Module 1: Introduction to Computer Security

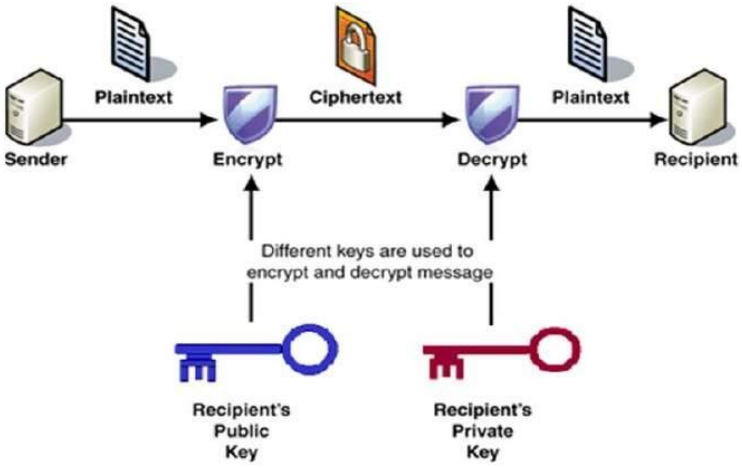
Vulnerabilities Threats Attacks and Control: (Reference: Pearson's)

- A computer-based system has three separate but valuable components: hardware, software, and data.
- Each of these assets offers value to different members of community affected by system.
- To analyze security, we can brainstorm about ways in which system or its information can experience some kind of loss or harm. For eg, we can identify data whose format or contents should be protected in some way.
- We want our security system to make sure that no data are disclosed to unauthorized parties.
- Neither do we want data to be modified in illegitimate ways. At same time, we must ensure that legitimate users have access to data. In this way, we can identify weaknesses in system.
- **A vulnerability is weakness in security system**, for example, in procedures, design, or implementation, that might be exploited to cause loss or harm. For instance, particular system may be vulnerable to unauthorized data manipulation because system does not verify user's identity before allowing data access.
- **A threat to computing system is set of circumstances that has potential to cause loss or harm o*r potential for violation of security, which exists when there is circumstance, capability, action, or event that could breach security and cause harm. That is, threat is possible danger that might exploit vulnerability.***
- To see difference between threat and vulnerability, consider, wall is holding water back. water to left of wall is threat to man on right of wall: water could rise, overflowing onto man, or it could stay beneath height of wall, causing wall to collapse. So threat of harm is potential for man to get wet, get hurt, or be drowned. For now, wall is intact, so threat to man is unrealized. However, we can see small crack in wall vulnerability that threatens man's security. If water rises to or beyond level of crack, it will exploit vulnerability and harm man. **There are many threats to computer system, including human-initiated and computer-initiated ones.** We have all experienced results of inadvertent human errors, hardware design flaws, and software failures. But natural disasters are threats, too; they can earthquake, for eg.
- **A human who exploits vulnerability perpetrates an attack on system. An attack can also be launched by another system, as when one system sends an overwhelming set of messages to another, virtually shutting down second system's ability to function.** Unfortunately, we have seen this type of attack frequently, as denial-of-service attacks flood servers with more messages than they can handle.
- How do we address these problems? We use control as protective measure. That is, control is an action, device, procedure, or technique that removes or reduces vulnerability.
- The man is placing his finger in hole, controlling threat of water leaks until he finds more permanent solution to problem. In general, we can describe relationship among threats, controls, and vulnerabilities in this way:
- **A *threat* is blocked by *control of vulnerability*.** Much of rest of this book is devoted to describing variety of controls and understanding degree to which they enhance system's security.
- To devise controls, we must know as much about threats as possible. We can view any threat as being one of four kinds: **interception, interruption, modification, and fabrication.** Each threat exploits vulnerabilities of assets in computing systems.
- **An interception means that some unauthorized party has gained access to an asset.**
- The outside party can be person, program, or computing system. Examples of this type of failure are illicit copying of program or data files, or wiretapping to obtain data in network. Although loss may be discovered fairly quickly, silent interceptor may leave no traces by which interception can be readily detected.
- **In an interruption, an asset of system becomes lost, unavailable, or unusable.** An example is malicious destruction of hardware device, erasure of program or data file, or malfunction of an operating system file manager so that it cannot find particular disk file.
- If unauthorized party not only accesses but tampers with an asset, threat is modification. For eg, someone might change values in database, alter program so that it performs an additional computation, or modify data being transmitted electronically. It is even possible to modify hardware. Some cases of modification can be detected with simple measures, but other, more subtle, changes may be almost impossible to detect.
- Finally, unauthorized party might create fabrication of counterfeit objects on computing system.
- Intruder may insert spurious transactions to network communication system or + records to an existing database. Sometimes these additions can be detected as forgeries, but if skillfully done, they are virtually indistinguishable from real thing
- **These four classes of threats interception, interruption, modification, and fabricationdescribe kinds of problems we might encounter.**

- ***An assault on system security that derives from an intelligent threat; that is, an intelligent act that is deliberate attempt (especially in sense of method or technique) to evade security services and violate security policy of system is an attack.***

Public Key Cryptography (Reference: William Stallings for diagram and tutorials point for contet):

Unlike symmetric key cryptography, we do not find historical use of public-key cryptography. It is relatively new concept. Symmetric cryptography was well suited for organizations such as governments, military, and big financial corporations were involved in classified communication. With spread of more unsecure computer networks in last few decades, genuine need was felt to use cryptography at larger scale. symmetric key was found to be non-practical due to challenges it faced for key management. This gave rise to public key cryptosystems. The process of encryption and decryption is depicted in following illustration −



The most important properties of public key encryption scheme are −

- Different keys are used for encryption and decryption. This is property which set this scheme different than symmetric encryption scheme.
- Each receiver possesses unique decryption key, generally referred to as his private key.
- Receiver needs to publish an encryption key, referred to as his public key.
- Some assurance of authenticity of public key is needed in this scheme to avoid spoofing by adversary as receiver. Generally, this type of cryptosystem involves trusted third party which certifies that particular public key belongs to specific person or entity only.
- Encryption algorithm is complex enough to prohibit attacker from deducing plaintext from ciphertext and encryption (public) key.
- Though private and public keys are related mathematically, it is not be feasible to calculate private key from public key. In fact, intelligent part of any public-key cryptosystem is in designing relationship between two keys.

There are three types of Public Key Encryption schemes. We discuss them in following sections −

RSA Cryptosystem

This cryptosystem is one initial system. It remains most employed cryptosystem even today. system was invented by three scholars **Ron Rivest, Adi Shamir,** and **Len Adleman** and hence, it is termed as RSA cryptosystem.

We will see two aspects of RSA cryptosystem, firstly generation of key pair and secondly encryption-decryption algorithms.

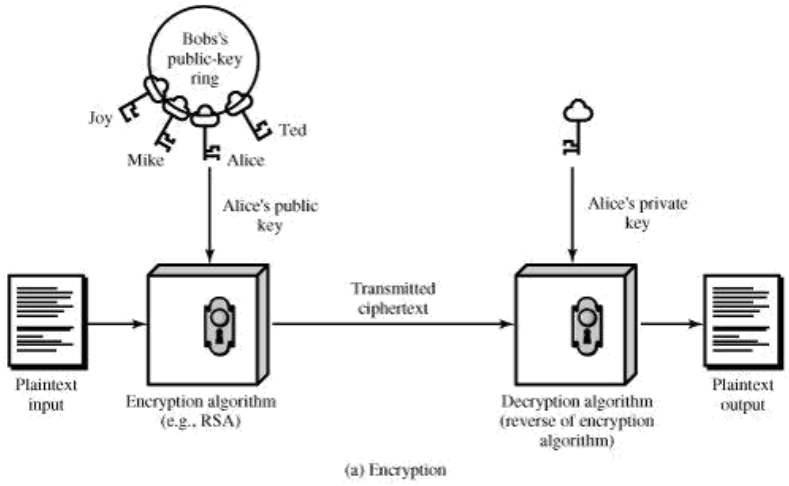
Generation of RSA Key Pair

Each person or party who desires to participate in communication using encryption needs to generate pair of keys, namely public key and private key. process followed in generation of keys is described below −

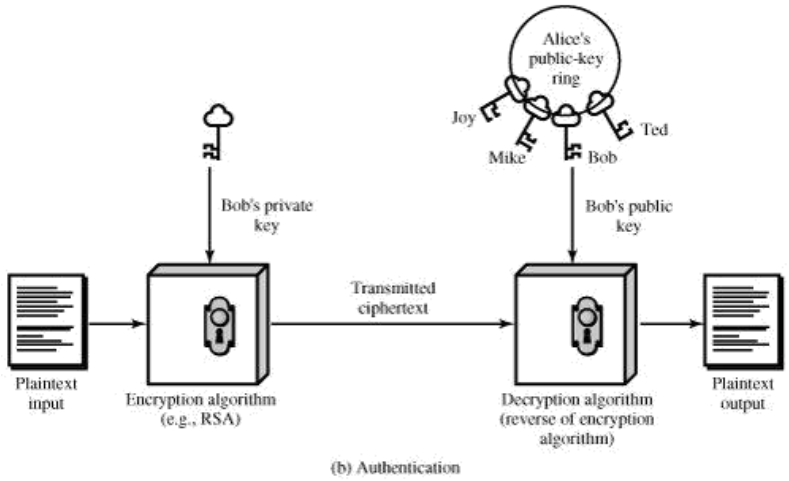
- **Generate RSA modulus (n)**
  - Select two large primes, p and q.
  - Calculate n=p*q. For strong unbreakable encryption, let n be large number, typically minimum of 512 bits.
- **Find Derived Number (e)**
  - Number **e** must be greater than 1 and less than (p − 1)(q − 1).
  - There must be no common factor for e and (p − 1)(q − 1) except for 1. In other words two numbers e and (p – 1)(q – 1) are coprime.
- **Form public key**
  - The pair of numbers (n, e) form RSA public key and is made public.
  - Interestingly, though n is part of public key, difficulty in factorizing large prime number ensures that attacker cannot find in finite time two primes (p & q) used to obtain n. This is strength of RSA.
- **Generate private key**
  - Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.
  - Number d is inverse of e modulo (p - 1)(q – 1). This means that d is number less than (p - 1)(q - 1) such that when multiplied by e, it is equal to 1 modulo (p - 1)(q - 1).
  - This relationship is written mathematically as follows −

$$ed = 1 \bmod (p - 1)(q - 1)$$

The Extended Euclidean Algorithm takes p, q, and e as input and gives d as output.



(a) Encryption



(b) Authentication

Example: An example of generating RSA Key pair is given below. (For ease of understanding, primes p & q taken here are small values. Practically, these values are very high).

- Let two primes be p = 7 and q = 13. Thus, modulus n = pq = 7 x 13 = 91.
- Select e = 5, which is valid choice since there is no number that is common factor of 5 and (p − 1)(q − 1) = 6 × 12 = 72, except for 1.
- The pair of numbers (n, e) = (91, 5) forms public key and can be made available to anyone whom we wish to be able to send us encrypted messages.
- Input p = 7, q = 13, and e = 5 to Extended Euclidean Algorithm. output will be d = 29.
- Check that d calculated is correct by computing −

$$de = 29 \times 5 = 145 = 1 \bmod 72$$

- Hence, public key is (91, 5) and private keys is (91, 29).

Encryption and Decryption: Once key pair has been generated, process of encryption and decryption are relatively straightforward and computationally easy. Interestingly, RSA does not directly operate on strings of bits as in case of symmetric key encryption. It operates on numbers modulo n. Hence, it is necessary to represent plaintext as series of numbers less than n.

RSA Encryption: Suppose sender wish to send some text message to someone whose public key is (n, e).

- The sender then represents plaintext as series of numbers less than n.
- To encrypt first plaintext P, which is number modulo n. encryption process is simple mathematical step as −

$$C = P^e \bmod n$$

- In other words, ciphertext C is equal to plaintext P multiplied by itself e times and then reduced modulo n. This means that C is also number less than n.
- Returning to our Key Generation example with plaintext P = 10, we get ciphertext C −

$$C = 10^5 \bmod 91$$

RSA Decryption: The decryption process for RSA is also very straightforward. Suppose that receiver of public-key pair (n, e) has received ciphertext C.

- Receiver raises C to power of his private key d. result modulo n will be plaintext P.

$$\text{Plaintext} = C^d \bmod n$$

- Returning again to our numerical eg, ciphertext C = 82 would get decrypted to number 10 using private key 29−

$$\text{Plaintext} = 82^{29} \bmod 91 = 10$$

RSA Analysis: The security of RSA depends on strengths of two separate functions. RSA cryptosystem is most popular public-key cryptosystem strength of which is based on practical difficulty of factoring very large numbers.

- **Encryption Function** − It is considered as one-way function of converting plaintext into ciphertext and it can be reversed only with knowledge of private key d.
- **Key Generation** − difficulty of determining private key from an RSA public key is equivalent to factoring modulus n. An attacker thus cannot use knowledge of an RSA public key to determine an RSA private key unless he can factor n. It is also one way function, going from p & q values to modulus n is easy but reverse is not possible.

If either of these two functions are proved non one-way, then RSA will be broken. In fact, if technique for factoring efficiently is developed then RSA will no longer be safe.

The strength of RSA encryption drastically goes down against attacks if number p and q are not large primes and/ or chosen public key e is small number.

The Security of RSA: Four possible approaches to attacking RSA algorithm are as follows:

Brute force: This involves trying all possible private keys.

Mathematical attacks: There are several approaches, all equivalent in effort to factoring product of two primes.
Timing attacks: These depend on running time of decryption algorithm.
Chosen ciphertext attacks: This type of attack exploits properties of RSA algorithm.
The defense against brute-force approach is same for RSA as for other cryptosystems, namely, use large key space. Thus, larger number of bits in *d*, better. However, because calculations involved, both in key generation and in encryption/decryption, are complex, larger size of key, slower system will run.

Knapsack Cryptosystem (Reference: https://asecuritysite.com/encryption/knap):

RSA is just one way of doing public key encryption. Knapsack is good alternative where we can create public key and private one. knapsack problem defines problem where we have number of weights and then must pack our knapsack with minimum number of weights that will make it given weight. In general problem is:

- Given set of numbers and number b.
- Find subset of which sums to b (or gets nearest to it).

So imagine you have set of weights of 1, 4, 6, 8 and 15, and we want to get weight of 28, we could thus use 1, 4, 8 and 15 (1+4+8+15=28).

So our code would become 11011 (represented by '1', '4', no '6', '8' and '15'.

Then if our plain text is 10011, with knapsack of 1, 4, 6, 8, 15, we have cipher text of 1+4+8+15 which gives us 28.

A plain text of 00001 will give us cipher text of 15.

With public key cryptography we have two knapsack problems. One of which is easy to solve (private key), and other difficult (public key).

We can now create super-increasing sequence with our weights where cur-rent value is greater than sum of preceding ones, such as {1, 2, 4, 9, 20, 38}.

Super-increasing sequences make it easy to solve knapsack problem, where we take total weight, and compare it with largest weight, if it is greater than weight, it is in it, otherwise it is not.

For example with weights of {1,2,4,9,20,38} with value of 54, we get:

```
Check 54 for 38? Yes (smaller than 54). [1] We now have balance of 16.
Check 16 for 20? No. [0].
Check 16 for 9? Yes. [1]. We now have balance of 5.
Check 5 for 4? Yes. [1]. We now have balance of 1.
Check 1 for 2? No. [0].
Check 1 for 1? Yes [1].
```

Our result is 101101.

If we have non-super-increasing knapsack such as {1,3,4,6,10,12,41}, and have to make 54, it is much more difficult. So non-super-increasing knapsack can be public key, and super-increasing one is private key.

We first start with our super-increasing sequence, such as {1,2,4,10,20,40} and take values and multiply by number n, and take modulus (m) of value which is greater than total (m - such as 120). For n we make sure that there are no common factors with any of numbers. Let's select an n value of 53, so we get:

```
1×53 mod(120) = 53
2×53 mod(120) = 106
4×53 mod(120) = 92
10×53 mod(120) = 50
20×53 mod(120) = 100
40×53 mod(120) = 80
```

So public key is: {53,106,92,50,100,80} and private key is {1, 2, 4, 10, 20,40}. public key will be difficult to factor while private key will be easy. Let's try to send message that is in binary code:

```
111010 101101 111001
```

We have six weights so we split into three groups of six weights:

```
111010 = 53 + 106 + 92 + 100 = 351
101101 = 53+ 92 + 50 + 80 = 275
111001 = 53 + 106 + 92 + 80 = 331
```

Our cipher text is thus 351 275 331. The two numbers known by receiver is thus 120 (m - modulus) and 53 (n multiplier). We need n-1, which is multiplicative inverse of n mod m, i.e. $n(n-1) = 1 \bmod m$. For this we find inverse of n:

```
n-1 = 53-1 mod 120
(53 x _n) mod 120 = 1
```

So we try values of n-1 in (53 x n-1 mod 120) in order to get result of 1:

n-1	Result
1	53
2	106
3	39
...	
75	15
76	68
77	1

So inverse is 77. [Calculate]. The coded message is 351 275 331 and is now easy to calculate plain text:

351×77 mod(120) = 27 = 111010 (1+2+4+20)
275×77 mod(120) = 55 = 101101
331×77 mod(120) = 47 = 111001

The decoded message is thus:

111010 101101 110001

which is same as our original message:

111010 101101 111001

The decoding was so easy, only thing we had was to find inverse which is not to difficult. Knapsack encryption provides good approach to creating public and private keys, where there private key is easy to use, while public key is difficult to compute. method was outlined by Ralph Merkle in his search for trap door function , but glory of sustainable trap door went to RSA, and soon cracks began to show when Adi Shamir published methods to crack it.

Module 2: Authentication

Password Based Authentication : User has decided to trust server, either without authentication or on basis of server authentication via SSL. The user requested resource controlled by server.

The server requires client authentication before permitting access to requested resource.

- In response to authentication request from server, client displays dialog box requesting user's name and password for that server. user supplies name and password separately for each new server user uses during work session.
- The client sends name and password across network, either in clear or over an encrypted SSL connection.
- The server looks up name and password in its local password database and, if they match, accepts them as evidence authenticating user's identity.
- Server determines whether identified user is permitted to access reqst'd resource, & if so, allows client to access it

With this arrangement, user supplies new password for each server, and administrator keeps track of name and password for each user. Password-based authentication is one of most popular approaches to authenticate user in various enterprise applications. But there are many problems associated with password based authentication systems and risks associated with using passwords as an authentication mechanism for enterprise applications is not completely secure. Considering all risks associated with password based authentication systems, there is strong need for enterprises to switch to stronger authentication system which provides security against various hacking attacks and also which is more convenient and easier to end user of system

Challenges with Password based Authentication:

1. Easy passwords can be cracked: The end users behavior such as choosing passwords that are easy to remember introduces majority of password weaknesses. For hacker, these passwords can easily be cracked or guessed. Surveys show that frequent passwords are word password, personal names of family members, names of pets, and dictionary words.

2. Random passwords cant be remembered: A random password should not have content, context, and should not be familiar. It can only be learned by using it over and over again. However, since repetition is weak way of remembering, users often completely ignore recommendations for pseudo-random passwords.

3. Remembering Multiple Passwords: The more passwords person has to remember, chances for remembering any specific password decreases. Having multiple passwords also increases chance of interference among similar passwords. This is especially true for systems that are not used frequently.

4. Problems with passwords that needs to be continuously changed: Computer systems require frequent password changes, to make system robust from various attacks. Users must think of new passwords that conform to all of organizations requirements but that are also easy to remember. System-enforced password policies, however, cannot guarantee password secrecy.

5. Security vs. Ease-of-Use for Passwords: To solve Password Problem, users will try to decrease memory burden at expense of security. Most commonly, user will write down passwords, raising potential of compromise of passwords. In case of multiple systems, users may choose only one password for all systems.

6. Shoulder Surfing Attack: Shoulder surfing is looking over someones shoulder when they enter password or PIN code. It is an effective way to get information in crowded places because it is relatively easy to stand next to someone and watch as they fill out form, enter PIN number at an ATM machine, or use calling card at public pay phone. Considering all above, there is growing trend among many enterprises globally to move to stronger authentication solution which provides high level of security with-out compromising users convenience.

Token Based Authentication: Token based authentication is prominent everywhere on web nowadays. With most every web company using an API, tokens are best way to handle authentication for multiple users. There are some very important factors when choosing token based authentication for your application. main reasons for tokens **Stateless and scalable servers, Mobile application ready, Pass authentication to other applications, Extra security. Who Uses Token Based Authentication?** Any major API or web application that you've come across has most likely used tokens. Applications like Facebook, Twitter, Google+, GitHub, and so many more use tokens.

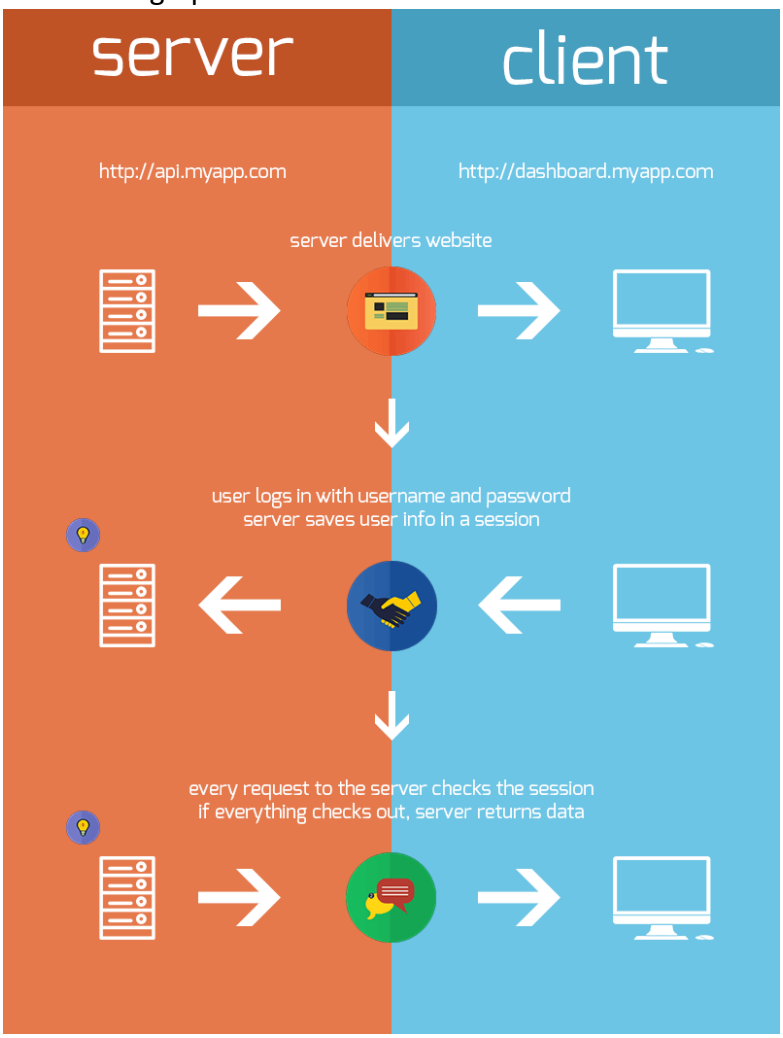
#Why Tokens Came Around Before we can see how token based authentication works and its benefits, we have to look at way authentication has been done in past.

Server Based Authentication (The Traditional Method)

Since HTTP protocol is *stateless*, this means that if we authenticate user with username and password, then on next request, our application won't know who we are. We would have to authenticate again.

The traditional way of having our applications remember who we are is to **store user logged in information on server**. This can be done in few different ways on session, usually in memory or stored on disk.

Here is graph of how server based authentication workflow would look:



As web, applications, and rise of mobile application have come about, this method of authentication has shown problems, especially in scalability.

#**The Problems with Server Based Authentication**

A few major problems arose with this method of authentication.

Sessions: Every time user is authenticated, server will need to create record somewhere on our server. This is usually done in memory and when there are many users authenticating, overhead on your server increases.

Scalability: Since sessions are stored in memory, this provides problems with scalability. As our cloud providers start replicating servers to handle application load, having vital information in session memory will limit our ability to scale.

CORS: As we want to expand our application to let our data be used across multiple mobile devices, we have to worry about cross-origin resource sharing (CORS). When using AJAX calls to grab resources from another domain (mobile to our API server), we could run into problems with forbidden requests.

CSRF: We will also have protection against cross-site request forgery (CSRF). Users are susceptible to CSRF attacks since they can already be authenticated with say banking site and this could be taken advantage of when visiting other sites.

With these problems, scalability being main one, it made sense to try different approach.

#**How Token Based Works**

Token based authentication is **stateless**. We are not storing any information about our user on server or in session. This concept alone takes care of many of problems with having to store information on server.

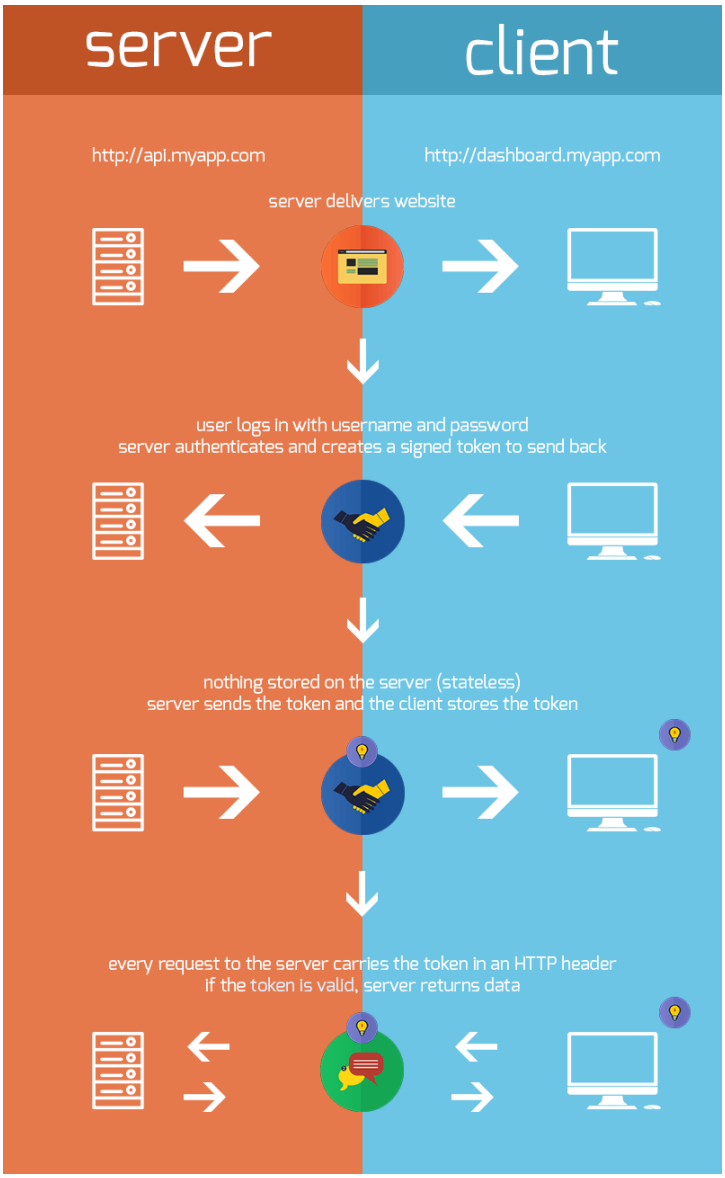
No session information means your application can scale and add more machines as necessary without worrying about where user is logged in.

Although this implementation can vary, gist of it is as follows:

1. User Requests Access with Username / Password
2. Application validates credentials
3. Application provides signed token to client
4. Client stores that token and sends it along with every request
5. Server verifies token and responds with data

Every single request will require token. This token should be sent in HTTP header so that we keep with idea of stateless HTTP requests. We will also need to set our server to accept requests from all domains using `Access-Control-Allow-Origin: *`. What's interesting about designating `*` in ACAO header is that it does not allow requests to supply credentials like HTTP authentication, client-side SSL certificates, or cookies.

Here's an infographic to explain process:



Once we have authenticated with our information and we have our token, we are able to do many things with this token. We could even create permission based token and pass this along to third-party application (say new mobile app we want to use), and they will be able to have access to our data -- **but only information that we allowed with that specific token**.

#[The Benefits of Tokens]

Stateless and Scalable: Tokens stored on client side. Completely stateless, and ready to be scaled. Our load balancers are able to pass user along to any of our servers since there is no state or session information anywhere. If we were to keep session information on user that was logged in, this would require us to keep sending that user to *same server that they logged in at*(called session affinity). This brings problems since, some users would be forced to same server and this could bring about spot of heavy traffic. Not to worry though! Those problems are gone with tokens since token itself holds data for that user.

Security: The token, not cookie, is sent on every request and since there is no cookie being sent, this helps to prevent CSRF attacks. Even if your specific implementation stores token within cookie on client side, cookie is merely storage mechanism instead of an authentication one. There is no session based information to manipulate since we don't have session! The token also expires after set amount of time, so user will be required to login once again. This helps us stay secure. There is also concept of token revocation that allows us to invalidate specific token and even group of tokens based on same authorization grant.

Extensibility (Friend of Friend and Permissions): Tokens will allow us to build applications that share permissions with another. For example, we have linked random social accounts to our major ones like Facebook or Twitter. When we login to Twitter through service (let's say Buffer), we are allowing Buffer to post to our Twitter stream. By using tokens, this is how we **provide selective permissions to third-party applications**. We could even build our own API and hand out special permission tokens if our users wanted to give access to their data to another application.

Multiple Platforms and Domains: We talked bit about CORS earlier. When our application and service expands, we will need to be providing access to all sorts of devices and applications (since our app will most definitely become popular!). Having our API just serve data, we can also make design choice to serve assets from CDN. This eliminates issues that CORS brings up after we set quick header configuration for our application.

Access-Control-Allow-Origin: *

Our data and resources are available to requests from any domain now **as long as user has valid token**.

Standards Based

When creating token, you have few options. We'll be diving more into this topic when we secure an API in follow-up article, but standard to use would be JSON Web Tokens.

This handy debugger and library chart shows support for JSON Web Tokens. You can see that it has great amount of support across variety of languages. This means you could actually switch out your authentication mechanism if you choose to do so in future!

Digital Certificate, X.509 Directory Services, PKI, Single Sign On, Federated Identity Management: Question Bank 1

Needham-Schroeder protocol reffers to communication protocol used to secure an insecure network. protocol got its name from creaters Roger Needham and Michael Schroeder.

There are two types of Needham-Schroeder protocol.

Needham-Schroeder protocol with symmetric key

Needham-Schroeder protocol with assymetric key

Needham-Schroeder protocol with symmetric key encryption because its one used in kerberos infrastructure.

This allows to prove identity of end users communicating, and also prents middle man from evesdropping.

- We will be using some terms in this document which needs to be understood first.
- Nonce: Nonce is randomly genrated string which is only valid for some period of time, This is used in encryption protocols to prevent replay attack. For example if somebody captures packet during communication between me and shopping website, he can resend packet without decrypting it, and server can accept packet and do operations on it. To prevent this, nonce(the random value generated) is added to data, so as server can check if that nonce is valid, or expired.
- Lets understand this protocol by taking an eg communication between 2 machines called Machine and Machine B.
- The main thing in this protocol is that there is trusted middle man or call him an arbitrator. This trusted middle man is server. If an X machine wants to communicate, with Y machine, then X has to contact middle man server, saying am interested in communicating with Y.
- Lets see how this works.

A = Machine A **B = Machine B**

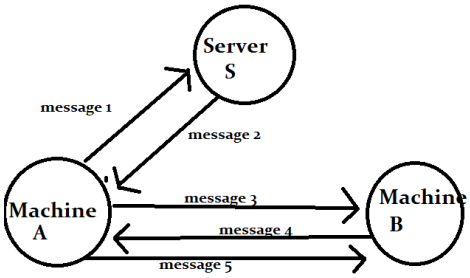
SK(AS) = this is symmetric key known to Machine and middle man Server named "S"

SK(BS) = this is symmetric key known to Machine B and middle man Server named "S"

NON(A) = Nonce generated by Machine A.

NON(B) = Nonce generated by Machine B

SK(S) = this is symmetric key/session key generated by server for both machine and Machine B.



Lets understand all messages above mentioned.

Initially before going ahead with explanation, make it clear that symmetric keys of both machine A, Machine B are already shared with Middle Man server. Also any other machine in network also shares its respective Symmetric keys with Middle Man server.

Message 1: Machine 1 sends message to Server S saying that i want to communicate with Machine B.

-> S: (this message contains and B and NON(A))

Message 2: Server S sends message 2 back to Machine containing SK(S), and also one more copy of SK(S) encrypted with SK(BS), this copy will be send to Machine B by Machine A.

Message 3: Machine forwards copy of SK(S), to Machine B, who can decrypt it with key it has because it was encrypted by Middle man server with Machine B's symmetric Key SK(BS).

Message 4: Machine B sends back Machine a nonce value encrypted by SK(S). to confirm that he has symmetric key or session key provided by middle man server.

Message 5: Machine performs simple operation on nonce provided by Machine B and resends that back to machine B just to verify Machine has key

There are still some vulnerability in this protocol for replay attacks which is fixed by timestamp implimentation in this, when used by kerberos.

Module 3: Access Control

Discretionary Access Control (DAC): DAC was developed to implement Access Control Matrices defined by Lampson in his paper on system protection.

- Discretionary policies defines access control based on identity of requestors and explicit access rules that determines who can, or cannot, execute particular actions on particular resources.
- In DAC users can be given authority to other users to access resources, where assigning and granting privileges is done by an administrative policy.
- Different types of DAC policies and models have been proposed in literature.

The access matrix model: It provides simple framework for implementing discretionary access control.

- It is proposed by Lampson for providing protection against unauthorized access to resources within operating systems and later it is refined by Graham and Denning, model was developed by Harrison, Ruzzo, and Ullmann (HRU model), to minimize complexity of access control policy.
- This model is called as access matrix. Access matrix holds authorization state at given time in system.
- It provides abstract representation of protection systems.

	File 1	File2	File3	Program-1
Jack	own read write	read write		execute
Tom	read		read write	
Kate			read	execute read

Fig 1: An example of access matrix

- To design an access control system first step is identification of objects which we have to be protected and executing access request and different activities to objects, and actions that can be executed on objects and that must be controlled.
- For eg, in os's, objects can be any programs, directories or files.
- The authorization state in access matrix model is defined by triple (S, O, A), where S is set of subjects, who can have access liberties; O is set of objects, on which access rights can be exercised (subjects may be considered as objects, in which case O); and is access matrix, In this rows represents subjects, columns represents objects, and entry A[s, o] reports access rights of s on o.

- The access control model simply provides framework where authorizations can be specified, model can contain different access rights or privileges.
- For example, read, write, and execute actions can be considered with ownership (i.e., property of objects by subjects), and control (to model father children relationships between processes) privileges.
- We can change state of system by executing different commands that can execute primitive operations on authorization state with some conditions.

Disadvantages of DAC: Global policy: DAC allows user to decide access control policies on their resources and these policies are global policies and therefore DAC has trouble to ensure consistency.

- **Malicious software/programs:** DAC is vulnerable from processes because it executing malicious programs. If it execute malicious programs exploiting authorizations of particular user on behalf of whom they are executing. For instance, Trojan Horses.
- **Information flow:** Once particular information is acquired by process, and then DAC do not have any control on flow of information. Information can be copied from one object to another; therefore it is possible to access copy even if owner does not provide access to original copy.

Mandatory Access Control (MAC): In MAC users do not have authority to override policies and it totally controlled centrally by security policy administrator.

- MAC is system-wide policy which defines who is allowed to have access; individual user cant change that access rule
- It totally relies on central system.
- MAC policies are defined by system administrator, and it is strictly enforced by OS or security kernel.
- Examples: According to law, court can access driving records without owners' permission.
- MAC mechanisms have been tightly coupled to few security models and it is mostly used in system where priority is based on confidentiality.
- For example Trusted Solaris, TrustedBSD , SELinux etc.)
- MAC can be classified in to following types:
  1. Multilevel Security 2. Multilateral Security

Multilevel Security

- In this, information and users are classified into different levels according to their sensitivity and trust.
- It will be classified into Confidential, Secret and Top Secret.
- This defines different levels such as clearance level, classification level and security level.
- Clearance level indicates how much trust or rights given to person with some clearance.
- The trust or given rights indicates highest level of classified information handled by users or device.
- Classification level indicates level of sensitivity to be given for particular resources or information. For instance, level may indicate degree of damage country if information is disclosed to an enemy.

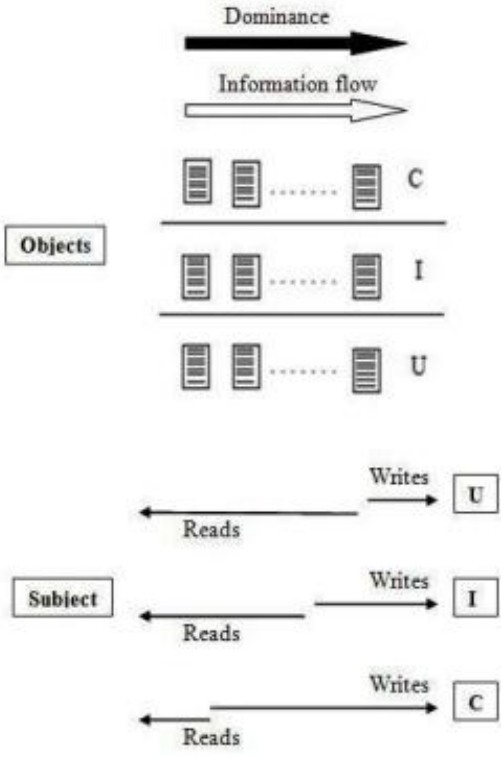


Fig 2: Integrity level for information flow

- Security level is general term for classification level or clearance level. In government and military facilities, MAC performs classification and then assigns label to each file system object. According to level of security it include confidential, secret and top secret.
- When user or device tries to access particular files or resources, OS or security kernel determine whether access will be granted or not.
- MAC requires continuous monitoring and careful planning to keep all resource objects' and users' classifications up to date.

- **The Bell-LaPadula Security Policy Model:** It is proposed by David Bell and Len Lapadula in 1973, to provide security for time-sharing mainframe systems. This model also called as MLS model.
- This model dealt with confidentiality. In this model, two types of security label are assigned to subjects and objects based on simple security property and *-property to verifiably ensure military classification policies that restrict information flow from more secure classification levels to less secure levels.
- Also referred as No read up and No write down.
- **Simple security property:** It states that process labeled with higher classification cannot access or read information or resources.

- That is, Subject is allowed to read object O only if class (O) ≤class (A).
- *-property: It does not allow processes from writing to lower classification.
- That is, Subject is allowed to write object O only if class (A) ≥ class (O).
- These two properties are enhanced by tranquility property, which is described in two types: strong and weak.
- In strong tranquility property, we cannot change labels during system operation.
- In weak tranquility property, however, we can change labels during system operation without violating defined security policies.
- **The main advantages of weak tranquility property are that it give rights in lowest security session while starting user session, have its classification level reduced and all objects created.**

The Biba model above mandatory access control model only dealt with confidentiality of information but not with integrity of information.

- To provide integrity of information new mandatory model is designed by Ken Biba, which controls flow of information and does not allow subjects to modify information directly.
- Important (I), and Unknown (U).The integrity level describes user's responsibility for modifying, inserting, or deleting information.
- Object replicate both degree of trust and damage that happen due to unauthorized modifications of information.
- Access control is described according to following two principles: No-read-down In this, subject is allowed to read an object only if object control access class of subject.
- Integrity levels can be defined as follows: Crucial (C), Important (I), and Unknown (U).
- The integrity level describes the user's responsibility for modifying, inserting, or deleting information.
- Object replicate both the degree of trust and the damage that happen due to unauthorized modifications of info.
- Access control is described according to the following two principles:
- No-read-down In this, subject is allowed to read an object only if the object control the access class of the subject.
- No-write-up In this, subject is allowed to write an object only if the subject control the access class of the object.
- Simple Integrity Property: In which low integrity subject cant have the rights to write or modify high integrity data
- *-Property: In which high integrity subject cannot have the rights to read low integrity data.
- Biba also proposed alternative criteria for safeguarding integrity, by providing more vibrant controls. These contain the following two policies.
- Low-water mark for subjects it control write operations according to the no-write up principle. No restriction is forced on read operations.
- Low-water mark for objects it controls the read operations according to the no-read down principle. No restriction is forced on write operations.

- **Multilateral Security** As we know that, mandatory policies provide better security than discretionary policies, therefore it could be used to control indirect information flows.
- Different policies are proposed as mixture of mandatory flow control and discretionary authorizations. Here we describe some of them.

Disadvantages of MAC MAC models put restrictions on user access that, and according to security policies, does not allow for dynamic alteration.

- MAC needs to place operating system and associated utilities outside access control frame work.
- MAC requires predetermined planning to implement it effectively.
- After implementing it needs high system management because due to constantly update object and account labels to collect new data.

- **Role-based access control (RBAC)** For providing access rights to user it is important to know user's responsibilities assigned by organization.
- But in DAC user rights of data plays an important part, are not good and in MAC, users have to take security clearances and objects need security classifications.
- RBAC try to reduce gap by combining forced organizational constraints with flexibility of explicit authorizations.
- RBAC mostly used for controlling access to computer resources. RBAC is very useful method for controlling what type of information users can utilize on computer, programs that users execute, and changes that users can make.
- In RBAC roles for users are assigned statically, which is not used in dynamic environment.
- It is more difficult to change access rights of user without changing specified roles of user.
- RBAC is mostly preferable access control model for local domain.
- Due to static role assignment it does not have complexity.
- Therefore it needs low attention for maintenance.
- Role is nothing but abstractions of user behavior and their assigned duties.
- These are used to assign system resources to departments and their respective members.
- To provide accessing control with security in particular software systems it will be beneficial to use role concept.
- It also reduces cost of authority management.
- Essentially, in role based access control policies need to identify roles in system, role can be defined as set of responsibilities and actions associated with particular working activity.
- In an Access control security model, role is considered as job related access rights which can be given to authorized users within an organization.
- It allows authorized user to achieve its associated responsibilities.
- In respect to RBAC model we describe two types of subjects: users that related to system and transactions which are executed on behalf of those users.
- Users can access particular objects by executing transactions on that object.
- A transaction can be referred as set of executable operations which causes consumption of system resource.
- For example, In bank Tellers are allow to execute deposit and withdraw transaction, for that it requiring read and write access to specific fields within account.
- An account supervisor has same or more rights to perform correction transactions.
- System protection is based on permission that describes given access right to particular object or set of objects.
- In RBAC model we are dealt with unauthorized access to computer system resources and data.
- Since we have consider only access rights that users required to execute particular transaction on particular object from defined set of objects.

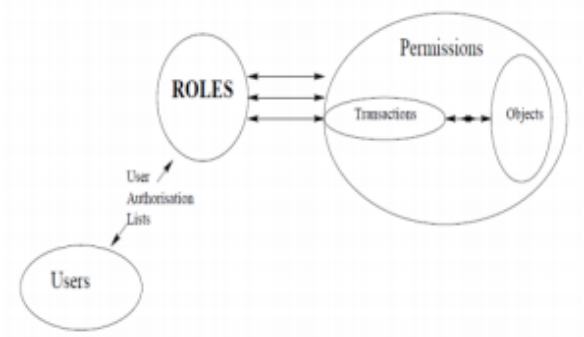


Fig 5: User-Role-Permission Mapping

A permission p is pair < trans, objset >, where trans represents transaction that executes on set of objects that is objset.

- Consider P indicate universal set of permissions, Trans indicate universal set of transactions, and Obj indicates set of objects.
- We can define association between permission/transaction and permission/object with following functions. TransP(p) : P → Tr, It gives associated transaction to specified permission p. ObP (p) : P → 2Obj ,It gives associated set of objects to specified permission p.

- A role is created by collecting permissions according to functional and logical requirements to this role should represent.
- Each role has name associated with this and it uniquely identifies this role in system. role r is pair of < rn, pset >, where rn indicates role name and pset indicates set of role permissions.
- The mapping between roles and permissions can be defined with following function: PR(r): R → 2 P , It gives associated set of permissions to specified role r.
- Here R indicates universal set of roles. While allocating permissions to roles it is need to ensure principle of least privilege that is each role should have only required rights for its functional requirements.

Advantages of RBAC: Authorization management: Role-based policies provide logical independence in specifying user authorizations. user authorizations task can be broken down into: i) assigning roles to particular users, and ii) assigning objects to roles.

- This make simpler to manage security policy: For example, when new user joins organization, administrator of system needs to grant particular roles as per job responsibilities; If user's job responsibilities get changed, then administrator needs to change roles associated with that user; when new task or program is added to security system, then administrator needs to decide which roles are provided to execute it.
- **Disadvantages of RBAC** In RBAC model, there is still some work to be done to cover up all requirements which may represent real world scenario. Defining roles in different context is difficult and it may result into large role definition. Sometimes it produces more roles than users.
- Now days, require fine grained results but RBAC not gives fine grained results.
- RBAC assigns roles statically to its user, which is not preferred in dynamic environment. It is difficult to implement when environment is dynamic and distributed. Due to this it is more difficult to change access rights of user without changing role of that user.
- Thus, RBAC not provide support for dynamic attributes such as time of day on which user permission is determined
- It maintains relation between users and its roles. It also maintains relation between permissions and roles.
- Therefore to implement RBAC model roles must be assigned in advance and it is not possible to change access rights without altering roles.

Module 4: Software Security

Malware regroups viruses, spyware, Trojans, and all sorts of small programs designed to harm your system, steal information, track your activities etc

The different types of malicious attacks are as follows:

- **Virus**: virus is form of malware that is capable of copying itself and spreading to other computers.

1. Viruses often spread to other computers by attaching themselves to various programs and executing code when user launches one of those infected programs.
2. Viruses can also spread through script files, documents, and cross-site scripting vulnerabilities in web apps.

- **Worm**: Computer worms are among most common types of malware. They spread over computer networks by exploiting operating system vulnerabilities.

1. Worms typically cause harm to their host networks by consuming bandwidth and overloading web servers.
2. Computer worms can also contain "payloads" that damage host computers.
3. Payloads are pieces of code written to perform actions on affected computers beyond simply spreading worm. Payloads are commonly designed to steal data, delete files, or create botnets.

- **Trojan Horse**: Trojan horse, commonly known as "Trojan," is type of malware that disguises itself as normal file or program to trick users into downloading and installing malware.

1. A Trojan can give malicious party remote access to an infected computer.
2. Once an attacker has access to an infected computer, it is possible for attacker to steal data (logins, financial data, even electronic money), install more malware, modify files, monitor user activity (screen watching, key logging, etc), use computer in botnets, and anonymise internet activity by attacker.

- **Spyware and Adware**: Spyware is type of malware that funct's by spying on user activity without their knowledge.

1. These spying capabilities can include activity monitoring, collecting keystrokes, data harvesting (account information, logins, financial data), and more.
2. Spyware often has additional capabilities as well, ranging from modifying security settings of software or browsers to interfering with network connections.
3. Spyware spreads by exploiting software vulnerabilities, bundling itself with legitimate software, or in Trojans.
4. Adware (short for advertising-supported software) is type of malware that automatically delivers advertisements.

5. Common examples of adware include pop-up ads on websites and advertisements that are displayed by software.

- **Rootkit**: rootkit is type of malicious software designed to remotely access or control computer without being detected by users or security programs.

1. Once rootkit has been installed it is possible for malicious party behind rootkit to remotely execute files, access/steal information, modify system configurations, alter software (especially any security software that could detect rootkit), install concealed malware, or control computer as part of botnet.

- **Zombies and Botnets**: zombie computer, usually known in short form zombie, is computer attached to Internet that has been compromised and manipulated without knowledge of computer owner.

1. Botnet refers to network of zombie computers that have been taken over and put under remote control of attacker.

- **Logic Bombs**: logic bomb is program code which is embedded in another program, and can be activated when certain predefined criteria are met.

1. For instance, time bomb will attack system and erase all data if licence key or another program code is not found in system. In some cases, logic bomb will inform attacker via Internet that bomb is ready to attack victim.

- **Trap Door**: trap door is secret entry point into program that is intentionally included in program code. While it can facilitate debugging during program development, it may be used for malicious purposes as well.

Salami Attack: salami attack is when small attacks add up to one major attack that can go undetected due to nature of this type of cyber crime. It also known as salami slicing. Although salami slicing is often used to carry out illegal activities, it is only strategy for gaining an advantage over time by accumulating it in small increments, so it can be used in perfectly legal ways as well .The attacker uses an online database to seize information of customers that is bank/credit card details deducting very little amounts from every account over period of time. customers remain unaware of slicing and hence no complaint is launched thus keeping hacker away from detection.

Salami Attack Incidents: In January 1993, four executives of rental-car franchise in Florida were charged with defrauding at least 47,000 customers using salami technique. In Los Angeles, in October 1998, district attorneys charged four men with fraud for allegedly installing computer chips in gasoline pumps that cheated consumers by overstating amounts pumped. In 2008, man was arrested for fraudulently creating 58,000 accounts which he used to collect money through verification deposits from online brokerage firms few cents at time. While opening accounts and retaining funds may not have been illegal by themselves, authorities charged that individual opened accounts using false names (including those of cartoon characters), addresses, and social security numbers, thus violating laws against mail fraud, wire fraud, and bank fraud.

How to identify salami attack: The only way to detect salami attack according to me is to perform rigorous white box testing by checking each and every line of code which is exhaustive but that's only way.

a) corporate has to update security of system as high as possible so that if attacker is taking advantage of any loophole than that bug is patched and attack is avoided.

b) Also those banks should advise customers on reporting any kind of money deduction that they aren't aware that they were part of. Whether small or big amount, banks should encourage customers to come forward and openly tell them that this could mean that an act of fraud could very well be scenario.

c) Most Important according to me is that Customers should ideally not store information online when it comes to bank details, but of course they can't help fact that banks rely on network that has all customers hooked onto common platform of transactions that require database. safe thing to do is to make sure bank/website is highly trusted and hasn't been part of slanderous past that involved fraud in any way.

This attack is not only on banks but also on any entity where slicing can be performed and people are made unaware of crime.

Lineraisation Attack: Extended Sparse Linearization (XSL) attack is method of cryptanalysis for block ciphers.

- Since AES is already widely used in commerce and govt for transmission of secret information, finding technique that can shorten amnt of time it takes to retrieve secret message without having key could have wide implications.
- The method has high work-factor, which unless lessened, means technique does not reduce effort to break AES in comparison to an exhaustive search. Therefore, it does not affect real-world security of block ciphers in near future.
- Overview:XSL attack relies on 1stanalyzing internals of cipher&deriving system of quadratic simultaneous equations. These systems of equations are typically very large, for eg 8,000 equations with 1,600 variables for 128-bit AES
- Several methods for solving such systems are known. In XSL attack, specialized algorithm, termed extended Sparse Linearization, is then applied to solve these equations and recover key.

- Attack is notable for requiring only handful of known plaintexts to perform; previous methods of cryptanalysis, such as linear and differential cryptanalysis, often require unrealistically large numbers of known or chosen plaintexts.
- **Example:** include <stdio.h>

```
int main(intargc, const char *argv[])
{          inti;
   char serial[9] = “S123N456\n”;
   for(I = 0;I < 8;++i)
   {  if(argv[1][i] ! = serial[i]) break;
   }       if(i==8)
   { printf(“\n Serial  number is   correct !\n\n”);}}
```

  - Program checks for serial number S123N456
  - For efficiency, check made one character at time
  - Correct string takes longer than incorrect
  - Attacker tries all 1 character strings
  - Finds S takes most time
  - Attacker then tries all 2 char strings S∗S∗ Finds S1 takes most time
  - Attacker is able to recover serial number one character at time.
  - **What is advantage of attacking serial number one character at time?**
    Suppose serial number is 8 characters and each has 128 possible values
    i. Then 1288=2561288=256 possible serial numbers
    ii. Attacker would guess serial number in about 255255 tries- lot of work
    iii. Using linearization attack, work is about 8*(128/2) = 2929 which is trivial
- **A real-world linearization attack**: TENEX (an ancient timeshare system)
  i. Passwords checked one character at time
  ii. Careful timing was not necessary
  iii. could arrange for “page fault” when next unknown character guessed correctly
  iv. page fault register was user accessible
  v. Attack was very easy in practice.

Module 5: Operating System Security

Linux Security Model: **Linux Security Modules** (**LSM**) is framework that allows Linux kernel to support variety of computer security models while avoiding favoritism toward any single security implementation.

- The framework is licensed under terms of GNU General Public License and is standard part of the
- Linux kernel since Linux 2.6. AppArmor, SELinux, Smack, TOMOYO Linux and Yama are currently accepted modules in official kernel. LSM was designed to provide specific needs of everything needed to successfully implement mandatory access control module, while imposing fewest possible changes to Linux kernel.
- LSM avoids approach of system call interposition as used in Systracebecause it does not scale to multiprocessor kernels and is subject to TOCTTOU (race) attacks. Instead, LSM inserts "hooks" (upcalls to module) at every point in kernel where user-level system call is about to result in access to an important internal kernel object such as inodes and task control blocks.
- The project is narrowly scoped to solve problem of access control to avoid imposing large and complex change patch on mainstream kernel. It is not intended as general "*hook*" or "*upcall*" mechanism, nor does it support Operating system-level virtualization.
- LSM's access control goal is very closely related to problem of system auditing, but is subtly different.
- Auditing requires that every attempt at access be recorded. LSM cannot deliver that, because it would require great many more hooks, so as to detect cases where kernel "*short circuits*" failing system calls and returns an error code before getting near significant objects.
- The LSM design is described in paper *Linux Security Modules: General Security Support for Linux Kernel* presented at USENIX Security 2002.
- At same conference was another paper which studied automatic static analysis of kernel code to verify that all of necessary hooks have actually been inserted into Linux kernel.

- Some Linux kernel developers dislike LSM for variety of reasons.
- LSM strives to impose least overhead possible, especially in case where no module is loaded, but this cost is not zero, and some Linux developers object to that cost.
- LSM is designed to provide only for access control, but does not actually prevent people from using LSM for other reasons, and so some Linux kernel developers dislike that it can be "abused" by being used for other purposes, especially if purpose is to bypass Linux kernel's GPL license with proprietary module to extend Linux kernel functionality.
- Some security developers also dislike LSM. author of grsecurity dislikes LSM because of its history, and that because LSM exports all of its symbols it facilitates insertion of malicious modules (rootkits) as well as security modules.
- The author of RSBAC dislikes LSM because it is incomplete with respect to needs of RSBAC.
- In particular, author of RSBAC argues that: "LSM is only about additional, restrictive access control.
- However, RSBAC system provides lot of additional functionality, e.g. symlink redirection, secure_delete, partial Linux DAC disabling.
- All this has to be patched into kernel functions in separate patch.".
- The Dazuko project argued that targeting LSM API is moving target, as it changes with each kernel release, leading to extra maintenance work.
- Other developers would like to have LSM modules stacked, e.g. developer of Yama LSM.

Fie system Security: Linux FS is quite simplistic in design, yet quite effective in controlling access to files and directorys

- Directories and files which are stored in them are arranged in hierarchical tree structure. Access can be controlled for both files and directories allowing very flexible level of access.

Linux Vulnerabilities: The list for Unix/Linux vulnerabilities currently includes (vulnerabilities that represent additional danger in large corporate environment due to number of servers with those applications installed):

- **Services like Webmin, phpmyadmin, Cpanel**, etc that provide Web-based interface for administrators and webmasters. They are typically attacked by script kiddies 24 x 7. As they typically they do not understand what they are doing this is just noise, but still it requires some means to hide ports and make URLs unique for site. You should never have generic URLs for those applications (Cpanel is an exception for obvious reasons).
  - **Web and Application Server Misconfiguration:** It is of critical importance that secure Web application have strong server configuration standard.
- **PHP Web applications misconfigurations or, more commonly, bugs in major PHP applications.** PHP is rarely used in enterprise environment, but it attracts lion share of exploits directed against Web sites. top ten are:
  - **Unvalidated Parameters:** Those intent on an attack can use this flaw to get at backside components through Web application. Information from Web requests is not validated before being used by Web application.
  - **Broken Access Control:** These flaws can be taken advantage of by hackers to access other users' accounts, view sensitive files or use unauthorized functions. flaws occur because restrictions on what authenticated users are allowed to do are not properly enforced.
  - **Broken Account and Session Management:** This vulnerability occurs when account credentials and session tokens are not properly protected. Attackers can assume other users' identities by compromising passwords, keys, session cookies or other tokens.
  - **Cross-Site Scripting (XSS) Flaws:** An attacker can use Web application as mechanism to transport an attack to an end user's browser. Successfully utilizing this method can disclose end users' session token, attack local machine or spoof content to fool user.
  - **Buffer Overflows:** Web application components in some languages that do not properly validate input can be crashed and, in some cases, used to take control of process. These components can include CGI, libraries, drivers and Web application server components.
  - **Command Injection Flaws:** Web applications pass parameters when they access external systems or local operating system. If an attacker can embed malicious commands in these parameters, external system may execute those commands on behalf of Web application.
  - **Error Handling Problems:** If an attacker is able to create errors that Web application does not handle, they can gain detailed system information, deny service, cause security mechanisms to fail or crash server.
  - **Insecure Use of Cryptography:** Web applications often use cryptographic functions to protect information and credentials. These functions, along with code to integrate them, have proven difficult to code properly, often resulting in weak protection.

- **Remote Administration Flaws:** Many Web applications allow administrators to access site using Web interface. If these administrative functions are not very carefully protected, an attacker can gain full access to all aspects of site.
- **Secure Shell (SSH) --** paradoxically ssh due to complexity of protocol and its implementation is most common way to get into remote system. Stream of remotely exploitable openssh vulnerabilities has long and ugly history.
- **BIND** Domain Name System (large corporations often use appliances for DNS, but that does not mean that they are more secure ;-).
- **Java.** Exploits against Java programs are actually far more common then exploits against Apache. See list of PHP vulnerabilities. It is fully applicable. Popular Java-based application can be attacked like popular PHP application with application-specific exploits.
- **Open Secure Sockets Layer (SSL)** -- like SSH it has history of nasty exploits. Patching is very important.
- **Remote Procedure Calls (RPC).** There were several famous malware worms instances that used those vulnerabilities to spread inside corporate environment.
- **Apache Web Server** (large corporations generally use HTTP proxy and firewall before any Web server, which limit impact of even unpatched exploits)
- **General UNIX Authentication problems such as accounts with no passwords or weak passwords.** In modern environment chances of cracking password via external login are close to zero, unless attacker has additional information about victim. But still prudence dictates necessity to two factor authentication for any Internet facing server. In this sense many US banks, brokers and financial companies are simply negligent. For you as customer ability to provide token is now an important criteria of choosing financial institution. ***For example both eTrade and Paypal provide tokens to customers.*** No externally facing corporate server can use just password authentication. Tokens are really cheap and increase security. CMS protocol and cell phones can be used instead. In this case CMS message is send and serve as second password after user authenticated, One time passwords , mascots , etc also increase security of authentication. Individual user mascots allow to distinguish real site from fake one.
- **Clear Text Services and shell vulnerabilities (predominantly bash). most recent was Shellshock (September, 2014).**
- **Sendmail and set of email based exploits including phishing.** Sendmail is outdated and poorly understood by administrator program with multiple vulnerabilities and dismal history of exploits that is still widely used in large corporate environment. program still has vulnerabilities that were not patched in common distributions but they are becoming more rate. Sendmail mainly is used as transport mechanism for malware and here weakest link is not software but users. Some phishing attempt are so ingenious, that they can full large number of users.
- **Simple Network Management Protocol (SNMP).** This is little understood by typical sysadmin protocol which if enabled often has mistakes in configuration. Generally it should be restricted to relevant IPs. Opening it to world is big mistake.
- **Misconfiguration of Enterprise Services like NIS (which is still widely used, but gradually is replaced with LDAP), NFS, Samba, etc. Often IP ranges used to open those services are way too wide.** additional problem with Samba is that Unix sysadmin often does not clearly understand mapping of Windows file attributed into extended attributes. Although there are thousands of security incidents each year affecting major Linux distribution, majority of successful attacks target one or more of these vulnerable services. Attackers usually are opportunistic. They take easiest and most convenient route and exploit best-known flaws with most effective and widely available attack tools. They count on organizations to be behind in patching, especially patching of application and protocols like SSL, not fixing well-known problems. They often attack indiscriminately, scanning Internet for any vulnerable systems. The best strategy for large corporate is avoidance. On Unix and Linux side, Berkeley Internet Domain Name (BIND) software remains top problem software. That means that large corporate should never try to run bind on Linux. Similarly Apache as an external web server should generally work via HTTP proxy. Generally apache is way too complex to be used as Internet facing Web server (but it can and should be used as an internal WEB server, due to its functional superiority over competitors). Running Sendmail on Linux is not recommended for same reason, as at number 6 it belongs to most vulnerable software on Unix/Linux servers. In major distribution it was replaced by postfix long ago, so this is only inertia that dictates continued use of Sendmail in enterprise environment.

Windows Security Architecture: Windows based Operating Systems is most common OS used in world. In this post we will gain some knowledge about Security Architecture of Windows which is very important before performing Hacks on Windows based machine. There are three components of Windows Security:

LSA (Local Security Authority): LSA is Central Part of NT Security. It is also known as Security Subsystem. Local Security Authority or LSA is key component of logon process in both Windows NT and Windows 2000. In Windows

2000, LSA is responsible for validating users for both local and remote logons. LSA also maintains local security policy. During local logon to machine, person enters his name and password to logon dialog. This information is passed to LSA, which then calls appropriate authentication package. password is sent in non-reversible secret key format using one-way hash function. LSA then queries SAM database for User's account information. If key provided matches one in SAM, SAM returns users SID and SIDs of any groups user belongs to. LSA then uses these SIDs to generate security access token.

SAM (Security Account Manager: Security Accounts Manager is database in Windows operating system (OS) that contains user names and passwords. SAM is part of registry and can be found on hard disk. This service is responsible for making connection to SAM database (Contains available user-accounts and groups). SAM database can either be placed in local registry or in Active Directory (If available). When service has made connection it announces to system that SAM-database is available, so other services can start accessing SAM-database. In SAM, each user account can be assigned Windows password which is in encrypted form. If someone attempts to log on to system and user name and associated passwords match an entry in SAM, sequence of events takes place ultimately allowing that person access to system. If user name or passwords do not properly match any entry in SAM, an error message is returned requesting that information be entered again. When you make New User Account with Password, it gets stored in SAM File.

Windows Security Files are located at "C:\Windows\System32\Config\SAM".

The moment operating system starts, SAM file becomes inaccessible.

SRM (Security Reference Monitor) Security Reference Monitor is security architecture component that is used to control user requests to access objects in system. SRM enforces access validation and audit generation. Windows NT forbids direct access to objects. Any access to an object must first be validated by SRM. For example, if user wants to access specific file SRM will be used to validate request. Security Reference Monitor enforces access validation and audit generation policy. The reference monitor verifies nature of request against table of allowable access types for each process on system. For example, Windows 3.x and 9x operating systems were not built with reference monitor, whereas Windows NT line, which also includes Windows 2000 and Windows XP, was designed with an entirely different architecture and does contain reference monitor.

Windows user account architecture

User account passwords are contained in SAM in Hexadecimal Format called Hashes.

Once Passwords converted in Hashes, you cannot convert back to Clear Text.

Windows Vulnerabilities:The understanding Windows based vulnerabilities, organizations can stay step ahead and ensure information availability, integrity, and confidentiality. **Listed below are Top 10 Windows Vulnerabilities:**

1. Web Servers –Misconfigurations, product bugs, default installations, and third-party products such as php can introduce vulnerabilities.

2. Microsoft SQL Server –Vulnerabilities allow remote attackers to obtain sensitive information, alter database content, and compromise SQL servers and server hosts.

3. Passwords –User accounts may have weak, nonexistent, or unprotected passwords. operating system or third-party applications may create accounts with weak or nonexistent passwords.

4. Workstations –Requests to access resources such as files and printers without any bounds checking can lead to vulnerabilities. Overflows can be exploited by unauthenticated remote attacker executing code on vulnerable device.

5. Remote Access (RA) –Users can unknowingly open their systems to hackers when they allow RA to their systems.

6. Browsers –Accessing cloud computing services puts an organization at risk when users have unpatched browsers. Browser features such as Active X and Active Scripting can bypass security controls.

7. File Sharing –Peer to peer vulnerabilities include technical vulnerabilities, social media, and altering or masquerading content.

8. E-mail –By opening message recipient can activate security threats such as viruses, spyware, Trojan horse programs, and worms.

9. Instant Messaging –Vulnerabilities typically arise from outdated ActiveX controls in MSN Messenger, Yahoo! Voice Chat, buffer overflows, and others.

10. USB Devices –Plug and play devices can create risks when they are automatically recognized and immediately accessible by Windows operating systems.