2-1 Journal: Defense in Depth (DiD)

Ryan DeBraal

CS-405-X6389 Secure Coding 21EW6

**How deep is too deep, and what’s the tradeoff?**

The number of redundant layers of security in place should be proportional to the classification of the data being utilized. For example, credit card information and other proprietary/confidential data needs to be treated with the maximum amount of assurances for its safe storage, retrieval, and transfer.  
  
There isn’t a *specific* number of levels that define a secure data network, but it is important that data be validated on each tier of an n-tier application.   
  
For example, an application with a frontend, business logic layer, data access layer, and an underlying relational database should have validations on all four layers. These include but are not limited to: form validation, datatype range checks, code injection checks, and null exception checks.

The two main tradeoffs when it comes to DiD are complexity and performance.

It is very possible to code something so dense that it becomes esoteric and not easily understandable. As Martin Fowler, a respected software developer, says: “Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

The second trade off is performance. Even though we live in an age where memory and storage are cheap we also live in a world where thousands, even millions of transactions take place within an application every second. Executing these coded instructions have a cumulative effect on the speed of an application.

**What are some time, money, reputation, and operational considerations?**

A flaw in the security of an application can have tremendous costs in all three areas. Any time a flaw in an application is exploited and people’s private data is leaked it is headline making news.

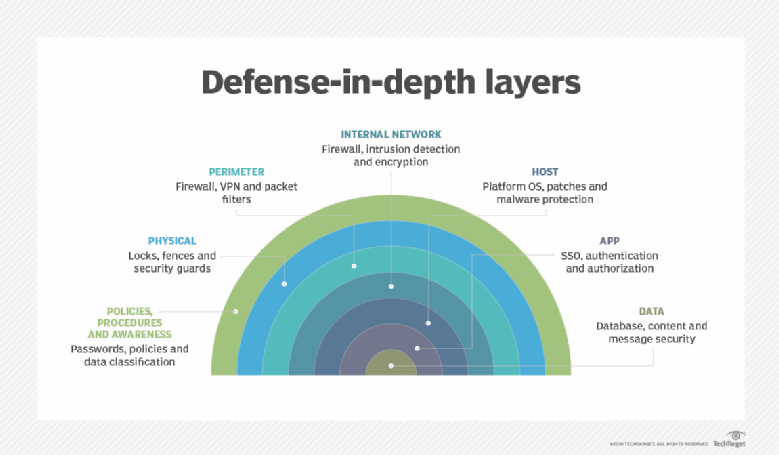
The time it takes to fix code, is time taken away from enhancing the code base. The reputational cost cannot be understated. A company’s reputation can be tarnished for years after a security incident. Finally, the monetary damages are often measured sometimes measured in the millions. For example, in 2013 a data breach in Yahoo! resulted in the exposure of over 3 billion records and an estimated loss of $350 million. (Perlroth, 2017)

**What are some additional aspects of DiD that make it unique for each situation?**

Each application needs to address its security differently. There is no ubiquitous way of protecting software from users with malicious intent. The strategy employed to protect an externally facing mobile application used for eCommerce will vary greatly from that of an internally facing data storage system used to retain restricted information.

The assumption when building an application should be that every layer of an application is vulnerable to exploit and therefore should be built with security in a proactive manner.   
  
Passwords should be long and inscrutable, physical locks should be in place around the hardware, firewalls should be running to help protect the network, encryption should be used when transferring data, fully patched operating systems should be running the latest malware protection, form validation should be used on the frontend, and multiple layers of redundant code checks should be in place on the backend and even in the stored procedures of the application.

Only then can an application we considered *somewhat* secure, and even then constant vigilance is required.

  
(Nolle, 2020)

**Citations**

Fowler, M. (1998, September). *Refactoring: Doing Design After the Program Runs*. Methods in Practice. <https://www.martinfowler.com/distributedComputing/refactoring.pdf>.

Nolle, T. (2020, December 18). The 4 rules of a microservices defense-in-depth strategy. SearchAppArchitecture. <https://searchapparchitecture.techtarget.com/tip/The-4-rules-of-a-microservices-defense-in-depth-strategy>.

Perlroth, N. (2017, October 3). All 3 Billion Yahoo Accounts Were Affected by 2013 Attack.  
The New York Times. <https://www.nytimes.com/2017/10/03/technology/yahoo-hack-3-billion-users.html>.