Appendix 1 – Cybersecurity First Principles

1. Domain Separation
    a. In the Robert Frost poem "Mending Fences", the last line states "Good fences make good neighbors". Domain separation is like this.
    b. A domain is a generic term. It could be a region governed by a king; it could be a website, or an area of control. At the Social Security Administration, there are different operational areas and job functions, each having different job responsibilities. In the programming department, there are programmers (developers and maintainers and many more), testers and system administrators. When the developers wish to test their code, they need test data that is similar to the real data, but obviously not live data. The test program must use test data in case the program reacts unexpectedly. Keeping the test data separate from the operational data is one example of domain separation.
    c. Inside a computer system, there are also domains. For example, most hardware microprocessors have a supervisor domain (sometimes referred to as a supervisor state or privileged state) and a user domain. In supervisor state, privileged hardware instructions can be executed. An operating system uses these hardware domains to implement mechanisms that protect it from interference by user written programs and purchased applications. The operating system code runs in supervisor state, while the user programs run in the user domain.
2. Process Isolation
    a. A process is a program running in a computer. Processes running in a computer have their own portion of memory called the address space. The address space is an area of memory that only one running program can access. If a word processor, a database and a browser are running on a computer, they are all running in different address spaces. This is done to ensure correct operation, security and protection. The word processor cannot access the memory of the browser or database. If two processes are running and one of the processes has a problem, that problem should be confined to the process with a problem and not affect the other process.
    b. In addition to process isolation, it is also possible to have operating isolation. Programs such as VMWare or Virtual Box enable multiple operating systems to execute on the same computer without interfering with other program. In this context, each OS is viewed as a process, to be kept separate from the other

processes (OSs). Program running in each OS, should not be aware of programs running on another OS, nor should they be aware of the existence of another OS.

3. Resource Encapsulation
    a. A computer has many resources. A resource can be hardware based such as memory, disk drives, or a monitor. It can also be system objects such as semaphores, a linked list, or shared memory.
    b. Encapsulation is an object oriented concept where all data and functions required to use the resource are packaged into a single self-contained component. The goal is to only allow access or manipulation of the resource in the way the designer intended. An example, assume a flag pole is the object. There are fixed methods on how the flag pole is to be used. Put the flag on, take the flag off, raise or lower the flag. Nothing else can be done to the flag pole.
    c. In addition to controlling what operations can be performed on the resource, the system can also control which users can perform these operations on the resource.
4. Least Privilege
    a. The principle of least privilege says to allow the minimum number of privileges necessary to accomplish the task.
    b. When a person gets a new computer, s/he installs or logs onto the computer using an administrative account. This account has privileges to install software, add users, add hardware, and add and delete almost any program or file. The account is all powerful and must be used wisely. If a person uses a browser to access a website that contains malware and they are running as administrator, it is more likely that malware could be installed. If the person was running as a regular user with minimal privileges, the malware would not have been installed.
5. Layering
    a. Layering in computer security implements multiple layers of computer security, each one having to be conquered before moving to the next.
    b. Consider a typical Windows-based workstation: At the core, you have a microprocessor of immense complexity with a defined interface (instruction set). Next is a layer of software running on top of the microprocessor that provides a "simple" interface to the operating system developer. This is called the Hardware Abstraction Layer (HAL) and it eliminates the need for the OS

developers to understand all the details of the microprocessor implementation. A microkernel then runs on top of the HAL and presents a relatively simple set of kernel calls to the operating system programmers. Thus, the OS programmer does not have to know all the details of the HAL. The OS then runs on top of the microkernel, hiding the kernel complexity and providing application developers with a simple system call interface. Applications run on top of the OS and provide the users with useful services without requiring the user to have any knowledge of the system call interface to the OS.

6. Abstraction
    a. An abstraction is a representation of an object or concept. It could be something such as a door, a speedometer, or a data structure in computer science. Abstraction decouples the design from the implementation. The gauges in an automobile are an abstraction of the performance of a car. A map is an abstraction of the earth.
    b. The goal in abstraction, from a computer security viewpoint is to remove any clutter that can distract and possibly be used in an incorrect way. Abstraction only provides the essential details of what is being modeled and provide the minimum information necessary to accomplish the task.
    c. Essentially, abstraction is about only providing the necessary details, and hiding all the "clutter" and reducing the details to a set of essential characteristics.

7. Information Hiding
    a. Information hiding is the technique that does not allow certain aspects of an object to be observed or accessed. Data and information hiding keeps the programmer from having complete access to data structures. It allows access to only what is necessary.
    b. In computer programming, manipulating a stack requires three operations. Push, pop and view the data item on the top of the stack. Information hiding allows the programmer to not be concerned with how the stack is implemented. The stack could be a linked list, tree structure, or an array. None of the details of how the stack is implemented are necessary.

8. Modularity
    a. Modular programming is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable

modules. Each module contains everything necessary to execute a unique part of the desired functionality through well designed interfaces. (Wikipedia.org) These well defined interfaces provide all the detail needed for one module to replace another and achieve the needed results.

b. In the 1980s and 1990s, desktop computers were more of a hobbyist effort. Computers could be modified to add hardware and increase performance. Memory could be added. There were different sound and video cards available. Disk drives came in different sizes. All of the components depended on having a well defined interface. Thus if a component from one manufacturer failed, it was easy to get another part from another manufacturer and replace it. Modularity allowed parts with the same interfaces to be interchangeable with others.

c. Current mobile phones are a good example of devices that are not modular. If a part breaks, the device will most likely have to be replaced. The Apple iPhone is a good example of this concept. If a hardware part breaks, it cannot easily be fixed.

9. Simplicity of Design

a. When designing a product, hardware of software, simplicity should be a goal. Simplicity is a design principle at Apple, especially with the iPhone. There is a single button. People understand how to use the device without having to read a manual. All unnecessary complexity has been removed.

b. Simplicity allows a person to better understand hardware and software. Without the clutter of unnecessarily complicated code and interfaces, the software will be more understandable by people that will update the code when requirements change. It will be easier to understand by the testers and they will be able to spot problems sooner. By keeping software as simple and as focused as possible, the reliability and security is greatly increased.