CEC14: OPERATING SYSTEMS

# Case Study on OpenBSD

Nipunika(2018UCO1550)
Meghna(2918UCO1564)

# TABLE OF CONTENTS

## Design Principles

OpenBSD is a UNIX - based operating system so it has basically the same design principles that UNIX has but it mainly focuses on SECURITY.

- The OpenBSD installer, a plain shell script, is very minimalistic and uncluttered which makes the installation process both easy and fast. It ensures the **ease of use principle** for a secure operating system.
- One of the goals of the OpenBSD project is the integration of facilities and software for **strong cryptography** into the core operating system.
- OpenBSD's top priority is to be **secure** and **reliable**, and it was designed with security in mind throughout every stage of development. OpenBSD 's well-deserved reputation of being an ultra-secure operating system is the byproduct of a no-compromise attitude valuing **simplicity**, **correctness**, and most importantly **proactivity**.
- The word "open" in the name OpenBSD refers to the availability of the operating system's source code on the Internet. It also refers to the wide range of hardware platforms, the system supports. It ensures the **open design** feature.
- The design of the system is not very complex and hence the system is easy to be tested and analysed and of course reliable (**Economy of mechanisms**).
- The system is built up on permissions (**Permission based**) and as the default scheme for the system there is a denial of access for resources in the system.It ensures security.

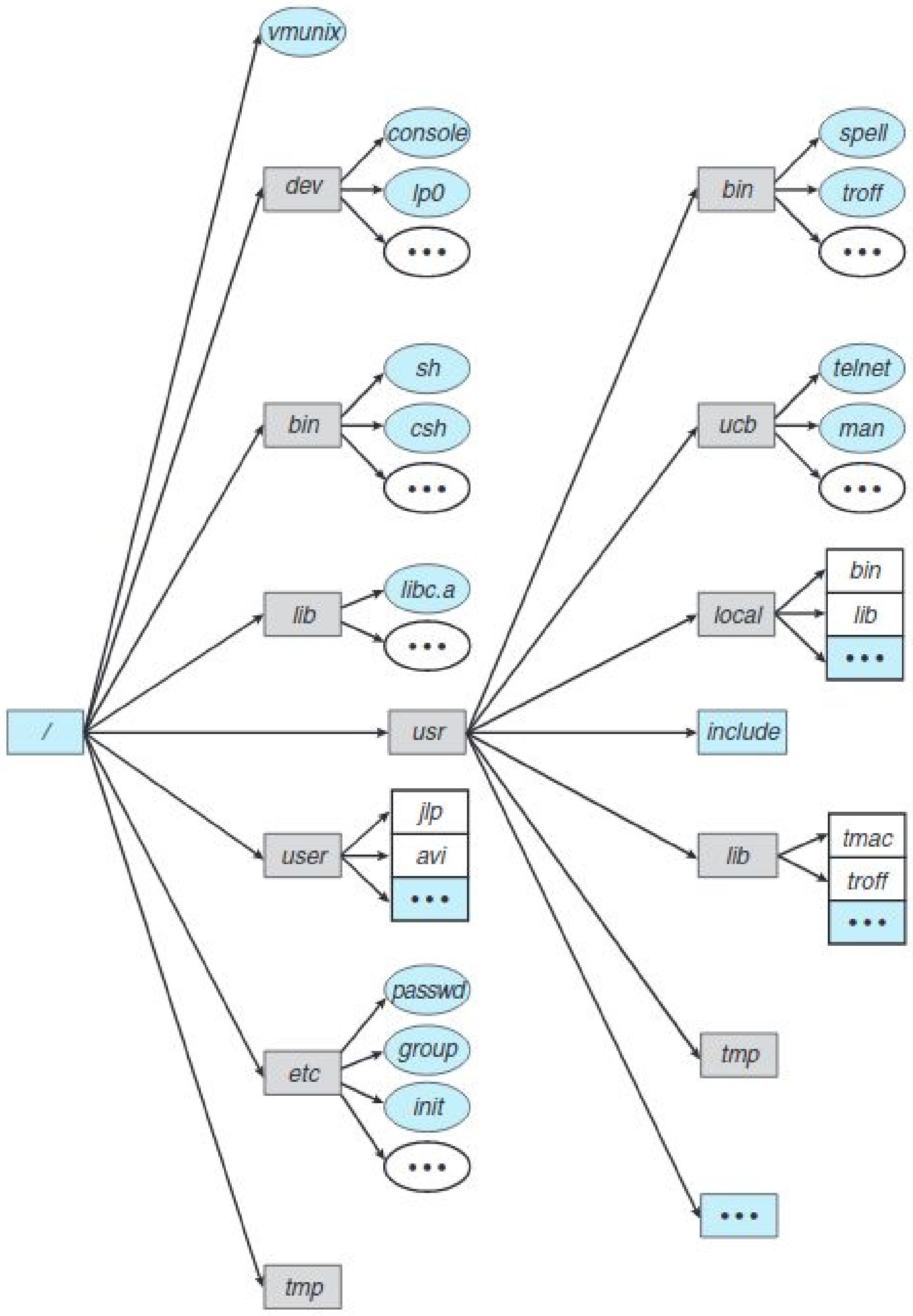
## Programmer Interface

This interface is defined by the set of system-calls. The set of user programs defines the User Interface. User and Programmer Interface together define the context of the functionalities that the kernel supports.

### File Manipulation

Everything in UNIX based systems can be treated as files. Files are organized in tree-structured directories. Directories themselves are files that contain information on how to find other files. A file may be known by multiple names called links. OpenBSD also supports symbolic links which are file names containing the pathname to the required file. Hardware devices are implemented as special files known to the kernel as device interfaces.

The figure shows the file structure in OpenBSD:



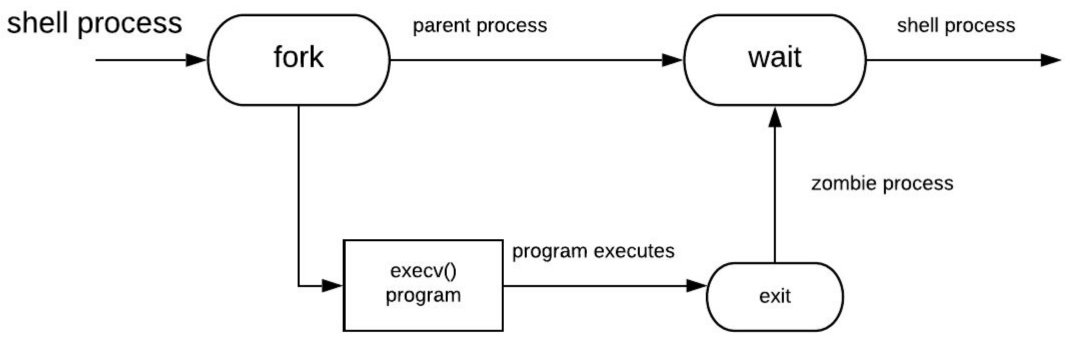
- System calls for basic file manipulation are creat, open, read, write, close, unlink, and trunc.
    - Creat sys call creates an empty file given a path and file name
    - Open sys call opens an existing file given the path and the modes and returns an integer called file descriptor
    - File descriptor, when passed to read or write sys call, helps perform data transfer
    - File descriptor, when passed to close syscall, closes the file
    - Trunc syscall reduces the length of the file to zero

    - File description is simply an index to the list of open files for a particular process
- There are several ways to change the information about a file
    - Stat syscall obtains the info about a file
    - Chown changes owner group
    - Chmod changes the mode
    - Rename changes the name
- Link syscall helps create a hard link for file with a new name, unlink can destroy the link and symlink helps make a symbolic link
- Syscalls for directory manipulation are:
    - Mkdir: makes a director
    - Rmdir: removes a directorycd: change directory
    - Opendir: open a directory
    - Closedir: close a directory

## Process Control

A process is a program in execution.

- fork() syscall is used to create a new child with the same address space the original program. Return code is 0 for the parent program and the PID for the child program.
- execv() transforms the calling process into a new process. The new process is constructed from an ordinary file, whose name is pointed to by path, called the new process file.
- exit() syscall can be used to terminate a process
- wait() syscall can be used by the parent process to wait for an event, wait3() performs the same function with the added ability to provide performance stats.
- Pipes are used for communication between processes. The pipe() function creates a pipe, which is an object allowing unidirectional data flow and allocates a pair of file descriptors. The first descriptor connects to the read end of the pipe, and the second connects to the write end so that data written to fildes[1] appears on (i.e., can be read from) fildes[0].



- Groups of related processes frequently cooperate to accomplish a common task. A process inherits its group from its parent but setgrp() command can change the group of the process

# Process Management

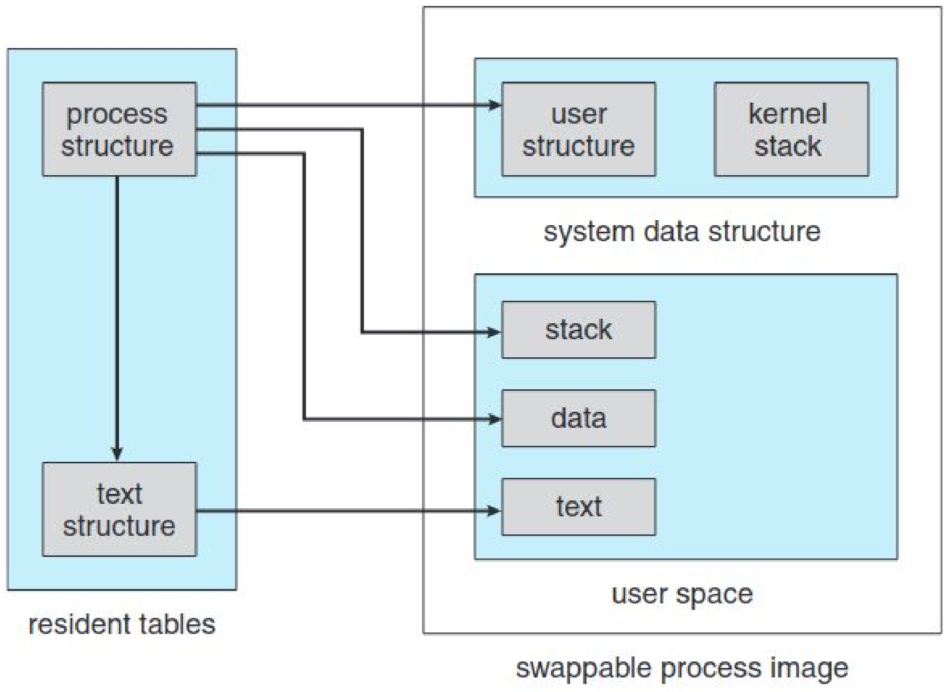
These processes are represented in UNIX by various control blocks.

## Process Control Blocks

A process structure contains everything that the system needs to know about a process when the process is swapped out.

- The process structures of ready processes are kept linked together by the scheduler in a doubly-linked list (the ready queue), and there are pointers from each process structure to the process's parent, to its youngest living child, and to various other relatives of interest, such as a list of process sharing the same program code.
- Every process with sharable text (almost all, underFreeBSD)has a pointer from its process structure to a text structure. The text structure records how many processes are using the text segment, including a pointer into a list of their process structures, and where the page table for the text segment can be found on disk when it is swapped.
- The page tables record information on the mapping from the processes of virtual memory to physical memory. The process structure contains pointers to the page table, for use when the process is resident in main memory, or the address of the process on the swap device when the process is swapped.

- Every process has both a user and a system mode. Most ordinary work is done in user mode, but when a system call is made, it is performed in system mode.



  - fork() syscall allocates new process structure for a child process and copies the user structure
  - vfork() syscall doesnt copy data and stack to the new process rather the new process only shares the page table of the parent process. This is a fairly dangerous syscall since any changes will take place for both parent and children process until the execve() syscall occurs.

## CPU Scheduling

- Processes are given small CPU time slices by a priority algorithm which reduces round-robin scheduling for CPU bound jobs.
- The more CPU time a process accumulates, the lower (more positive) its priority becomes, and vice versa. This negative feedback inCPUschedulingmakes it difficult for a single process to take all the CPU time. Process aging is employed to prevent starvation.
- OpenBSD reschedules processes every 0.1 seconds and recomputes the priorities every second.
- There is no preemption of one process by another in the kernel. A process may relinquish the CPU because it is waiting for I/O or because its time slice has expired.

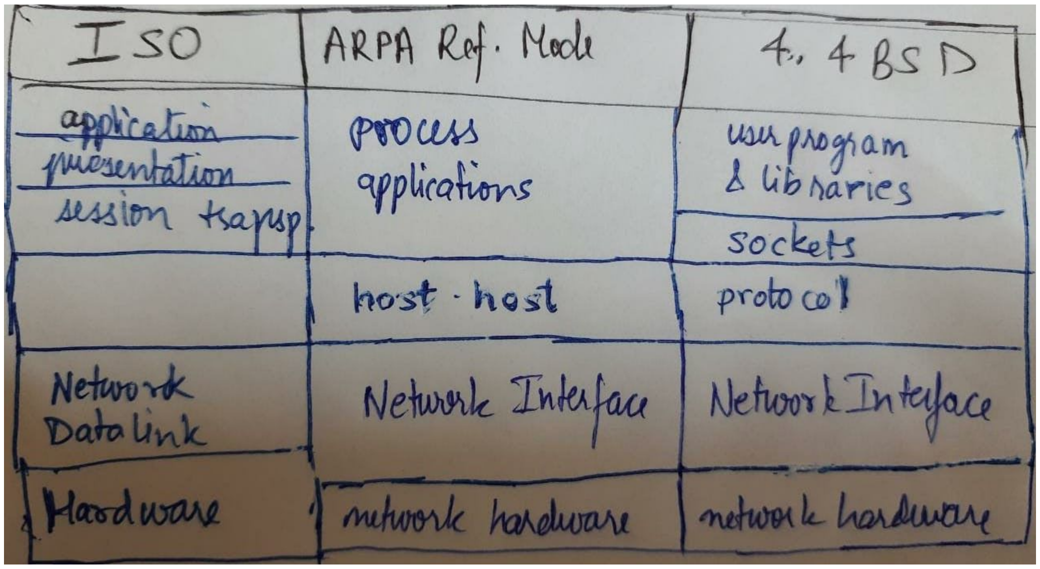
## Interprocess Communication

### Socket

- A pipe permits a reliable unidirectional byte stream between two processes. In openBSD, pipes are implemented as a special case of socket mechanism.
- Pipes can be used by two processes related through fork() syscall but sockets can be used by numerous unrelated processes.
- A socket is an endpoint of communication. A socket in use usually has an address bound to it. The nature of the address depends on the communication domain of the socket
- The following socket types are supported by OpenBSD:
    - Stream Sockets
    - Sequenced Packet Sockets
    - Datagram Sockets
    - Reliably delivered message sockets
    - Raw Sockets
- The select system call can be used to multiplex data transfers on several file descriptors and/or socket descriptors. It can even be used to allow one server process to listen for client connections for many services and to fork a process for each connection as the connection is made.

### Network Support

- OpenBSD supports the DARPA internet protocols UDP, TCP, IP, and ICMP, wide range of Ethernet, token-ring, andARPANETinterfaces. The networking framework is most closely related to ARPA Reference Model (ARM).
- Following four protocol layers are supported by the ARM Framework:
    - Process/Application
    - Host-Host
    - Network Interface
    - Network Hardware
- Following is a comparison of ISO Model, ARM, and the protocol and services of 4.4BSD so that you can easily understand the equivalent layer in each model:



| ISO | ARPA Ref. Mode | 4.4 BSD |
|---|---|---|
| application<br>presentation<br>session transp | process applications | user program & libraries<br>sockets |
|  | host · host | protocol |
| Network<br>Data Link | Network Interface | Network Interface |
| Hardware | network hardware | network hardware |

## Memory Management

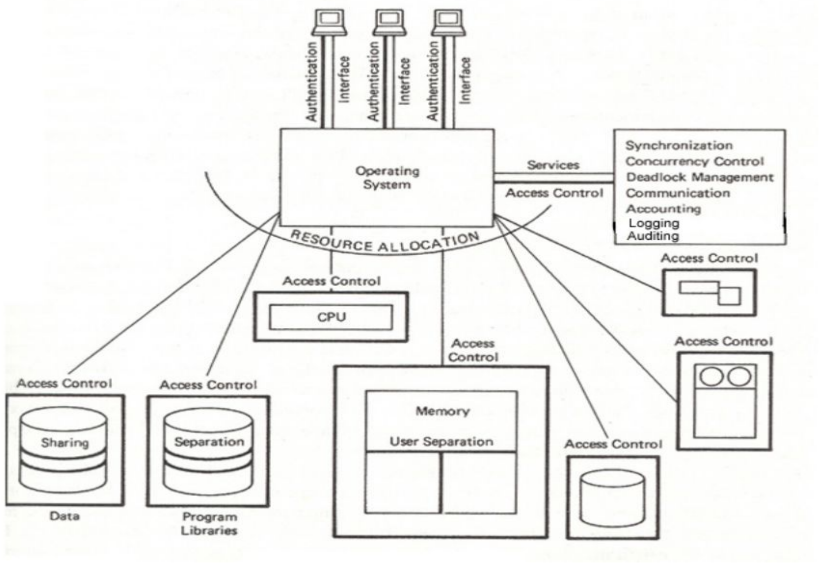
OpenBSD is demand paged Virtual Memory system. Paging eliminates external memory fragmentation and allows more jobs to be kept in memory minimizing elimination. Demand paging is done in a straightforward manner. When a process needs a page and the page is not there, a page fault to the kernel occurs, a frame of main memory is allocated, and the proper disk page is read into the frame.
Optimizations include:

- A desired page which is still in main memory but has been marked invalid can be validated without any I/O transfer.
- Pages can easily be retrieved from the list of free Pages as mostly happens when processes start.

OpenBSD uses a modification of the second chance algorithm. .When the clock hand reaches a given frame, if the frame is marked as being in use by some software condition (for example, if physical I/O is in progress using it) or if the frame is already free, the frame is left untouched, and the clock hand sweeps to the next frame. Otherwise, the corresponding text or process page-table entry for this frame is located. If the entry is already invalid, the frame is added to the free list; otherwise, the page-table entry is made invalid but reclaimable (that is, if it has not been paged out by the next time it is wanted, it can just be made valid again).

## Security

- The OpenBSD operating system focuses on security and the development of security features. According to author Michael W. Lucas, OpenBSD "is widely regarded as the most secure operating system available anywhere, under any licensing terms."
- OpenBSD is a distribution based on BSD 4.4. The goals of the OpenBSD project are to have correctness, security, standardization and portability.
- The cryptography is exported with OpenBSD as default. OpenBSD is said to be secure by default, which means that novice users do not have to learn everything directly.
- It is also said that all services that are not needed to run the system have been disabled as default.



Following are some of the security features:

## Authentication

- Authentication manages the access to the system itself in different ways that is verifying the identity of a user.
- The authentication in all tested systems is based on login name and password, both through console and remote connection services like SSH.

- OpenBSD provides one time password support for users, which means that the users get a key output at the login session and the users generate the passwords from that key in another application.

### Network Security

- There is a support for rlogin and telnet ( these services do not provide any encryption of the connection ), they are not started as default in any of the systems.
- OpenBSD has configured SSH (Secure Socket Shell) by default to let root login over a network with SSH. This is more secure because it is not hard to see if a new connection over SSH is initialized by scanning the network with a packet sniffer.
- The root login should be disabled for SSH in OpenBSD by default.

## Access Control

- In OpenBSD there is currently no Access Control List support for the file system.
- The OpenBSD system is using the three fields protection scheme.
    - It is a lightweight protection scheme. In this protection scheme there are only three fields that specify the access control for each file or directory. Each field is a collection of bits, which each prevent or allow access.
    - Read, write and execute are the most common bits that can be set (rwx).
        - R - read access
        - W - write access
        - X- file execution access
- The three fields that contain these sets of bits are for owner, group and universe.
- The disadvantage with the three field's protection scheme is that it is not very fine-grained access control.

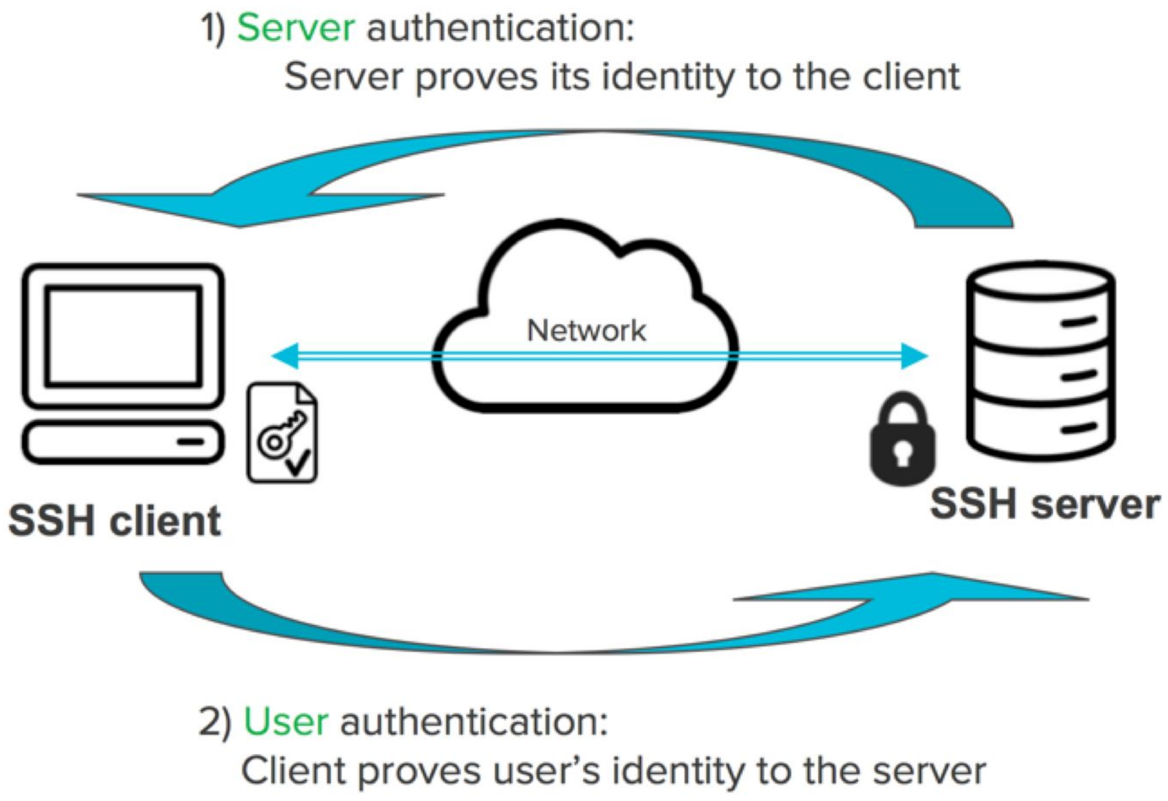
## Cryptography

- Cryptography is all about securing information.
- OpenBSD uses OpenSSH, which is supporting SSH version 1, 1.5 and 2. All cryptographic restrictive components in the client have been removed and support for Kerberos authentication and ticket passing is included.

- To protect sensitive information such as passwords from leaking on to disk, where they can persist for many years, OpenBSD supports encryption of the swap partition. The swap space is split up into many small regions that are each assigned their own encryption key: as soon as the data in a region is no longer required, OpenBSD securely deletes it by discarding the encryption key.This feature is enabled by default in OpenBSD 3.9 and later.

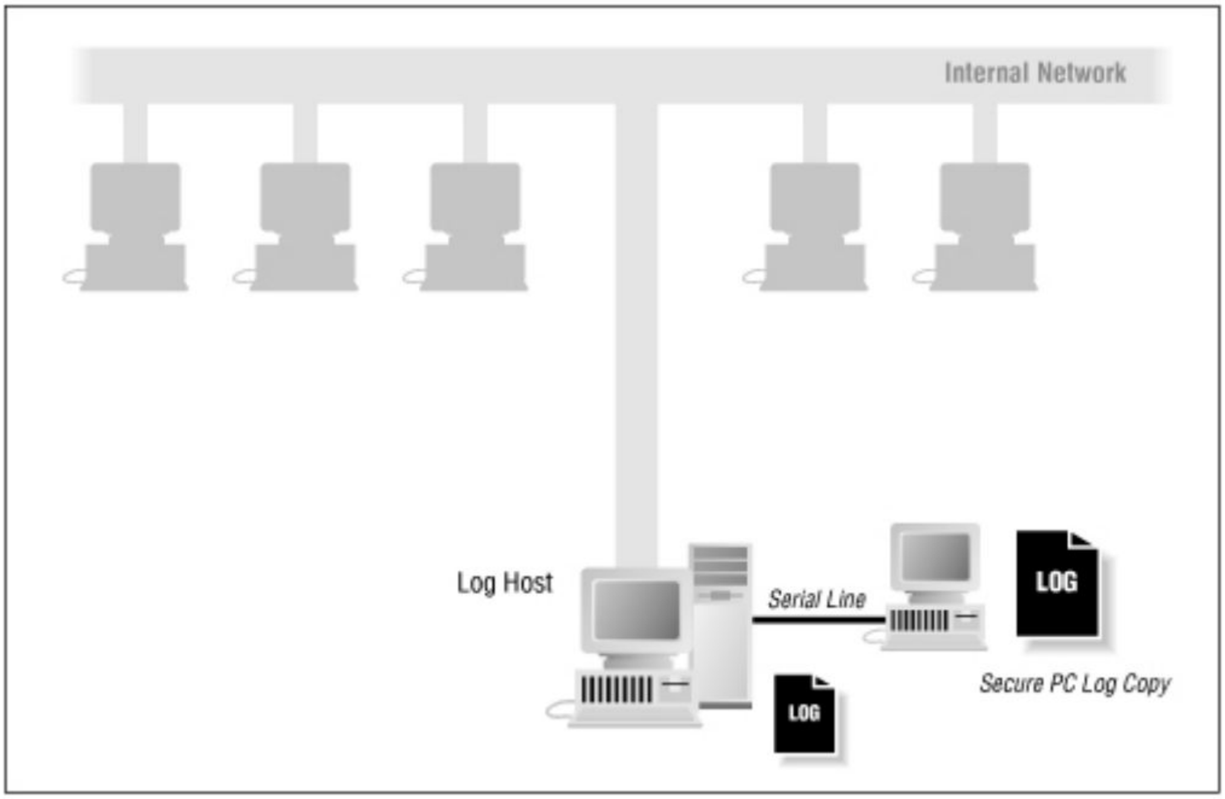
## Network Security

- The OpenBSD project is using IPsec(IP security)  and Kerberos V for network security in the system.In OpenBSD the algorithms MD5, SHA-1 are used for this purpose
- IPSec is included in OpenBSD but needs to be compiled into the kernel before it could be used.
- There is heavy use of randomization with the help of network stack to increase security and reduce the predictability of various values that may be of use to an attacker, including TCP initial sequence numbers and timestamps.


1) Server authentication:
Server proves its identity to the client
Network
SSH client
SSH server
2) User authentication:
Client proves user's identity to the server

## Auditing and Logging

- Auditing and Logging is a way to collect information about what is happening or what has happened in the system and for different services that are running on the system. These logs may contain information about possible attacks or security breaches. The logs are also a source of information to see what caused the failure and perhaps what to do to insure that it will not happen again.
- OpenBSD uses the syslog service to provide log functionality to the system.
  - Syslog provides a logging service for all applications that are used in the system and logs can be created on local machines as well as over network on another system.
  - The logs are created in the file system directory /var/log/.
  - In the default installation of the system there is already a default configuration for the syslog service in the file syslog.conf which is located in /etc/.
  - OpenBSD supports 8 layers of importance for the syslog service.



## Firewalls

- Firewalls are used to defend both systems and whole computer networks. The main function of a firewall is to restrict the information flow between two networks, often local and global network.
- Packet Filter is OpenBSD's firewall system. Pack filter is a fully-featured firewall with Quality of Service functionality.
- OpenBSD is using Packet Filter (PF) to filter TCP/IP traffic and to do network address translation (NAT).
- PF also has integrated quality of service and can, therefore, control the bandwidth and packet prioritization.
- PF is included as default in the OpenBSD release, since version 3.0.
- To get the PF firewall to work, it needs to be enabled in the boot configuration file (rc.conf.local).

Security in OpenBSD can be summarized in the following table.

| AUTHENTICATION | ACCESS CONTROL | CRYPTOGRAPHY | AUDITING AND LOGGING | FIREWALLS |
|---|---|---|---|---|
| Medium: Password | Low: DAC | High: AES | Medium: syslog | High: PF |