



Link Mountain, LLC PO Box 182 Port Sanilac Michigan, 48469

---

## Link Mountain Secure Coding Training for Developers

Link Mountain developer training is intended to insure that developers come away with a full understanding of basic application security best practices, and is based on the current OWASP top ten, although it does not necessarily teach them in order. It also includes quite a bit of information not included in the OWASP top ten, and the training is organized to flow in a 'teachable' manner.

### Passive Recon

A brief treatment of passive reconnaissance activities and how these activities can be used by attackers to gain information about target applications and networks. Whois queries, Search engine hacks and other web resources are discussed, with emphasis on harvesting of potential user names and information that could aid in dictionary attacks, phishing and social engineering attacks as well as cached error messages and other information accessible through passive recon techniques. The goal is twofold:

1. Insure that developers understand how much information can be gained, how it can be used and the advantages it gives to an attacker.
2. Insure that developers understand what measures they can take to limit the amount and value of information available through passive recon.

### Information Disclosure – Robots.txt, Listable directories, Hidden HTML fields, Comments, etc

Developers are taught how attackers further their knowledge of an application by probing for Robots.txt files and listable directories, examining hidden fields, javascript and html comments, web server banners and other sources of information before launching attacks against applications.

### Session Handling

Discussion includes the dangers of home grown session tokens, use of session tokens in GET requests, mixed use of encrypted and unencrypted transport and other session related issues.

### Encryption

Discussion includes the difference between obfuscation and encryption, the danger of using home grown encryption schemes or simple encoding schemes. Discussion includes choosing appropriate transport encryption, either SSLv3 or TLSv1, and ensuring that only high strength ciphers are allowed to be negotiated.

### Authentication – Logon Logic, Password Recovery, Account Creation

Various common faults in logon logic are addressed. Brute force and dictionary attacks are discussed. Account lockout functionality, password recovery and account creation logic are discussed as common areas for errors, weakness and disclosure of user accounts.

### Access Controls – Logical Faults, Path Traversal, Predictability, etc

We discuss the difference between authentication and access control. Common dangerous assumptions when handling access control between application boundaries. File system access control is discussed – traversal and canonical attacks are examined. Discussion of how error conditions can impact access control function.

### Input Validation Issues – SQL Injection, Command Injection, etc

We discuss the need for input validation, and identify the many vectors for input into the application – including header names and values, dynamic parameter names and values, cookie parameter pairs, etc. We discuss the importance of well considered architecture for validation functions. We focus first on the dangers of SQL and command injection, and end with an introduction to reflection that carries us into the next segment.



Link Mountain, LLC PO Box 182 Port Sanilac Michigan, 48469

---

### **Reflection Issues – XSS, Response Splitting, etc**

We engage in a thorough discussion of reflection and the different attack opportunities that can be presented. Discussion includes persistent and non persistent reflection, and reflection occurring in headers, html tags, scripts, xml documents, cookies and even active X controls and flash content. The challenges of sanitation and validation are discussed in light of the many vectors for reflection. We end this segment by ensuring that developers are taught to recognize 'fuzzing' techniques in application logs.

### **Error Handling**

By this point in the training, every dev should be familiar with fuzzing techniques and should recognize how easily an entire application can be probed for untrapped error conditions, and should understand that error pages can be cached by search engines retained indefinitely. We discuss the danger of stack traces in terms of information disclosure and vulnerability research. We discuss the importance of well considered architecture for error handling functions.

### **Language and Architecture Specific**

Training concludes with language and architecture specific details that are appropriate to the environment. Buffer overflows are discussed where either native code or libraries could be vulnerable, inheritance, state management, serialization and other issues are discussed where relevant to the code base in use.