make changes to application code
3. Release control – after new code is written, it is reviewed and then moved into production.

Code changes go through different environments:

Development -> test -> staging -> production

The Open web application security project (OWASP) maintans a list of common web security issues. The OWASP top ten are:

1. Injection flaws – insert unwanted transaction code. Can be prevented with:
   1. Input validation – protects against unsafe user input by checking it on the server before executing commands
   2. Parameterized SQL – precompiles SQL code on the database server to prevent user input from altering query structure.
2. Broken authentication code – Session hijacking – if cookies are guessable, the attacker can use it to take over a user’s session on a website. Can be solved by using random cookies and make sure they are encrypted.
3. Sensitive data exposure – discloses confidential information
4. XML external entities -allow remote code execution. Code execution attacks – occur when an attacker exploits vulnerability in a system that allows the attacker to run commands on that system. This can be avoided by:
   1. Limit administrative access
   2. Patch systems and applications
5. Broken access control – allow unauthorized access
6. Security misconfiguration – occur in many possible location
7. Cross site scripting – inserts malicious scripts on website. Cross site request forgery(CSRF) also exists, it involves taking advantage of the fact that users are logged on to multiple sites at the same time and use one site to trick the browser into sending malicious requests to another site without the user’s knowledge. CSRF are client side. This can be prevented with:
   1. Input validation – don’t allow <script> tags in user supplied input(cross site scripting)
   2. Rearchitect web applications(CSRF)
   3. Prevent the use of HTTP GET requests(CSRF)
   4. Advise users to log out of sites (CSRF)
   5. Automatically log out users after an idle period(CSRF)
8. Insecure deserialization – allow api exploitation
9. Using vulnerable components – jeopardizes application security. Driver manipulation – can be done with refactoring (modifying a driver to carry out malicious activities) or shimming (wraps a legitimate driver with a malicious shim). This can be avoided with:
   1. Code signing
10. Insufficient logging – frustrating security analysis
11. Overflow attacks – overloading the memory allocation of a data entry. Can be avoided by limiting the amount of characters in a field.
12. Privilege escalation – involve buffer overflow vulnerabilities. Can be avoided with:
    1. Input validation
    2. Patch OS, platforms, and applications
    3. Enforce the least privilege
13. Memory vulnerabilities :
    1. Memory leak- fails to release memory for reuse.
    2. Memory pointer –attackers can use null pointer dereferencing.
    3. DLL injection attacks tricks an application into using malicious code(DLLs)

**Application hardening -** Application hardening takes a finished, well-built application and both manipulates existing, and injects new, code to shield the application against static and dynamic attacks far beyond "hygienic" vulnerabilities like those created by not verifying a sender or destination or message format.