Sicurezza e siti web: cosa si nasconde dietro al tuo sito?

Quando puoi veramente essere certo di aver pubblicato un ottimo sito? Quando il navigatore casuale di passaggio riesce ad usarlo senza dover leggere di istruzioni, suggerimenti e guide all'uso in modo naturale, concentrandosi sui contenuti anziché sulle meccaniche dell'interfaccia utente. La semplicità è il primo dono del web - ma l'apparenza inganna, e spesso non inganna solo l'ignaro visitatore, ma anche chi il sito lo produce, che sottostima le possibili fonti di problemi di sicurezza, quanto chi lo compra, che ne sottostima la dimensione in termini di mezzi e persone necessari al funzionamento.

Tu, conosci davvero il tuo sito?

La verità, è che molto spesso conosciamo perfettamente, in modo inconscio, la torre contorta e insidiosa su cui appoggiamo il nostro connubio d'arte e tecnologia, ma tendiamo ad ignorarla concentrandoci (spesso giustamente, almeno in certi momenti di vita del progetto) sugli aspetti funzionali ed estetici del nostro sito. In fondo al cliente vendiamo composizioni visive, florilegi, interfacce utente e moduli di contatto!

Abbiamo già visto le ragioni che ci devono spingere a considerare il problema sicurezza nello sviluppo dei nostri siti web, non importa quanto siano complessi o chi li ha commissionati (["Sicurezza e siti web: cosa significa, e perché non devi sottovalutarla su internet?"]), adesso passiamo ad esaminare in dettaglio il funzionamento dei siti stessi, per ricercarne i possibili punti d'attacco.

Il punto di vista del cliente

Dall'altro lato del problema, molto spesso il cliente vede il prodotto-sito acquisito come una specie di semplice **"presentazione Powerpoint"** glorificata, se non **"tipo il menu di un DVD"**, qualcosa a se stante che per qualche ragione strana non sempre **"possiamo metterlo su CDROM"** e che per motivazioni incomprensibili **"non funziona se ci scolleghiamo dalla rete"**, ma che tende ad accettare per fiducia, opportunità o consiglio, o perché così fan tutti ...

A volte siamo noi stessi, dalla prima riunione e per valide ragioni - non spaventiamolo, che poi non ci accetta il preventivo! - ad inculcare il concetto di semplicità. Peccato che, in realtà, nella strada che separa il navigatore dal sito si trovino centinaia o migliaia di persone addette ai lavori con i loro problemi di vita, di chilometri di cavi di varia natura che corrono sotto terra o sott'acqua ed in ambienti poco confortevoli, server di ogni tipo e apparecchiature con nomi esoterici e funzionalità oscure e misteriose ai più - molto spesso, anche a chi le apparecchiature le deve seguire ...

Ed **ognuna di quelle persone**, **ognuno di quei server** e di quelle apparecchiature **è un possibile punto d'attacco** che può pregiudicare il funzionamento del sito: altro che presentazione glorificata! Quindi, dobbiamo necessariamente conoscere i passi che il nostro sito web compie nella strada che conduce la nostra bella collezione di html e immagini sul server agli occhi del visitatore. Prima di tutto, per poter corazzare il nostro sito nei vari passaggi, ed in secondo luogo per poter spiegare al cliente le ragioni delle nostre scelte di sviluppo o, nel malaugurato caso, le sorgenti di un attacco avvenuto.

Il cliente ci vede e ci vedrà sempre come responsabili di quanto consegnato. È ragionevole, in fondo, è un nostro prodotto: quello che dobbiamo fargli capire, è che **il funzionamento e la sicurezza di quanto gli abbiamo consegnato è sì in parte nostra, ma coinvolge anche persone e apparecchiature in modo poco controllabile**: magari noi, come programmatori web, conosciamo le persone e le apparecchiature della web farm e possiamo avere garanzie di sicurezze, ma molto raramente andiamo oltre. Nessuno può garantirci che l'addetto di un determinato punto di smistamento in un lato oscuro della rete stia rivendendo informazioni estratte "sniffando" il gateway che dovrebbe invece monitorare!

Come sempre, ricordiamoci che la sicurezza su web è soprattutto una questione di massima resistenza ed efficace ripristino, non di inviolabilità assoluta: con il dovuto tempo ed i relativi mezzi si può sgretolare qualunque muro o demolire qualunque porta, quindi il nostro obiettivo è di allungare il più possibile quel tempo per rendere l'impresa non interessante o accessibile a pochi e darci tempo di predisporre contromisure, chiudere le falle o rimuovere il tesoro, prima che diventi bottino!

Ad ognuno il suo

Analizziamo passo per passo il funzionamento di un sito web:

1. abbiamo un'interfaccia disegnata in HTML e CSS: menu, scrollbar, pulsanti, testi e così via.

Quello che abbiamo venduto nel nostro progetto web!

2. l'interfaccia è gestita tramite Javascript, solitamente tramite framework stile jQuery (a cui mi riferirò: ma si tratta, comunque, di concetti applicabili alla stragrande maggioranza dei frameworks);
3. i frameworks gestiscono effetti speciali visuali, per la gran gioia del reparto grafico, ma soprattutto comunicano con i servizi gestiti sul server - integrazione di feed in tempo reale da twitter o altri social networks, interrogazioni su database, chiamate a server REST, ed il classico e comunissimo "modulo da compilare", la form che ci accompagna da sempre - tramite chiamate AJAX eseguite dal browser;
4. il browser invia le richieste ricevute tramite AJAX, traducendole nei metodi http classici: GET e POST per le applicazioni classiche, cui si aggiungono PUT e DELETE per le chiamate a server REST. Per farla semplice: il browser impacchetta i nostri dati e li spedisce al server, indicando cosa deve farne;
5. ... i nostri pacchetti viaggiano nella rete, migliaia di km di cavi e decine di apparecchiature controllate e sorvegliate da computer, personale, litri di caffè e kg di cioccolata ...
6. ... e raggiungono il server, dove vengono passati alla nostra applicazione PHP, che li elabora, magari pescando da un bel database postgres o mysql, e di contro ci rimanda indietro un altro bel pacchetto, con i risultati delle nostre richieste ...
7. ... che ripassano per la rete ...
8. ... arrivano al browser, che risponde al framework ...
9. ... che aggiorna l'interfaccia, visualizza il feed, aggiorna la tabella, cambia l'immagine, avverte l'utente che l'operazione è stata completata ...
10. ... per l'utente, si tratterà solo di un nuovo testo da leggere, di una nuova immagine o di un modulo inviato: nulla di complesso!

Ora rivediamo la stessa lista, in termini di problemi di sicurezza e con il massimo della paranoia:

1. l'HTML che si vede nel sito potrebbe essere alterato: potrebbero essere state introdotte modifiche da malintenzionati direttamente nella web farm, o modificato in tempo reale da malware sul PC del navigatore sostituendo links, cambiando immagini, etc. A volte sono cambi sottili, un link, un testo, altre volte sono cose assolutamente senza possibilità d'errore: al posto della galleria d'arte, troviamo un sito di poker clandestino (e, più frequentemente, altri tipi di sito ...), o una bella tag stile graffiti sul muro ... il nostro sito è vittima di un atto chiamato defacing;
2. il Javascript può essere disabilitato, o alterato in tempo reale tramite console e tool di sviluppo, gli stessi che ci consentono di svilupparle: il buon vecchio Firebug. E a volte basta tentare senza neppure guardare, compilare un campo di una form con caratteri particolari come gli accenti, o le & commerciali o le tag html, o persino query SQL: in questo caso, si tratta di un attacco code injection, che può portare rapidamente al defacing se non alla compromissione dei dati;
3. i frameworks traducono le chiamate ai server impacchettando i dati in formati xml o json, a loro volta facilmente tracciabili e modificabili, come replicabili, tramite l'ignaro Firebug

stesso: un'altra possibile fonte di code injection;
4. le richieste a questo livello sono virtualmente identiche alle richieste classiche di qualunque browser, quindi intercettabili da malware installato sul PC del navigatore che può salvarle su log per poi inviarle a malintenzionati, in modo simile al keylogging;
5. ... in qualche punto tra quelle decine, indistinguibile se considerato sulla base dei caffè bevuti o della cioccolata, potrebbe esserci qualcuno dedito od incaricato o con la necessità di guardare o censurare quello che avete appena trasmesso, perché magari non state usando nessuna encryption ...
6. ... trasmissione che comunque raggiunge il server, che può essere stato compromesso alla stregua del PC del navigatore, o controllato da malintenzionati che stanno sorvegliando quello che fate come sopra, e potrebbe fornirvi risposte impreviste o tali da ingannare il navigatore ...
7. ... altra possibilità di venire intercettati a metà strada, con in aggiunta una chance per alterare i dati che vi vengono forniti ...
8. ... altra possibilità di essere loggati dal malware sul PC del navigatore prima di passare al framework ...
9. ... che aggiorna l'interfaccia, magari con dati compromessi ...
10. ... che l'utente non riconosce come falsi, e su cui magari prende decisioni. Come, ad esempio, procedere con il pagamento con carta di credito ...

(Un consiglio... nelle questioni di sicurezza siate paranoici, fa solo bene, ma nello stesso modo cercare di controllare la paranoia con un calcolo di rischi e benefici di quello che fate per proteggervi: proteggete ogni risorsa in modo proporzionale al suo valore, ma **non cercate la protezione assoluta, perché è impossibile, o impraticabile, o superiore al valore dell'intera impresa!**)

Ce n'è da mettersi le mani nei capelli - e abbastanza per scriverci un libro. In pratica ogni ingranaggio, ogni snodo del nostro sito è attaccabile, ed è la terza legge della sicurezza: **tutto è in mano al nemico**. Significa: ogni volta che dobbiamo valutare la sicurezza dei nostri siti, consideriamo sempre cosa succederebbe se una o più delle parti in gioco fosse compromessa ed in mano a malintenzionati, valutiamo i rischi che corriamo e dove possiamo, turiamo le falle, dove non possiamo, prepariamo un bel piano di emergenza e ripristino - ed in ogni caso mai, mai fidarsi completamente della sicurezza di parti e persone di cui non abbiamo controllo.

Le buone fondamenta

Cosa vuol dire per noi sviluppatori web?

Vuol dire sviluppare i nostri siti considerando il fatto che **il navigatore potrebbe avere un PC** o il browser **infetto o controllato**, o **il server web potrebbe essere invaso e compromesso** dall'esterno quanto dall'interno e quindi distrutto, modificato o monitorato. Che l'utente che noi pensiamo che sia **in realtà è qualcun altro** che ha preso possesso del suo username e della

password del sito (e quindi della sua identità!), o che **controlla la sua casella di posta**. Che l'utente che si trova davanti al nostro sito non stia apprezzando la bellezza del nostro design e dei nostri effetti speciali, ma che stia osservando il nostro javascript, cercando di capire come fare a raggiungere i nostri database o i nostri files php a suo uso, abuso e consumo.

Scoraggiato? La cattiva notizia è che è comunque troppo tardi per cambiare mestiere e non importa che nessuno ce l'abbia detto quando l'abbiamo scelto, oramai ci siamo dentro, e vediamo come comportarci e trasformare i rischi in virtù: **conoscere e sapere affrontare i problemi della sicurezza sul web è sempre un ottimo biglietto da visita** - purché lo si presenti con l'umiltà di chi sa di non dover promettere fortezze inviolabili, ma solo muri molto solidi.

Non arrenderti - e non ignorare!

Sarebbe bello pensare solo ad accostare colori, fondini e gradienti del layout ed a perfezionare la fluidità delle animazioni nell'intro page - al più, quello è un privilegio del reparto grafico, fortunati loro! Purtroppo, per noi non è così, ed è importantissimo averlo ben chiaro al momento della scrittura della prima linea di HTML, dalle fondamenta a salire. Quindi:

1. scriviamo codice tale da evitare ogni tipo di code injection (come? Ne parleremo in abbondanza presto);
2. prepariamo le directories del sito in modo che non siano scrivibili se non in modo molto controllato, in modo che il nostro HTML non sia modificabile trivialmente. **Queste vanno gestite con cura estrema, specialmente se sono accessibili anche tramite URL!**
3. Isoliamo i files che non devono essere direttamente accessibili tramite URL (es. files di inclusione, files di dati, sql, files temporanei o di upload, etc) da quelli che invece lo devono essere: questo significa tanto organizzare la struttura del sito in anticipo e scrivere codice PHP corretto quanto imporre regole ad Apache che impediscano o controllino gli accessi a determinate directories (ne parla egregiamente Maurizio Tarchini nell'articolo "Come modificare l'accesso ai file ed alle cartelle?");
4. controlliamo allo stesso modo anche gli accessi ftp o ssh - **cosa succederebbe se le aree del mio sito fossero leggibili tramite ftp**, magari con l'account di altri utenti?
5. per tutte le nostre operazioni AJAX ed ogni altra comunicazione col server, incluso il semplice submit delle form in vecchio stile, valutiamo sempre quali possono essere gli effetti se terzi potessero monitorare le transazioni, leggerne il contenuto. Si tratta di dati sensibili, possono essere sfruttati per altri fini?
6. nel caso ci andasse male, o fossimo attaccati su parti di cui non abbiamo il controllo, **siamo in grado, e se si di cosa abbiamo bisogno, di ricostruire il sito che è stato attaccato e danneggiato?** Siamo in grado di riconoscere se un sito è stato compromesso in modo molto subdolo? Abbiamo backup dei database utili?

L'introduzione finisce qui: dal prossimo articolo partiremo con esempi, parti di codice (soprattutto da non fare!) e suggerimenti mirati a risolvere od evitare i problemi di cui abbiamo parlato sin'ora -

finalmente! Una riflessione da fare, a piede di tutte queste considerazioni: **quanto pesano i problemi di sicurezza e della complessità nascosta nei rapporti con i tuoi clienti?**

Indice