WEB APPLICATION SICURE ATTRAVERSO L'HARDENING DI APACHE

Giuseppe Augiero

Agenda

Quante volte avete dovuto mettere in esercizio una applicazione web scritta da terze parti, una di quelle applicazioni di cui non si conosce nulla o che è implicitamente insicura?

Come fare ad aumentare il "grado di sicurezza" di una web app?

Come dormire sonni tranquilli e tirare un sospiro di sollievo?

Partiamo da lontano...

I tempi cambiano…

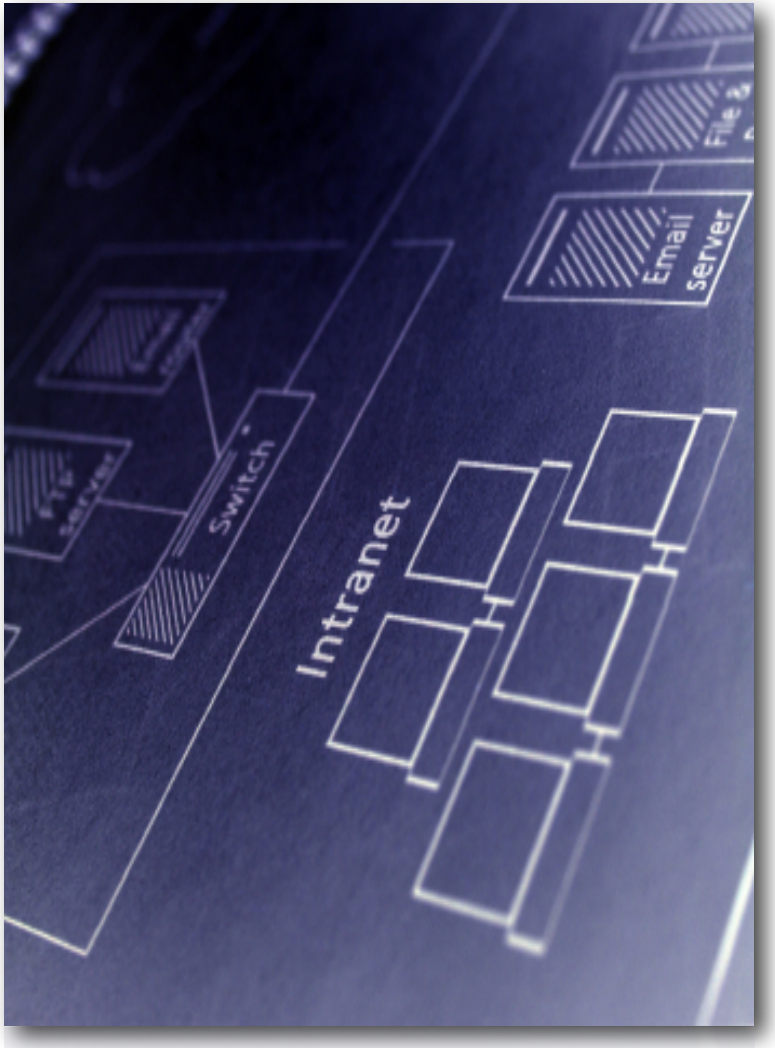
7 maggio 2010 - Area della Ricerca di Pisa © Giuseppe Augiero - giuseppe@augiero.it

Le architetture tradizionali basate su modelli di rete fortemente strutturate su base layer:

- Layer 2 (**switching**) nella LAN
- Layer 3 (**routing**) nella WAN
- **Firewall** sul confine

Tendono a sparire

Il mondo delle reti sta adottando **modelli dinamici** e flessibili che prevedono la localizzazione delle attività di routing e switching **dove servono** effettivamente.



Niente più confini

Tecnologie (wireline e wireless) di nuova generazione iniziano a cancellare dal vocabolario della rete la parola **Perimetro**.

La logica di "Demarcazione del proprio territorio" viene a mancare e cambia integralmente la gestione della sicurezza aziendale.

La rete aziendale, interconnessa con fornitori e clienti, è sempre più una rete di tipo **bordless network**.

Da chi dobbiamo difenderci?

Con l'abbattimento dei confini viene a mancare la teoria secondo la quale "i buoni sono dentro e i cattivi sono fuori".

I dispositivi wireless possono essere una vera minaccia.

La maggior parte degli attacchi informatici arrivano dall'interno della struttura da proteggere.

Alcune volte l'attacco a un server diventa "uno sport" o "un modo per protestare".

Da cosa dobbiamo proteggerci?

Con il passare degli anni gli attacchi sono diventati sempre più mirati e aggressivi.

L'attacante può essere interessato alle informazioni presenti all'interno dalla nostra infrastruttura o solo interessato ad utilizzarci come punto di rilancio per una attacco di entità maggiore.

Spesso si cerca un obiettivo facilmente attaccabile.

Il web

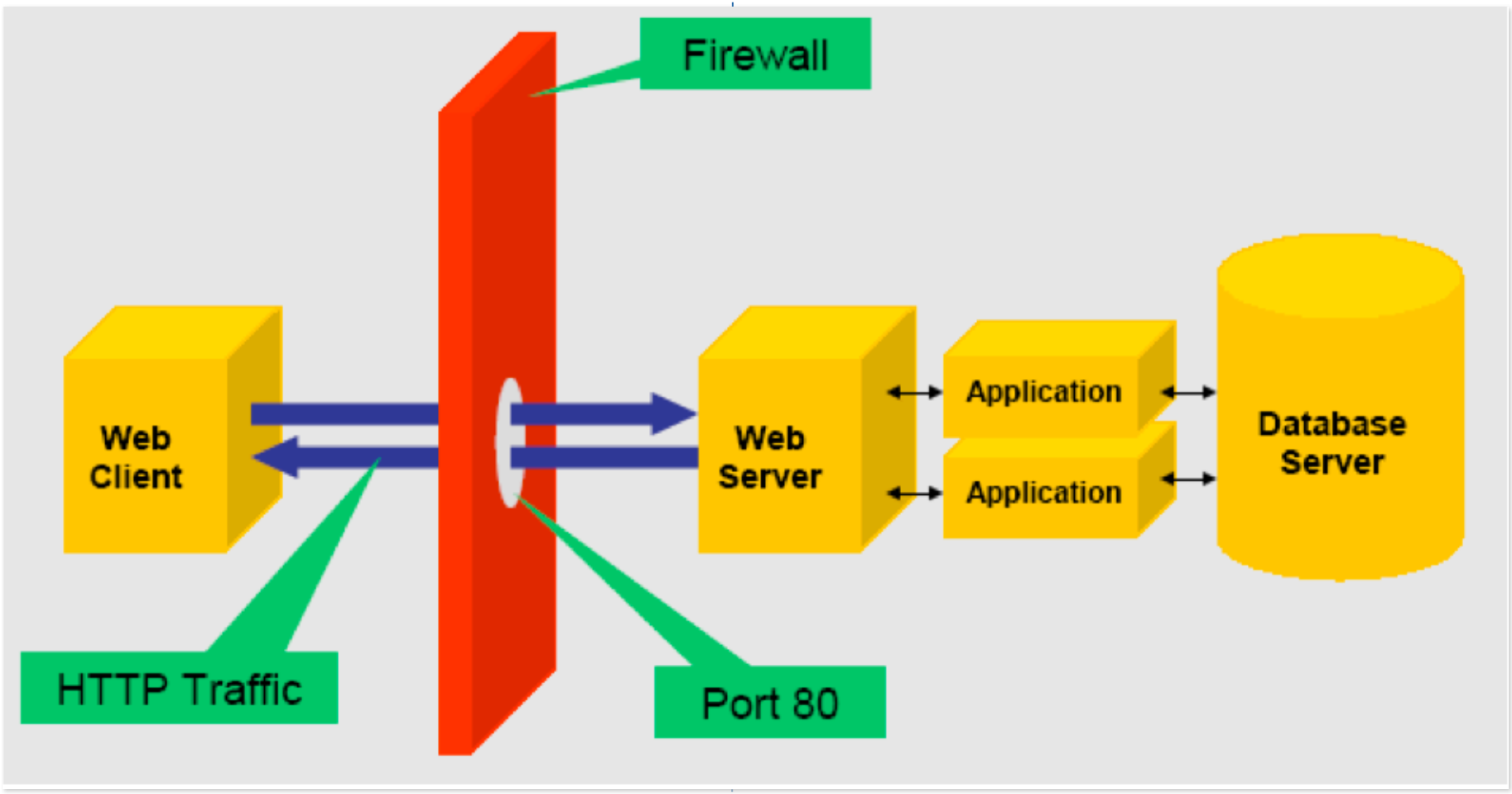
Il 70% degli attacchi informatici arrivano dal Web.

Il protocollo **HTTP** durante Defcon'06 è stato ribattezzato protocollo UBFP.

UBFP = Universal Bypass Firewall Protocol.

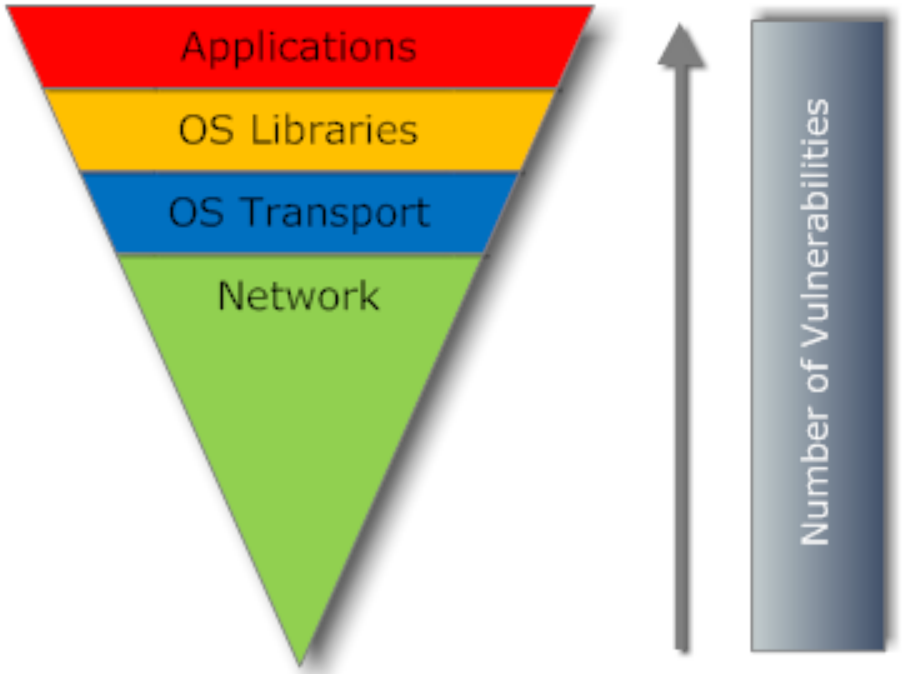
Le applicazioni Web sono insicure: i programmatori sembra abbiano dimenticato tutto quello imparato in 20 anni di evoluzione del software.

Firewall (Layer 3)



Vulnerabilità

- Più si sale con i livelli dello stack Tcp/Ip e più aumenta il numero di **vulnerabilità** dell'intero sistema.
- I **Firewall** tradizionali lavorando a Layer 3 non possono fare analisi e protezione dei livelli sovrastanti.
- Le applicazioni web non sempre sono sviluppate con criteri di sicurezza e testate in ambienti opportuni.



Vulnerabilità Web

•Le maggiori vulnerabilità del mondo Web sono:

- **Cross-Site Scripting** (70%)
- **Predictable Resource Location** (25%)
- **Content Spoofing** (25%)
- **Insufficient Authentication** (20%)
- **Sql Injection** (20%)

Cerchiamo un rimedio

Fondamenta Sicure

Per offrire un servizio sicuro, l'intero sistema (e quindi le singole parti) deve essere messo in sicurezza.

Il server che offre il servizio che vogliamo erogare deve essere "**bastionizzato**" sia dal punto di vista del sistema operativo e sia da quello dei demoni che erogano il servizio voluto.

E' falsa l'affermazione "funziona, allora meglio non toccare nulla".

Permessi minimi

E' inutile e veramente insicuro far girare apache con permessi da **superutente**.

Nel caso in cui l'attaccante riuscisse ad accedere al server attraverso il web si ritroverebbe con i diritti di root e la possibilità di fare qualsiasi cosa.

Meglio far girare apache come utente **nobody** (debian) o un utente ad hoc creato per questo scopo.

Implementare geo-firewall

E' una buona norma limitare il servizio offerto unicamente a coloro che realmente devono usufruirne.
Se l'applicazione non deve essere erogata all'intera Internet e' possibile utilizzare dei **geo-firewall** per limitare l'accesso.

E' possibile definire geo-regole anche per i più comuni firewall (ip.ludost.net).

I Moduli di Apache

Mod_Evasive

Il modulo Evasive permette di prevenire attacchi di tipo Dos (**Denial of Service**) attraverso il protocollo Http.

Il modulo conta quante richieste vengono effettuate dai singoli ip dei client connessi al server.

Al superamento di una soglia prestabilita l'indirizzo ip viene messo in black-list.

Mod_Antitamper

Il modulo Antitamper permette di evitare alcune forme di manipolazione degli **indirizzi url** o dei **cookies**.

In particolare il modulo si pone sulla catena di uscita di apache per firmare con sha1 i link degli url della pagina e i cookies inviati al client web.

Mod_Rewrite

Attraverso il modulo Rewrite è possibile riscrivere gli indirizzi url del nostro server web.

E' possibile mascherare, con semplici regole, il linguaggio di programmazione server-side.

E' possibile forzare l'utilizzo del protocollo ssl (**https**) al posto del canonico http.

Mod_Security

Mod_Security

Mod_security è un modulo di Apache **preposto ad aumentare la sicurezza delle applicazioni web**.

Si propone di fare intrusion detection and prevention all'interno del Web server.

Lavora ad application layer e svolge le funzioni di application firewall.

Mod_Security (II)

Mod_Security è sviluppato da **Ivan Ristic** e si è evoluto fino alla versione 2.5.

Due licenze per la distribuzione del codice: **Gpl** e commerciale.

Pacchettizzazione per le maggiori distribuzioni di Linux e per Windows.

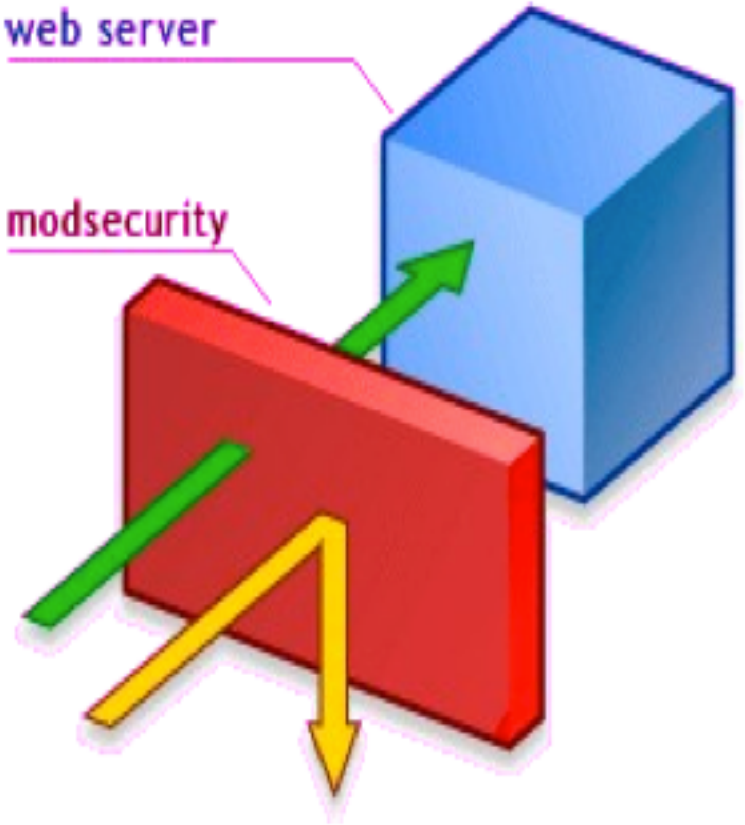
Sito di riferimento: www.modsecurity.org

Come lavora

Mod_Security analizza tutte le richieste in arrivo al Web server attraverso il protocollo http.

Permette di controllare tutte le variabili che vengono scambiate in una navigazione (Get, Post, Cookie) ed i file uploadati.

Può normalizzare le richieste per smascherare le tecniche di evasione del riconoscimento degli attacchi.



Input

Ogni richiesta HTTP viene '**normalizzata**' cioè viene elaborata ogni codifica (es: %20 = spazio).
La richiesta del client web viene sezionata in diverse variabili come nome server, file richiesto, variabili Get, variabili Header...

Per ogni variabile può essere definita una regola per controllare la sua coerenza con l'applicazione.

Output

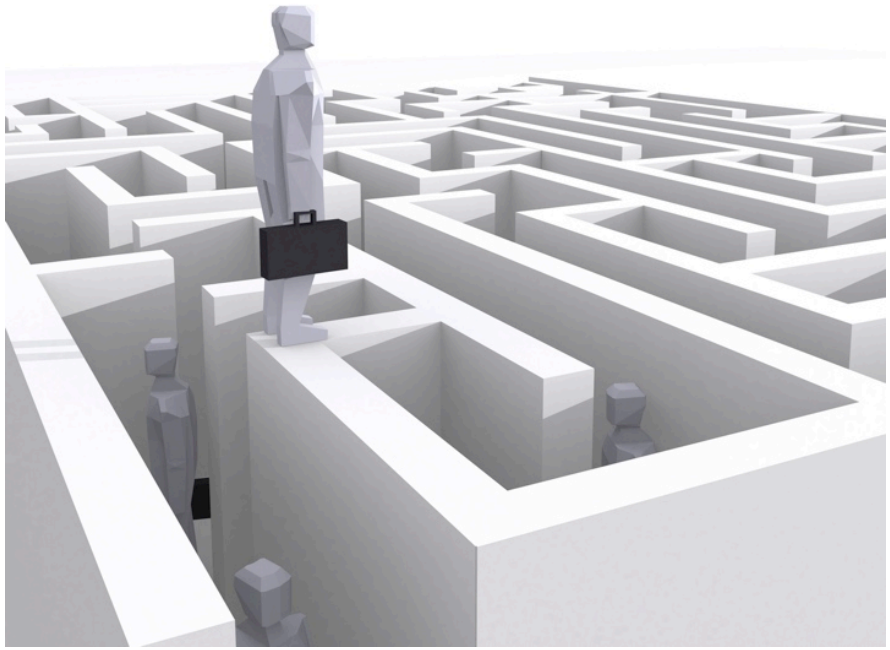
L'output dell'applicazione web può essere controllato o/e filtrato.

Un tipico esempio del controllo dell'output è il mascheramento di un errore applicativo che nel 99% dei casi rivela dati preziosi all'attaccante: percorsi di file o errori sul DB.

A ognuno le sue regole

Le direttive di configurazione vengono esplicitate nel file di configurazione di Apache attraverso delle **regole**.

Le regole possono essere definite a livello globale o per risorsa (directory o file) e generalmente vengono ereditate dalla directory padre.
Le regole possono essere concatenate e prevedono un'**azione** per ognuna di essa.



Dove trovo regole già pronte?

All'interno del pacchetto di mod_security sono già presenti alcuni insiemi di regole scritte per proteggere le più comuni applicazioni web.

E' possibile scaricare altri set di regole dal sito www.gotroot.com.

Scrivere nuove regole non è una operazione difficile e nel breve tempo è possibile definire un set di regole personalizzate.

Azioni

Nella definizione di una singola regola è possibile esplicitare quale azione intraprendere.

Alle azioni di base (**Pass** e **Deny**) si aggiungono le seguenti azioni:

- ➔ status
- ➔ redirect
- ➔ exec
- ➔ log
- ➔ pause
- ➔ chain

Server Signature

Attraverso Mod_Security è possibile modificare la firma (**signature**) del server web, che viene allegata ad ogni risposta HTTP.

In questo modo è possibile disorientare l'attaccante inviandogli dei dati fuorvianti che gli dovrebbero far perdere inutilmente del tempo nella ricerca di vulnerabilità probabilmente inefficaci per il webserver protetto.

Proteggere altri web server

L'accoppiata dei moduli **Mod_Security** e **Mod_Proxy** permette di anteporre il firewall applicativo a qualsiasi web server. Potrete proteggere ogni web server (Apache, IIS, Sun One) visto che il modulo proxy reinstraderà ogni richiesta come è pervenuta al back-end.

A questo si sommano i vantaggi di un comune proxy: caching – network isolation – single point of access

Accounting

Analisi dei log

E' fondamentale l'analisi dei log per capire cosa sta succedendo al nostro web server.

La concatenazione degli eventi può portare al riconoscimento di "attacchi invisibili".

L'analisi permette di effettuare **anomaly detection**.

Host Intrusion Detection

E' buona norma utilizzare un host-ids sul server web che vogliamo proteggere.

Ossec offre le seguenti funzionalità:

- Analisi dei log.
- Controllo dell'integrità dei file.
- Identificazione di Rootkit.
-Offre informazioni per eventuali analisi forensi.

DOMANDE?
RISPOSTE!

Giuseppe Augiero – giuseppe@ftgm.it - giuseppe@augiero.it

Grazie !!!

Giuseppe Augiero – giuseppe@ftgm.it - giuseppe@augiero.it

Licenza di utilizzo

Giuseppe Augiero – giuseppe@ftgm.it - giuseppe@augiero.it