

Security within a development lifecycle

Enhancing product security through development process improvement



Who I am

- Working within a QA environment, with a focus on security for 10 years
- Primarily web based and e-commerce based applications
- Worked for financial institutions, operating system manufacturers, security companies
- A 'hacker' (what does this mean? Who should hire hackers?)

Who you are?

- How many people work within the development cycle in a corporation (internal or external applications)
- How many of you are customers for applications that store and/or serve sensitive data
- How many people work on projects or products that use databases to store potentially sensitive information?

To be covered

- Being a discerning customer, demanding quality software from vendors
- When to find security issues
- The Dev team and its roles
- How to find them
- Documenting/analysing them
- Policy questions
- Legal Issues
- Database Concerns

What I want you to get, what I want

- As a customer, a way of articulating your demands for secure software, feeling assured that your vendor is taking your security concerns seriously
- As a developer, a realization and help to incorporate security commitments and responsibilities into your development cycle, visibly increasing product security
- Personally, I want the world's software environment to be more secure, so I can live in a more secure world.

Why Security during development?

Same reasons a company would care about security at all

- Reputation (worked for companies that lost sleep due to 'brand tarnishing' nightmares)
- Loss of customers
- security issues cost more than traditional bugs
- Liability and other legal issues
- Not contributing to the problem

Buying 'secure' software

- Are there outside audits? (are you willing to pay for them, or negotiate against product price?). When were the audits done?
- What are there internal security evaluations? Are they entirely positive and mention no potential vulnerabilities, or are they.
- Can you afford it?
- As a development house all your dependencies should go through a security evaluation phase

Politics, Money & Security

- Politics - Ownership of security responsibility should be outside of the development organization
- Money - If you can't make a business case for a security issue, then you can't make any case for it.

When/Where to find security issues?

- As early as possible!
- Business Model
- Requirements phase
- Architectural Documents
- Product Specifications
- Implementation bugs in product
- Other products

Understanding bug severity

- Level of knowledge needed (deep product knowledge, will the manual give needed knowledge, looking at config files?)
- Level of access needed (user level account on machine, any external web user, physical access to machine, physical access of network)
- What is compromised (DoS, financial data)
- Scenario of intruder

Development team and security

- Everyone has a role to play
- Doc – doc best practices, known issues (see disclosure discussion), db access rights, OS access requirements
- PM – security specification review (or assignment of such), security requirements, security release requirements
- Operations (if applicable) – give input and feedback during dev, determine security concerns and/or develop workarounds. May assist in detecting known vulnerabilities as well (custom IDS signatures, for example)
- Dev – write secure code, fix bad code, improve & follow best coding practices
- QA – Develop & Execute sec test cases, work on generalizing bugs into best practices document
- Having security advocates and experts in any of these roles will be beneficial to the security of your product

Hiring hackers

- Conventional wisdom is that financial/security/military does not hire hackers
- Only a fool would hire someone because they call themselves a hacker
- What is the impact of NOT having some people that continually think of the system as something to break?

Database Security Issues

- Connection (what is auth type, how is auth info stored?)
- Documentation – are database minimal permission levels documented
- cryptographic protection of records. Not simplistic, potentially very effective (go ahead, steal my database, little good will it do you). Potential limiter of liability and disclosure laws (CA, USA in particular)
- Database Auditing
- Think of the database as your ‘heart of gold’ at the end of your security rainbow. Important part of your ‘security onion’ (layered security model).
- Application level auditing (do you want it, do you want to implement tamper evident logging?)
- May be useful for threat modeling to scenario plan database attacks, poisonings, and malicious modifications

Security & User Interface

- Often overlooked
- Default values
- Understandability of settings

Typical Bugs and reactions

- Application authentication credentials sitting plaintext on machine – “but he can’t get into the box”, “but if he owns the box, it’s all over anyways!”
- Authentication credentials cached plaintext on client machine, “it’s his responsibility to secure the machine”
- Sensitive data plaintext in database, “if they hack the machine, it’s all over anyways”

Internal exploit code

- Automated tools may end up exploit code
- Should be unnecessary to write

Internal data handling

- Until fixed, security bugs are very sensitive information
- If your bug tracking software is not secure it may be wise to track security issues outside of your normal communication channels
- Need to know good, too small need to know very bad

Legal Implications

- Failure to disclose
- Government disclosure regulations
- 3rd party auditor deception issues
- False representation/false advertising issues
- Fitness of product laws

Policy questions

- Whose call is it to fix security issues
- What is the disclosure policy (to customers, to internal parties)

Who to tell: the disclosure question

- Vendor disclosure doesn't seem common today
- Either ship software without known security issues (possible, costly, harder to do this with a 'hard ship date') or give your customer software with known issues
- If you decide to disclose to a customer, limit the internal distribution of that data.
- If you honestly/competently self-audit, there will be a disclosure question

Implement best coding practices

- Great developers are not automatically great security developers
- Many times security functionality is being implemented without thought or knowledge
- All issues found should be transferred into this document, preventing future mistakes
- Educate your developers regarding security
- Do not implement homegrown crypto
- Do not trust user input
- Leave no plaintext confidential data –anywhere
- Be careful when components interact
- Disallow privilege escalation
- Audit appropriately, safely

References & Additional Reading

- Cost of software Bugs - NIST Study
<http://www.nist.gov/director/prog-ofc/report02-3.pdf>.
- Cost of when bugs are found -
<http://www.cutter.com/research/2000/crb000111.html>
- Usability in Privacy applications -<http://www.blackhat.com/presentations/bh-europe-03/bh-europe-03-sassaman.pdf>
- Liability Issue: Search for Stephen Wu at
<http://cyberlaw.stanford.edu/blogs/>
- Web Application security best practices guide-
<http://www.owasp.org/documentation/guide>
- Database security - Honeytoken paper -
<http://www.securityfocus.com/infocus/1713>