**Module 2: Journal**

Zane Milo Deso

Southern New Hampshire University

CS-405: Secure Coding

**Journal**

**Introduction**

As a Computer Science undergraduate attending Southern New Hampshire University (SNHU), I have been tasked with writing a journal entry that explores and analyzes trade-offs within defense in depth (DiD) in software, identifies resource constraints, and expanding on the aspects of DiD that make it unique for any scenario.

**Defense in Depth (DiD)**

DiD describes the method of establishing multiple layers of defense so if one layer fails then the next layer ideally would catch the exploit, or bad actor. DiD initially was used by the Romans to organize their troops in such a manner to maintain the same idea as outlined above, except in their case the exploit and bad actor is an actual physical barrage.

As times modernized, all industries have adopted and polymorphed DiD to best serve their own use cases. For example, a web development company that also handles administration, security, and so forth, may have input validation for the fields on their web sites. They may also use static typing, obfuscation, encryption, and more to add layer upon layer of defense for each feature available.

**Trade-offs and Constraints of DiD**

As with anything, implementation of DiD takes time, costs money, and needs to be carried out by some person(s). This of course takes coordination, which will involve meetings, deadlines, shifting of deadlines, new constraints, changes to resources, and management of personnel. So, it shows that the work must be done up-front and continuously to effectively integrate DiD.

One of the main glaring aspects of this implementation is that guidance is required. A person(s) who has never touched a chisel cannot be expected to build an ark in any reasonable amount of time, or without strenuous effort. This alludes to requiring consultants, hiring of experts, or upfront time spent training key personnel to be able to do so. The other aspect that should be mentioned is some combination of the three.

So, if we examine the trade-offs that are required to carry out proper DiD, we can gather that some balance between time, money and personnel must be maintained. When outsourcing takes place, spending money on contracts and oversite, this costs money. In respect to training in-house, this will cost money and time, since a credible source or trainer will be necessary. Lastly, we will only ever have a certain number of personnel available to oversee, plan, implement, and maintain DiD, which will limit how we can leverage those team members in another area of production.

**Uniqueness of DiD in Alternate Circumstances**

As mentioned previously, DiD will need to be retrofitted to the use case in which it is guarding. Each industry will have legal regulations, ethical dilemmas and other novel situations that necessitate polymorphic DiD architecture. Healthcare, for example, requires extraneous care for patients' data thus the trade off to spend resources is more of a constraint that must be met. The overhead in each scenario needs to be thought of upfront for each situation due to such constraints.

**Conclusion**

In conclusion, DiD is not only necessary in most cases but may even be essential in certain use cases such as in the healthcare industry. DiD comes with the debt of forethought to plan for tradeoffs that will become roadblocks if not considered carefully or routinely. DiD comes with the weight of constraints, as with all things, planning, implementing, integrating, and finally maintaining uses limited resources such as time, money and personnel. With all of these areas considered, one can see how easy it may be to take it too far or naively fall into technical debt due to lack of planning upfront. DiD remains a tried-and-true method of security and defense transcendentally across domains if proper precautions are heeded.