6 PRINCIPALI VULNERABILITÀ DERIVANTI DA ERRORI DI PROGRAMMAZIONE: OVERVIEW

Nel presente capitolo viene fornita un overwiev delle principali vulnerabilità, ad oggi conosciute, che scaturiscono da errori di programmazione indicando le buone pratiche che, indipendentemente dal linguaggio di programmazione utilizzato, è necessario adottare al fine di ridurre il rischio (common best practices).
A tal fine, si evidenzia che il 90% delle vulnerabilità nel software deriva da due distinte macro-categorie di errori di programmazione:

- una poco accorta gestione dell'input utente;
- controlli erronei o assenti durante l'allocazione delle aree di memoria adibite a contenere i dati.

A queste macro-categorie vanno ad aggiungersi:

- le problematiche di gestione delle sessioni utente;
- l'assenza di meccanismi crittografici a protezione dei dati scambiati in rete o conservati su disco;
- le vulnerabilità correlate al controllo degli accessi.

Vi è inoltre un fattore di media entità che, seppur non infici in via diretta la sicurezza di un software o di un sistema, consente a una minaccia esterna di acquisire informazioni preziose sullo stato dell'applicazione e di ottenere utili spunti per progredire gradualmente verso tecniche di attacco più complesse e sempre più finalizzate all'accesso fraudolento o al trafugamento dei dati. Queste tematiche, congiuntamente a quelle circostanze possono indurre al blocco del sistema o del software

6.1 Validazione dell'input

Il programmatore, spesso, non si pone il problema che gli utenti autorizzati, che possiedono una regolare password d'accesso, potrebbero non essere gli unici coinvolti a interagire con l'applicazione e si dà per scontato che l'input acquisito, in ingresso, dal programma sarà sempre conforme e pertinente al caso.
Le vulnerabilità di Input Validation scaturiscono proprio dall'assenza di controlli o da errori nella gestione dei dati inviati dall'utente e/o da un processo esterno al dominio di analisi. Le conseguenze di tali vulnerabilità consistono in una serie di tecniche di attacco differenti, solitamente finalizzate all'esecuzione di comandi remoti o alla visualizzazione di dati importanti.
È necessario quindi, verificare che l'input dell'utente e la sua rappresentazione non contenga caratteri o sequenze di caratteri che possono essere sfruttati in modo malizioso.
La validazione dell'input deve essere implementata utilizzando espressioni regolari, o algoritmi di filtro, dopo aver definito la lista di ciò che può essere accettato. La white list, contentente solo i valori ammissibili, è da preferire alla black list, che elenca i valori non ammissibii, poichéil continuo evolversi degli attacchi rende l'insieme delle stringhe 'non accettabili', di fatto, infinito.
Le problematiche di Input Validation sono comuni a tutti gli ambienti, ma trovano la loro espressione massima nelle applicazioni Web. Di seguito sono trattate le principali vulnerabilità, causate dal mancato filtro dei dati utente, nelle quali un aggressore può imbattersi sul Web, presentate da script, Servlet o CGI.

6.1.1 Shell Execution Command

Se nella casistica degli Overflow la vulnerabilità di riferimento è lo Stack Overflow, nelle applicazioni Web è senza dubbio lo Shell Execution Command. Le problematiche di Shell Execution Command, infatti, rientrano nella sfera delle vulnerabilità più note e più sfruttate di sempre . Si manifesta quando i parametri acquisiti in input vengono passati all'interprete di shell senza essere filtrati. L'esecuzione di un comando non è spesso possibile in modo diretto (ovvero semplicemente specificando ciò che si desidera eseguire), ma viene causata da una precisa condizione. Sui sistemi Unix è, ad esempio, possibile utilizzare il carattere ";" per concatenare più comandi fra loro, mentre in molti altri casi la condizione scatenante può essere causata da caratteri differenti come: