Holistic Security

A few years ago, security was about keeping the bad guys out of your network. Network security relied extensively on perimeter defenses such as firewalls, demilitarized zones (DMZ) and bastion hosts to protect applications and data that were within the organization's network. These perimeter defenses are absolutely necessary and critical, but with globalization and the changing landscape in the way we do business today, wherein there is a need to allow access to our internal systems and applications, the boundaries that demarcated our internal systems and applications from the external ones are slowly thinning and vanishing. This warrants that the hosts (systems) on which our software run are even more closely guarded and secured. Having the need to open our networks and securely allow access now requires that our applications (software) are hardened as well, in addition to the network or perimeter security controls.

The need is for secure applications running on secure hosts (systems) in secure networks. The need is for holistic security, which is the first and foremost software security concept that one must be familiar with. It is pivotal to recognize that software is only as secure as the weakest link. In this day and age, software is rarely deployed as a stand-alone business application. It is often complex, running on host systems that are interconnected to several other systems on a network. A weakness (vulnerability) in any one of the layers may render all controls (safeguards and countermeasures) futile. The application, host and network must all be secured adequately and appropriately. For example, a Structured Query Language (SQL) injection vulnerability in the application can allow an attacker to be able to compromise the database server (host) and from the host, launch exploits that impact the entire network. Similarly an open port on the network can lead to the discovery and exploitation of unpatched host systems and vulnerabilities in applications. Secure software is characterized by the securing of applications, hosts and networks holistically, so there is no weak link, i.e., no Achilles Heel as depicted in *Figure 1.1*.

Implementation Challenges

Despite the recognition of the fact that the security of networks, systems and software is critical for the operations and sustainability of an organization or business, the computing ecosystem, today seems to be plagued with a plethora of insecure networks and systems and more particularly insecure software. In today's environment where software is rife with vulnerabilities, as is evident in full disclosure lists, bug tracking databases and hacking incident reports, software

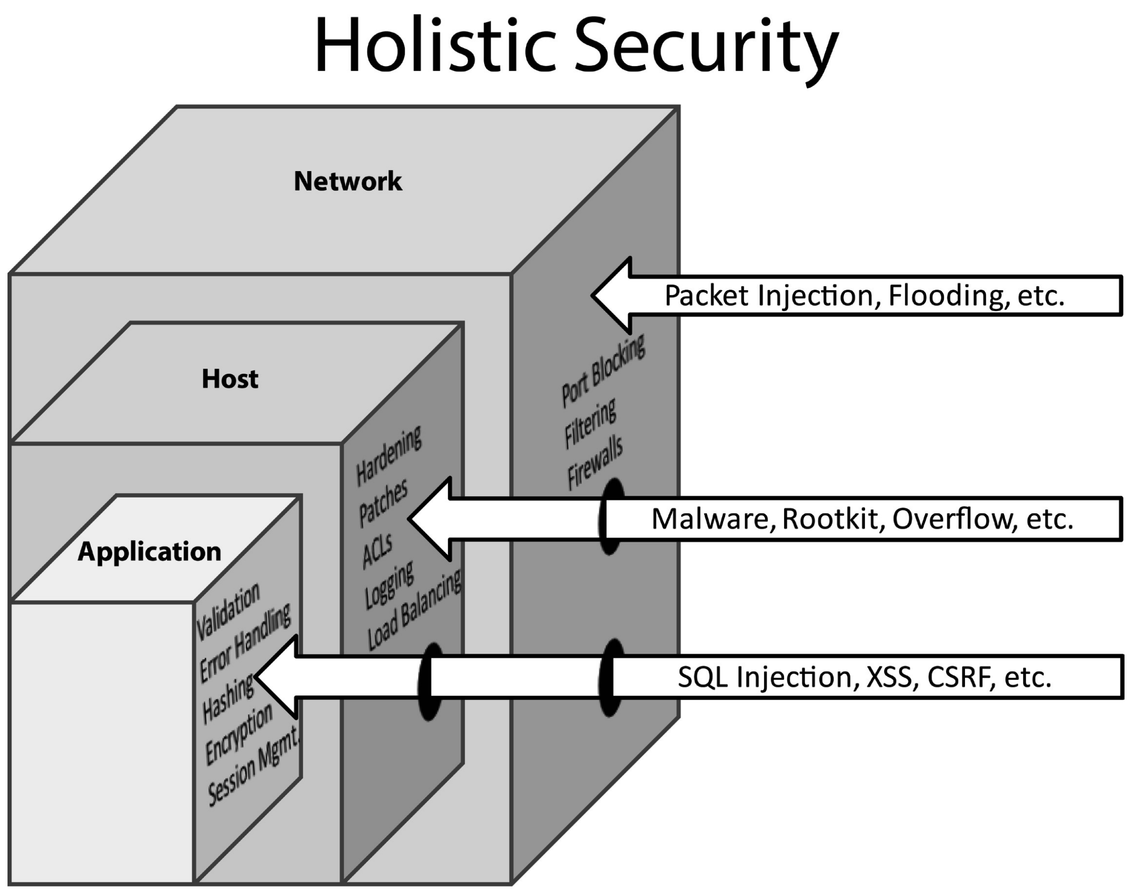

Figure 1.1 – **Securing the Network, Hosts and Application Layer**

security cannot be overlooked, but it is. Some of the primary reasons as to why there is a prevalence of insecure software may be attributed to the following –

- Iron Triangle Constraints
- Security as an Afterthought
- Security versus Usability

Iron Triangle Constraints

From the time a solution to solve a business problem using software is born to the time that, that solution is designed, developed and deployed, there is a need for time (schedule), resources (scope) and cost (budget). Resources (people) with appropriate skills and technical knowledge are not always readily available and are costly. The defender is expected to play vigilante 24x7, guarding against all attacks while being constrained to play by the rules of engagement, while the attacker has the upper hand since the attacker needs to be able to exploit just one weakness and can strike anytime without the need to have to play by the rules. Additionally, depending on your business model or type of organization, software development can involve many stakeholders. To say the least, software development in and of itself is a resource, schedule (time) and budget intensive process. Adding the need to incorporate security into the software is seen as having the need to do 'more' with what is already deemed 'less' or insufficient. Constraints in scope, schedule and budget, the components of the Iron Triangle as shown in *Figure 1.2*, are often the reasons why security requirements are left out of the software. If the software development project's scope, schedule (time), and budget are very rigidly defined (as is often the case), it gives little to no room to incorporate even the basic, let alone additional security requirements into the software and unfortunately what is typically overlooked are elements of software security.

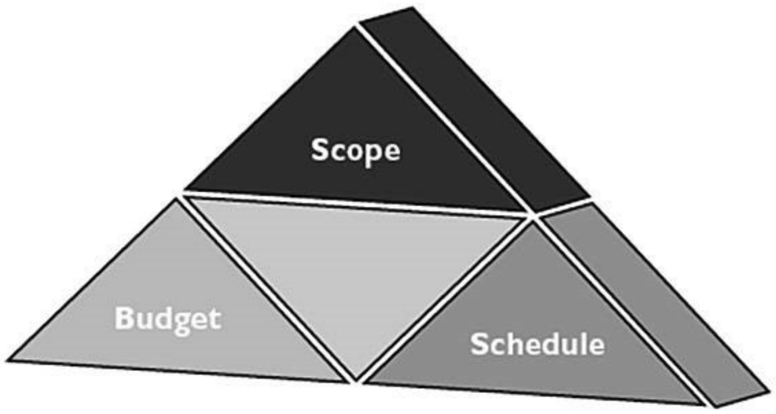

*Figure 1.2 - I***ron Triangle**

Security as an Afterthought

Developers and management tend to think that security does not add any business value since it is not very easy to show a one-to-one return on security investment. Iron triangle constraints often lead to add-on security, wherein secure features are bolted on and not built into the software. It is important that secure features are built into the software, instead of being added on at a later stage, since it has been proven that the cost to fix insecure software earlier in the software development life cycle (SDLC) is insignificant when compared to having the same issue addressed at a later stage of the SDLC, as depicted in *Figure 1.3*. Addressing vulnerabilities just before a product is released is very expensive.

Security vs. Usability

Another reason as to why it is a challenge to incorporate secure features in software is that the incorporation of secure features is viewed as rendering the software to become very complex, restrictive and unusable. For example, the human resources organization needs to be able to view payroll data of employees and the software development team has been asked to develop an intranet web application that the human resources personnel can access. When the software

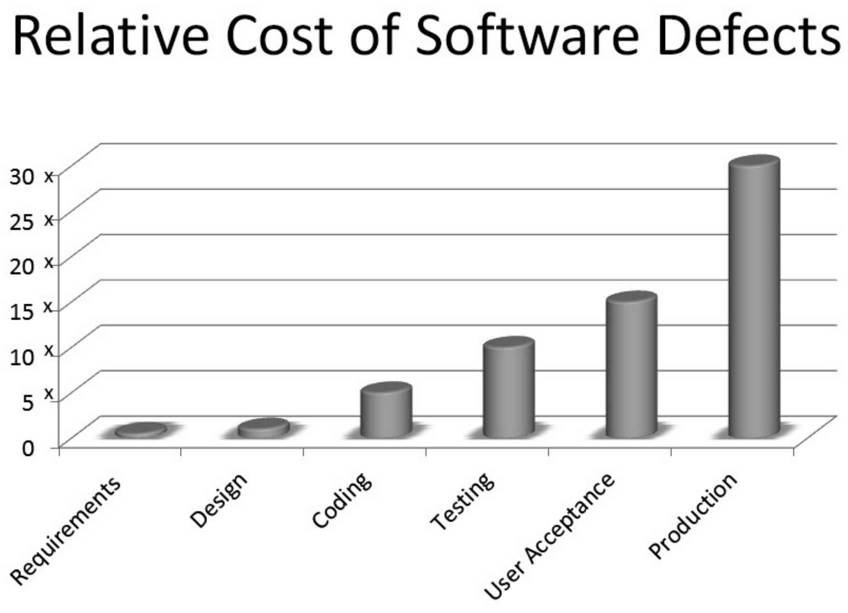

Figure 1.3 – **Relative cost of fixing code issues at different stages of the SDLC**

development team consults with the security consultant, who is a CSSLP, the security consultant recommends that such access should be granted to only those who are authenticated and authorized and that all access requests must be logged for review purposes. Furthermore, in the true sense of security, the security consultant advises the software team to ensure that the authentication requirements involve the use of passwords that are at least fifteen characters long, requires upper case and lower case characters, and has a mix of alpha-numeric and special characters, which will need to be reset every thirty days. Once designed and developed, this software is deployed for use by the human resources organization. It is quickly apparent that the human resources personnel are writing their complex passwords down on sticky notes and leaving them in insecure locations such as their desk drawers or in some cases, even their system monitors. They are also complaining that the software is not usable since it takes a lot of time for each access request to be processed, since all access requests are not only checked for authorization but also audited (logged). There is absolutely no doubt that the incorporation of security comes at the cost of performance and usability. This is true if the software design does not factor in the concept known as psychological acceptability. Software security must be balanced with usability and performance. We will be covering 'Psychological Acceptability' in detail along with many other design concepts in the Secure Software Design chapter.