

Software Security

Where are today's development projects going wrong?
by Daniel Cuthbert



Have you tested your software for bugs? Scoured it for vulnerabilities of every shape and form and then eliminated them? If not, then your product is not ready to ship!!!

(Steven B. Lipner, Senior Director of security engineering strategy at Microsoft)

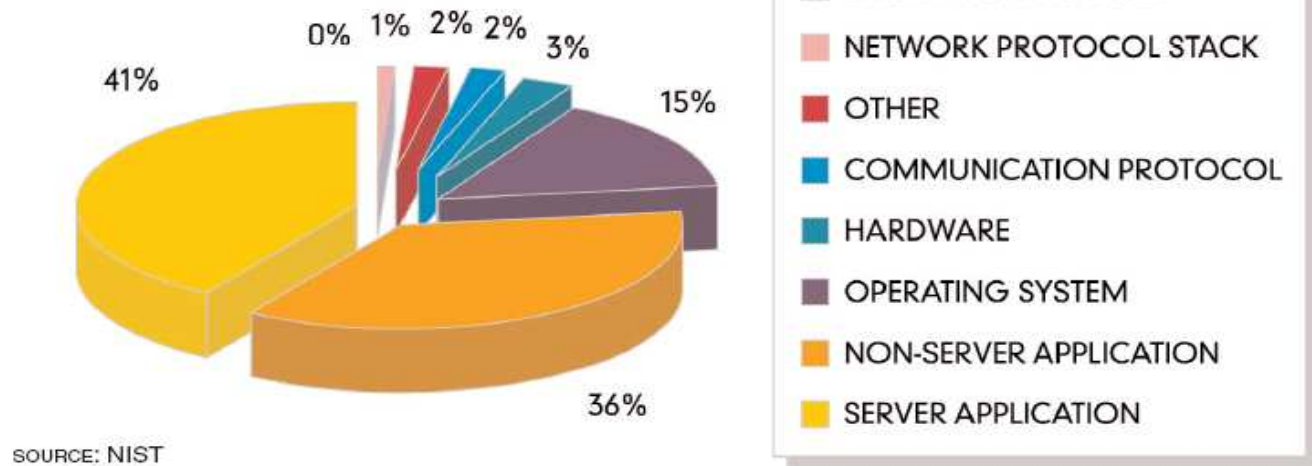
About me

- One of the OWASP Leaders
- Headed up the OWASP Testing Guide
- Principal Consultant for Corsaire

The hard facts

- How Many Vulnerabilities Are Application Security Related?

92% of reported vulnerabilities are in applications, not networks



The band-aid approach

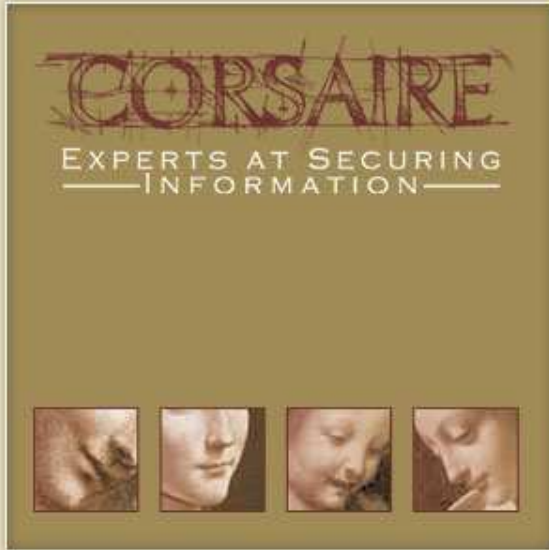
- Effective security requires not only prioritising security at the beginning of a project, but also increasing its visibility throughout the lifecycle of the project.
- Application firewalls DO NOT SOLVE THE ISSUE!
- Solving security issues at the end of the QA period is much more expensive.
- Developers and architects need to design and understand secure coding techniques and how attackers exploit applications.

Microsoft's approach

- Microsoft applied a dramatic security overhaul when developing Windows 2003.
- Development was stopped and all focus was shifted to security.
- What followed was two months of:
 - Threat Modeling
 - Risk Analysis
 - Penetration Testing
 - Source Code Reviews
 - Production not resuming until all of the bugs were found and eliminated.

Why security fails on projects

- Increase of programmer's job scope:
 - Traditional programming discipline has been driven by the challenge of making products optimally efficient.
 - Developers have been focused on providing robust functionality to their customers.
 - Developers often learn insecure programming methods during their education phase.
 - Attackers leverage this way of development to perform their malicious activity.
- Separation of security as a new domain discipline:
 - Security isn't a new problem, it's an old problem that now has global media coverage.



Corsaire Application Trends Research



The state of the industry today

- The results from a representative sample of application security assessments conducted over the last 12 months have been collated to analyse the prominent issues identified, and how application security has developed as a result of application security programmes being introduced.
- Corsaire Application Security Assessments are typically performed on the following interdependent areas from a security standpoint:
 - Authentication and Authorisation
 - Session Management
 - Data Validation
 - Data Transport Security
 - Presentation Tier

General findings

Description of Finding	Application Exhibiting Finding (%)
Flawed authentication mechanisms	91%
Weak access control (authorisation) methods	45%
Insecure session management	73%
Inadequate data validation	95%
Insufficient data transport security	95%
Poor error handling	95%
Application denial of service	41%

Authentication and Authorisation

- Authentication and authorisation:
 - In over two thirds (68%) of the tested applications, the password format in use was weak, increasing the risk of password guessing or brute-force attacks.
 - The risk of unauthorised access to user's accounts was increased by the fact that in 41% of applications it was possible to enumerate valid usernames through the error messages returned, and in nearly one third of cases there were no mechanisms to lockout accounts or IP addresses involved in brute-force attacks, or escalating timeouts to mitigate them.
 - In just under 10% of the applications, the HTTP Basic Auth schema was used for authentication – a mechanism which is generally recognised to be inadequate from a security perspective for most modern applications.

Session Management

- In nearly three quarters (73%) of cases, weaknesses in session management or the session tokens themselves exposed the application to attack.
- In nearly a quarter (23%), it was possible to manipulate the session management.
- In 45% of the applications, timeouts were either not implemented or could be overridden by a malicious client to assume the identity of another user within the system.

Data Validation

- Whilst data validation problems were identified in nearly all (95%) of the applications, in a third of cases the inadequacies resulted in un-exploitable application errors or other minor problems.
- However, nearly two-thirds were susceptible to temporary or persistent Cross Site Scripting (XSS) errors and 27% of cases were susceptible to SQL Database Injection attacks, critically exposing the application and data.

Data Transport Security

- Cache-controls were inadequate in just under half (45%) of the tested applications, potentially exposing user or session data – particularly where shared terminals were used (such as libraries or web-café).
- Mechanisms for enforcing encryption at all times were not present.
- In all cases, data was (or could be) passed unencrypted in 55% to 68% of the tested applications.

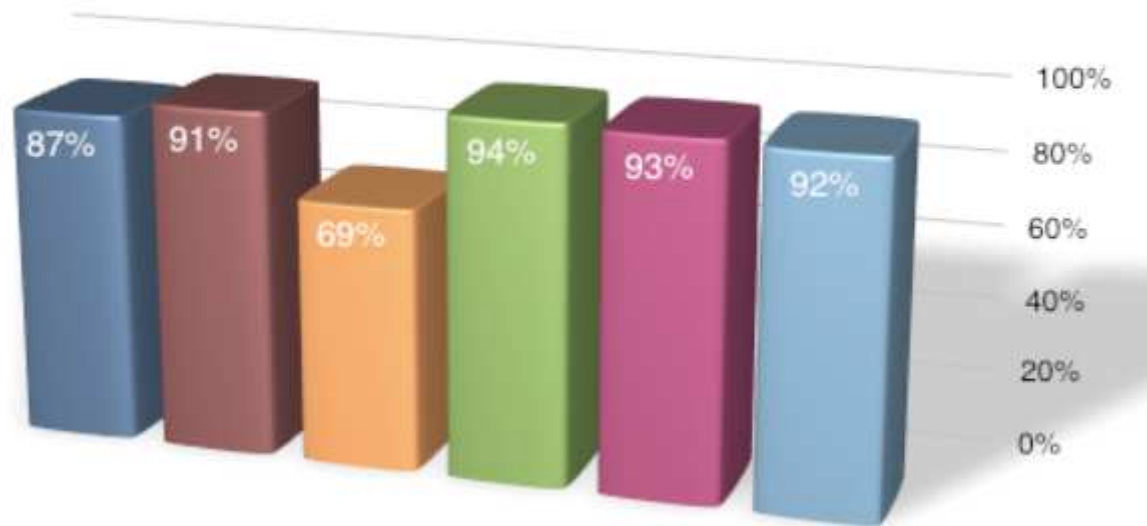
Presentation Tier

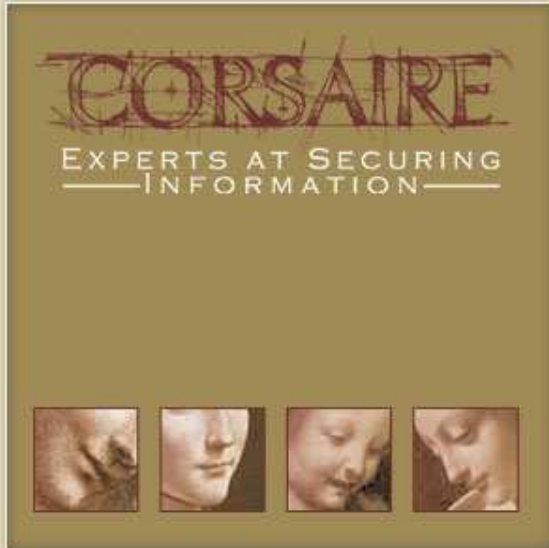
- In almost all of the applications (95%) a variety of un-trapped error messages were returned, allowing an attacker to enumerate the type and structure both of the web-application and the presentation platform.
- The types of error returned included script and server messages, and in 23% of cases potentially sensitive data was returned.
- This included debug information about how the application processed and filtered the input, or internal server data, such as version or patch levels, physical paths or internal IP addresses.

Something isn't working is it?



■ Authentication ■ Session Management ■ Authorisation
■ Data Validation ■ Data Transport Security ■ Presentation Tier



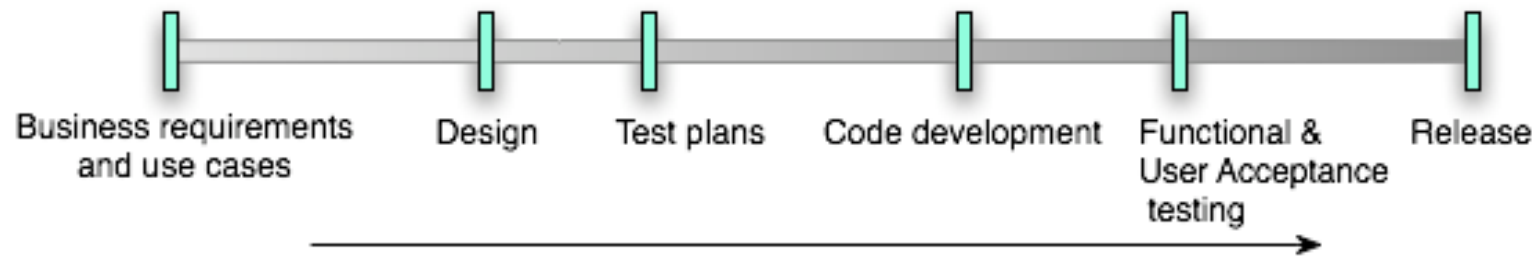


Defining Secure Development

What is secure development

- In order to mitigate the risk of attack through the bespoke applications in an environment, it is vital both to build secure applications and regularly validate their security through testing.
- Secure development is the term largely associated with the process of producing reliable, stable, bug and vulnerability free software.
- There are a number of ways that this can be undertaken within traditional application development but the most common procedures involve phased security assessments and reviews that encompass knowledge share; design and implementation assessment and regular security health checks.

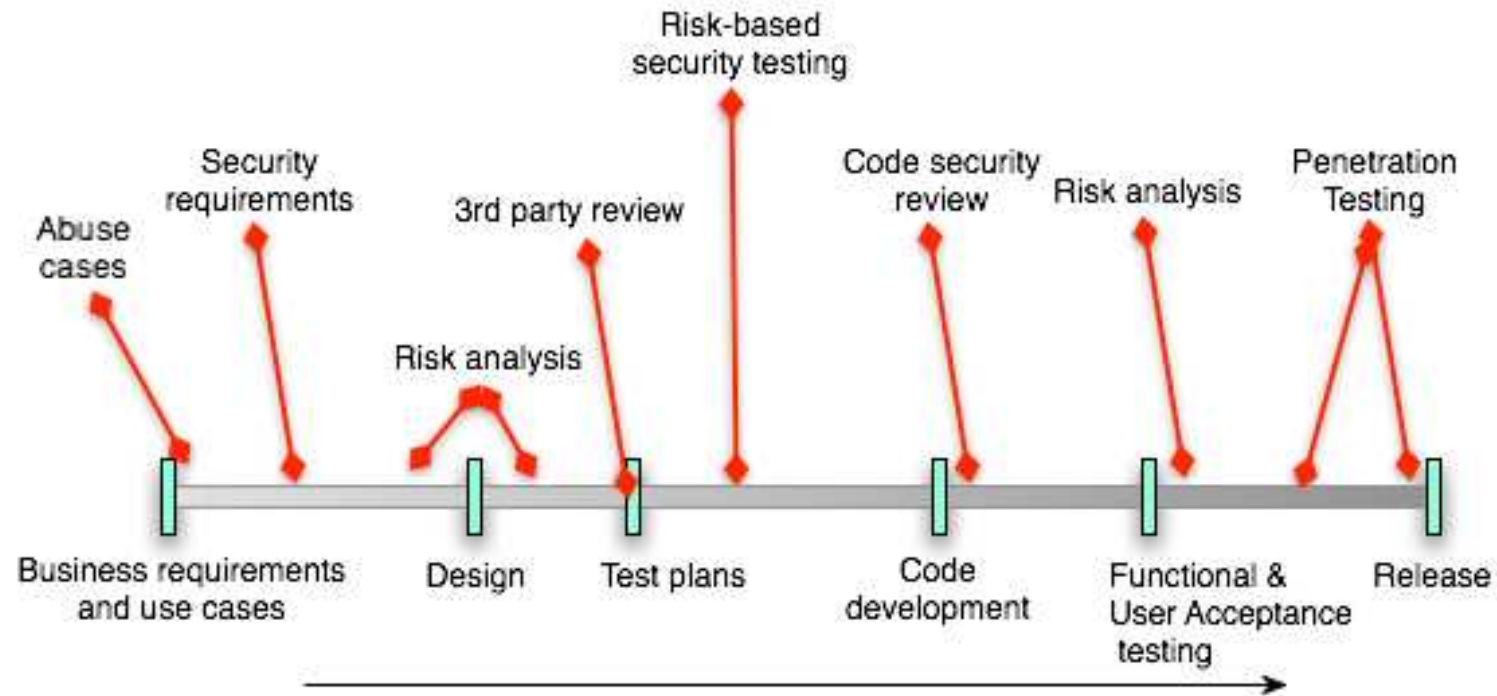
Typical development lifecycle



Current development lifecycle problems

- Security is considered at the last stage of most projects.
- Any assessment on the application, or code, is normally performed days before the intended go live date, giving hardly any time to solve issues.
- Only 30% of possible security vulnerabilities can be discovered this way.
- There is no stage to include the web application security standards, if any, to the project design.
- Developers are not made aware of any insecure practices they are doing.

Secure Development Lifecycle (SDLC)



Introducing security into the SDLC

- Security is included as part of the design and implementation phase.
- Each stage now has to include some form of security testing or thought process.
- Developers are constantly thinking of how the code/application could be manipulated.
- The end result is an application which has far fewer security vulnerabilities than previous efforts.

Successful implementation

- A Secure Development Programme should be integrated with all phases of the organisation's software development lifecycle.
- It ensures that security is a consideration at all stages of a development project – from risk analysis of the business objectives through design and implementation to deployment in production environments.
- Secure Development involves the systematic analysis of the security controls already in place in the organisation's lifecycle.
- It will typically involve the integration of phased security workshops, reviews and assessments during the development process.

Drawing on expertise

- In order to successfully integrate security to the development process a comprehensive understanding of the potential issues and failures is required, together with intrinsic knowledge of the existing development processes.
- The objectives of quality assurance and user acceptance testing are often at odds with those of security testing.
- Acceptance testing is performed with a mind to ensuring that the application behaves as it was designed to, from the point of view of a user.
- Tests from the point of view of a user with malicious intent, are often not carried out and form the core of the security testing mindset.

Understanding threats

- New threats need to be identified and mitigated all the time.
- Different types of businesses will be more sensitive to different threats and will have different security goals to mitigate those threats.
- Understanding threats is important in determining a systems security goals.

Agreeing deliverables

- As with all risk management, development of secure applications requires a delicate balance between investment and tangible return.
- Applications *can* be secured before, during and after development but re-architecting to remove serious flaws discovered in production systems may be prohibitively expensive.

Security architecture review

- The Security Architecture Review is a logical review of the high-level designs and processes associated with the project and how it integrates with the existing environment and third parties.
- Through this phase, process design flaws and intrinsic security risks are identified and presented to the business in order to find resolutions prior to implementation.
- Systems and software architects are primarily concerned with the functionality of the application and whether or not it meets the business needs.
- Security considerations, and particularly, defending data from users with malicious intent is often overlooked.

Security workshops and review

- The next phase occurs during the technical specification development phase and consists both of logical reviews of designs and interactive workshops with the project stakeholders.
- Potential security issues and design flaws can be engineered out of the technical design specification prior to the costly implementation phase.
- The workshops also raise security awareness in the project team prior to decisions being made which may be difficult to reverse later.

Secure programming workshop

- Early in the implementation phase, secure programming workshops provide the developers with specific advice relating to typical insecure implementation practice and common flaws.
- These workshops are tailored to the specific project and environment, e.g. the type of software being developed and the development technologies.
- Workshops include discussion on defining the data sets and common security libraries and routines that will be used by the application.
- Possible attack scenarios and areas of risk within the application are identified and mitigation techniques planned.

Application security assessments

- An Application Security Assessment is designed to identify and assess threats to the organisation through bespoke, proprietary applications or systems.
- These applications may provide interactive access to potentially sensitive materials, for example. It is vital that they be assessed to ensure that, firstly, the application doesn't expose the underlying servers and software to attack, and secondly that a malicious user cannot access, modify or destroy data or services within the system.
- Even in a well-deployed and secured infrastructure, a weak application can expose the organisation's crown jewels to unacceptable risk.

Transforming organisational attitudes

- Prepare Software Managers for the program. They should appreciate the value of the process and understand the importance of allocating time in the schedules for the software developments team's participation in certain aspects of the phased security assessments.
- Build phased security assessments into the project schedule and remember to factor in time for the inevitable rework that will follow each phase.
- Inform the participants and if appropriate your customer on the benefits of secure development implementation.
- Encourage a team culture of mutual respect; the process of secure development should be seen as non-threatening.
- Have a local champion who preaches the merits of a security development program, trains others as they get started, and strives to improve the overall software development process.

Choosing your secure development partner

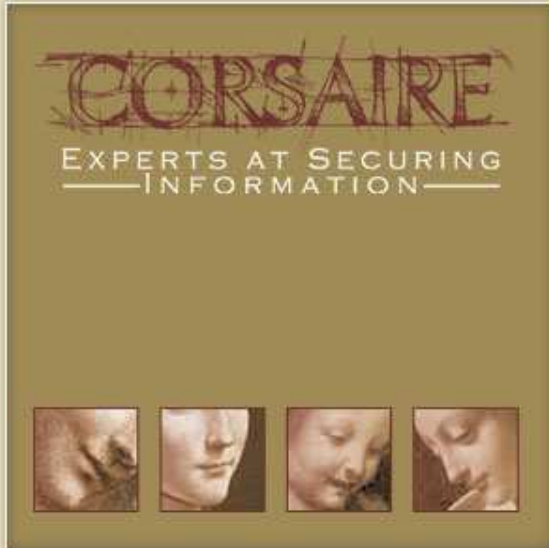
- When considering a security supplier, first outline your goals for the project and your expectations of the supplier.
- You will want to choose a supplier with experience in your industry and one that has all the expertise you require.
- Speak to at least two references; both those that have worked with the supplier and those that are still working with the supplier.
- In order to assist in your secure development program, the supplier must have experience of integrating security controls with organisations existing development processes.
- They must have extensive experience of assessing and auditing application security at all levels – from design and architecture through prototyping and coding to analysis of full production environments.

Education and awareness

- Educating developers about current secure development guidelines and good practice:
 - OWASP Guide to building secure web applications.
 - IEEE P1074 Standard for Developing Project Lifecycle Processes.
- Ensuring developers are familiar with ISO 15408 Common Criteria Threat Categories:
 - Allowing for research into latest vulnerabilities and why they happened.
 - Allowing developers to give input on any possible issues found during the development process.
- Making everyone aware that security is no longer a add-on:
 - Using security as one of the selling points of the application/product.

SDLC training

- Last 24 months has seen an influx of “secure development” courses enter the market.
- Ensure you fully understand what is being offered and also how it fits into your companies development lifecycle.
- The trainers qualifications are paramount.



Web 2.0 SDLC Case Study

Web 2.0 SDLC case study

- You are in charge of your companies Web 2.0 application push.

What is Web 2.0

- Web 2.0 is billed as an improved WWW (as opposed to the web 1.0).
- Key principles of Web 2.0 Applications;
 - The web as the platform
 - Data as the driving force
 - Users owning the data on the site and the control over that data
 - A rich, interactive, user-friendly interface (AJAX)
 - More acronyms than ever before!!!

Understanding threats with Web 2.0

- Web 2.0 lacks precise definition, often used as a term for websites which aren't static html sites.
 - Web 2.0 sites are more interactive, allowing users more freedom than traditional applications.
 - AJAX allows this interaction but at the same time opens up the application for attack vectors.
 - AJAX documentation* includes errors, so developers are learning bad practice before any code has been written.
 - New development, old mistakes.
-
- *Foundations of AJAX included various errors in the sample code provides.

Web 2.0 attack vectors

- Cross Site Scripting (XSS)
- Race Conditions
- Insecure Randomness
- Poor Error Handling
- XML Poisoning
- Malicious AJAX code execution
- WSDL Scanning
- SOAP parameter manipulation
- Modification of data and data streams

Designing and deploying requirements

- Close attention paid to how systems interact with their environment.
- Developers must write about what the systems must do and also must not do.
- Use cases need to include misuse cases.
- Design aspects need to include methods of reducing the applications attack surface.
- Implementation needs to have correct error handling methods, avoiding dangerous code constructs, a strong implementation of validation and encryption.

Security requirements

- Document key security objectives:
 - e.g.: systems should disallow unauthorised access
 - Password mechanism should enforce strong passwords
 - Password mechanism should time out after 3 failed attempts
 - Password generation mechanism should use randomly generated passwords, which are harder to brute force
- For every Use Case, write a Misuse Case:
 - Use cases describe legitimate users and how they interact, misuse cases describe attackers and how they misuse the system.

Risk analysis

- Threat Modeling is an engineering technique used to describe threats to the software application.
- A well implemented threat model will profile the assets that need to be protected, the attacks that could be used to realise these threats and the conditions under which the attack could be successful.
- Example of a threat of a malicious user being able to gain extra credit on their account:
 - Threat: A malicious user is able to gain extra credit on their account.
 - Attack: User sets the `_Credit` field to a higher value.
 - Condition: Server-side validation isn't present on the field.

Code development

- All developers should adhere to rigid code standards.
- Obtaining the standards and keeping them updated, and developers updated on the latest techniques is paramount.
- Coding standards for security should include:
 - Handling of strings and integers.
 - Input validation and output validation.
 - Handling of temporary files.
 - Secure authentication and authorisation methods.
 - Proper error handling.

Code review

- All code developed should be security reviewed by experienced code reviewers.
- Key objectives of a code review are:
 - Design goals are being met.
 - Security objectives are being met.
 - Implementation is robust.
- Careful attention should be paid attention to:
 - Hard coded secrets.
 - Injection (XML/SOAP).
 - Information disclosure.
 - XSS.
 - Input validation weaknesses.

Testing and deployment phase

- Looking for what is not there.
- Security testing is the process of thinking like an attacker and trying to exploit the application.

It is possible.....

CORSAIRE

Questions?

CORSAIRE