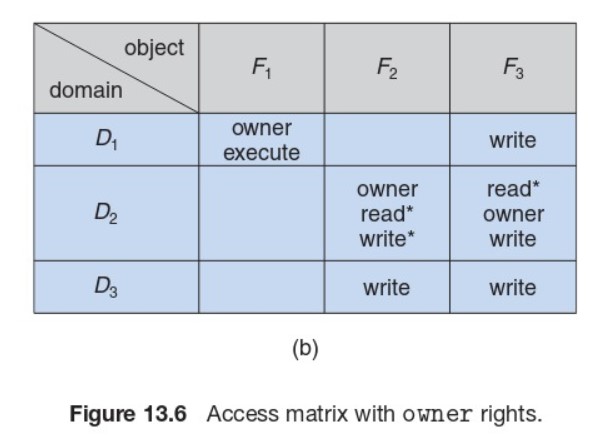
**Week 5 Interactive Assignment: Outline the goals and principles of a domain- and language-based protection in a modern computer system, and describe how an access matrix is used to protect specific resources a process can access.**

Computers need methods to avoid both accidental, as well as malicious, attempts to violate access restrictions. Additionally, strict policies are needed for access to the resources in a computer system. Providing the mechanism for deploying both of these kinds of protection is the operating system. In other words, the operating system solves the protection problem—that each object (such as a file) is accessed correctly, and only by processes (or users) with specific access rights to that object (Silberschatz, Galvin, & Gagne, 2014). One way that the operating system deploys this protection is through the principle of least privilege. This principle states that users should receive the least amount of privileges possible that will allow them to complete the necessary tasks. An example of this type of protection would be creating separate accounts for users, each consisting of customized privileges relative to the user’s role and responsibilities. A second principle on which system protection is based is the “need to know” principle. The basis of this concept is that a process should only be allowed to access resources that it currently requires (Silberschatz, Galvin, & Gagne, 2014).

System protection may further be subdivided into domain-based and language-based. Language-based protection refers to the code written in a programming language specifying policies for the allocation of resources. Language-based protection is deployed through software and is used to initiate system calls corresponding to protection policies to employ resources for an operation (Silberschatz, Galvin, & Gagne, 2014). Domain-based protection works hand-in-hand with language-based protection. Domains consist of objects, such as memory segments, printers, or files, as well as a set of operations that can be performed on those objects. Domains can either be static (unchanging over the life of a process or system) or dynamic (changed as needed). Additionally, domain switching allows a process to switch from one domain to another and allowing for the escalation of privileges as needed (Silberschatz, Galvin, & Gagne, 2014). These characteristics of domains allow flexibility in terms of protection and access.

The ability to perform a specific operation on a given object is known as an access right. In the example below (Figure 13.6), the user associated with domain D1 is the owner of file F1 (an object). Therefore, they possess the access right to execute the file. The user associated with domain D2 is the owner of file F2 and can both read and write to the file. While the user associated with domain D3 can also write to file F2, they have more limited access rights, since they are not the owner of the file. Access rights are determined by the owner, and can include the ability to read, write, append, insert, execute, delete, lock, modify rights, set owner, create group, and add a member to the file (UNC, n.d.). These access rights are controlled through what is known as an access matrix. An access matrix is the mechanism for establishing and referencing access rights for objects in relation to domains. Domains can be users, processes, or procedures. In an access matrix, the access rights of objects are arranged into columns, while domains are arranged into rows (Silberschatz, Galvin, & Gagne, 2014). Figure 13.6: Operating System Concepts Essentials (2nd ed.) (2014).

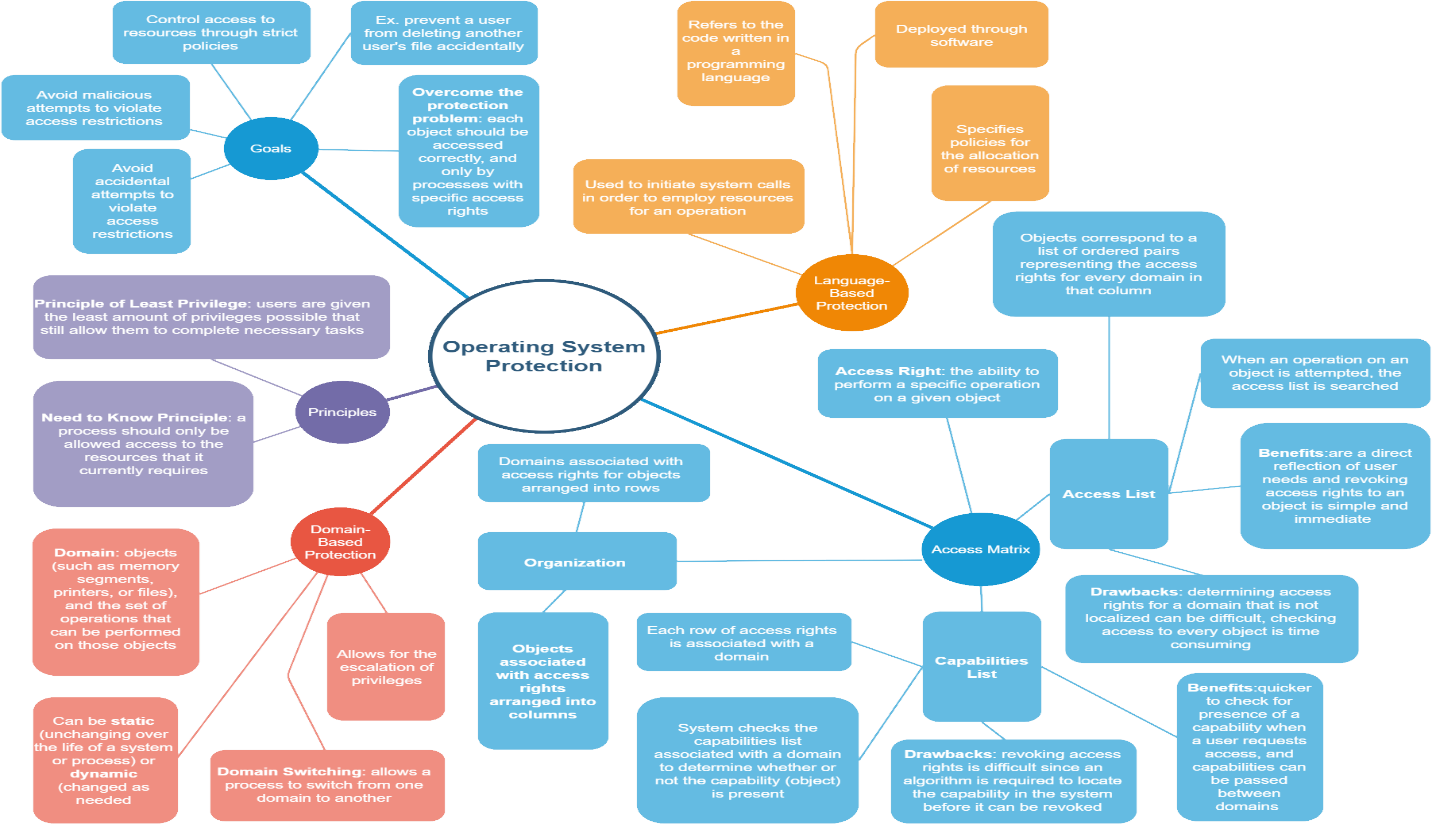


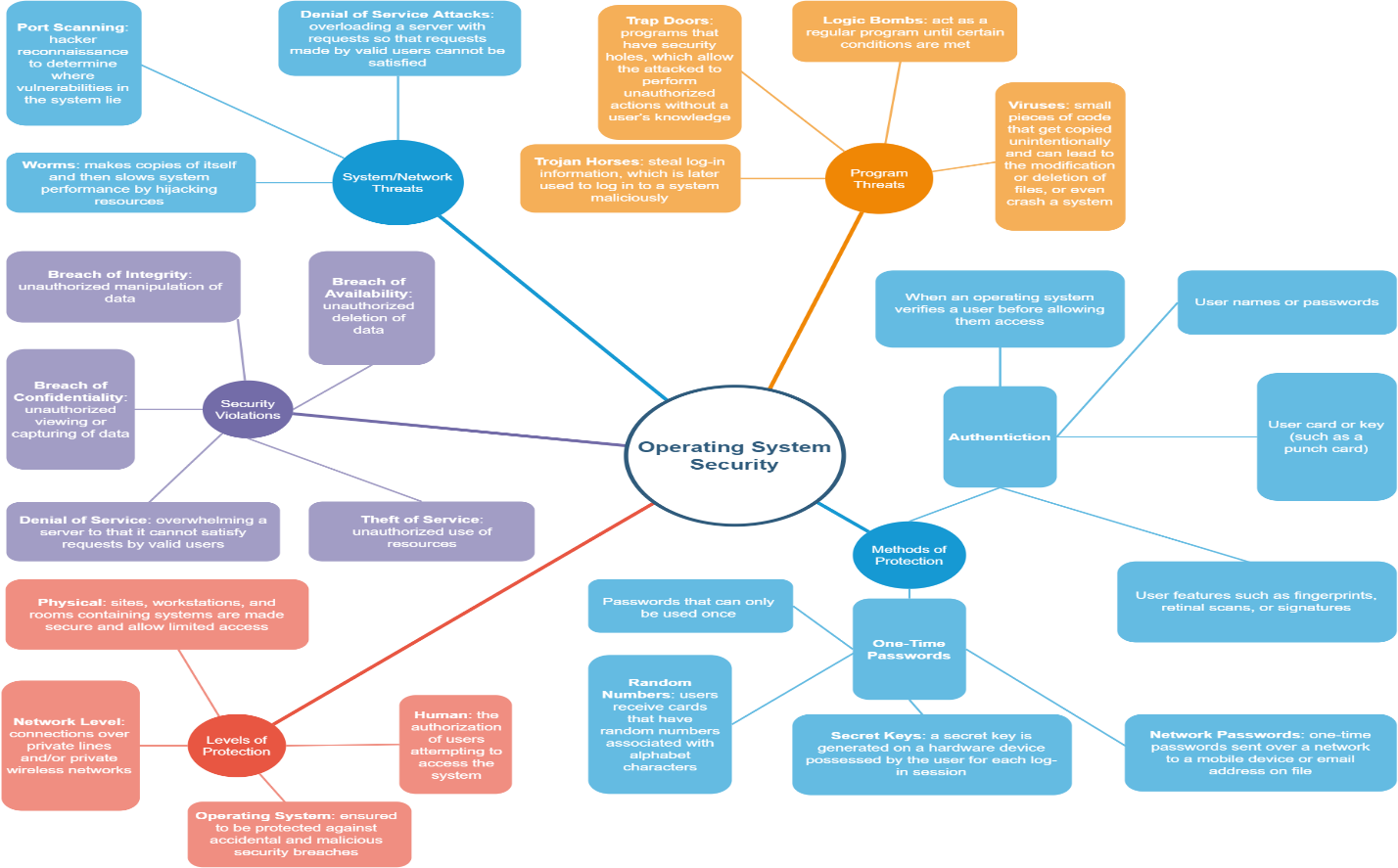
Access matrices can be implemented in two different ways, either through access lists or through capability lists. Implementing an access matrix through access lists refers to a model in which objects (files) correspond to a list of ordered pairs that represent the access rights for every domain falling under the column of that object. When an operation on an object is attempted (i.e., writing to a file), the access list for that file is searched to determine whether or not it contains an access right for the domain. If it does, the operation executes. If there is no access right for the domain found in the access list, then the default set is checked. If it is still not found, the operation is denied (Silberschatz, Galvin, & Gagne, 2014). The other method of access matrix implementation utilizes capability lists. In this model, each row of access rights is associated with a domain. Therefore, if a user wants to write to a file, the system checks the capability list associated with a domain to determine whether the capability (the object) is present. If the domain possesses the capability, the operation can execute. Capability lists are treated as protected objects, and are not directly accessible by the domain (Silberschatz, Galvin, & Gagne, 2014).

**Describe how security is used to protect programs, systems, and networks from threats.**

While protection refers to the internal control of access to data and resources, security aims to guard a system from potential external threats. There are five main categories of security violations that operating systems are designed to protect against. These categories are breach of confidentiality (unauthorized viewing or capturing of data), breach of integrity (unauthorized manipulation of data), breach of availability (unauthorized deletion of data), theft of service (unauthorized use of resources), and denial of service (overwhelming a server so that it cannot satisfy requests by valid users) (Silberschatz, Galvin, & Gagne, 2014). While security threats are organized into these five broad categories, operating systems are often designed to protect against specific threats such as those designed to attack via programs or those which create vulnerabilities in the system or network. Some examples of program threats include Trojan horses (steal log-in information, later used to log in to a system maliciously), trap doors (programs that have security holes, which allow the attacker to perform unauthorized actions without a user’s knowledge), logic bombs (act as regular programs until certain conditions are met), and viruses (small pieces of code that get copied unintentionally and can lead to the modification or deletion of files, or even crash a system). Some examples of system threats include worms (copies itself repeatedly, slowing system performance by hijacking resources), port scanning (hacker reconnaissance to determine where vulnerabilities in the system lie), and denial of service attacks (overloading a server with requests so that requests by valid users cannot be satisfied) (Tutorials Point, n.d.).

To protect against these threats, systems are protected at four different levels: physical (sites, workstations, and rooms containing systems are made secure and allow limited access), human (the authorization of users attempting access to the system), operating system (protected against accidental and malicious security breaches), and at the network level (connections over private lines or private wireless networks) (Silberschatz, Galvin, & Gagne, 2014). In addition, systems often employ authentication processes and one-time passwords to increase security. Authentication refers to the process in which an operating system verifies a user before allowing them access to the system. Methods of authentication can include usernames or passwords, a user card or key (such as a physical punch card), or even user features such as fingerprints, a retinal scan, or a signature. One-time passwords refer to passwords that can only be used once. They can be implemented through random numbers (users receive cards that have random numbers associated with alphabet characters), secret keys (a secret key is generated on a hardware device possessed by the user for each log-in session), and network passwords (one-time passwords sent over a network to a mobile device or email address on file) (Tutorials Point, n.d.). My bank implements one-time passwords when I want to log in to my account on a new device by sending a code to my cell phone.





**Figure 1: Concept Maps**

**References**

Silberschatz, A., Galvin, P. B., & Gagne, G. (2014). Operating system concepts essentials (2nd ed.). Retrieved from https://redshelf.com/

Tutorials Point. (n.d.). Operating System- Security. Retrieved on October 11, 2018 from https://www.tutorialspoint.com/operating\_system/os\_security.htm

University of North Carolina, Chapel Hill, Department of Computer Science. (n.d.). COMP 730/242 Class Notes: Section 8: Protection. Retrieved on October 10, 2018 from http://www.cs.unc.edu/~dewan/242/s07/notes/prot.PDF