CS4650 Topic 21:
Cloud Security Mechanisms

Cloud Security Mechanisms

- This chapter presents several cloud security mechanisms and protocols.
- These can be used to counter the security threats explored in Topic 20.

Encryption

- The most natural form for storing or transmitting data is in a readable format.
- There are several readable formats.
- These are easy to use, because the formats are fairly simple.
- Since they are readable, they are vulnerable to several of the attacks.
- Data in any of these readable formats is called *plaintext*.
- *Encryption* is the process of converting data from plaintext into a form that is unreadable unless the secret is known for reversing the process, known as *decryption*.
- The algorithm for performing encryption is called a *cipher*.
- The encoded data is called *ciphertext*.

Encryption

- The cipher is a standardized algorithm which takes the plaintext data and an *encryption key*, then produces the ciphertext.
- The cipher can also take the ciphertext and a *decryption key* to produce the original plaintext.
- The encryption and decryption keys are the 'secret' mentioned previously.
- Ideally, the cipher algorithm is strong enough so that a person that has the ciphertext and the algorithm but not the decryption key cannot feasibly deduce the plaintext.
- The converse is also a goal, that with the cipher and the plaintext, but without the encryption key, the ciphertext cannot be deduced.

Encryption

- There are several variations of encryption, with different characteristics.
- The goal of encryption is to preserve the confidentiality of data: Someone without the proper keys cannot understand the data.
- In some sense, encryption can also preserve the integrity of data:
    - A third party cannot feasibly modify the ciphertext in a way that an authorized party decrypting the data will be led to think the data was not corrupted. If the third party modifies the ciphertext, the decryption of the modified data will yield unrecognizable garbage. In this case the data's integrity is not preserved.
    - However, the recipient of the data will be able to see that the data has been modified, and so will reject the data. The sender will need to re-send the data.
- Encryption can help counter the traffic eavesdropping, malicious intermediary, insufficient authorization, and overlapping trust boundaries security threats.

Symmetric Encryption

- With *Symmetric Encryption*, the encryption and decryption keys are identical (the process is symmetrical).
- This is also known as *secret key cryptography*.
- Messages that are encrypted with the key can only be decrypted using the same key.
- Therefore, if only authorized users have this key, then only authorized users can send and receive these messages.
- Any message which is successfully decrypted using the key is known to have been sent by an authorized user. This is a basic form of authentication.

Symmetric Encryption

- As long as the keys are kept safe, so that only authorized entities have access, confidentiality is preserved.
- Symmetric Encryption is simpler and faster than Asymmetric encryption, so it is frequently used for data transmission.
- Symmetric Encryption does not have the characteristic of non-repudiation, since you can verify that an authorized entity created the message, but you cannot guaranteed that a *particular* entity created it. For example, the receiver could have created the message, then later claim that the sender sent it.
- One issue with Symmetric Encryption is *secure key exchange*: If parties A and B wish to start communicating, how can they securely pass the key?

Asymmetric Encryption

- The second major form of encryption is *asymmetric encryption*.
- With this form, the encryption key and decryption key are *different*.
- Usually one of these keys is called the *private key* and the other key is called the *public key*.
  - The private key is kept private. The owning entity does not share this key with anybody.
  - The public key can be publicly known: 'everyone' can see this key.
- A vital requirement is that anyone with access to the public key cannot feasibly deduce the private key.
- Why would you want a key to be public?

Asymmetric Encryption

- Suppose the owner of a document wants to prove the integrity of the document:
    - The owner of the document encrypts the document with the private key before publishing the document.
    - Anyone who wants to verify the integrity decrypts the document with the owner's public key. If the document successfully decrypts, this shows that the original document must have been provided by the owner.
- This adds the integrity characteristic, but not the confidentiality characteristic (since everyone has access to the public key, so everyone can decrypt the message).

Asymmetric Encryption

There are a couple of interesting and useful properties of asymmetric encryption:

- The public and private keys can be used for either encryption or decryption.
  - A message encrypted using a private key can only be decrypted using the matching public key.
  - A message encrypted using a public key can only be decrypted using the matching private key.
- A message can be encrypted more than once!
  - If a message is encrypted first using key A then using key B, it can only be decrypted using key B' then by using key A'.

Asymmetric Encryption

- Suppose the owner (A) of a document wants to prove the confidentiality of the document so that only a particular entity (B) can decrypt it:
  - The owner of the document encrypts the document with the public key of B before publishing the document.
  - Entity B decrypts the message using B's private key. Only entity B can successfully decrypt the message since only B has that private key.
- This adds the confidentiality characteristic, but not the integrity characteristic.
  - Since everyone has access to the public key, everyone can encrypt the message. So we can't prove who it came from (no integrity).
  - Since only B has the private key, only B can decode the message (confidentiality)

Asymmetric Encryption

- How can a message have confidentiality and integrity?
- By using the second interesting property: encrypting twice.
- Entity A first encrypts with A's private key, then encrypts that result with B's public key.
- Entity B first decrypts using B's private key (only B can do this step), then decrypts that result with A's public key (so only A could have sent the message).
- This combined approach provided confidentiality, integrity, authenticity, and non-repudiation.

Combined Encryption

- Symmetric encryption is computationally faster, but has the difficulty of secure key exchange.
- Asymmetric encryption is slower, but does not require a secure key exchange, as the public keys are always freely available and the private keys are never exchanged.
- Using a hybrid approach gives the advantages of both techniques:
    - Entity A wants to initiate an exchange with B. Entity A creates a new *session key*, a symmetric key used for this communication.
    - A then uses asymmetric encryption to send the session key to B.
    - A and B can then communicate securely and at a faster speed using this shared session key.

Hashing

- *Hashing* is a mechanism which computes a *signature* of a message.
- The signature cannot be used to recreate the message. The computation is strictly one-way.
- The signature can be used to verify the integrity of the data: If the data has been changed in any way, the signature of the new version of the data will not match the signature of the original version of the data.

Hashing

- One use of hashing is for storing passwords.
- Originally, user's passwords were stored as plaintext in a file buried in the computer's OS.
- Very quickly people found this file, and so were able to discover everyone's password on the system.
- This same problem exists on many on-line databases.
- Instead, the password entered by a user is first hashed to form a signature, and this signature is stored in the database.
  - Given the signature, it is infeasible to determine the original password, or even to compute an alternative password that would create the same signature.
  - When someone logs in, the password is again hashed and the resulting signature is matched to the stored value.

Hashing

- Hashing is used other places to verify the integrity of a document.
- For example, in many websites that host programs for download, in addition to the program, these sites also post the corresponding signatures.
- These sites suggest that after downloading the program's installer, but before running the program, the hash of the installer be computed and compared with the published signature. If the values are different, the user should *not* install the program.
- Consequently, if a hacker modified the program to include malware, the users would be able to detect this.
- *Unless, of course, the hacker not only changed the data file, but also 'hacked' the published signature!*

Digital Signature

- In many cases, people are interested in authentication, integrity, and non-repudiation, but confidentiality is not essential. For example, a person is publishing a document, but want to prove that authorship, and that document has not been altered.
- One thought would be to hash the document, then append the signature to the document. The flaw with this approach is that a malicious agent could alter the document, compute the new signature, then append that on the document. The document would appear to have integrity, but it has in fact been changed.

Digital Signature

- The improved approach is to first perform the hash, then to encrypt the hash with the author's private key. The resulting string is the *digital signature*. This is the value that is appended to the document.
- A party that wants to verify the integrity and authenticity of the document would compute the hash of the document, then decode the digital signature using the author's public key. If the two results match, then this shows that the document did come from the author, and that the document was not altered.

Digital Signature

- Why use a digital signature? Why not just encrypt the whole file with the author's private key?
- In many cases, we want the document to be freely readable. For example, a legal contract. The contract should be available for anyone to read.
- However, if anyone is concerned about the authenticity of the document, at that point the digital signature can be checked.
- The digital signature can be used to mitigate the malicious intermediary, insufficient authorization, and overlapping trust boundaries security threats.

Public Key Infrastructure

- In the discussion of asymmetric encryption, it was stated that an entity would create two keys, a private key and a public key. The private key would be kept private, and the public key would be made public, freely available.
- However, there is an issue with the public key: How do we know that this key which is claimed to be the public key of entity A is actually the public key of entity A.
- Suppose there was just a list of public keys. Anyone could place a key into that list, and anyone could view these keys.
- But a malicious agent could insert a new key into the list claiming that the key belonged to A, even though it didn't.

Public Key Infrastructure

- The entries in the public list consist of a couple of items:
  - The entity's public key
  - The entity's data, such as name, contact information, and so on.
- What is missing is some guarantee that this package is valid, that the entity identified is in fact the owner of that key.
- What is needed is a *certificate authority*.
  - A certificate authority is a trusted organization that performs the same role as a notary public.
  - The certificate authority first verifies that the applicant is in fact who they claim to be.
  - Once satisfied with the applicant's identity, the certificate authority then creates a *certificate* for the applicant.
  - The certificate assures that the certificate authority verified that the public key does belong to the named entity.

Public Key Infrastructure

- So what is the certificate? The certificate consists of three parts:
    - The entity's data (name, contact, etc)
    - The entity's public key
    - In many cases, an expiration date, beyond which the certificate is deemed invalid and should be replaced by a more up-to-date certificate.
    - A digital signature signed by the certificate authority showing that all of the above information has been unaltered.
- The resulting certificate can be publicly displayed, possibly on the owner's website or other forums.
- To verify the certificate, the digital signature is decrypted with the certificate authority's public key, and the hash of the remaining values is computed.
- If the results match, then the user is confident that the public key contained in the certificate is indeed the public key of the owner.

Public Key Infrastructure

- The key to the public key infrastructure is the certificate authority.
  - How can we trust the certificate authority?
  - At some point we need to receive the certificate authority's public key!
- There are a few trusted certificate authorities, such as VeriSign and Comodo.
- Lower tier certificate authorities exist, their public keys are verified through certificates issued by the main authorities!

Single Sign-On (SSO)

- In some concepts, there may be several related cloud services, which may be hosted on a single cloud, or they may be hosted on multiple clouds.
- The relationship between these services is that they use the same user accounts, so a particular user may have access to a variety of these services.
- It would be cumbersome for a cloud service consumer to need to log into each of these services, especially if the consumer might be using several services simultaneously, or moving data between multiple services.
- The *Single Sign-On (SSO)* mechanism allows the user to sign-in one time to a *security broker service*.

Single Sign-On (SSO)

- The user would sign in to the security broker service. Upon successful log-in, the customer receives a security token.
- The security token identifies the consumer, and is signed by a digital signature from the security broker service.
- The token would be valid for a single session or for a limited period of time.
- Each of the cloud services enabled by this SSO simply need verify the token using the digital signature from the broker.

Hardened Virtual Server Images

- As previously discussed, a virtual server is created from a template configuration called a virtual server image.
- The cloud service provider customizes the virtual server image to have the resources required for the cloud application.
- Ideally, these virtual server images should only have installed the programs and tools necessary for the application.
- One reason is that if the program is not going to be used, why have it installed? It just bloats everyone's server with useless code.
- *Hardening* is the process that takes this elimination of programs further.

Hardened Virtual Server Images

- In addition to removing unused programs, the following steps are performed:
  - Closing unnecessary server ports
  - Disabling unused services
  - Removing internal root accounts
  - Removing guest access
  - Establishing memory quotas
- Hardening the Virtual Server Image results in a significantly more secure virtual server.
- This helps counter the denial of service, insufficient authorization, and overlapping trust boundaries threats.

Wrap-Up

- The ideas presented in this chapter are some of the common tools to help securing the on-line environment.
- Additional tools and techniques are available, and new approaches are frequently being added to the tool set!