1. Domain Separation

- What is a Domain?
  - In a computer, this word refers to a *collection of data or instructions* that warrant protection. Outside of a computer, a domain can be an area of responsibility or control. My house is my domain. My school is another domain. My country is a domain.
  - Separating domains allows for enforcement of rules governing the entry and use of domains by entities outside the domain. Examples include the lock on the door to a house, a parking permit to park at school, and entering the country with a passport.
- Examples
  - Most computer processors run in two types of states (actually four or eight – depends on how you count) states. The supervisor domain (operating system) and the user domain (applications). The processor, when in the supervisor domain, can directly access memory (i.e. RAM) or manipulate access control tables in a primitive file system. When in the user domain, the processor cannot access memory that belongs to other programs or the operating system.
  - A virtual machine (or a container) is a domain that is separate from other virtual machines (or containers)
  - If you can understand and limit the domain, you have a better chance of protecting it
- Implications
  - Crossing domain boundaries requires access control. Transactional overhead and maintenance of access control rules are side-effects.
  - Stakeholders should negotiate the level of granularity at which domains require separation.
- Related principles
  - Process isolation - similar but distinct – fine yet important distinction between them
  - Resource encapsulation
  - Layering
  - Abstraction

2. Process Isolation

- What is a Process?
  - A process is a program running on a computer. Memory and data are also encompassed in the definition of process. A process could have multiple programs in it. Each process has a region of the memory (address space), which only it can access. Important to the definition of a process is the notion of state. Different users might be running the same program on the same computer – that program might be in the same memory. But each user has own registered state that gives them their own unique program, which makes the process different from each other.

    - Isolating the process address space from other address spaces prevents tampering or interference from/by other processes.
- Examples
    - A word processor, a database, and a browser running on a computer are all running in different addresses spaces. Process isolation ensures that each one cannot influence the others address space.
    - A non-technical example of process isolation is when a prosecutor and defense attorney run their cases in court. It would be a problem if either had access to each other's work. Keeping their work separate protects it from misuse by the other party. Other examples are two football teams protecting their own playbooks or two restaurant chains protecting their recipes.
- Implications
    - Processes have to use defined communications mediated by the operating system to communicate with other processes.
    - Processes work together so there has to be some trust between them. For example, when you get an email, the email is sent through many processes before you read it. What is important is validating the inputs from other processes.
    - Malware can act like a valid process and trick another process into performing a task. SQL Injection is an example.
- Related principles
    - Domain separation
    - Resource encapsulation

3. Resource Encapsulation

- What is a Resource?
    - A computer has many resources. A resource can be the memory, disk drive, network bandwidth, battery power, or a monitor. It can also be system objects such as shared memory or a linked list data structure.
- What is encapsulation?
    - Encapsulation -allows access or manipulation of the resource's data in only the ways the designer intended. Files – file system application is only thing that allows you to read them – files and directories might be better way to talk about encapsulation. A directory is a file that acts as a folder for other files. A directory can also contain other directories (subdirectories); a directory that contains another directory is called the parent directory of the directory it contains.
- Examples
    - The application logic of a website only allows access and manipulation of database records in defined ways. Here the database is a resource encapsulated by the website application logic.
    - Text messaging and email on phones run programs that allow access to the data and limited operations for those programs.
    - A flag pole only allows certain operations (raise the flag, lower the flag, unhook the flag). No one needs to know how the flag pole works internally, just that they

can use it only in certain ways. And for security, no one can change the operations or manipulate the data in unauthorized ways.

- Implications
  - Programs or users have to be aware of the interface for the resource and only communicate with it in a defined manner.
  - Programs or Users can abuse/take control of a resource and not allow the other programs/users gain access (starvation). What if a process is allowed to monopolize CPU time? Other important processes may not be able to function.
- Related principles
  - Domain separation
  - Process Isolation
  - Modularization
  - Abstraction
  - Simplicity (tension)

4. Least Privilege

- What is a privilege?
  - In a computer, a privilege is a right for the user to act on managed computer resources.
  - Also, programs might have a privilege. Some programs have privileges assigned to a specific user and when they run that program it is has those privileges.
- Why should privileges be minimized?
  - Minimizing the number of privileges granted to a user for accomplishing assigned duties improves accountability and limits accidental misuse. The operating system must also disable privileges when not required.
  - Also, what are the rules for permissions taken away or denied? What happens if a user has a database open and walks away? The database should timeout and close itself. If not, an unauthorized user could take a look which is also a violation of least privilege.
- Examples
  - When a person gets a new computer, they install or log onto it using an administrative account. This account has privileges to install software, add users, add hardware, and add and delete almost any program or file. Now, if the person opens a malicious phishing attachment, the malware will run with administrative privileges. Instead, if the person lowered their privileges to a regular user immediately after the initial installation is over, the malware would be less likely to acquire administrative privilege.
  - If a user doesn't need a permission, why give it to them? Should a military radio operator have permission to access a nuclear silo?
  - Ending an employee's email account after they leave the company so they no longer receive internal communications
  - Emergency Exit Doors – You can exit the door from inside. But an outsider cannot enter from that door.

    - Entering a bedroom gives you access to all the dresser drawers and closets. If you don't want someone in them, you need another lock.
    - Someone unlocks your phone and you gain access to all their applications.
    - The use of a sandbox to confine viruses/malware.
    - Room keys in the school – some teachers have their classroom and the faculty lounge key; some teachers also have a key to the gym; and the principal and janitor have the only master keys.
- Implications
    - The operating system may frequently prompt users to elevate their privilege for tasks that have high consequences, such as installing software or deleting a system file. Or you can't do what you need to do and you waste time.
- Related principles
    - Minimization
    - Simplicity (tension)

5. Layering

- What is a Layer?
    - In the context of computer security, a layer is a separate level that must be conquered by an attacker to breach a system.
    - Layering slows down an attacker. The attacker needs to conquer each layer before moving on to the next.
- Examples
    - A moat is an outer layer that protects a castle. The next layer that an intruder has to go through is the high walls. All of this has to be done by the intruder while avoiding the watchful guards. Finally, the intruder needs to scale the inner walls before getting to the king's treasure.
    - Firewall, intrusion detection systems, internal encryption, access control and personnel controls are examples of layers typically employed to protect enterprise data and programs.
- Implications
    - Multiple independent layers require separate management and integration to get the full benefits of layered protection.
    - Cost/Value Assessment. Do you spend $100 to protect $10?
- Related principles
    - Domain separation
    - Resource encapsulation
    - Abstraction
    - Data Hiding

6. Abstraction

- What is abstraction?

- Abstraction is the concept that something complicated can be thought of and represented more simply, as well with richness. All models are abstractions - since they reduce the complexity of an object into something that is understandable. Abstraction can also provide a richer presentation of simple things.
- How does abstraction contribute to cybersecurity?
    - Remove/reduce any clutter that can distract the user or programmer from using a resource correctly.
    - Only provide the necessary details, while reducing the complexity to a set of essential characteristics.
    - Excess complexity may hide malicious behaviors.
    - Allows solutions to be transferred to other contexts
- Examples
    - The gauges in a car are an abstraction of the car's performance.
    - A map is an abstraction of an area.
    - A model airplane is an abstraction of a real airplane and may be used to test aerodynamics.
    - Any formula is an abstraction of quantities and can be transferred to other contexts
- Implications
    - Different users/roles may need different abstractions.
    - The abstraction could be wrong. For example, GPS is often wrong and takes me to the wrong house. An abstraction could purposely lead someone in the wrong direction.
- Related principles
    - Layering
    - Data Hiding
    - Resource encapsulation

7. Data Hiding

- How does data hiding contribute to cybersecurity?
    - Only allow necessary aspects of a data structure or a record to be observed or accessed. Log all access attempts.
    - Cannot change data you are not authorized to change.
    - Can help prevent users/programmers/processes from updating/changing data in invalid ways or by mistake.
    -
- Examples
    - A stack data structure exposes only the data at the top of the stack using simple push and pop instructions. The operating system applies access control to different regions of the stack.
    - In OOP, the data is private and there are public methods to access the data (constructors, getters or setters). If a programmer/user needs to access the data, they need to call the getter or setter. That way the data can only be changed in the way it was intended.

    - Websites don't need to load all of a user's data to show a list of usernames - they only need the username, the rest of the record fields can be hidden.
- Implications
    - Can crash a system if data is changed in a way that the code is not expecting
    - Can update data to make the code do something it should not do for you
- Related principles
    - Resource encapsulation
    - Abstraction
    - Least Privilege

8. Modularity

- What is modularity?
    - Modularity is a design technique that separates the functionality of a program into independent components.
    - Each component/module is self-sufficient and capable of executing a unique part of the desired functionality through well-designed interfaces.
    - A component cannot be modular without a well-designed interface and an important aspect of the well designed interface is not just how you interact with it but that it has no or minimum side effects.
- Purpose is so a very complex system can be broken down into components so it can be more easily understood – thus better maintained and less likely to have errors that can exploited
- How does modularity contribute to cybersecurity?
    - One function per component is easier to test and maintain, therefore more secure.
    - Compartmentalization is possible using modularization. It contains damage to a single module.
    - Using modules means that you can swap out a bad part. If batteries weren't modules, any time a battery died you would need to throw out the entire electronic device it was in.
    - Modularity allows us to make sure that everyone is using the best implementation of a given function (bubble sort or cryptography)
- Examples
    - Electronic circuits. (separate components on a mother board)
    - Lego blocks.
    - Network nodes.
    - Car parts – a car is a big complicated machine but you can replace almost any part from the manufacturer or a third party supplier
- Implications
    - Well defined interfaces allow a system designer to replace one module with another.
    - Prevents vendor lock-in by writing a protocol that all vendors can follow.

    - Encourages re-use of well-designed modules by other modules. Rather than having several different parts of a system that require similar functions, the goal is to have one modular version.
- Related principles
    - Domain separation
    - Process isolation
    - Resource encapsulation
    - Abstraction

9. Simplicity

- How does simplicity contribute to cybersecurity?
    - The lack of complexity allows system designers and programmers to identify unwanted access paths.
    - Easier to understand, maintain and test so more secure.
    - Users can easily translate their general protection goals to appropriate system security configurations.
    - The simpler it is the less that can go wrong
- Examples
    - Interface designs that allow correct application of security features.
    - Intuitive and straightforward access control rules
    - Easy to follow and maintain program statements.
- Implications
    - Testers will be able to cover all possible combinations and spot problems sooner.
- Related principles
    - Abstraction
    - Minimization
    - Modularity
    - Layering

10. Minimization

- What is minimization?
    - Having the least functionality necessary in a program or device
    - Finding the shortest path through a network/maze
- How does minimization contribute to cybersecurity?
    - Decrease the number of ways in which attackers can exploit a program or device.
    - Time efficiency and space efficiency of algorithm design is minimization.
    - Minimizing the attack surface
    - Minimization decreases the opportunity to find an exploitable vulnerability in the system
- Examples
    - Reduce the amount of code. The goal is to implement the function in the least amount of code.
- Implications

    - Expanding feature sets in future versions can be difficult.
    - Reduce test combinations.
- Related principles
    - Simplicity
    - Abstraction
    - Least Privilege

This document is the product of a series of interviews with 6 people on the definition examples, and implications of the Cybersecurity First Principles. It was based on a list of the same developed by Dr. Matt Hale from the University of Nebraska Omaha. The other subject matter experts interviewed include: Dr. Matt Bishop, UC Davis; Dr. Suzanne Mello-Stark, Rhode Island College; Mark Emry, Washington High School; Steve LaFountain, National Security Agency; Grant Wagner, National Security Agency, Dr. Yesem Peker, Columbus State University.