SEZG566/SSZG566

Secure Software Engineering

Web Application Security

T V Rao

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

- *The slides presented here are obtained from the authors of the books, product documentations, and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified slides to suit the requirements of the course.*

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Web Application Security

Web Security

Web has two distinct components

- Server
  - Has resources to serve. Content(resources) delivered to the client based on URL etc.
  - Generally managed by an administrator.
  - Self-service models reduce control
- Client/Browser
  - The component that is likely to approach a server for content.
  - Meant for end-user(non-technical)

Security

- Need to consider both sides for protecting users

Goals of web security

Safely browse the web

- Users should be able to visit a variety of web sites, without incurring harm:
  - No stolen information
  - Site A cannot compromise session at Site B

Secure web applications

- Applications delivered over the web should have the same security properties we require for stand-alone applications

Web Security

During early stages of web, entire security focus was on servers.

Today browsers have acquired lot of capabilities:

- JavaScript: Allows a page to execute client--side code.
- DOM model Provides a JavaScript interface to the page's HTML, allowing the page to add/remove tags, change their styling, etc.
- XMLHttpRequests (AJAX): Asynchronous HTTP requests.
- Web sockets: Full--duplex client--server communication over TCP.
- Web workers: Multi--threading support.
- Multimedia support: (video), web cams, screen--sharing.
- Geolocation: Browser can determine your location by examining GPS units. Firefox can also locate you by passing your WiFi information to the Google Location Service.
- <canvas> and WebGL: Bitmap manipulation and interactive 2D/3D graphics.
  - WebGL (Web Graphics Library) is a JavaScript API for rendering graphics within any compatible web browser without the use of plug-ins
- Nacl (Native Client): Allows browsers to run native code!

Browser Vulnerabilities

Whenever a browser communicates with a website,

- the website, as part of that communication, collects some information about the browser (in order to process the formatting of the page to be delivered, if nothing else).
- If malicious code has been inserted into the website's content, then vulnerabilities specific to a particular browser can allow this malicious code to run processes within the browser application in unintended ways (one of the bits of information that a website collects from a browser communication is the browser's identity- allowing specific vulnerabilities to be exploited).
- Once an attacker is able to run processes on the visitor's machine, then exploiting known security vulnerabilities can allow the attacker to gain privileged access to the victim's machine or network

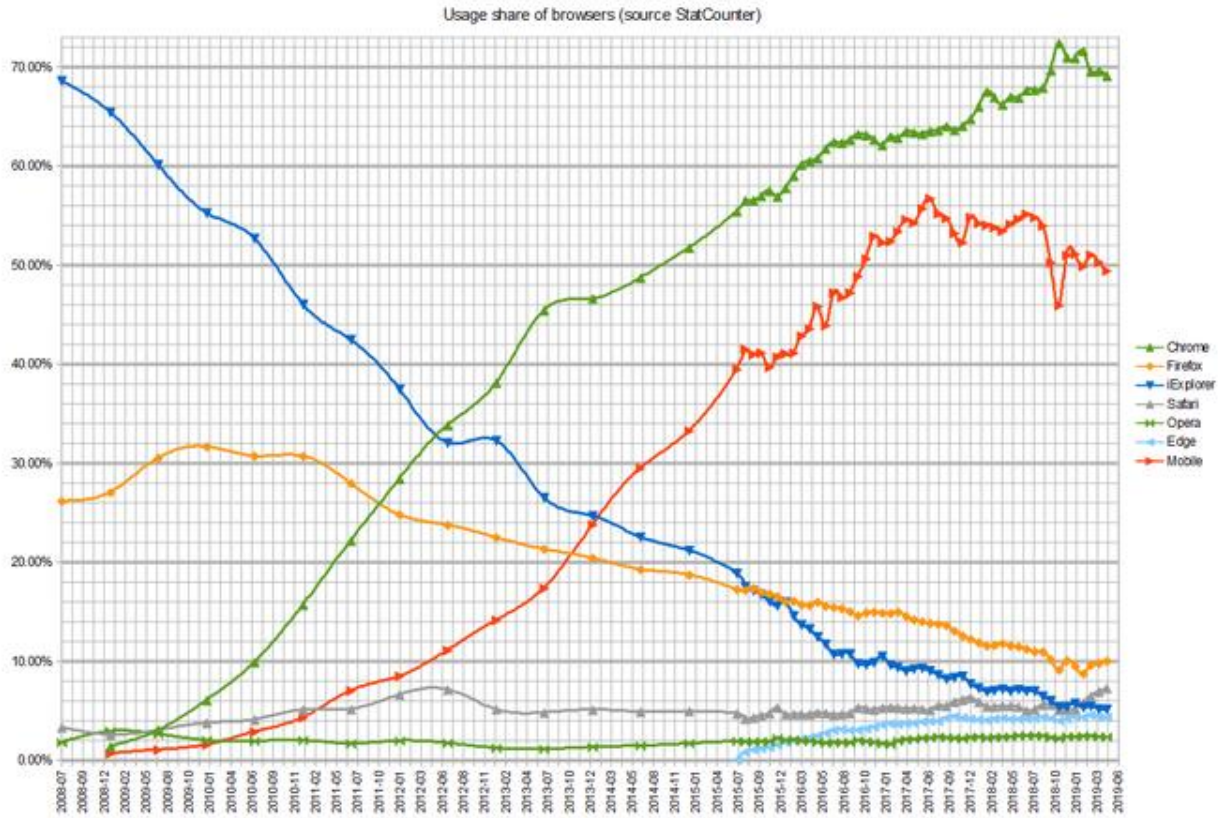
Securing Web Browser

Web browsers can be breached in one or more of the following ways:

- Operating system is breached and malware is reading/modifying the browser memory space in privilege mode
- Main browser executable can be hacked
- Browser components may be hacked
- Browser plugins can be hacked
- Browser network communications could be intercepted outside the machine

https://en.wikipedia.org/wiki/Browser_security

Usage Share of Browsers (Wikipedia)



Browser security

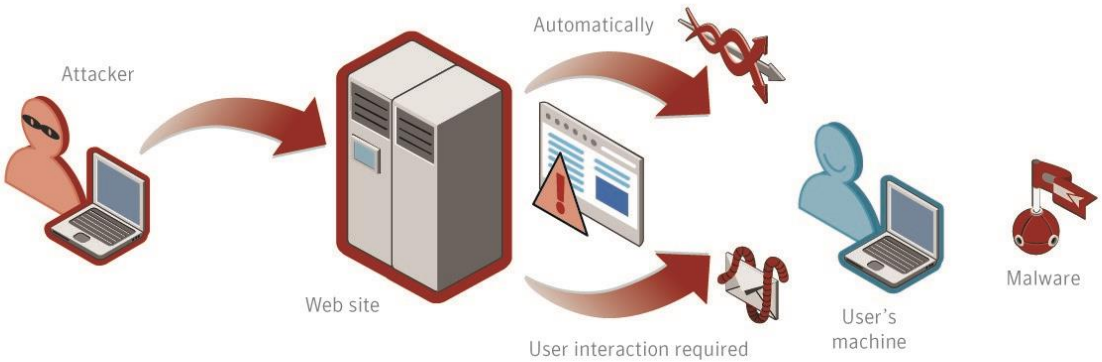
- Vulnerabilities in the web browser software itself can be minimized by keeping browser software updated, but will not be sufficient if the underlying operating system is compromised, for example, by a rootkit.
- Some subcomponents of browsers such as scripting, add-ons, and cookies are particularly vulnerable
- Improper Input Validation (CWE-20) and Improper Access Control (CWE-284) are the most occurring root causes for security vulnerabilities
- Hundreds of vulnerabilities are reported every year. Large number of vulnerabilities occurred in Chrome because of reusing or importing vulnerable versions of third party libraries

https://en.wikipedia.org/wiki/Browser_security

Large Attack Surface

- Web has become a complex platform for distributed computation
- Developed a Huge Attack Surface
- A single web application spans multiple programming languages, Operating Systems, hardware platforms, throwing up emergent vulnerabilities.
  - E.g. might be running Chrome on Windows interacting with a Linux server running Apache and interfacing with MySQL
  - Difficult (almost impossible) to verify end--to--end correctness
- The web specs are very long, very complex, sometimes contradictory, and constantly evolving (quirks at quirksmode.org and several security information sites)

Anatomy of Web Attack



1. How does Malware make its way onto Web sites?
2. How does Malware make its way onto user's machines?
3. Once on the user's machines what does Malware do?

www.Symantec.com white paper

Anatomy of Web Attack

- Typically attacker breaks into a legitimate Web site and posts malware
  - malware is not exclusive to malicious web sites. Often mainstream web sites are made to act as parasitic hosts that serve up malware to their unsuspecting visitors. Due to the complexity of modern web sites there are several techniques by which they are compromised.
- Attacking enduser machines
  - malware on a web site makes its way down on to a user's machine when that user visits the host Web site.
  - Some of the techniques enable this to happen with no user interaction – 'drive-by-download'.
  - Some techniques which do require some input from the user.
- Leveraging end user machines for malicious activity
  - The most malicious activities begin once new malware has established a presence on a user's machine.

Web : Risks & Defenses

- Risk #1: we don't want a malicious site to be able to trash my files/programs on my computer
  - Browsing to danger.com (or evil.com) should not infect my computer with malware, read or write files on my computer, etc.
  - Defense: Javascript is sandboxed; try to avoid security bugs in browser code; privilege separation; automatic updates; etc.
- Risk #2: we don't want a malicious site to be able to spy on or tamper with my information or interactions with other websites
  - Browsing to evil.com should not let evil.com spy on my emails in Gmail or buy stuff with my Amazon account
  - Defense: the same-origin policy - A security policy grafted on after-the-fact, and enforced by web browsers
    - A web browser permits scripts contained in a first web page to access data in a second web page, but only if both web pages have the same origin
      - An origin is defined as a combination of URI scheme, hostname, and port number
      - This policy prevents a malicious script on one page from obtaining access to sensitive data on another web page through that page's Document Object Model
- Risk #3: we want data stored on a web server to be protected from unauthorized access
  - Defense: server-side security

Chromium Architecture

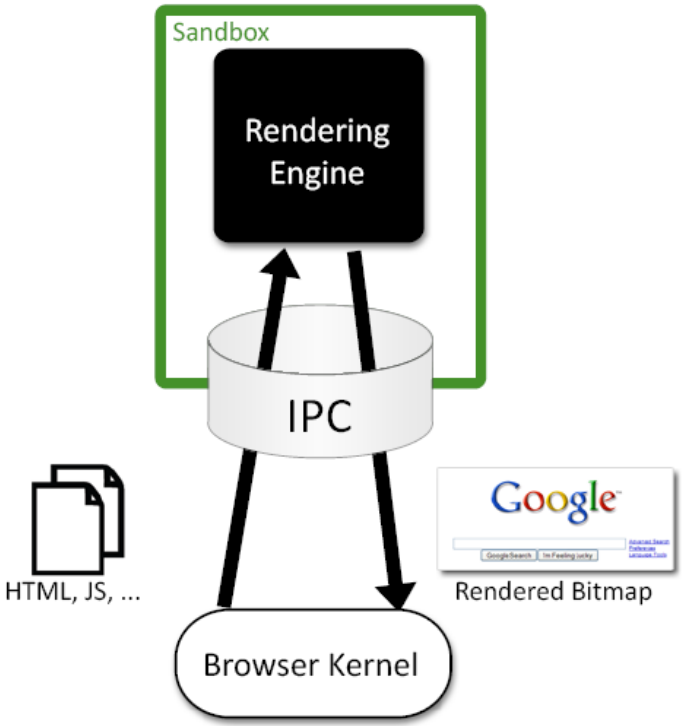
Chromium's architecture has two modules: a rendering engine and a browser kernel.

- The rendering engine is responsible for converting HTTP responses and user input events into rendered bitmaps.
- The browser kernel is responsible for interacting with the operating system.

The browser kernel is trusted to act as the user, whereas the rendering engine is trusted only to act as the web.

The Security Architecture of the Chromium Browser, Adam Barth et al
https://seclab.stanford.edu/websec/chromium/chromium-security-architecture.pdf

Chromium Architecture



The browser kernel treats the rendering engine as a black box that parses web content and emits bitmaps of the rendered document.

The Security Architecture of the Chromium Browser, Adam Barth et al

The assignment of tasks between the rendering engine and the browser kernel.

Rendering Engine	Browser Kernel
HTML parsing CSS parsing Image decoding JavaScript interpreter Regular expressions Layout Document Object Model Rendering SVG XML parsing XSLT	Cookie database History database Password database Window management Location bar Safe Browsing blacklist Network stack SSL/TLS Disk cache Download manager Clipboard
Both URL parsing Unicode parsing	

The Security Architecture of the Chromium Browser, Adam Barth et al

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

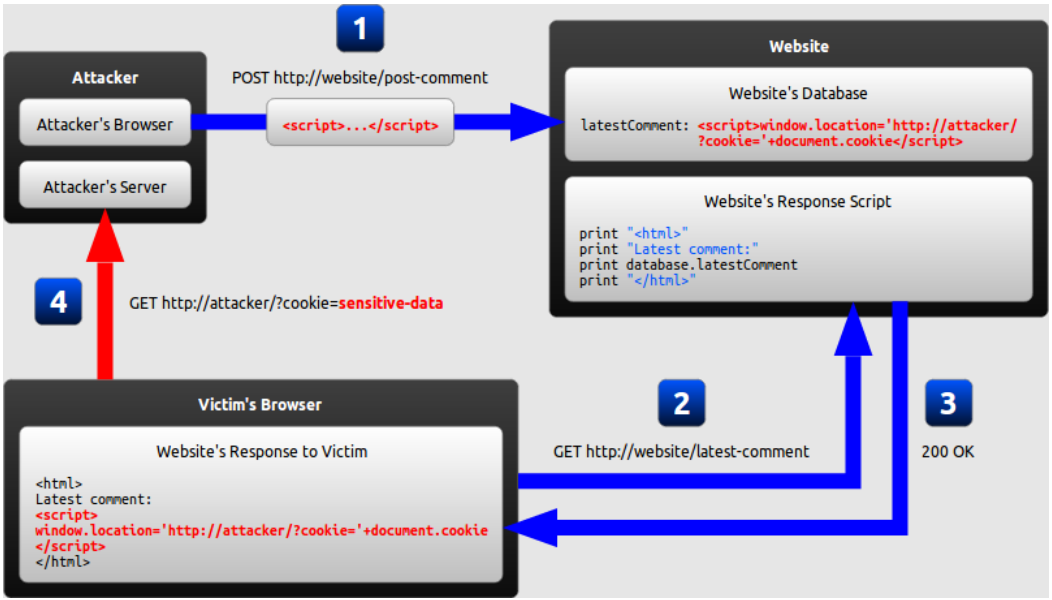
Cross-site Scripting

Cross-site Scripting attack

An XSS attack needs three actors — **the website**, **the victim** and **the attacker**

- The attacker injects a payload in the website's database by submitting a vulnerable form with some malicious JavaScript
- The victim requests the web page from the website
- The website serves the victim's browser the page with the attacker's payload(malicious javascript) as part of the HTML body.
- The victim's browser will execute the malicious script inside the HTML body. In this case it would send the victim's cookie to the attacker's server. The attacker now simply needs to extract the victim's cookie when the HTTP request arrives to the server, after which the attacker can use the victim's stolen cookie for impersonation.

Cross-site Scripting attack



http://www.acunetix.com/websitesecurity/cross-site-scripting/

What can an attacker can do with JavaScript?

JavaScript has access to the following

- Malicious JavaScript has access to all the same objects the rest of the web page has, including access to cookies. Cookies are often used to store session tokens, if an attacker can obtain a user's session cookie, they can impersonate that user
- JavaScript can read and make arbitrary modifications to the browser's DOM (within the page that JavaScript is running).
- JavaScript can use XMLHttpRequest to send HTTP requests with arbitrary content to arbitrary destinations.
- JavaScript in modern browsers can leverage HTML5 APIs such as accessing a user's geolocation, webcam, microphone and even the specific files from the user's file system. While most of these APIs require user opt-in, XSS in conjunction with some clever social engineering can bring an attacker a long way.

The consequences of malicious JavaScript

Cookie theft:

- The attacker can access the victim's cookies associated with the website using document.cookie, send them to his own server, and use them to extract sensitive information like session IDs.

Keylogging:

- The attacker can register a keyboard event listener using addEventListener and then send all of the user's keystrokes to his own server, potentially recording sensitive information such as passwords and credit card numbers.

Phishing:

- The attacker can insert a fake login form into the page using DOM manipulation, set the form's action attribute to target his own server, and then trick the user into submitting sensitive information.

XSS Examples

The wife of the former prime minister Gordon Brown, who has more than a million followers on Twitter, unknowingly sent a link which contained malicious code that would redirect anyone who moved their mouse over it - but didn't click it - to an evil site.

- The problem arises because users are able to post chunks of Javascript program code inside tweets - and because Twitter has not taking precautions to disable the code by "escaping" the relevant characters, the Javascript becomes active.
- The specific code being used is onMouseOver, which carries out a function when you move the mouse over the link. Users don't have to click the link to be redirected.

https://www.theguardian.com/technology/blog/2010/sep/21/twitter-bug-malicious-exploit-xss

With a day to go before a critical Pennsylvania Democratic primary, Barack Obama's team has been busy patching security holes.

- According to Netcraft, a hacker exploited security flaws in Obama's site to redirect traffic to Hillary Clinton's site. Anyone that visited Obama's community blogs section of the site was sent to Clinton.

http://www.zdnet.com/article/obama-site-hacked-redirected-to-hillary-clinton

More sophisticated ones: https://brightsec.com/blog/xss-attack/

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

XSS Variants

Variants of XSS

Cross-site scripting vulnerabilities may be of two broad types:

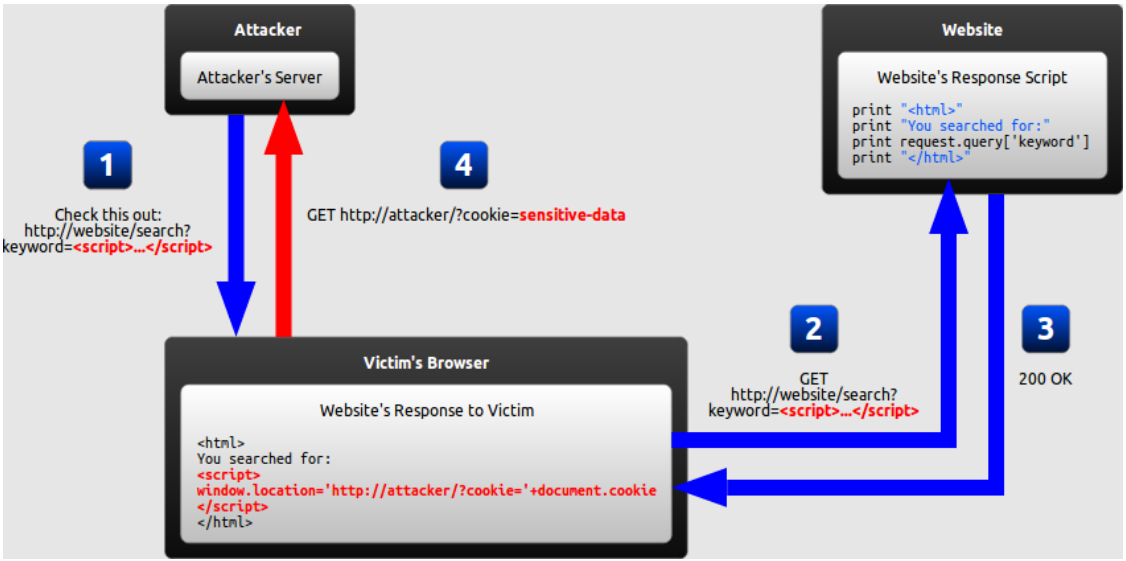
- Persistent
  - Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the malicious code.
- Non-persistent
  - Non-persistent attacks (and DOM-based attacks) require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form

They may further be divided into two varieties:

- Traditional (caused by server-side code flaws) and
- DOM-based (in client-side code)

Reflected XSS (non-persistent)

In a reflected XSS attack, the malicious string is part of the victim's request to the website. The website then includes this malicious string in the response sent back to the user



http://excess-xss.com/

Reflected XSS

1. The attacker crafts a URL containing a malicious string and sends it to the victim.
2. The victim is tricked by the attacker into requesting the URL from the website.
3. The website includes the malicious string from the URL in the response.
4. The victim's browser executes the malicious script inside the response, sending the victim's cookies to the attacker's server.

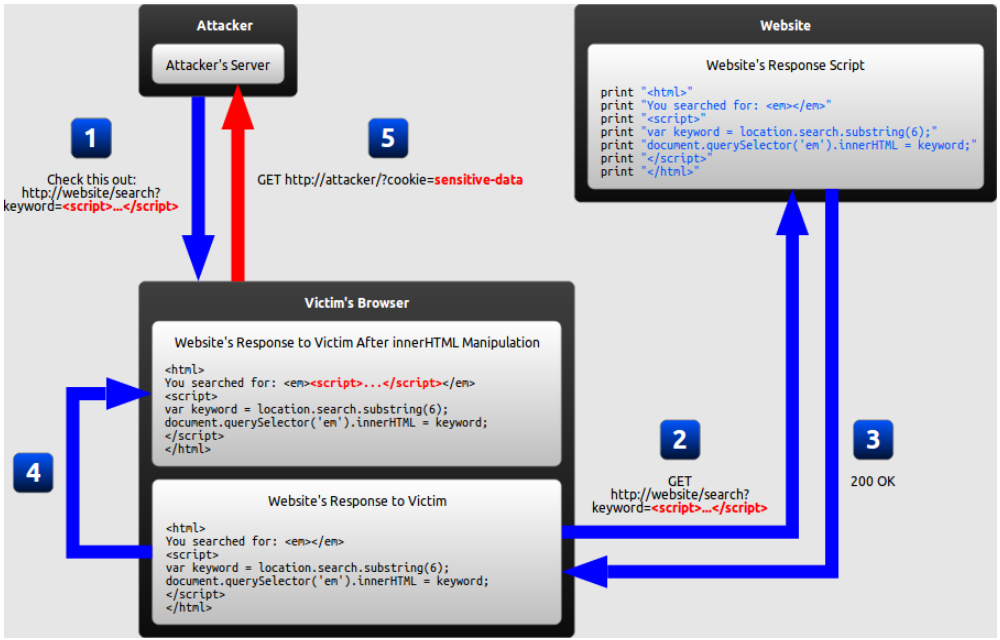
How can reflected XSS succeed?

- Reflected XSS requires the victim to actually send a request containing a malicious string, hence it may seem unlikely attack.
- Two common ways of causing a victim to launch a reflected XSS attack against himself
  - If attacker targets a specific individual, he can send the malicious URL to the victim (using e-mail or instant messaging, for example) and trick him into visiting it.
  - If the attacker targets a large group of people, he can publish a link to the malicious URL (on his own website or on a social network, for example) and wait for visitors to click it.
- With the use of a URL shortening service, which masks the malicious string from users, the chances of successful attack increase.

DOM-based XSS

- DOM-based XSS is a variant of both persistent and reflected XSS.
- Here the legitimate website does not send attacker's script
- In a DOM-based XSS attack, the malicious string is parsed by the victim's browser after the website's legitimate JavaScript is executed
- The legitimate script directly makes use of user input in order to add HTML to the page
- Since the malicious string is inserted into the page using innerHTML, it is parsed as HTML, causing the malicious script to be executed
- Even with completely secure server-side code, the client-side code might still unsafely include user input in a DOM update after the page has loaded

DOM-based XSS



DOM-based XSS Sequence

1. The attacker crafts a URL containing a malicious string and sends it to the victim.
2. The victim is tricked by the attacker into requesting the URL from the website.
3. The website receives the request, but does not include the malicious string in the response.
4. The victim's browser executes the legitimate script inside the response, causing the malicious script to be inserted into the page.
5. The victim's browser executes the malicious script inserted into the page, sending the victim's cookies to the attacker's server

BITS Pilani

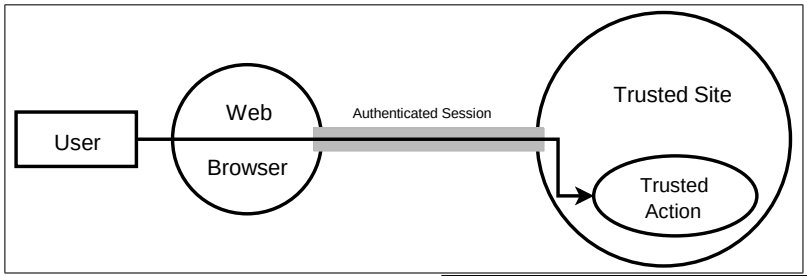
Pilani | Dubai | Goa | Hyderabad

Cross-Site Request Forgery

Cross-Site Request Forgery

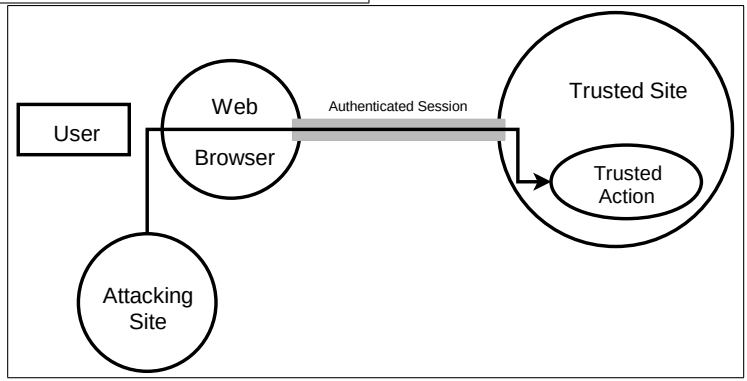
- Cross-Site Request Forgery (CSRF) is an attack that forces a victim to execute unwanted actions on a web application on which he/she is currently authenticated.
- CSRF attacks specifically target state-changing requests, (no direct theft of data), since the response to the forged request goes to victim.
- With social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing (including fund transfers etc.)

Cross-Site Request Forgery (CSRF)



A valid request.

A CSRF attack



Common ways to perform a CSRF attack

Common ways to execute CSRF attacks is by using

a HTML image tag,

e.g. **IMG SRC**

```
<img src="http://host/?command">
```

or JavaScript image object

e.g. **SCRIPT SRC**

```
<script src="http://host/?command">
```

CSRF Attacks

- Vulnerability discovered in January 2007 which allowed a attacker to steal a GMail user's contact list.
- Discovered in Netflix which allowed an attacker to change the name and address on the account, as well as add movies to the rental queue etc.

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

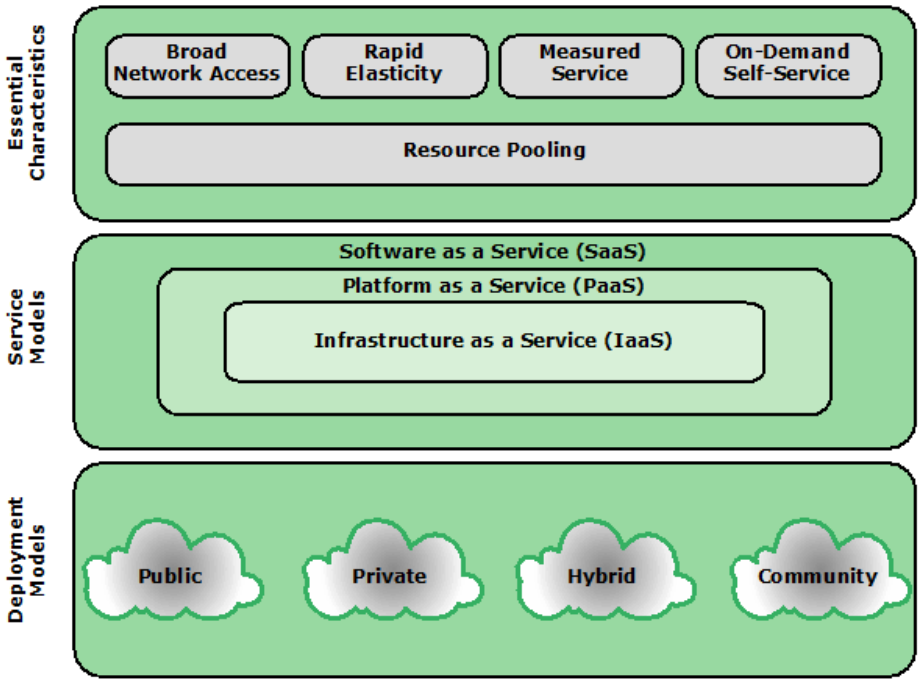
Cloud Security

Cloud Computing

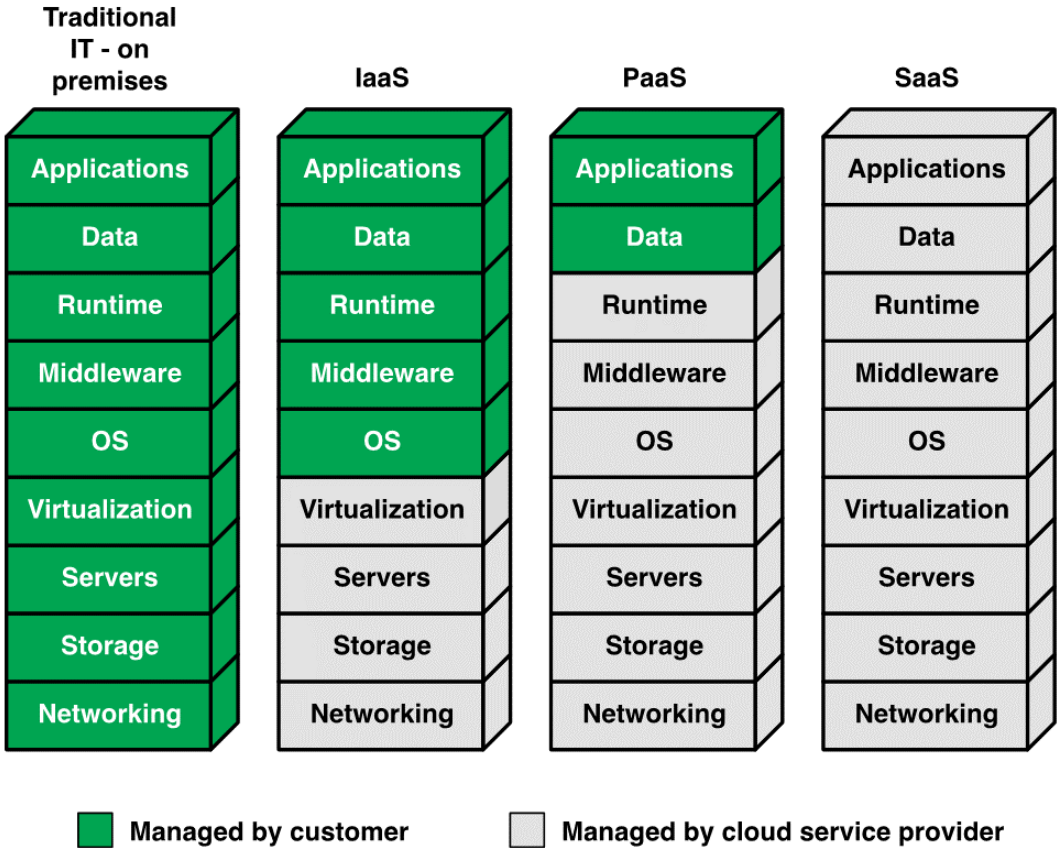
As per NIST,

"Cloud computing is A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics,

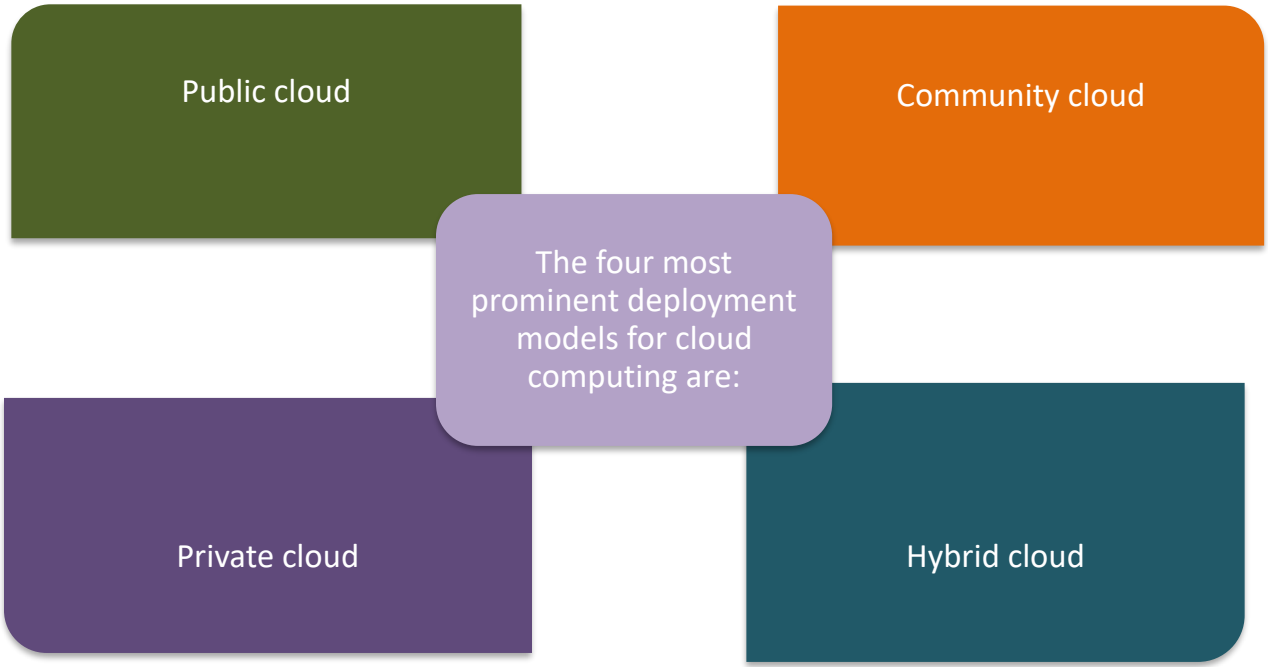
three service models, and four deployment models."



Cloud Service Models



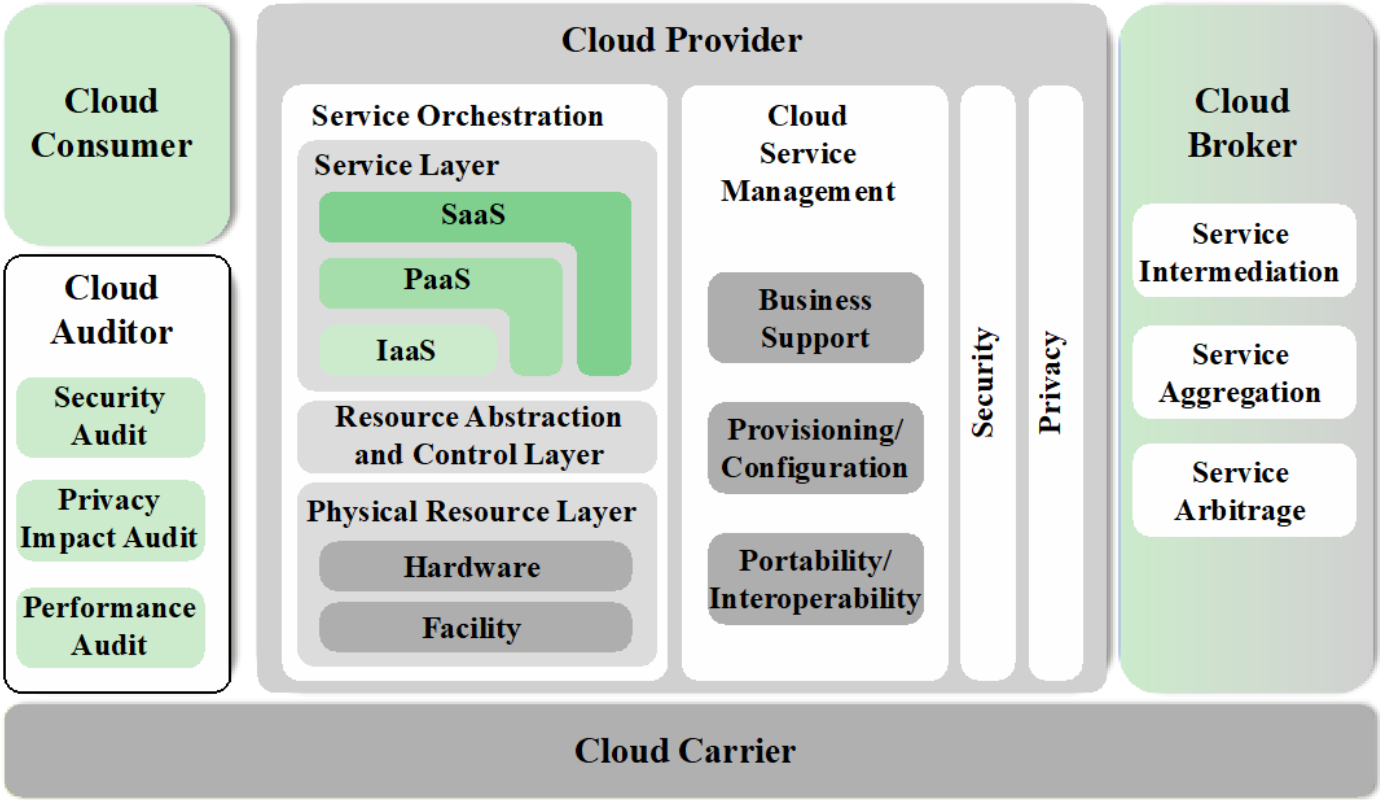
Cloud Deployment Models



Cloud Deployment Models

Public Cloud	Private cloud	Hybrid cloud	Community cloud
Makes it possible for anybody to access systems and services. Infrastructure in this cloud model is owned by the provider. **Minimal Investment** **No setup cost** **Infrastructure Management is not required** **No maintenance:** **Dynamic Scalability**	The cloud platform is implemented in a secure environment under the supervision of an organization's IT department. The private cloud gives the greater flexibility of control over cloud resources. **Better Control** **Data Security and Privacy** **Supports Legacy Systems** **Customization**	Bridges the public and private clouds with a layer of proprietary software. Organizations can move data and applications between different clouds depending on their needs. **Flexibility and control** **Cost** **Security**	Systems and services to be accessible by a group of organizations. The infrastructure could be shared between the organization which has shared concerns or tasks, e.g. healthcare, banking **Cost Effective** **Security** **Shared resources** **Collaboration and data sharing**

NIST Cloud Reference Architecture



NIST Cloud Reference Architecture

The NIST reference architecture, defines five major actors in terms of the roles and responsibilities:

- **Cloud service consumer** (CSC): A person or organization that maintains a business relationship with, and uses service from, cloud providers.
- **Cloud service provider** (CSP): A person, organization, or entity responsible for making a service available to interested parties.
- **Cloud auditor**: A party that can conduct independent assessment of cloud services, information system operations, performance, and security of the cloud implementation.
- **Cloud broker**: An entity that manages the use, performance, and delivery of cloud services, and negotiates relationships between CSPs and cloud consumers. A cloud broker can offer three areas of support:
  - **Service intermediation:** These are value-added services, such as identity management, performance reporting, and enhanced security.
  - **Service aggregation:** The broker combines multiple cloud services to meet consumer needs not specifically addressed by a single CSP, or to optimize performance or minimize cost.
  - **Service arbitrage:** The broker has the flexibility to choose services from multiple agencies. The cloud broker can use parameters to measure and select an agency with the best provider.
- **Cloud carrier:** An intermediary that provides connectivity and transport of cloud services from CSPs to cloud consumers.
  - Typically, a CSP will set up service level agreements (SLAs) with a cloud carrier to provide services

NIST Cloud Reference Architecture

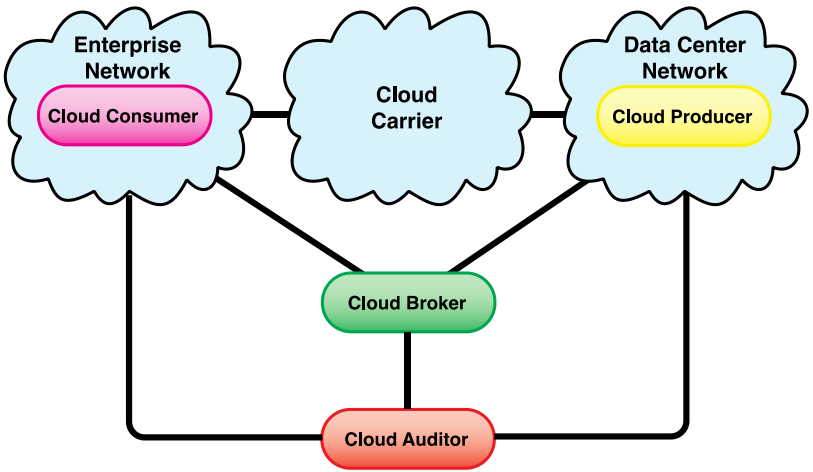


Figure 13.4 Interactions Between Actors in Cloud Computing

Security Issues for Cloud Computing

- Security is a major consideration when augmenting or replacing on-premises systems with cloud services
  - Allaying security concerns is frequently a prerequisite for further discussions about migrating part or all of an organization's computing architecture to the cloud
- Availability is another major concern
- Auditability of data must be ensured
- Businesses should perform due diligence on security threats both from outside and inside the cloud
  - Cloud users are responsible for application-level security
  - Cloud vendors are responsible for physical security and some software security
  - Security for intermediate layers of the software stack is shared between users and vendors
  - Cloud providers must guard against theft or denial-of-service attacks by their users and users need to be protected from one another
  - Businesses should consider the extent to which subscribers are protected against the provider, especially in the area of inadvertent data loss

Security Issues for Cloud Computing

https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-pandemic-eleven/

https://www.infosecurity-magazine.com/opinions/ransomcloud-ransomwares-cloud/

Data Protection in the Cloud

The threat of data compromise increases in the cloud, due to the number of, and interactions between, risks and challenges that are either unique to the cloud or more dangerous because of the architectural or operational characteristics of the cloud environment

Even with these precautions, corruption and other denial-of-service attacks remain a risk

Data must be secured while at rest, in transit, and in use, and access to the data must be controlled

For data at rest, the ideal security measure is for the client to encrypt the database and only store encrypted data in the cloud, with the CSP having no access to the encryption key

The client can employ encryption to protect data in transit, though this involves key management responsibilities for the CSP

The client can enforce access control techniques, but CSP is involved to some extent depending on the service model used

Cloud Security as a Service

- In the context of cloud computing, cloud security as a service, designated SecaaS, is a segment of the SaaS offering of a CSP
- The CSA defines SecaaS as the provision of security applications and services via the cloud either to cloud-based infrastructure and software, or from the cloud to the customers' on-premise systems
- The CSA has identified the following SecaaS categories of service:
  - Identity and access management
  - Data loss prevention
  - Web security
  - E-mail security
  - Security assessments
  - Intrusion management
  - Security information and event management
  - Encryption
  - Business continuity and disaster recovery
  - Network security

BYOK vs KYOK
(Bring Your Own Key vs Keep Your Own Key)

- Nearly half (48%) of all corporate data is stored in the cloud according to the 2019 Thales Global Cloud Security Study conducted by the Ponemon Institute.
  - Organizations admitted that on average, only about half (49%) of the data stored in the cloud is secured with encryption and
  - Only one-third (32%) believe protecting data in the cloud is their responsibility
- The question is “Who is responsible for cloud security, the cloud provider or organizations consuming cloud services?”
  - According to the shared security model, the answer is both.
- Bring Your Own Key (BYOK) allows enterprises to encrypt their data and retain control and management of their encryption keys. The HSM (Hardware Security Module) stays with CSP.
- Keep Your Own Key (KYOK) allows the enterprise to retain the physical ownership and logical control of customer managed encryption keys.

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Encryption for Security

Cryptography Basics

Cryptography is the science of secret, or hidden writing

- It has two main Components:

1. Encryption
   - Practice of hiding messages so that they can not be read by anyone other than the intended recipient
2. Authentication & Integrity
   - Ensuring that users of data/resources are the persons they claim to be and that a message has not been surreptitiously altered

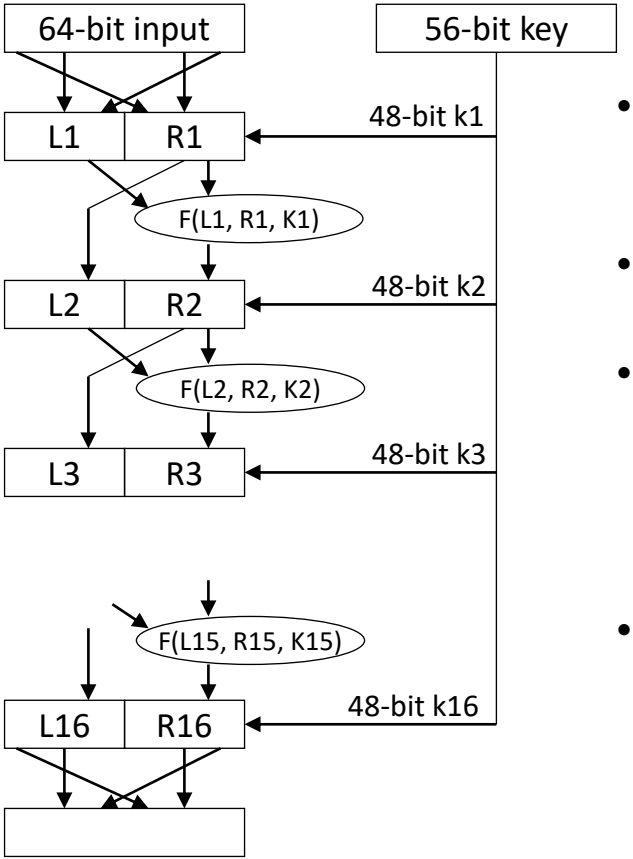
Properties of Trustworthy Encryption Systems

- Based on sound mathematics.
  - Good cryptographic algorithms are derived from solid principles.
- Analyzed by competent experts and found to be sound.
  - Since it is hard for the writer to envisage all possible attacks on the algorithm
- Stood the “test of time.”
  - Over time people continue to review both mathematical foundations of an algorithm and the way it builds upon those foundations.
  - The flaws in most algorithms are discovered after their release.

Data Encryption Standard (DES)

- DES completely scrambles block of data so that every bit of cipher text depends on every bit of data and ever bit of key
- DES is a block Cipher Algorithm
  - Encodes plaintext in 64 bit chunks
  - One parity bit for each of the 8 bytes thus it reduces to 56 bits
- It is (along with variants) widely used algorithm
  - Standard approved by US National Bureau of Standards for Commercial and non-classified US government use in 1993

Data Encryption Standard (DES)



- From original 56-bit key, 16 subkeys are generated
- DES runs in reverse to decrypt
- Cracking DES
  - 1997: 140 days
  - Today: Minute

- TripleDES uses DES 3 times in tandem
  - Output from 1 DES is input to next DES

Triple DES

- First standardized for use in financial applications in 1985
- Incorporated in DES FIPS PUB 46-3 standard of 1999
- Uses three keys & three DES executions:
  - $C = E(K_3, D(K_2, E(K_1, P)))$
- Decryption same with keys reversed
- Use of decryption in second stage gives compatibility with original DES users
- Effective 168-bit key length, slow, secure
- AES to eventually replace 3DES

Encryption Algorithms (symmetric)

Algorithm	Type	Key Size	Features
DES	Block Cipher	56 bits	Most Common, Not strong enough
TripleDES	Block Cipher	168 bits (112 effective)	Modification of DES, Adequate Security
Blowfish	Block Cipher	Variable (Up to 448 bits)	Excellent Security
AES	Block Cipher	Variable (128, 192, or 256 bits)	Replacement for DES, Excellent Security
RC4	Stream Cipher	Variable (40 or 128 bits)	Fast Stream Cipher, Used in most SSL implementations

Symmetric Encryption - Key Strength

- Strength of algorithm is determined by the size of the key
  - The longer the key the more difficult it is to crack
- Key length is expressed in bits
  - Typical key sizes vary between 48 bits and 448 bits
- Set of possible keys for a cipher is called key space
  - For 40-bit key there are 2^{40} possible keys
  - For 128-bit key there are 2^{128} possible keys
  - Each additional bit added to the key length doubles the security
- To crack the key the hacker has to use brute-force
  (i.e. try all the possible keys till a key that works is found)
  - Super Computer can crack a 56-bit key in 24 hours
  - It will take 2^{72} times longer to crack a 128-bit key
  (billions of years)

Symmetric Encryption - Limitations

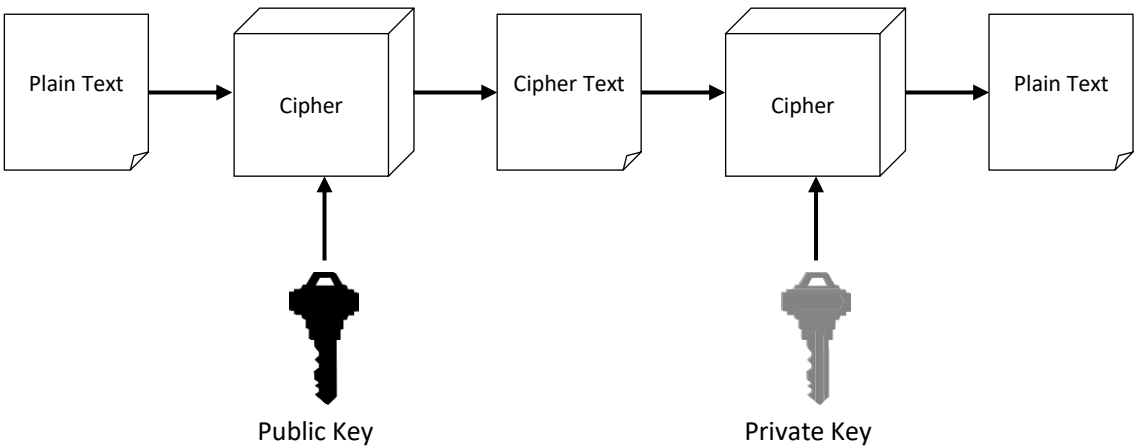
- Any exposure to the secret key compromises secrecy of ciphertext
- A key needs to be delivered to the recipient of the coded message for it to be deciphered
  - Potential for eavesdropping attack during transmission of key

Stream Ciphers

- Processes input elements continuously
- Key input to a pseudorandom bit generator
  - produces stream of random like numbers
  - unpredictable without knowing input key
  - XOR keystream output with plaintext bytes
- Are faster and use far less code
- Design considerations:
  - encryption sequence should have a large period
  - keystream should approximate random number properties
  - use a sufficiently long key (to guard against brute-force attacks)

Asymmetric Encryption

- Uses a pair of keys for encryption
  - Public key for encryption
  - Private key for decryption
- Messages encoded using public key can only be decoded by the private key
  - Secret transmission of key for decryption is not required
  - Every entity can generate a key pair and release its public key



Asymmetric Encryption Types

- Two most popular algorithms are RSA & El Gamal
  - RSA
    - Developed by Ron Rivest, Adi Shamir, Len Adelman
    - Both public and private key are interchangable
    - Variable Key Size (512, 1024, or 2048 bits)
    - Most popular public key algorithm
  - El Gamal
    - Developed by Taher ElGamal
    - Variable key size (512 or 1024 bits)
    - Less common than RSA, used in protocols like PGP

RSA

- Choose two large prime numbers p & q
- Compute n=pq and z=(p-1)(q-1)
- Choose number e, less than n, which has no common factor (other than 1) with z
- Find number d, such that ed – 1 is exactly divisible by z
- Keys are generated using n, d, e
  - Public key is (n,e)
  - Private key is (n, d)
- Encryption: $c = m^e \bmod n$
  - m is plain text
  - c is cipher text
- Decryption: $m = c^d \bmod n$
- Public key is shared and the private key is hidden

RSA Example

- P=5 & q=7
- n=5*7=35 and z=(4)*(6) = 24
- e = 5
- d = 29 , (29x5 –1) is exactly divisible by 24
- Keys generated are
  - Public key: (35,5)
  - Private key is (35, 29)
- Encrypt the message using ($c = m^e \bmod n$)
  - Assume that the alphabets are between 1 & 26

Numeric Representation	m^e	Cipher Text ($c = m^e \bmod n$)
12	248832	17
15	759375	15
22	5153632	22
5	3125	10

RSA Example

- Decrypt the message using ($m = c^d \bmod n$)
  - n = 35, d=29

Cipher Text (c)	c^d	($m = c^d \bmod n$)
17	481968572106750915091411825223072000	12
15	127834039488589391112327575683594 00	15
22	852643319086537701956194499721110000000	22
10	10000000000000000000000000000	5

Timing Attacks

- Paul Kocher, a cryptographic consultant, demonstrated that a snooper can determine a private key by keeping track of how long a computer takes to decipher messages
- Timing attacks are applicable not just to RSA, but also to other public-key cryptography systems
- This attack is alarming for two reasons:
  - It comes from a completely unexpected direction
  - It is a ciphertext-only attack

Timing Attack Countermeasures

innovate achieve lead

Constant exponentiation time

- Ensure that all exponentiations take the same amount of time before returning a result
- This is a simple fix but does degrade performance

Random delay

- Better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack
- If defenders do not add enough noise, attackers could still succeed by collecting additional measurements to compensate for the random delays

Blinding

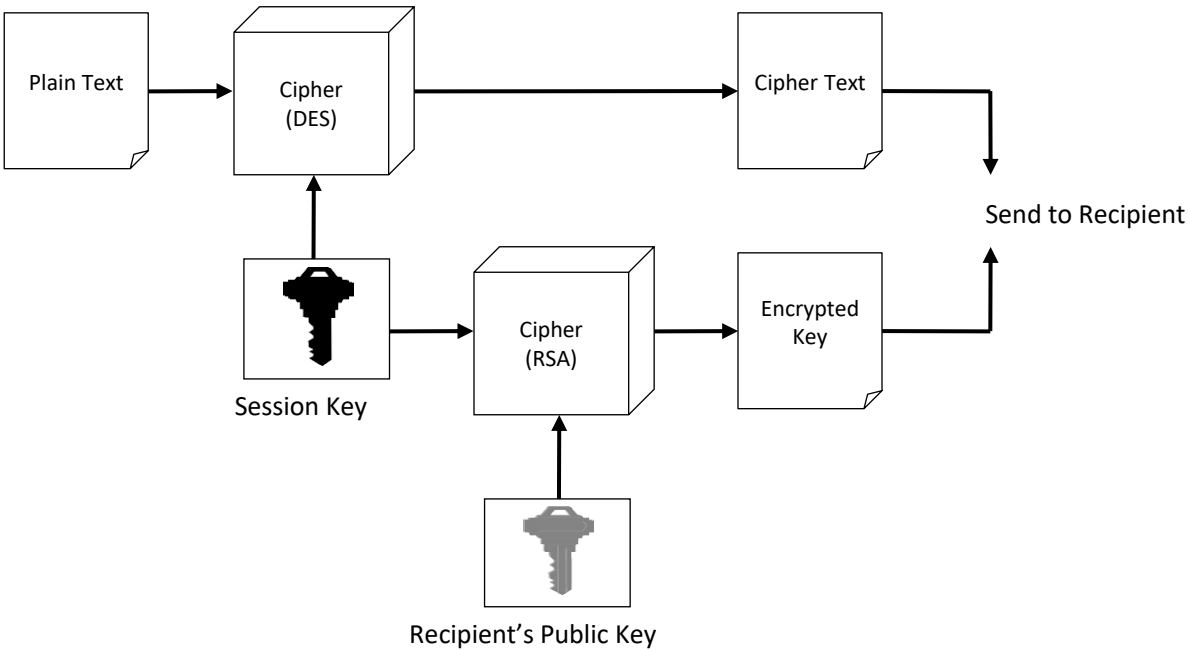
- Multiply the ciphertext by a random number before performing exponentiation
- This process prevents the attacker from knowing what ciphertext bits are being processed inside the computer and therefore prevents the bit-by-bit analysis essential to the timing attack

Asymmetric Encryption Weaknesses

- Efficiency is lower than Symmetric Algorithms
  - A 1024-bit asymmetric key is equivalent to 128-bit symmetric key
- Potential for man-in-the middle attack
- It is problematic to get the key pair generated for the encryption

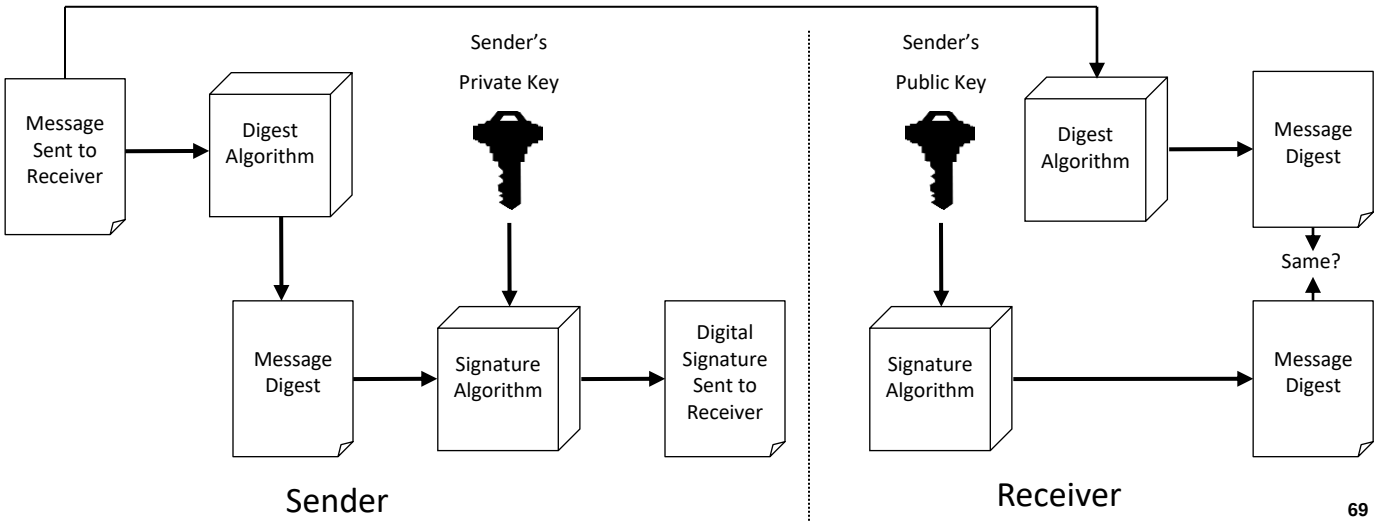
Session-Key Encryption (Asymmetric Encryption)

- Used to improve efficiency
  - Symmetric key is used for encrypting data
  - Asymmetric key is used for encrypting the symmetric key



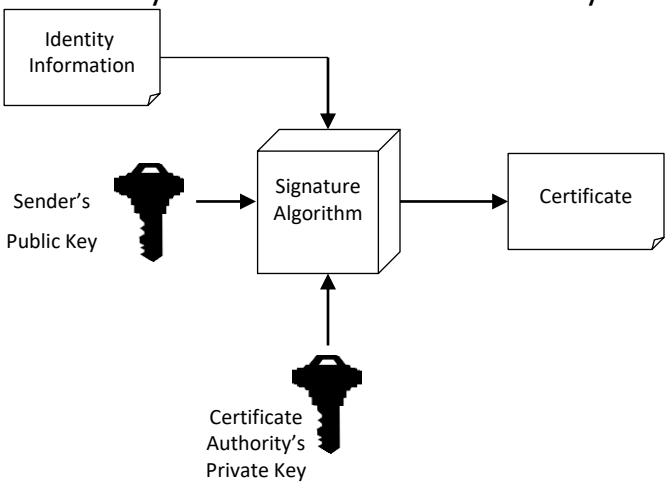
Authentication using Digital Signatures

- A digital signature is a data item which accompanies or is logically associated with a digitally encoded message.
- It has two goals
  - A guarantee of the source of the data
  - Proof that the data has not been tampered with



Authentication - Digital Certificates

- A digital certificate is a signed statement by a trusted party that another party's public key belongs to them.
  - This allows one certificate authority to be authorized by a different authority (root CA)
- Top level certificate must be self signed
- Any one can start a certificate authority
  - Name recognition is key to some one recognizing a certificate authority
  - Verisign is industry standard certificate authority



Computer Security: Principles and Practice by William Stallings, and Lawrie Brown Pearson, 2020.
www.cigital.com
sei.cmu.edu/cert
www.owasp.com

Thank You!