Chapter 3: Security Principles to Live By

1. Secure by design: security is built in to product
    a. Someone should be in charge of security – they say yes or no before any further steps are taken in product development
    b. All personnel should receive security training (MS example)
    c. Threat models should be used
    d. Design and code guidelines should be followed and *must* be enforced
    e. Fix all bugs ASAP
    f. Guidelines should evolve as new threats are identified and new methods of mitigating threats are developed
    g. Regression testing should be an integral part of the process (Agile methods have this built in via TDD)
    h. Simplify code (once again, Agile)
    i. Penetration test: have people try and break code (break in) – carefully choose who does this – novices won't do a good job of pointing out vulnerabilities
2. Secure by default: product should be secure out of the box (Win 2K was *NOT*)
    a. Turn of unneeded capabilities in default configuration
    b. Use principle of least privilege
    c. Make sure sensitive data is protected (ACLs, encryption, different location, etc.)
3. Secure in Deployment: system is maintainable in a secure fashion once deployed
    a. Security functionality should be easy to administer via app
    b. Good, quality patches should be produced and distributed ASAP
    c. Documentation should be provided on proper use of system
4. Security Principles: everyone should be aware of them and actively incorporate and promote them
    a. Learn from mistakes (there is only one thing more painful than learning from experience and that is not learning from experience)
        i. Approach every bug as a learning opportunity
        ii. If you don't learn from mistake, you will make it again
    b. Minimize attack surface (depending on what your app does):
        i. Limit number of open sockets (MS example)
        ii. Limit number of open named pipes
        iii. Limit number of RPC endpoints
        iv. Number of services
        v. Number of services running by default
        vi. Number of services running with elevated privileges
        vii. Number of dynamic-content web pages
        viii. Number of accounts in an admin group
        ix. Number of files with weak ACLs
    c. Employ secure defaults: choose appropriate features for base set of users and make sure they are secure
        i. A feature that is not running is not vulnerable to attack
        ii. Side note: more features means more memory use – perf hit

d. Employ defense in depth: multiple security features should be used to dissuade an attacker – depending on your app, these features will vary
e. Use least privilege: use only the privileges needed to get job done
    i. List resources that must be accessed when developing app and determine privileges required for each
    ii. Try and avoid running as admin as much as possible
    iii. Side note: don't run your machine as admin (can use "Run As…" on XP)
f. Avoid backward compatibility whenever possible (this has haunted MS for years)
g. Assume external systems are insecure: never trust outside data
h. Plan on Failure: fail securely
    i. Disclose as little information as possible
    ii. Accept on the things you know are good – all else is bad
    iii. Nice document on failing securely: http://web.mit.edu/Saltzer/www/publications/protection/
i. Remember that Security Features != Secure Features
j. Do not depend on security through obscurity: assume attacker can view your code
k. Don't mix code and data – it is trivial for an attacker to find the data
l. Fix security issues correctly and look for more of the same elsewhere in the code (Agile can help avoid this) – cure the problem not the symptoms