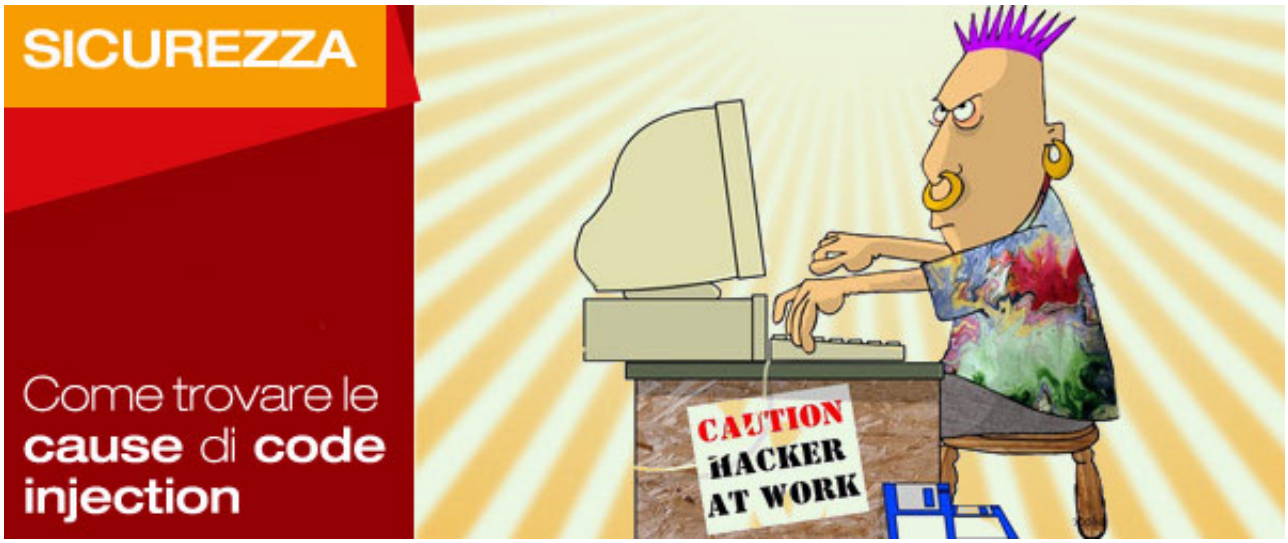
Sicurezza e siti web: come trovare le cause di code injection, tecniche di validazione



E così, dopo aver analizzato il funzionamento "completo" di un sito web, abbiamo scoperto quanto sia vulnerabile, e da quanti punti lo sia. Abbiamo parlato di code injection, e visto le origini di questo tipo di debolezza, magari anche scoperto che alcuni nostri siti sono vulnerabili. Dato che il compito di questi articoli non è solo valutare le debolezze, ma anche irrobustire le difese - come trovare ed eliminare le vulnerabilità dai nostri siti?

"Tutto è nelle mani del nemico. Non fidarti di nessuno" - Fox Mulder sarebbe stato un eccellente programmatore!

Di sicuro, una notizia di cattivo auspicio. La buona notizia è che possiamo evitare per intero il code injection controllando accuratamente il codice, e praticando tecniche di programmazione note come **validazione, escaping** e **indirezione**.

Nel resto dell'articolo, per mantenere quanto possibile la semplicità farò riferimento esclusivamente a tecnologie basate su **Javascript** e **php**, ma i concetti sono assolutamente estensibili anche a **ASP** e **VBScript**, **JSP**, **Java**, **Ruby** o qualunque altro linguaggio usato a livello di browser ed a livello di server.

Facciamoci aiutare - dal server

Per prima cosa, assicuriamoci di una cosa molto importante - che il nostro server PHP abbia il flag **register_globals** su *off*. Molti ne avranno sentito parlare ma, in realtà, a cosa serve e perché è "pericoloso" se settato su *on*?

Il flag **register_globals** indica come l'interprete PHP tratta le variabili globali: principalmente, tutte quelle variabili che provengono dalla url, o vengono passate tramite **post**, o anche i cookies e le variabili di sessione (in pratica, tutte le **variabili esterne**), definite come **globali** perché accessibili da ogni funzione senza essere dichiarate.

Se il flag è impostato su **on**, come era nelle prime versioni di PHP, tutte le variabili esterne diventano automaticamente variabili php. Se è invece, come di norma nelle ultime versioni, impostato su **off**, le variabili esterne sono accessibili esclusivamente tramite i vari array $_POST, $_GET, $_REQUEST, $_SESSION, etc.

Perchè questo è un problema? Perché molto spesso in php si usa il valore di una variabile senza inizializzarla, perché sappiamo che il php **la inizializza "come vuota" per noi**. Un esempio è il codice seguente:

L'idea è che se invochiamo questo script così:

*http://www.vergognosi.it/script.php?**flag=yes***

mi viene visualizzata la scritta "flag attivo". Se invece lo invocassimo così:

*http://www.vergognosi.it/script.php?**flag=no***

non mi verrebbe visualizzato niente. Ora, se il flag **register_globals** fosse su **on**, io potrei fare una cosa di questo genere:

*http://www.vergognosi.it/script.php?**messaggio=imprevisto***

e mi visualizzerebbe la scritta "imprevisto"!

Questo perchè il PHP registrerebbe la variabile "messaggio", che ho passato nell'url, automaticamente come variabile globale PHP. E non avendo inizializzato tale variabile prima, ho effettivamente creato un punto vulnerabile al code injection. Ora, ovviamente nel caso di cui sopra i danni sono limitati, ma immaginiamo cosa succederebbe se la variabile di cui sopra fosse usata per comunicare con un database o scrivere un file: a questo punto, avrei creato un passaggio diretto dall'url al mio database!

Si può notare immediatamente che questa tecnica d'aggressione è efficace soprattutto quando si conosce il codice php. Già sappiamo che basare la sicurezza sulla segretezza è un errore: il codice potrebbe essere stato letto, magari da qualche personaggio poco fidato nella web farm, o esposto da qualche problema del server web - o, addirittura, essere visibile a tutti perché **open source**.

In certi casi, la compromissione avviene banalmente perché il programmatore ha condiviso parti di codice su forum per programmatori, condivisione da cui elementi malintenzionati sono in grado di capire lo schema con cui inventiamo i nomi delle variabili e dedurli per applicarli ad altre nostre creazioni, e notare i lati deboli del nostro stile di programmazione per poi forzarli.

Molto spesso, il problema è subdolo:

In questo caso, lo script sarebbe (quasi, a parte la mailto ... ma ne parleremo più avanti) sicuro se il flag **register_globals** fosse su off, ma se è su on, a me basta invocarlo così:

http://www.vergognosi.it/script.php?email=il%20tuo%20indirizzo@email

e far felice qualche spammer: ho appena creato un mail gateway, che posso sfruttare per spedire mail a mia insaputa, a mio costo (o, peggio, del cliente che è molto meno accomodante ...) a mezzo mondo. I mail gateway tramite php, per inciso, sono tutt'altro che rari o non usati nel mondo dello spam.

Una cosa importante da notare: il flag **register_globals** non è "cattivo" in se. Nessuno dei problemi di cui sopra sorgerebbe se io inizializzassi tutte le variabili, che rimane comunque una pratica da seguire assolutamente. Mettendolo su off, però, sono protetto da questo tipo di problemi - eliminando i problemi alla radice, faccio solo del bene al mio codice.

Una checklist per il server

1. verifichiamo, tramite la funzione phpinfo, che il flag **register_globals** sia su off;
2. se è su on, settiamolo a off. A livello Apache (non possiamo farlo altrove!), basta aggiungere nell'.htaccess sulla directory principale del nostro sito: **php_value register_globals off**
3. se per qualche ragione non posso metterlo su **off** (verifichiamo sempre tramite phpinfo che il valore cambi, e se siamo puntigliosi verifichiamo all'avviamento dei nostri script in automatico), chiediamo alla web farm di farlo per noi;
4. se per qualche ragione non possono farlo a livello globale (vecchi siti scritti in PHP4 potrebbero non funzionare) e non posso farlo a livello locale perchè magari non posso modificare i .htaccess (e, soprattutto, non posso passare a provider migliori!), controlliamo ed inizializziamo tutte le variabili in uso dal nostro php. Un aiuto può venirci tramite la funzione **error_reporting**, messa al massimo dettaglio ci dirà se vengono usate variabili non inizializzate.

Faccio notare che l'uso del flag di **error_reporting** al massimo dettaglio è una pratica salutare e consigliata, anche laddove non ci siano problemi nella gestione del flag di **register_globals**. Un codice che non produce errori di nessun genere è un codice migliore, quindi meno soggetto ad

imprevisti e, di conseguenza, meno suscettibile ad aggressioni.

Tecniche di programmazione

Una volta che ci siamo assicurati la "collaborazione" del server, è ora di mettere ordine al nostro codice. Parlo di mettere in ordine ma, normalmente, questa fase viene effettuata **durante la scrittura del codice**, non dopo: è parte integrante. L'idea di scrivere il codice prima "alla veloce" per poi essere rinforzato dopo è foriera di problemi - non farlo se non sei davvero costretto e solo per cose molto piccole e molto temporanee, è meglio se integri da subito qualche buona tecnica e "pensi" il codice già al livello di paranoia necessario.

Le tre tecniche che andremo a vedere sono:

1. la **validazione**, ovvero ci assicuriamo che i dati siano validi sia da un punto di vista sintattico, numeri fatti di cifre, nomi fatti di lettere, che dove possibile semantico: date composte da numeri validi, indirizzi di mail composti dalle varie componenti, etichette esistenti che corrispondono a quelle che vogliamo, stringhe non vuote dove non devono essere vuote. Sebbene non sia propriamente corretto come definizione, questa fase include tacitamente anche la **conversione** dei valori: se leggo un numero nel mio codice **php**, lo converto in formato numerico, se leggo una data in Javascript la converto in oggetto **Date**: non li lascio rispettivamente come stringa o array di numeri;
2. l'**escaping,** ovvero sostituire simboli e caratteri che hanno un significato particolare in un dato linguaggio, pensiamo ad esempio ai caratteri ' e " nelle stringhe SQL e Javascript, o a e & in HTML, con quelle che vengono definite storicamente "sequenze di escape": ad esempio, nelle stringhe SQL si sostituiscono tutti i **'** con **\'**. Quel primo \ è detto **carattere di escape**. A livello di HTML, per esempio, la sequenza di escape di