

Sicurezza Web Publishing

<http://escher07.altervista.org>

Generalità

In questo documento si ipotizza di dover mettere su web un applicativo con tecnologia adeguata (ASP/IIS, PHP/Apache etc...). Si vogliono discutere pregi e difetti delle varie possibili soluzioni.

Prospetto Riassuntivo

Soluzione	Scenario	Difetti	Prerequisiti	Scenario di utilizzo
Unico Server	<ul style="list-style-type: none">Molti/Pochi client fidati collegati alla LAN	<ul style="list-style-type: none">Solo ambito intranetNessuna protezione globale dalla intranet	<ul style="list-style-type: none">Funzionalità di Web Server + DB dipendente dall'applicazione e.	
Separazione DB/Application	<ul style="list-style-type: none">Molti possibili client non fidati	<ul style="list-style-type: none">Apertura http DMZ-INSIDE per il server web su una porta applicativa	<ul style="list-style-type: none">Apache o IIS sul server esterno.Un IP pubblico per servizio esposto.	<ul style="list-style-type: none">Almeno due server distinti (DB e Application)Applicazione Sicura a seconda dei dati.
Reverse Proxy	<ul style="list-style-type: none">Come Sopra	<ul style="list-style-type: none">Single Point Of Failure;Apertura http DMZ-INSIDE per il server web sulla porta 80.	<ul style="list-style-type: none">Apache o IIS+ISA sul server esterno.Al limite un IP pubblico per tutti i servizi esposti.	<ul style="list-style-type: none">Come Sopra
VPN	<ul style="list-style-type: none">Pochi client fidati collegati ad INTERNET	<ul style="list-style-type: none">Occorre conoscere chi sono i client;Tempi di configurazione;Consumo banda;	<ul style="list-style-type: none">Presenza di VPN concentrator e AAA server;	<ul style="list-style-type: none">Un VPN concentrator;Un AAA server;Un application+DB server;
Duplicazione DB e Application	<ul style="list-style-type: none">Molti possibili client non fidati	<ul style="list-style-type: none">Da gestire il dialogo DB esterno/interno;Ridotta operatività dei client;	<ul style="list-style-type: none">Come in Separazione DB/Application	<ul style="list-style-type: none">Almeno due DB server distintiApplicazione Sicura a seconda dei dati.

Unico Server

E' la soluzione più semplice ed è quella normalmente adottata in contesti intranet. Esiste un solo server dove risiedono l'applicativo ed il DB, entrambi in INSIDE. I due livelli possono essere separati ma solo per motivi prestazionali, ovvero possono essere usati un Application Server ed un DB Server ma entrambi sulla LAN. E' ovvio che essendo per ipotesi la INSIDE un'area con politiche omogenee per entrambi le protezioni sono solo quelle locali (sistema operativo e applicativi).

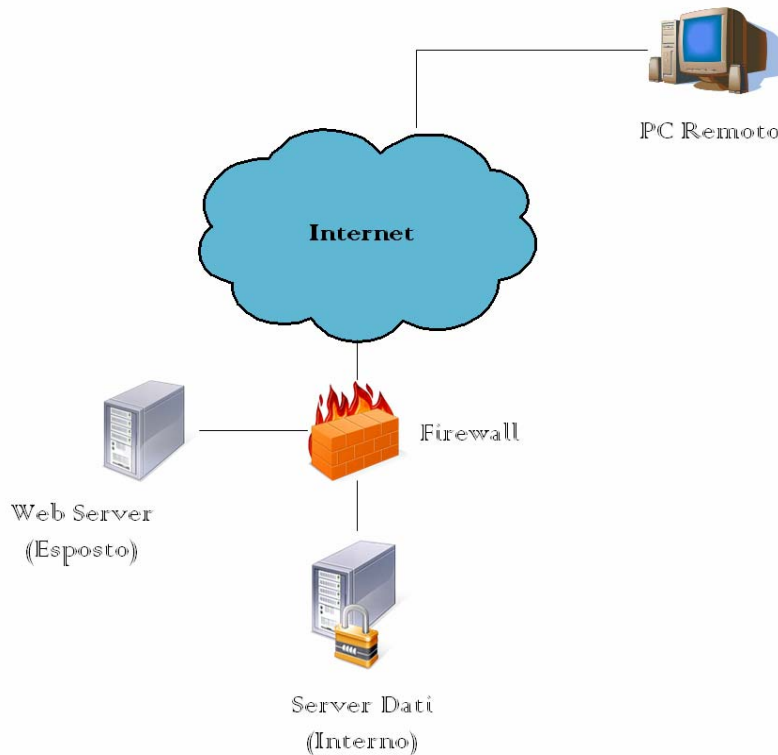
I dati viaggiano in chiaro quindi non c'è alcuna protezione nei confronti di possibili sniffing o tentativi di accesso/Dos dalla INSIDE. Del resto si suppone che per politiche aziendali e/o sorveglianze interne e/o scarse capacità tecniche degli utenti etc... questi eventi siano estremamente improbabili.

Separazione DB/Application

Ipotizziamo che la stessa applicazione di cui in precedenza debba ora essere resa disponibile su internet. Rispetto al caso precedente ci sono alcune differenze di fondo, ovvero:

- presenza di un firewall
- i servizi esterni devono essere posizionati in una opportuna area (detta DMZ) ed essere accessibili tramite uno o più IP pubblici (fisici o tipicamente NAT operati dal firewall stesso)

Dunque ciò che deve essere pubblico deve andare in un'area "demilitarizzata". La prima idea è quella di sostituire l'unico server precedente con due (o più) server uno più esposto ed uno meno. Cosicché se crolla il primo almeno i danni sono limitati alla sola DMZ e non comportano perdita dei dati almeno finché non crolla anche il secondo. In pratica lo scenario è questo:



In sostanza il server web esterno raccoglie una richiesta http destinata ad un server dati (tipicamente in INSIDE, dove interno=senza indirizzo pubblico) soddisfacendola e restituendo al PC remoto l'HTML giusto. E' da evidenziare che il server esterno naviga dentro la LAN al posto del PC remoto, quindi la differenza rispetto ad un normale link è che i diritti a livello di firewall saranno solo su di lui e non sul PC, cosa (impensabile) che comporterebbe l'apertura del server interno a tutti.

Ci sono vari possibili modi con cui il server esterno dialoga con l'interno. Il più semplice è quello di interrogare direttamente il servizio interno interessato : ad esempio se quest'ultimo è un DB Oracle opererà una richiesta sulla porta 1521. Ci sono poi metodi alternativi per cui la richiesta generica è impacchettata in una di tipo http (Web Service/Reverse Proxy). In tutti i casi il firewall deve avere una regola di questo tipo:

Permit Web server -> Server Dati Port=P0

...dove P0 è rispettivamente 1521 o 80 nei nostri esempi.

In una situazione di questo tipo la sicurezza viene demandata essenzialmente

- Alla stabilità del Web Server Esterno
- Al servizio che raccoglie le richieste sul server dati (Oracle Listener o Servizio WWW)

Un eventuale crash del primo apre le porte della DMZ. Un eventuale crash del secondo apre le porte della LAN, dato che evidentemente conquistare un server di una certa area

corrisponde (a meno delle sicurezze locali, ovvero firewall applicativi e password) ad avere a disposizione tutta quell'area.

In questo caso la sicurezza dell'applicazione è certo importante ma il quanto questa debba essere sicura dipende soprattutto dai dati. In altre parole gli eventi più pericolosi possono essere questi:

1. l'intruso esegue sniffing sui dati trasmessi su internet
2. l'intruso manda in crash l'applicativo
3. l'intruso riesce ad eseguire del codice sul web server e invia comandi dannosi per il server interno

questi eventi sono in ordine crescente di difficoltà e di pericolosità. In altre parole se l'applicazione invia dati "delicati" deve sempre implementare tecniche tipo SSL (per proteggersi dal primo evento) : detto questo gli altri due eventi riguardano tutti i possibili siti web anche se qui c'è da valutare bene il rapporto costo/beneficio di spendere risorse in un adeguato tuning/hardening del sistema.

Reverse Proxy

E' in pratica un caso particolare del precedente nel senso che la richiesta DMZ-INSIDE non è generica ma di tipo http ed è rivolta quindi ad un application server in INSIDE, che normalmente accorpa anche le funzioni di DB (ovvero non lo fa ma solo per motivi prestazionali, tanto una volta su quello in LAN già ci saremmo). Quindi introduce un livello aggiuntivo di sicurezza nel senso che per far crollare questa architettura occorre "sfondare" il Web Server, poi l'application tramite codice inviato sulla porta 80, poi il DB, eventualmente su una terza macchina.

Da notare che in ambiente Unix il servizio Apache può fare anche funzioni di reverse proxy mentre in ambiente Windows non vale lo stesso per IIS : in tal caso occorre utilizzare ISA, ovvero il proxy di Microsoft.

Il reverse proxy ha anche un altro vantaggio rispetto alla generica architettura a "due livelli" di cui in precedenza : non è indispensabile avere un indirizzo IP per ogni servizio esposto ma, al limite un solo indirizzo pubblico e un Web Server che supporta il virtual hosting (sia Apache che IIS lo fanno).

Il principale problema di questa soluzione è la presenza di un singolo point of failure. Un attacco DoS al Web Server provoca una interruzione di servizio di tutti i servizi per cui questo fa reverse proxy.

Inoltre il reverse proxy non protegge in alcun modo dallo sniffing di dati, dato che non opera alcuna crittografia che deve essere implementata dall'application server. Se questo non lo fornisce ed i dati sono "sensibili" non va usato.

Virtual Private Network

Questa soluzione serve tipicamente per espandere l'utilizzo di un applicativo intranet (o addirittura client/server) ad una serie di sedi remote della stessa organizzazione.

In pratica la sede remota tramite adeguato client (esempio Cisco VPN Client) instaura un dialogo con la logica PKI con un adeguato apparato hardware della rete interna (detto VPN Concentrator) , tipicamente il firewall.

Questi eseguita l'autenticazione tramite proprie access list o più spesso tramite il dialogo con un server dedicato detto AAA (Authentication Authorization Accounting *server*) crea un canale criptato in cui viaggiano i dati.

Fisicamente tramite internet ma virtualmente in modo punto-punto fra PC remoto e application server (se come di norma la VPN è configurata per poter accedere solo a quella macchina e a quel servizio ossia porta).

Questo tipo di architettura replica in sintesi lo scenario intranet fra sedi connesse tramite internet, per cui l'application server (+db o meno a seconda delle prestazioni) è di norma in INSIDE, ovvero in una DMZ dove si trovano tutti i server per la intranet se di è decisa questa architettura.

Le principali criticità di questa soluzione sono :

- il numero comunque limitato di sedi collegabili, necessariamente note a priori;
- la necessità di apparati dedicati (VPN Concentrator, AAA server)
- il traffico che si genera e l'impegno per gli apparati che eseguono la decrittazione;

Il vantaggio principale è che diventa abbastanza irrilevante la sicurezza intrinseca dell'applicazione : anzi si può dire che questa soluzione ha senso solo se il dialogo client/server avviene in chiaro (come avviene per tutte le applicazioni originariamente pensate per contesti LAN), altrimenti si sovrappone inutilmente crittografia (VPN) a crittografia (SSL) oltre a portarsi dietro tutti i vincoli di cui sopra.

Duplicazione DB e Application

Questa è la soluzione normalmente attuata in presenza di molti client che accedono da internet e di dati delicati, come ad esempio succede nei progetti e-commerce.

La duplicazione del DB porta evidentemente dei costi di licenze significativi (nel caso di software non Open Source) ma permette di essere sempre abbastanza tranquilli sul fatto che i dati più delicati non siano compromessi.

In pratica in DMZ si trova un application server che normalmente implementa tecniche tipo SSL ed un DB server che contiene i suoi dati, che costituiscono solo una piccola parte dell'archivio (ad esempio gli ordini raccolti in giornata). Application e DB possono trovarsi anche sulla stessa macchina e/o sono separati più per motivi prestazionali che per altro. Questo application consente in genere funzionalità estremamente limitate quali ad esempio l'inserimento dell'ordine ma non la sua modifica e/o cancellazione sia per motivi funzionali (in nessun marketplace il cliente può cambiare idea su ciò che vuole quando vuole) ma anche per motivi tecnici (il DB in DMZ contiene proprio per ragioni di sicurezza solo le minime informazioni possibili).

In INSIDE si trova invece un secondo application e un secondo DB, eventualmente anche qui sulla stessa macchina. Il dialogo fra questi due servizi avviene con le politiche tipiche della LAN e quindi di norma con piene funzionalità del client (es. inserimento, modifica, cancellazione) e con dati che viaggiano in chiaro.

In tempi prestabiliti i due DB dialogano fra loro : in genere è quello interno che attraverso una impostazione di questo tipo:

Permit Application Interno -> Application Esterno Port=80

... ovvero (se si utilizzano stored procedures o simili) di questo tipo:

Permit DB Interno -> DB Esterno Port=P0

... preleva i dati da quello esterno, portandoli sulle proprie tabelle e facendoli entrare nel "giro" (es. contabilità) ordinario.