1. CONFINEMENT PRINCIPLE:

Designers of protection systems are usually preoccupied with the need to safeguard data from unauthorized access or modification, or programs from unauthorized execution. The customer will want to ensure that the service cannot access (i.e. read or modify) any of his data except those items to which he explicitly grants access. If he is cautious, he will only grant access to items which are needed as input or output for the service program.

Even when all unauthorized access has been prevented, there remain two ways in which the customer may be injured by the service: (1) it may not perform as advertised; or (2) it may leak, i.e. transmit to its owner the input data which the customer gives it.

Or if you have downloaded a program from the internet, you are not very certain whether you can trust that program, you are not very certain that if you run that program your system may got compromised. So, these programs are called untrusted programs (entire application untrusted or part of application untrusted).

Example: let us assume that you are running an application and you use a third-party module or a third-party library which is needed for that application to execute. You downloaded that third-party module or third-party library but may be a malicious code present in that library, so the possible solutions can be:

1) Air gapped systems

So the first is to use air gapped systems, so what we mean by this is that you could have a computer which is totally isolated from all the other computers present in your LAN, in your network, you can install the required operating system and all other software present in this special dedicated system and run your untrusted application. (hardware level). This is very expensive.

2) Virtual machines

Another option which is adopted quite often is cloud computing environments is to have virtual machines. So, with this solution you could create a virtual machine in your system, install a guest operating system in that virtual machine and then run the untrusted programs on that virtual machine. (isolate OS's on a single machine)

3) Containers

Containers or dockers are not as strong as Air gapped systems and Virtual machines but yet is able to provide some level of isolation, in which you could run your untrusted applications. (Process (treated as virtual machine): system call interposition (wrapper around each process)). Example: sandboxing.

So we see that all of these solutions would provide some degree of isolation in which untrusted can execute however all of these are very coarse grain solutions, so what we mean by this is that each of these solutions would require a dedicated entity to be present for example with air gap systems, you would require a complete dedicated computer just to run the untrusted program.

With virtual machines you still would require an entire virtual machine to be present in your system while similarly with containers you would require a container such as docker to be present in your system. (Threads: software fault isolation- isolating threads sharing same address space (thread local storage is used to create SFI) and interchange lots of information).

APPLICATION: e.g; Browser- based confinement

- Discretionary Access Control

- Mandatory Access control
- Role-based Access control

UNIX AND WINDOWS ACCESS CONTROL SUMMARY

1.1. Permissions

Unix-like systems implement three specific permissions that apply to each class:

- The read permission grants the ability to read a file. When set for a directory, this permission grants the ability to read the names of files in the directory, but not to find out any further information about them such as contents, file type, size, ownership, permissions.
- The write permission grants the ability to modify a file. When set for a directory, this permission grants the ability to modify entries in the directory, which includes creating files, deleting files, and renaming files. Note that this requires that execute is also set; without it, the write permission is meaningless for directories.
- The execute permission grants the ability to execute a file. This permission must be set for executable programs, in order to allow the operating system to run them. When set for a directory, the execute permission is interpreted as the search permission: it grants the ability to access file contents and meta-information if its name is known, but not list files inside the directory, unless read is set also.

The effect of setting the permissions on a directory, rather than a file, is "one of the most frequently misunderstood file permission issues".

When a permission is not set, the corresponding rights are denied. Unlike ACL-based systems, permissions on Unix-like systems are not inherited. Files created within a directory do not necessarily have the same permissions as that directory.

1.2. Changing permission behavior with setuid, setgid, and sticky bits

Unix-like systems typically employ three additional modes. These are actually attributing but are referred to as permissions or modes. These special modes are for a file or directory overall, not by a class, though in the symbolic notation (see below) the setuid bit is set in the triad for the user, the setgid bit is set in the triad for the group and the sticky bit is set in the triad for others.

The set user ID, setuid, or SUID mode. When a file with setuid is executed, the resulting process will assume the effective user ID given to the owner class. This enables users to be treated temporarily as root (or another user).

The set group ID, setgid, or SGID permission. When a file with setgid is executed, the resulting process will assume the group ID given to the group class. When setgid is applied to a directory, new files and directories created under that directory will inherit their group from that directory. (Default behaviour is to use the primary group of the effective user when setting the group of new files and directories, except on BSD-derived systems which behave as though the setgid bit is always set on all directories (See Setuid).)

The sticky mode. (Also known as the Text mode.) The classical behavior of the sticky bit on executable files has been to encourage the kernel to retain the resulting process image in memory beyond termination; however, such use of the sticky bit is now restricted to only a minority of Unix-like operating systems (HP-UX and UnixWare). On a directory, the sticky permission prevents users from renaming, moving or deleting contained files owned by users other than themselves, even if they have written permission to the directory. Only the directory owner and superuser are exempt from this.

1.3. Detour Unix user IDs process IDs and privileges

A system which is based on authentication or authorization uses the notion of some principals (IDentification) and subjects.

In UNIX the **subject** is a program (a program or application executing on behalf of some principal) and a **principal** could be a password, user name or any unique token or any device which allows you to get into the system to authenticate yourself and if you go through the authentication and if you have authorization to do certain things, then you are allowed to do those things.

An **object** is anything on which a subject can perform operations (mediated by rights of subjects). Usually objects are passive things, such as file, directory or memory segment, sockets or ports etc. But subjects can also be objects sometimes, with operations e.g. a process is normally a subject which access those objects but some time we kill a process or suspend or resume a process, so in that case subject is an object.

Process user ID model in modern UNIX systems:

- Each process has three user IDs
  - Real user (ruid): owner of the process
  - Effective user ID (euid)(mostly same as ruid): used in most access (invoked to create a process)
  - Saved user ID (suid): used to remember if you change the euid temporarily, (to restore euid back)

Each user account has a unique UID:

- The UID 0 means the super user (system admin).
  - A user account belongs to multiple groups. Subjects are processes–associated with uid/gid pairs, e.g., (euid, egid), (ruid, rgid), (suid, sgid).

Almost all objects are modeled as files:

- Files are arranged in a hierarchy
- Files exist in directories
- Directories are also one kind of files
  - Each object has–owner–group–12 permission bits
  - rwx for owner, rwx for group, and rwx for others
  - suid, sgid, sticky

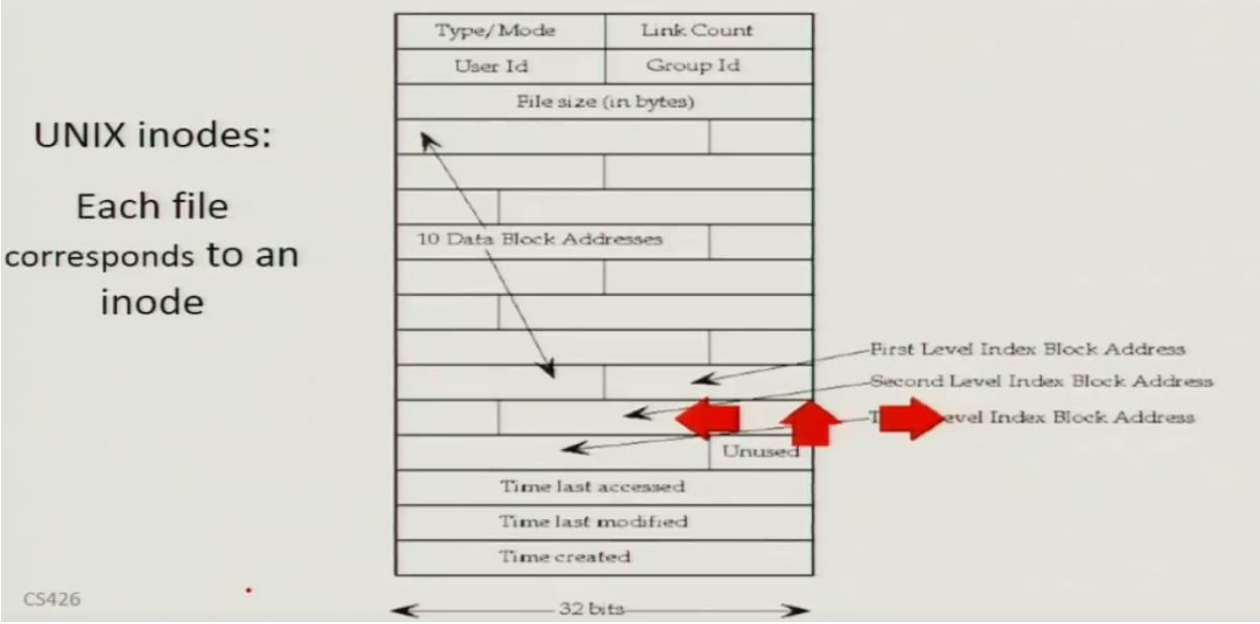
The suid, sgid, sticky bits

	suid	sgid	sticky bit
non-executable files	no effect	affect locking (unimportant for us)	not used anymore
executable files	change euid when executing the file	change egid when executing the file	not used anymore
directories	no effect	new files inherit group of the directory	only the owner of a file can delete

Permission bits on directories: (chroot)

- Read bit allows one to show file names in a directory

- The execution bit controls traversing a directory
  - Does a lookup, allows one to find inode # from file name
  - chdir to a directory requires execution
- Write and execution control creating/deleting files in the directory
  - Deleting a file under a directory requires no permission on the file
- Accessing a file identified by a path name requires execution to all directories along the path
- No one can implement negative authorization in Unix, e.g. only members of a particular group are not allowed to access a file.
- Only owner or root user can change the permission bits.
- Superuser or privileged can change the owner.
- Operations such as shutting down the system, mounting a new file system, listening on a low port are traditionally reserved for the root user.



Jailkit:

Jailkit is a set of utilities to limit user accounts to specific files using chroot() and or specific commands. Setting up a chroot shell, a shell limited to some specific command, or a daemon inside a chroot jail is a lot easier and can be automated using these utilities.

Jailkit is a specialized tool that is developed with a focus on security. It will abort in a secure way if the configuration, the system setup or the environment is not 100% secure, and it will send useful log messages that explain what is wrong to syslog.

Jailkit is known to be used in network security appliances from several leading IT security firms, internet servers from several large enterprise organizations, internet servers from internet service providers, as well as many smaller companies and private users that need to secure cvs, sftp, shell or daemon processes.

Process user ID model in modern Unix systems:

a) When a process is created by fork.

A process can be created by ***fork.*** Fork is a system call used for creating a new process, which is called child process, which runs concurrently with the process that makes the fork() call (parent process). After a new child process is created, both processes will execute the next instruction following the fork() system call. A child process uses the same pc(program counter), same CPU registers, same open files which use in the parent process.
It takes no parameters and returns an integer value. Below are different values returned by fork().
Negative Value: creation of a child process was unsuccessful.
Zero: Returned to the newly created child process.
Positive value: Returned to parent or caller. The value contains process ID of newly created child process.

b) When a process executes a file by exec

It keeps its three user IDs unless the set-user-ID bit of the file is set, in which case the effective uid and saved uid are assigned the user ID of the owner of the file.

c) A process may change the user IDS via system call

2. SYSTEM CALL INTERPOSITION:

In computing, a system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. This may include hardware-related services (for example, accessing a hard disk drive), creation and execution of new processes, and communication with integral kernel services such as process scheduling. System calls provide an essential interface between a process and the operating system.
In most systems, system calls can only be made from user space processes, while in some systems, OS/360 and successors for example, privileged system code also issues system calls.
System call interposition is a powerful method for regulating and monitoring program behavior and block unauthorized calls. A wide variety of security tools have been developed which use this technique. However, traditional system call interposition techniques are vulnerable to kernel attacks and have some limitations on effectiveness and transparency. Analyze the binary to determine if it truly is malware or just run the code without letting it affect your system.
With System Call Interposition you can rewrite operating system calls but unless you use interposition to create a complete container you need to be able to identify which system calls are dangerous before allowing them through. This may not be obvious. Even using Interposition to create a container may fail if the code exploits a subtle kernel bug. System call interposition isolates a process in a single operating system.

Complications:

- If app forks (create child process), monitor must also fork
  - Forked monitor monitors forked app
- If monitor crashes, app must be killed
- Monitor must maintain all OS state associated with app
  - Current working directory (**cwd**), **UID, EUID, GID**
  - When app does “cd path” monitor must update its CWD
    - Otherwise: relative path requests interpreted incorrectly

ptrace:

ptrace is a system call found in Unix and several Unix-like operating systems. By using ptrace (the name is an abbreviation of "process trace") one process can control another, enabling the controller

to inspect and manipulate the internal state of its target. ptrace is used by debuggers and other code-analysis tools, mostly as aids to software development.

Systrace:

Recording device activity over a short period of time is known as system tracing. System tracing produces a trace file that can be used to generate a system report. This report helps you identify how best to improve your app or game's performance.

The Android platform provides several different options for capturing traces:

- System Tracing app
- Systrace command-line tool
- Perfetto command-line tool

The System Tracing app is an Android tool that saves device activity to a trace file. On a device running Android 10 (API level 29) or later, trace files are saved in Perfetto format (see below). On a device running an earlier version of Android, trace files are saved in the Systrace format.

Systrace is a legacy platform-provided command-line tool that records device activity over a short period of time in a compressed text file. The tool produces a report that combines data from the Android kernel, such as the CPU scheduler, disk activity, and app threads.

Perfetto is the new platform-wide tracing tool introduced in Android 10. It is a more general and sophisticated open-source tracing project for Android, Linux, and Chrome. It offers a superset of data sources compared to Systrace and allows you to record arbitrarily long traces in a protobuf-encoded binary stream. You can open these traces in the Perfetto UI.

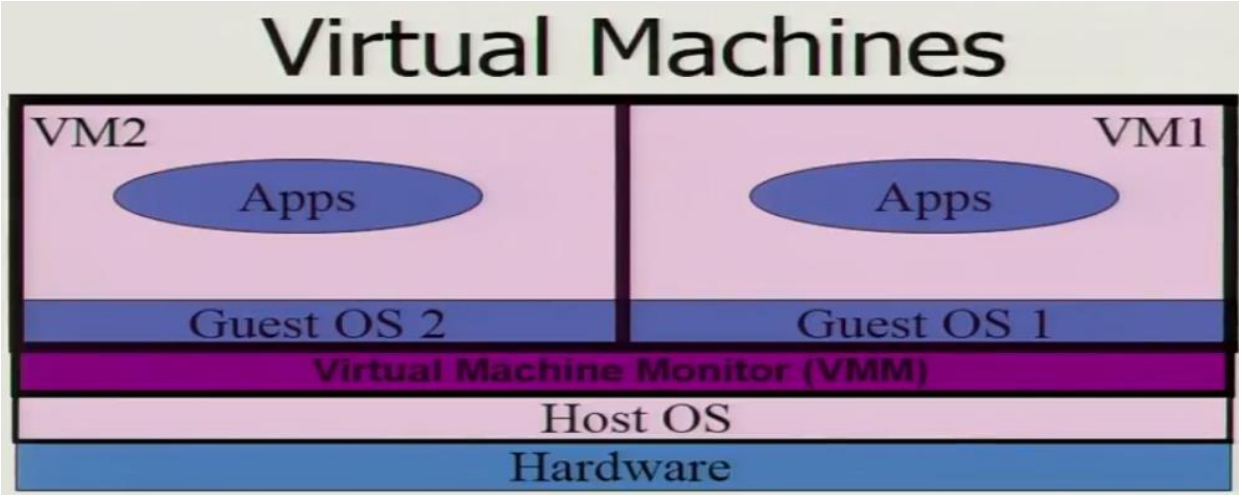
Ostia:

Ostia implements a delegating sandbox architecture. As described in the previous section, it is composed of three primary components.

- **Kernel module:** A small kernel module enforces Ostia's static policy of denying any call that provides direct access to sensitive system resources. This is done simply by preventing a fixed set of system calls from executing. (As a belt-and-suspenders measure to ensure that access to the file system is denied, sandboxed processes are chrooted to an empty directory if Ostia is run as root.) It also provides a trampoline mechanism that redirects delegated calls back into the emulation library as discussed below. Finally, it implements an fexecve call because execve cannot be delegated to another process, for obvious reasons.
- **Emulation library:** Ostia uses a callback mechanism in the kernel module to redirect system calls. (Ostia evolved from an earlier delegating system we built that relied on shared library replacement to redirect system calls. We note this to emphasize that system call redirection—or virtualization—can be done multiple ways.) When a sensitive system call reaches the kernel entry point, it calls back into the handler in a special emulation library in the program's address space. The emulation library transforms the system call into a request to the agent. To speed up subsequent system calls from the same point in the code, the handler also examines the machine instructions that made the call and, if they take the expected form, patches them inplace to jump directly to the handler, avoiding subsequent round trips through the kernel.

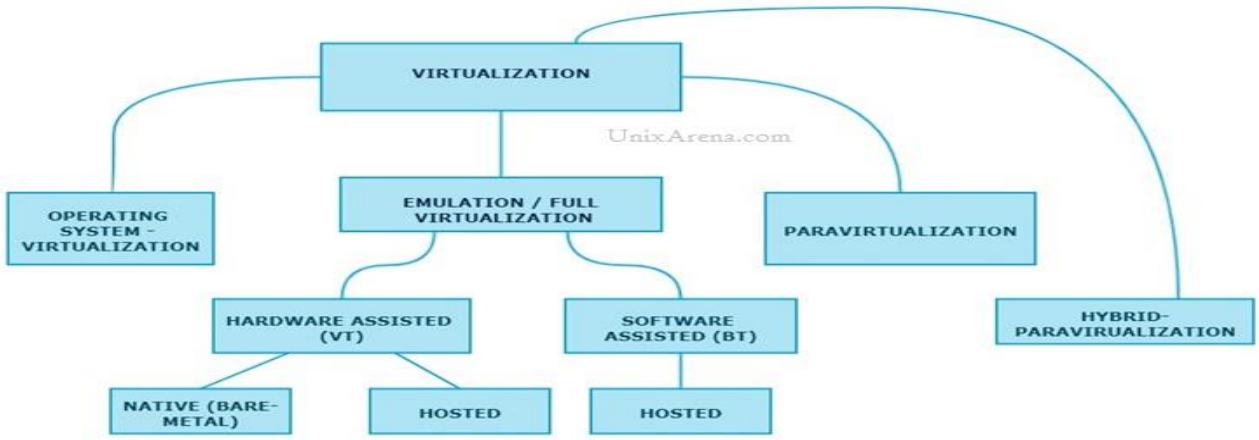
3. VM BASED ISOLATION

VMs would present a smaller attack surface than OS kernels and since modern CPUs provide hardware support should protect even against undocumented opcodes.
Example: **NetTop** is an NSA project to run Multiple Single-Level systems (Multiple single-level or multi-security level (MSL) is a means to separate different levels of data by using separate computers or virtual machines for each level. It aims to give some of the benefits of multilevel security without needing special changes to the OS or applications, but at the cost of needing extra hardware.) with a Security-Enhanced Linux host running VMware with Windows as a guest operating system.



Virtualization is nothing but abstracting operating system, application, storage or network away from the true underlying hardware or software. It creates the illusion of physical hardware to achieve the goal of operating system isolation. In last decade, data centers were occupied by a large number of physical servers, network switches, storage devices. It consumed a lot of power and manpower to maintain the data centers. In that period, there were many companies were researching about the hardware emulation/simulation like QEMU, virtual PC etc. It's very hard to list all the virtualization types here. So, I have just listed down only the server virtualization types.

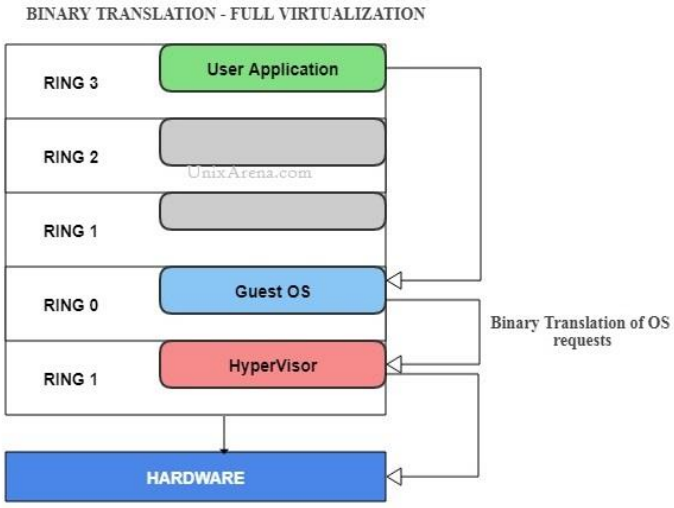
- Full Virtualization (Hardware Assisted/ Binary Translation)
- Paravirtualization
- Hybrid Virtualization
- OS level Virtualization



FULL VIRTUALIZATION:

Virtual machine simulates hardware to allow an unmodified guest OS to be run in isolation. There is two type of Full virtualizations in the enterprise market. On both full virtualization types, guest operating system's source information will not be modified.

- Software assisted full virtualization
- Hardware-assisted full virtualization



1) **Software Assisted – Full Virtualization (BT – Binary Translation)**

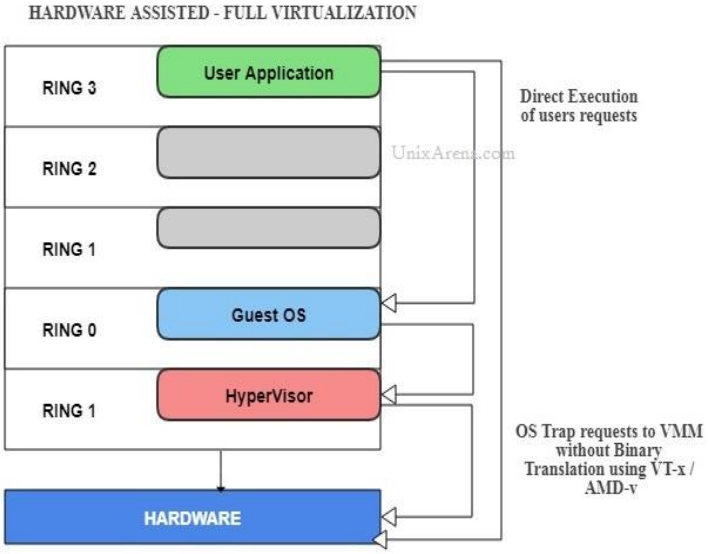
   It completely relies on binary translation to trap and virtualize the execution of sensitive, non-virtualizable instructions sets. It emulates the hardware using the software instruction sets. Due to binary translation, it often criticized for performance issue. Here is the list of software which will fall under software assisted (BT).

- VMware workstation (32Bit guests)
- Virtual PC
- VirtualBox (32-bit guests)
- VMware Server

2) **Hardware-Assisted –Full Virtualization (VT)**

Hardware-assisted full virtualization eliminates the binary translation and it directly interrupts with hardware using the virtualization technology which has been integrated on X86 processors since 2005 (Intel VT-x and AMD-V). Guest OS's instructions might allow a virtual context execute privileged instructions directly on the processor, even though it is virtualized.

Here is the list of enterprise software which supports hardware-assisted – Full virtualization which falls under hypervisor type 1 (Bare metal)



- VMware ESXi /ESX
- KVM
- Hyper-V
- Xen

PARAVIRTUALIZATION:

Paravirtualization works differently from the full virtualization. It doesn't need to simulate the hardware for the virtual machines. The hypervisor is installed on a physical server (host) and a guest OS is installed into the environment. Virtual guests aware that it has been virtualized, unlike the

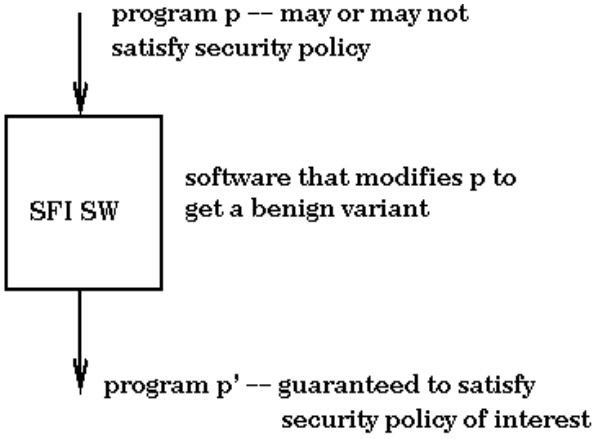
full virtualization (where the guest doesn't know that it has been virtualized) to take advantage of the functions. In this virtualization method, guest source codes will be modified with sensitive information to communicate with the host. Guest Operating systems require extensions to make API calls to the hypervisor. In full virtualization, guests will issue a hardware calls but in paravirtualization, guests will directly communicate with the host (hypervisor) using the drivers.

HYBRID VIRTUALIZATION: (HARDWARE VIRTUALIZED WITH PV DRIVERS):
In Hardware assisted full virtualization, Guest operating systems are unmodified and it involves many VM traps and thus high CPU overheads which limit the scalability. Paravirtualization is a complex method where guest kernel needs to be modified to inject the API. By considering these issues, engineers have come with hybrid paravirtualization. It's a combination of both Full & Paravirtualization. The virtual machine uses paravirtualization for specific hardware drivers (where there is a bottleneck with full virtualization, especially with I/O & memory intense workloads), and the host uses full virtualization for other features.

OS LEVEL VIRTUALIZATION:
Operating system-level virtualization is widely used. It also knowns "containerization". Host Operating system kernel allows multiple user spaces aka instance. In OS-level virtualization, unlike other virtualization technologies, there will be very little or no overhead since its uses the host operating system kernel for execution. Oracle Solaris zone is one of the famous containers in the enterprise market. Here is the list of other containers.

- Linux LCX
- Docker
- AIX WPAR

4. SOFTWARE FAULT ISOLATION



to get programs to behave in a manner consistent with a given security policy is by "brainwashing." That is, modify the programs so that they behave only in safe ways. This is embodied by a recent approach to security known as software-based fault isolation (SFI).
So far, the environment has been responsible for policy enforcement, where the environment is either the OS/kernel or the hardware. Hardware methods include addressing mechanisms (e.g. virtual memory); OS methods include having two modes (where the supervisor mode has access to everything). The new approach we discuss today is to construct a piece of software that transforms a given program p into a program p', where p' is guaranteed to satisfy a security policy of interest.
This SFI SW transformation could be any number of things. It could be a piece of the compiler or of the loader. It could also involve a separate pass over machine language code before execution commences. The point is that we are modifying the program before it is executed.

When would the SFI approach be?

- If we are trying to run programs on a machine that lacks suitable protection hardware, then with this method we can achieve the functionality that the hardware doesn't support.
- Another application is when hardware doesn't support the security policy of interest. This is likely to become increasingly important, especially in situations where security policies

relate to higher-level abstractions (e.g. paragraphs in a document being assigned security ratings). If the hardware or the OS software doesn't know about a particular abstraction, we can add a program as described above to handle it.

- The original motivation for this approach was performance. We can use SFI to achieve memory protection at a low cost. Most operating systems/HW provide security by protecting disjoint address spaces. However, programs operating in these isolated address spaces need to communicate with each other. An inter-process communication (IPC) provided by the operating system is invariably 10-100 times as costly as a procedure call. An IPC needs to trap, copy arguments, save/restore arguments, change the address space and then repeat all of the above on the way out. We would like to avoid this costly mechanism and can replace it with SFI.
- Another use might be for programs that support plug-ins. We would like to protect the main program (e.g. a browser) from buggy plug-ins. This is a form of "sub-address space" protection, since the browser and the plug-in execute in the same address space.
- Finally, this approach is useful for programs comprising many "objects." Such programs would profit from having sub-address space level protection. If one object "goes bad," it would be nice if that behavior didn't trash everything else. However, an IPC-like operation that would accompany operating system address space protection is likely to be much too expensive for use with every method invocation.

With SFI you might run into a problem where the suspected malware uses an undocumented opcode and the SFI therefore refuses to run the code as it cannot analyze it.

5. ROOTKITS

A rootkit is a malicious software that allows an unauthorized user to have privileged access to a computer and to restricted areas of its software. A rootkit may contain a number of malicious tools such as keyloggers, banking credential stealers, password stealers, antivirus disablers, and bots for DDoS attacks. This software remains hidden in the computer and allow the attacker remote access to the computer.

The term rootkit is derived from the combination of two words – "root" and "kit". "Root" refers to the administrator account in Unix and Linux operating systems, which is an all-powerful account with full privileges and unrestricted access. It is equivalent to the administrator account in Windows systems. The term "kit" refers to the programs that allow a threat actor to obtain unauthorized root/admin-level access to the computer and restricted areas. The rootkit enables the threat actor to perform all these actions surreptitiously without the user's consent or knowledge.

How the Attacker Installs Rootkits

The threat actor tries to obtain root/administrator access by exploiting known vulnerabilities, or by stealing administrator privilege credentials. Cyber criminals employ social engineering techniques to obtain credentials. Root access allows installation of rootkits or any other malware. Installation of the rootkit enables the threat actor to access the computer from remote to install other malware, steal data, observe activities and even control the computer. Rootkits are sophisticated malware, and most antivirus solutions and antimalware solutions do not detect rootkits. Rootkits are also able to hide their intrusion, and hence once they are in, they are practically undetectable.

Since rootkits have complete control over the system, they can modify software and the cyber security solutions such as the antivirus that could detect rootkits. As even the detection solutions are modified, it is difficult to detect and remove rootkits.

Threat actors use rootkits for many purposes:

- **Stealth capabilities:** Modern rootkits add stealth capabilities to malicious software payloads (such as keyloggers and viruses) to make them undetectable.
- **Backdoor access:** Rootkits permit unauthorized access through backdoor malware. The rootkit subverts the login mechanism to also accept a secret login access for the attacker. Standard authentication and authorization mechanisms are bypassed to provide admin privileges to the attacker.
- **DDoS attacks:** Rootkits allow the compromised computer to be used as a bot for distributed-denial-of-service attacks. The attack would now be traced to the compromised computer and not to the attacker's system. These bots are also called as zombie computers and are used as part of bot networks to launch the DDoS attacks, and other malicious activities such as click fraud and spam email distribution.

The functionality of rootkits is also used for good causes, such as:

- in a honeypot to detect attacks
- to enhance emulation software
- to enhance security software – it enables the software to secure itself from malicious actions
- digital rights management enforcement
- device anti-theft protection - BIOS-based rootkit software enables monitoring, disabling and wiping of data on mobile devices when they get lost or stolen

There are five types of rootkits:

- User-mode rootkits
- kernel-mode rootkits
- bootkits
- hypervisor rootkits
- firmware rootkits.

How to Detect Rootkits?

A behavioral-based approach proves to be effective in detecting rootkits. Cyber security solutions such as the Comodo Advanced Endpoint Protection (AEP) utilize their Host Intrusion Prevention Systems to effectively detect and remove rootkits in computer systems.

6. ERROR 404 DIGITAL HACKING IN INDIA:

Julian Assange, the founder of WikiLeaks, came out with the list of tools and exploits which were used by the CIA, which is the central intelligence agency of the US government. They were used for offensively hacking into targets. The CIA is not just one example. Some report specifically cites evidence linking a cyber threat called APT1 to china's second bureau of the people's liberation army. A devastating hack, crippling one of the most powerful entertainment studio in the world.

All of these are examples of how the governments around the world are pro-actively investing and engaging more on the cyber front. Like the Army, who is serving for the country, there is another army or kind of Cyber Army. It is said, that the third world war will be a cyber war.

- **Critical Infrastructure**:

  It is the body of systems, networks and assets that are so essential that their continued operation is required to ensure the security of a nation, the economy and the public's health.

In 2012, India had one of world's largest black outs. The problem in the northern grids affected 20 of India's 28 states. It started in one line that got overloaded and once it got too overloaded, it got disconnected. Then all the power flowed to the other lines and then these lines heated up and got disconnected and this propagated. Then at some point, the northern, western and eastern grids tripped each other and black-out happened. But it can be done through a cyber-attack. All you have to do is hack into the relays, and relays can be used to disconnect the lines. So, instead of reaching a thermal limit, you are going to disconnect the lines through hacking. Ukraine and Israel's power grids, New York, hydro-power plant and Iran's nuclear power plant hit by a big hack attack/worm. It's being called one of the worst cyber attacks ever.

Bangladesh based group hacked into nearly 20,000 Indian websites including Indian Border Security Force. The first virus that could crash power grids or destroy oil pipelines is available online for anyone to download and tinker with. There is no way of knowing who will use it or what they will use it for. If there was to be an attack on the critical infrastructure of India, it would mean chaos. There would be no electricity, no water supply, no phone network, no satellite network and no cash or banking facilities. An attack on any of its critical infrastructures can cripple a country. Israel is one of those classic examples where the PM says this publicly that they have a fourth division in their defence system which is the cyber warfare division. Where we not only defend our borders we also go and pro-actively offend because offence is defence in a borderless cyber world. We can say that state sponsored hacks are only going to go up in the near future, without a doubt. The reason is that the governments have realized that data is the new oil. You will be able to take decisions both pro-actively and reactively, in case there is a situation, based on a more informed set of data entries and points, which would be far more accurate than just diplomatic talks or reading articles.

Biggest unawareness:

- The tremendous lack of awareness in the government, they don't believe in insider threats. And that means is that they will be caught unaware when an insider does something.
- Other problem is our hardware. Because all our hardware comes from abroad. And there is a concern that while manufacturing these chips someone can add some extra circuitry tool, which can be triggered at a certain time and some harm can happen.

India saw its biggest data breach when the SBI debit card breach happened. When this happened, banks were initially in a state of denial. But subsequently they had to own up to the biggest cyber security breach that took place in Indian history.

The ATMs are not manufactured by the banks. There are popular OEMs which manufacture the physical ATM's and then you put in a Windows system in that. It can be windows XP, 7 or 8. And on top of that you load up a software. It's basically a software that lets you select the type of account, enter the amount, etc. what was observed was that there were multiple transactions happening in China and close to Rs. 1.3 crore was withdrawn using certain VISA and MASTERCARD cards which were specifically used in a few selected ATMs. These ATM machines are actually connected by a network, to some sort of a control center. It means, that we are completely in the hands of others, where they can do various things.

The debate in about the national security versus the privacy. Here, two people have to play the main role: the citizens and the government. But it's a fundamental right of any person to chose if he wants to be anonymous on the internet or not which is known as anonymity.

7. ACCESS CONTROL CONCEPTS:

Your Security Needs an Access Control when it comes to protecting your home or business, as well as the building's occupants, access control is one of the best ways for you to achieve peace of mind. Understanding cornerstone access control concepts, including confidentiality, integrity, and availability (and their mirror opposites: disclosure, alteration, and destruction), and subjects and objects are a critical foundation to understanding access control. Outlined below are overviews of the three basic types of access control systems that are available to your company so you can see which are best suited for your day-to-day operations.

The Three Types of Access Control Systems:

In brief, access control is used to identify an individual who does a specific job, authenticate them, and then proceed to give that individual only the key to the door or workstation that they need access to and nothing more. Access control systems come in three variations: Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC).

7.1. Discretionary Access Control (DAC)

Discretionary Access Control is a type of access control system that holds the business owner responsible for deciding which people are allowed in a specific location, physically or digitally. DAC is the least restrictive compared to the other systems, as it essentially allows an individual complete control over any objects they own, as well as the programs associated with those objects. The drawback to Discretionary Access Control is the fact that it gives the end user complete control to set security level settings for other users and the permissions given to the end user are inherited into other programs they use which could potentially lead to malware being executed without the end user being aware of it.

7.2. Mandatory Access Control (MAC)

Mandatory Access Control is more commonly utilized in organizations that require an elevated emphasis on the confidentiality and classification of data (i.e. military institutions). MAC doesn't permit owners to have a say in the entities having access in a unit or facility, instead, only the owner and custodian have the management of the access controls. MAC will typically classify all end users and provide them with labels which permit them to gain access through security with established security guidelines.

7.3. Role-Based Access Control (RBAC)

Also known as Rule-Based Access Control, RBAC is the most demanded in regard to access control systems. Not only is it in high demand among households, RBAC has also become highly sought-after in the business world. In RBAC systems, access is assigned by the system administrator and is stringently based on the subject's role within the household or organization and most privileges are based on the limitations defined by their job responsibilities. So, rather than assigning an individual as a security manager, the security manager position already has access control permissions assigned to it. RBAC makes life much easier because rather than assigning multiple individuals' particular access, the system administrator only has to assign access to specific job titles.

7.4. Centralized access control

Centralized access control is concentrated at one logical point for a system or organization. Instead of using local access control databases, systems authenticate via third-party authentication servers.

Centralized access control can be used to provide Single Sign-On (SSO), where a subject may authenticate once and then access multiple systems.

7.5. Decentralized access control

Decentralized access control allows the IT administration to be closer to the mission and operations of the organization. With it, an organization spans multiple locations, and the local sites support and maintain independent systems, access control databases, and data. Decentralized access control is also called distributed access control.

7.6. Access control protocols and frameworks

Both centralized and decentralized models may support remote users authenticating to local systems. A number of protocols and frameworks may be used to support this need, including RADIUS, Diameter, TACACS/TACACS+, PAP, and CHAP.

8. OTHER ISSUES IN ACCESS CONTROL:

Discretionary Access Control (DAC) gives subjects full control of objects they have created or been given access to, including sharing the objects with other subjects. Subjects are empowered and control their data. Standard UNIX and Windows operating systems use DAC for file systems: subjects can grant other subjects access to their files, change their attributes, alter them, or delete them.

If a subject makes a mistake, such as attaching the wrong file to an email sent to a public mailing list, loss of confidentiality can result. Mistakes and malicious acts can also lead to a loss of integrity or availability of data.

An access-control list (ACL), with respect to a computer file system, is a list of permissions attached to an object. An ACL specifies which users or system processes are granted access to objects, as well as what operations are allowed on given objects. Each entry in a typical ACL specifies a subject and an operation. For instance, if a file object has an ACL that contains (Alice: read, write; Bob: read), this would give Alice permission to read and write the file and Bob to only read it.

The Portable Operating System Interface (POSIX) is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems. POSIX defines the application programming interface (API), along with command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems. POSIX 1003.1e/1003.2c working group made an effort to standardize ACLs.

Many kinds of operating systems implement ACLs, or have a historical implementation:

1) Filesystem ACLs:

A filesystem ACL is a data structure (usually a table) containing entries that specify individual user or group rights to specific system objects such as programs, processes, or files. These entries are known as access-control entries (ACEs) in the Microsoft Windows NT,[2] OpenVMS, Unix-like, and Mac OS X operating systems. Each accessible object contains an identifier to its ACL. The privileges or permissions determine specific access rights, such as whether a user can read from, write to, or execute an object. In some implementations, an ACE can control whether or not a user, or group of users, may alter the ACL on an object.

2) Networking ACLs:

On some types of proprietary computer-hardware (in particular routers and switches), an access-control list provides rules that are applied to port numbers or IP addresses that are available on a host or other layer 3, each with a list of hosts and/or networks permitted to use the service. Although it is additionally possible to configure access-control lists based on network domain names, this is a questionable idea because individual TCP, UDP, and ICMP headers do not contain domain names. Consequently, the device enforcing the access-control list must separately resolve names to numeric addresses. This presents an additional attack surface for an attacker who is seeking to compromise security of the system which the access-control list is protecting. Both individual servers as well as routers can have network ACLs. Access-control lists can generally be configured to control both inbound and outbound traffic, and in this context, they are similar to firewalls. Like firewalls, ACLs could be subject to security regulations and standards such as PCI DSS.

3) SQL implementations:

ACL algorithms have been ported to SQL and to relational database systems. Many "modern" (2000s and 2010s) SQL-based systems, like enterprise resource planning and content management systems, have used ACL models in their administration modules.

The main alternative to the ACL model is the role-based access-control (RBAC) model. In modern SQL implementations, ACLs also manage groups and inheritance in a hierarchy of groups. So "modern ACLs" can express all that RBAC express, and are notably powerful (compared to "old ACLs") in their ability to express access-control policy in terms of the way in which administrators view organizations.

9. INTRODUCTION TO BROWSER ISOLATION:

Browser isolation is a cybersecurity model used to physically isolate an internet users web browser and their browsing activity away from the local machine and network, it is the underlying model and technology that supports a remote browsing platform. Browser isolation technologies are one of the most effective ways that an enterprise can reduce web-based attacks.

Browser isolation was an invention borne out of necessity, our current security tools (anti-virus, firewall, intrusion detection and prevention) are failing to protect us from malware, ransomware and browser based cyber-attacks. Browser based attacks are increasing in frequency, through their browsers as they use the internet normally.

a) Web Isolation:

- Provides a safe visual stream of the original web site in the user's browser.
- End users browse the site as if the site was running directly in the browser.
- Preserves original browsing experience with full usability and control.
- Only safely rendered information arrives at the user’s browser.

b) Protect Endpoints from Attack

As stated above, Web Isolation executes web sessions away from endpoints, sending only safely rendering information to users' browsers, thereby preventing malware from reaching your network and devices. Web pages are rendered and isolated as graphics for display on the end user’s browser.

- Isolating websites, emails, and documents so that no malicious content can ever reach the end-point.
- Preventing malware and fraud.

- Protecting against drive-by infection, malvertising, and ransomware.
- Blocking malware Command & Control (C&C), and exfiltration communication

Web Isolation executes web sessions away from endpoints.

We know that most cyberattacks begin with the browser, we understand that the browser is the window through which cyber-criminals climb into your local machine and networks, but our browsers are so useful to our everyday work and lives that we leave the window open. There is a better way, you can physically isolate your browsers and the associated cyber risks away from your local computer and networks by putting an air-gap between you and the internet. We call this model browser isolation cybersecurity and the WEBGAP remote browser platform is underpinned by this browser isolation cybersecurity model.

Browser isolation solutions provide their users with 'disposable' (non-persistent) browsers, once a browsing session is closed the entire browser environment is destroyed and its resources returned to the pool for use by other users. Any malicious code encountered during that session is thus prevented from reaching the endpoint or persisting within the network, regardless of whether or not any threat was detected. In this way, browser isolation proactively combats both known, unknown and zero-day threats, complementing other security measures and contributing to a defense-in-depth, layered approach to web security.

A quick and easy way to physically isolate your user's browsers is to give each of them their own cloud hosted remote browser, a fully functional browser hosted on a cloud server and delivered to the user over the internet. Users logon to their hosted remote browsers and browse the internet through them rather than the browser on their local machines, physically isolating themselves from web based cyber-attacks. This is a highly effective model which allows you to quickly shut down the most common infiltration points on your networks, but without worrying about setting up and configuring the associated server infrastructure.

10. WEB SECURITY LANDSCAPE:

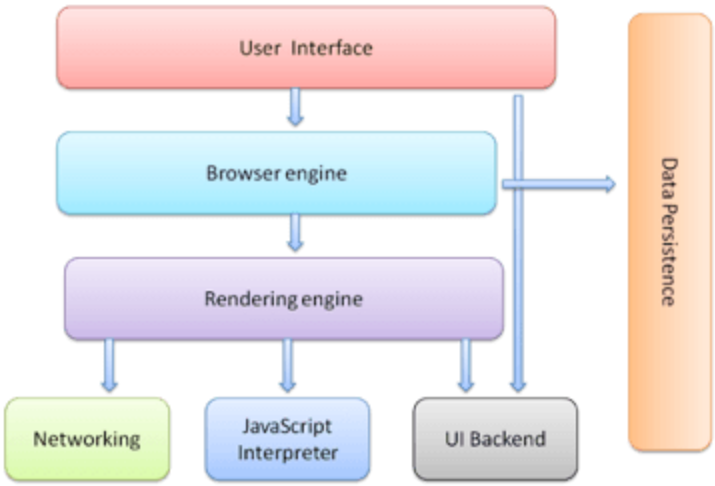
In one type of hijacking (also known as a man in the middle attack), the perpetrator takes control of an established connection while it is in progress. The attacker intercepts messages in a public key exchange and then retransmits them, substituting their own public key for the requested one, so that the two original parties still appear to be communicating with each other directly. The attacker uses a program that appears to be the server to the client and appears to be the client to the server. This attack may be used simply to gain access to the messages, or to enable the attacker to modify them before retransmitting them.

Another form of hijacking is browser hijacking, in which a user is taken to a different site than the one the user requested. There are two different types of domain name system (DNS) hijacking. In one, the attacker gains access to DNS records on a server and modifies them so that requests for the genuine Web page will be redirected elsewhere - usually to a fake page that the attacker has created. This gives the impression to the viewer that the Web site has been compromised, when in fact, only a server has been. In February 2000, an attacker hijacked RSA Security's Web site by gaining access to a DNS server that was not controlled by RSA. By modifying DNS records, the attacker diverted requests to a spoof Web site. It appeared to users that an attacker had gained access to the actual RSA Web site data and changed it - a serious problem for a security enterprise. This type of hijacking is difficult to prevent, because administrators control only their own DNS records, and

have no control over upstream DNS servers. In the second type of DNS hijack, the attacker spoofs valid e-mail accounts and floods the inboxes of the technical and administrative contacts. This type of attack can be prevented by using authentication for InterNIC records.
In another type of Web site hijack, the perpetrator simply registers a domain name similar enough to a legitimate one that users are likely to type it, either by mistaking the actual name or through a typo. This type of hijack is currently being employed to send many unwary users to a pornographic site instead of the site they requested.

11. HTTP CONTENT RENDERING:

The primary function of a web browser is to request resources from the web and display them inside of a browser window. Typically, a browser will request HTML, CSS, JavaScript and image content from a server and interpret them based on web standards and specifications. They follow standards because it allows websites to behave the same way across all browsers, and creates less work and fewer headaches for web developers.



There are 7 key components of any web browser:

1. Layout Engine – this takes input from the browser (URL bar, search box, mouse clicks and key presses) and passes them to the rendering engine

2. Rendering Engine – this takes HTML code and interprets it into what you see visually. For instance, a tag would be interpreted by the rendering engine as a set of instructions to make the text inside of the element bold

3. User Interface – This is the visual presentation of controls in the browser, for instance the back and forward buttons, bookmarks, and all chrome that appears around the main browser window where web pages display.

4. JavaScript Engine – This engine takes JavaScript code, parses it, executes it, and returns the results.

5. Network Layer – This is a function of the browser that happens behind the scenes and handles network functions such as encryption, http and ftp requests, and all network settings such as timeouts and the handling of HTTP status codes.

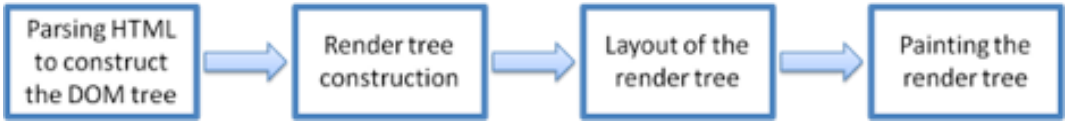
6. Storage – Browser's must store some data which can include cached files, cookies, and recently browsers have updated to be able to store data and objects created with JavaScript

7. Operating System Interface – The browser must interact with the operating system to draw out several elements of the page like drop down boxes and the chrome of a window (close, maximize, and minimize buttons)

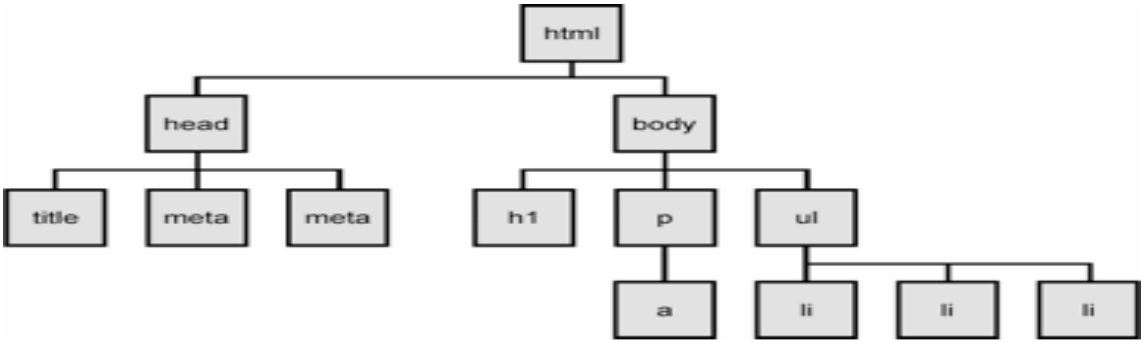
Server rendering is not a silver bullet - its dynamic nature can come with significant compute overhead costs. Many server rendering solutions don't flush early, can delay TTFB or double the

data being sent (e.g. inlined state used by JS on the client). In React, renderToString() can be slow as it's synchronous and single-threaded. Getting server rendering "right" can involve finding or building a solution for component caching, managing memory consumption, applying memoization techniques, and many other concerns. You're generally processing/rebuilding the same application multiple times - once on the client and once in the server. Just because server rendering can make something show up sooner doesn't suddenly mean you have less work to do.
Server rendering produces HTML on-demand for each URL but can be slower than just serving static rendered content. If you can put in the additional leg-work, server rendering + HTML caching can massively reduce server render time. The upside to server rendering is the ability to pull more "live" data and respond to a more complete set of requests than is possible with static rendering. Pages requiring personalization are a concrete example of the type of request that would not work well with static rendering.
Client-side rendering (CSR) means rendering pages directly in the browser using JavaScript. All logic, data fetching, templating and routing are handled on the client rather than the server.

The Rendering Engine:
The rendering engine has a very important job as it displays what you see on your screen. It communicates with the networking layer of the browser to grab HTML code and other items passed from a remote server. Then it follows these steps:



1. Parsing HTML and creating the DOM Tree – HTML is a hierarchal structure that begins with a <html> tag, usually contains a <head> and <body> tag, and elements can be nested within elements. These HTML elements are parsed and turned into a “DOM tree” by the rendering engine. It is a tree like structure made out of the HTML, where each tag is a branch starting at the root element.



2. Render Tree Construction – CSS attributes (style attributes) are also parsed and combined with the DOM tree to create a “render tree”. This is a tree of visual elements such as height/width and color ordered in the hierarchy in which they are to be displayed in the browser.
3. Layout Process – Once the render tree is constructed, the rendering engine recursively goes through the HTML elements in the tree and figure out where they should be placed on the screen.

This starts at the top left in position 0,0 and elements and attributes are mapped to coordinates on the screen.

4. Painting – Each node (branch) of the render tree is drawn out on the screen by communicating with the Operating System Interface which contains designs and styles for how UI elements should look.

A search engine spider cannot "see" a web page the way we do and will instead use a set of rules programmed into it to construct the DOM tree and understand the elements that are a part of it. For Google to understand that a word on the page is in the body context of the page, it must be able to process the entire page.

12. BROWSER ISOLATION:

Browser isolation is a cybersecurity model for web browsing that can be used to physically separate an internet user's browsing activity from their local machine, network and infrastructure. With this model, individual browser sessions are abstracted away from hardware and direct internet access, trapping harmful activity inside the disposable environment. Browser isolation may also be referred to as remote browser isolation, web isolation or remote browsing.

A major weakness in popular security tools is protection from web or browser-based attacks, malware and ransomware. The development of browser isolation technology was meant to combat that weakness. By separating browsing activity from endpoint hardware, the device's attack surface is reduced, sensitive data is protected and malware or other known and unknown security threats are minimized. This is an evolution of the cybersecurity concepts of security through physical isolation and air-gapping.

How it works:

Browser isolation works by providing users with a disposable, non-persistent environment for browsing. This can be executed through a variety of methods but typically involves virtualization, containerization or cloud browsing. When a user closes the browsing session or the session is timed out, the isolated environment is reset or discarded. Additionally, any malicious code or harmful traffic is discarded as well, preventing it from ever reaching the endpoint device or network.

The browser isolation method treats all websites, files and content equally by labeling them as untrusted or blacklisted unless otherwise specified. Within the isolated environment, files can be rendered remotely or sanitized without the need to download them. This is different from other security methods that do not treat information equally and filter content based on potential threatening signs.

Browser isolation vendors:

Browser isolation technology is typically delivered to customers through a cloud browser, a container, a virtual machine or browser isolation technology hosted on a server. A few popular vendors in the browser isolation sector include:

- Apozy.
- Authentic8.
- Ericom.
- Menlo Security.
- Symantec.
- WEBGAP.

Advantages and disadvantages of browser isolation:

The primary benefit to browser isolation is reducing the spread of malware through web browsers. This has proven to be more effective than other anti-virus application methods since it does not need to be programmed to find specific threats or risks. However, the installation of browser isolation technology can be complex or expensive. This usually means an organization has to hire IT professionals with the right expertise or contracted service providers to oversee and troubleshoot isolation efforts. Additionally, browser isolation may cause users to experience slight delay or lag times when browsing.

13. COOKIES FARMES AND FRAME BUSTING:

Cookies are messages that web servers pass to your web browser when you visit Internet sites. Your browser stores each message in a small file, called cookie.txt. When you request another page from the server, your browser sends the cookie back to the server. These files typically contain information about your visit to the web page, as well as any information you've volunteered, such as your name and interests.

The term "cookie" is an allusion to a Unix program called Fortune Cookie that produces a different message, or fortune, each time it runs.

Examples of cookies

Cookies are most commonly used to track website activity. When you visit some sites, the server gives you a cookie that acts as your identification card. Upon each return visit to that site, your browser passes that cookie back to the server. In this way, a web server can gather information about which web pages are used the most, and which pages are gathering the most repeat hits.

Cookies are also used for online shopping. Online stores often use cookies that record any personal information you enter, as well as any items in your electronic shopping cart, so that you don't need to re-enter this information each time you visit the site.

Servers can use cookies to provide personalized web pages. When you select preferences at a site that uses this option, the server places the information in a cookie. When you return, the server uses the information in the cookie to create a customized page for you.

Security concerns

Only the website that creates a cookie can read it, so other servers do not have access to your information. Additionally, web servers can use only information that you provide or choices that you make while visiting the website as content in cookies.

Webmasters have always been able to track access to their sites, but cookies make it easier to do so. In some cases, cookies come not from the site you're visiting, but from advertising companies that manage the banner ads for a set of sites (such as DoubleClick.com). These advertising companies can develop detailed profiles of the people who select ads across their customers' sites.

Accepting a cookie does not give a server access to your computer or any of your personal information (except for any information that you may have purposely given, as with online shopping). Also, it is not possible to execute code from a cookie, and not possible to use a cookie to deliver a virus.

What are Cookies?

Cookies are small text files we put in your browser to track usage of our site but they don’t tell us who you are and are harmless.

Can I delete or control my Cookies?

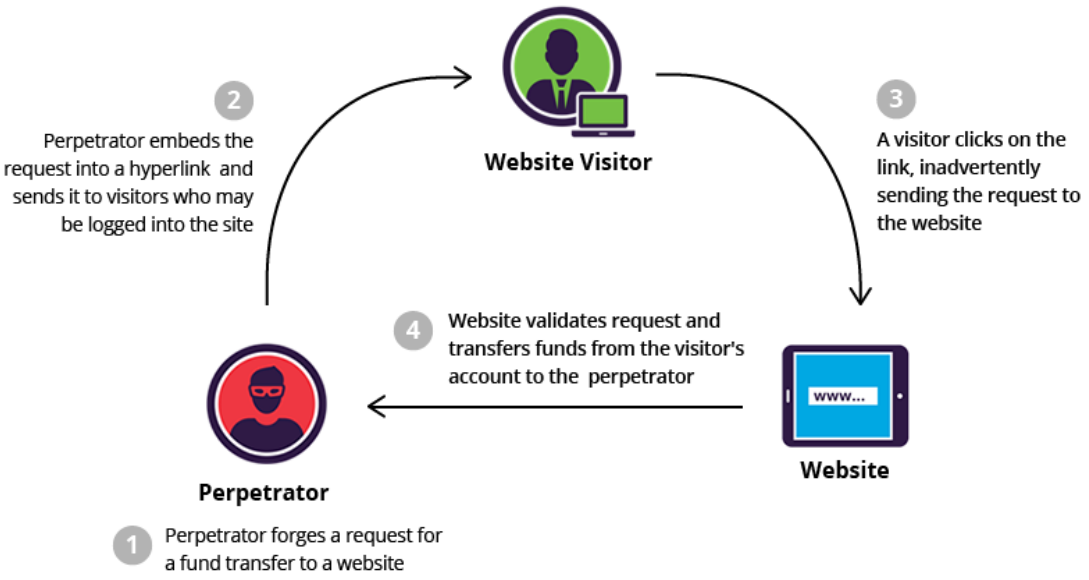
If you wish to delete any cookies that are already on your computer or device please refer to instructions for your file management software to locate the file or directory that stores cookies. Please note that by deleting our cookies or disabling further cookies you may not be able to shop the online store as the session cookies retain your basket information.

Frame busting:

Web framing attacks such as clickjacking use iframes to hijack a user's web session. The most common defense, called frame busting, prevents a site from functioning when loaded inside a frame. While many popular web sites on the Internet use frame busting to defend against clickjacking, very few mobile sites use frame busting. Similarly, few embedded web sites such as those used on home routers use frame busting. In this paper we show that framing attacks on mobile sites and home routers can have devastating effects. We develop a new attack called tap-jacking that uses features of mobile browsers to implement a strong clickjacking at- tack on phones. Tap-jacking on a phone is more powerful than traditional clickjacking attacks on desktop browsers. For home routers we show that framing attacks can result in theft of the wi-fi WPA secret key and a precise geo- localization of the wi-fi network. Finally, we show that overlay-based frame busting, such as used by Facebook, can leak private user information.

14. CROSS SITE REQUEST FORGERY:

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request.



With a little help of 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. If the victim is a normal

user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

What is CSRF

Cross site request forgery (CSRF), also known as XSRF, Sea Surf or Session Riding, is an attack vector that tricks a web browser into executing an unwanted action in an application to which a user is logged in.

A successful CSRF attack can be devastating for both the business and user. It can result in damaged client relationships, unauthorized fund transfers, changed passwords and data theft—including stolen session cookies.

CSRFs are typically conducted using malicious social engineering, such as an email or link that tricks the victim into sending a forged request to a server. As the unsuspecting user is authenticated by their application at the time of the attack, it's impossible to distinguish a legitimate request from a forged one.

What can be done with CSRF?

Most of the functionality allowed by the website can be performed by an attacker utilizing CSRF. This could include posting content to a message board, subscribing to an online newsletter, performing stock trades, using an shopping cart, or even sending an e-card. CSRF can also be used as a vector to exploit existing Cross-site Scripting flaws in a given application. For example imagine an XSS issue on an online forum or blog, where an attacker could force the user through CSRF to post a copy of the next big website worm. An attacker could also utilize CSRF to relay an attack against a site of their choosing, as well as perform a Denial Of Service attack in the right circumstances.

What are common ways to perform a CSRF attack?

The most popular ways to execute CSRF attacks is by using a HTML image tag, or JavaScript image object. Typically, an attacker will embed these into an email or website so when the user loads the page or email, they perform a web request to any URL of the attackers liking. Below is a list of the common ways that an attacker may try sending a request.

HTML Methods

IMG SRC

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

SCRIPT SRC

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

IFRAME SRC

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

JavaScript Methods

'Image' Object

```
<script>
var foo = new Image();
foo.src = "http://host/?command";
</script>
```

'XMLHTTP' Object (See "Can applications using only POST be vulnerable?" for when this can be used) **IE**

```
<script>
var post_data = 'name=value';
var xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
xmlhttp.open("POST", 'http://url/path/file.ext', true);
xmlhttp.onreadystatechange = function () {
if (xmlhttp.readyState == 4)
{
alert(xmlhttp.responseText);
}
};
xmlhttp.send(post_data);
</script>
```

Mozilla

```
<script>
var post_data = 'name=value';
var xmlhttp=new XMLHttpRequest();
xmlhttp.open("POST", 'http://url/path/file.ext', true);
xmlhttp.onreadystatechange = function () {
if (xmlhttp.readyState == 4)
{
alert(xmlhttp.responseText);
}
};
xmlhttp.send(post_data);
   </script>
```

Many other ways exist in HTML/VBScript/JavaScript/ActionScript/JScript and other markup languages to make the users browser perform remote requests.

Is this vulnerability limited to browsers?

Absolutely not. An attacker could embed scripting into a word document, Flash File, Movie, RSS or Atom web feed, or other document format allowing scripting. Applications utilizing XML documents use XML parsers to quickly parse through data. Certain tags within an XML document may tell the XML parser to request additional documents from a URI. Browsers will be the dominant way to execute these attacks but aren't the only way.

15. CROSS SITE SCRIPTING (XSS):

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user’s browser has no way to know that the script should not be trusted, and will execute the script. Because it

thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

Cross-Site Scripting (XSS) attacks occur when:

1) Data enters a Web application through an untrusted source, most frequently a web request.
2) The data is included in dynamic content that is sent to a web user without being validated for malicious content.

The malicious content sent to the web browser often takes the form of a segment of JavaScript, but may also include HTML, Flash, or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but they commonly include transmitting private data, like cookies or other session information, to the attacker, redirecting the victim to web content controlled by the attacker, or performing other malicious operations on the user's machine under the guise of the vulnerable site.

- Basic 3 types of XSS are defined as follows:

Stored XSS (AKA Persistent or Type I)

Stored XSS generally occurs when user input is stored on the target server, such as in a database, in a message forum, visitor log, comment field, etc. And then a victim is able to retrieve the stored data from the web application without that data being made safe to render in the browser. With the advent of HTML5, and other browser technologies, we can envision the attack payload being permanently stored in the victim’s browser, such as an HTML5 database, and never being sent to the server at all.

Reflected XSS (AKA Non-Persistent or Type II)

Reflected XSS occurs when user input is immediately returned by a web application in an error message, search result, or any other response that includes some or all of the input provided by the user as part of the request, without that data being made safe to render in the browser, and without permanently storing the user provided data. In some cases, the user provided data may never even leave the browser (see DOM Based XSS next).

DOM Based XSS (AKA Type-0)

As defined by Amit Klein, who published the first article about this issue[1], DOM Based XSS is a form of XSS where the entire tainted data flow from source to sink takes place in the browser, i.e., the source of the data is in the DOM, the sink is also in the DOM, and the data flow never leaves the browser. For example, the source (where malicious data is read) could be the URL of the page (e.g., document.location.href), or it could be an element of the HTML, and the sink is a sensitive method call that causes the execution of the malicious data (e.g., document.write)."

Types of Cross-Site Scripting

For years, most people thought of these (Stored, Reflected, DOM) as three different types of XSS, but in reality, they overlap. You can have both Stored and Reflected DOM Based XSS. You can also have Stored and Reflected Non-DOM Based XSS too, but that’s confusing, so to help clarify things, starting about mid 2012, the research community proposed and started using two new terms to help organize the types of XSS that can occur:

- Server XSS
- Client XSS

Server XSS

Server XSS occurs when untrusted user supplied data is included in an HTML response generated by the server. The source of this data could be from the request, or from a stored location. As such, you can have both Reflected Server XSS and Stored Server XSS.

In this case, the entire vulnerability is in server-side code, and the browser is simply rendering the response and executing any valid script embedded in it.

Client XSS

Client XSS occurs when untrusted user supplied data is used to update the DOM with an unsafe JavaScript call. A JavaScript call is considered unsafe if it can be used to introduce valid JavaScript into the DOM. This source of this data could be from the DOM, or it could have been sent by the server (via an AJAX call, or a page load). The ultimate source of the data could have been from a request, or from a stored location on the client or the server. As such, you can have both Reflected Client XSS and Stored Client XSS.

16. DEFENSES AND PROTECTIONS AGAINST XSS:

Recommended Server XSS Defenses

Server XSS is caused by including untrusted data in an HTML response. The easiest and strongest defense against Server XSS in most cases is:

- Context-sensitive server side output encoding

The details on how to implement Context-sensitive server side output encoding are presented in the OWASP XSS (Cross Site Scripting) Prevention Cheat Sheet in great detail.

Input validation or data sanitization can also be performed to help prevent Server XSS, but it's much more difficult to get correct than context-sensitive output encoding.

Recommended Client XSS Defenses

Client XSS is caused when untrusted data is used to update the DOM with an unsafe JavaScript call. The easiest and strongest defense against Client XSS is:

- Using safe JavaScript APIs

However, developers frequently don't know which JavaScript APIs are safe or not, never mind which methods in their favorite JavaScript library are safe. Some information on which JavaScript and jQuery methods are safe and unsafe is presented in Dave Wichers' DOM Based XSS talk presented at OWASP AppSec USA in 2012: Unraveling some of the Mysteries around DOM Based XSS

If you know that a JavaScript method is unsafe, our primary recommendation is to find an alternative safe method to use. If you can't for some reason, then context sensitive output encoding can be done in the browser, before passing that data to the unsafe JavaScript method. OWASP's guidance on how do this properly is presented in the DOM based XSS Prevention Cheat Sheet. Note that this guidance is applicable to all types of Client XSS, regardless of where the data actually comes from (DOM or Server).

How do I detect if a website is vulnerable?

If your website allows performing a site function using a static URL or POST request (i.e. one that doesn't change) then it is possible. If this command is performed through GET then it is a much higher risk. If the site is purely POST see "Can applications using only POST be vulnerable?" for use cases. A quick test would involve browsing the website through a proxy such as Paros and record the requests made. At a later time perform the same action and see if the requests are performed in an identical manner (your cookie will probably change). If you are able to perform the same function using the GET or POST request repeatedly then the site application may be vulnerable.

Methods of CSRF mitigation:

A number of effective methods exist for both prevention and mitigation of CSRF attacks. From a user's perspective, prevention is a matter of safeguarding login credentials and denying unauthorized actors access to applications.
Best practices include:

- Logging off web applications when not in use
- Securing usernames and passwords
- Not allowing browsers to remember passwords
- Avoiding simultaneously browsing while logged into an application

For web applications, multiple solutions exist to block malicious traffic and prevent attacks. Among the most common mitigation methods is to generate unique random tokens for every session request or ID. These are subsequently checked and verified by the server. Session requests having either duplicate tokens or missing values are blocked. Alternatively, a request that doesn't match its session ID token is prevented from reaching an application.
Double submission of cookies is another well-known method to block CSRF. Similar to using unique tokens, random tokens are assigned to both a cookie and a request parameter. The server then verifies that the tokens match before granting access to the application.
While effective, tokens can be exposed at a number of points, including in browser history, HTTP log files, network appliances logging the first line of an HTTP request and referrer headers, if the protected site links to an external URL. These potential weak spots make tokens a less than full-proof solution.

Can CSRF be prevented by implementing referrer checking?

No for two reasons.
First there are many ways that a Referer header can be blanked out or modified such as via web filtering software, parental control software, privacy software, proxies, or DOM trickery. This makes the referer header unreliable by nature.
Second Referer headers can be spoofed using XMLHTTP and by using flash as demonstrated by Amit Klein and rapid7. While these particular methods have been patched by the vendors, not every user visiting your website has applied these patches. Even if they did the first issue would still exist.

Has a vulnerability in a major site been discovered?

A vulnerability in GMail was discovered in January 2007 which allowed a attacker to steal a GMail user's contact list. A different issue was 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...

What can I do to protect myself as a user?

Nothing. The fact is as long as you visit websites and don't have control of the inner architecture of these applications you can't do a thing.

What can I do to protect my own applications?

The most popular suggestion to preventing CSRF involves appending non predictable challenge tokens to each request. It is important to state that this challenge token MUST be associated with the user session, otherwise an attacker may be able to fetch a valid token on their own and utilize it in an attack. In addition to being tied to the user session it is important to limit the time period to which a token is valid. This method is documented in multiple documents however as pointed out in mailing list postings an attacker can utilize an existing browser vulnerability or XSS flaw to grab this session token.