LINEE GUIDA DI PROGRAMMAZIONE

SICUREZZA DELLE APPLICAZIONI WEB

Ver. 1.0

TABELLA DELLE VERSIONI

Data	Versione	Descrizione	Cap. /Sez. modificati
Febbraio 2004	1.0	Nascita del documento	tutti

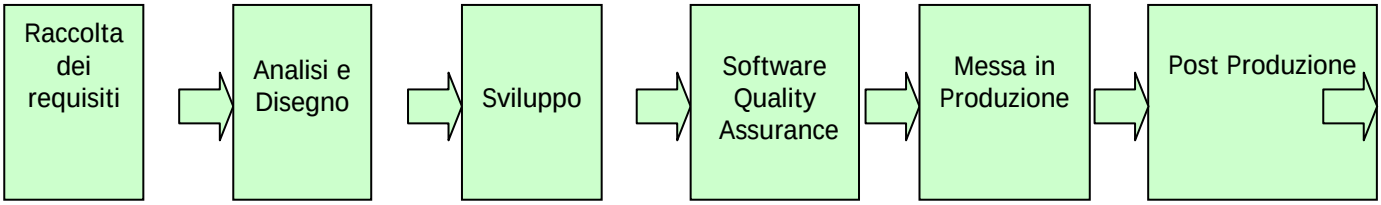
INDICE

1 Introduzione

Le attuali tendenze dello sviluppo delle applicazioni web del MEF richiedono una sempre maggior attenzione agli aspetti di sicurezza nello sviluppo ed utilizzo delle stesse. Infatti, nell'ultimo anno, ad uno scenario d'applicazioni esclusivamente ad uso interno (dominio Tesoro) si è progressivamente sostituito uno scenario d'applicazioni che offrono servizi all'esterno sia ad utenza extranet sia ad utenza internet. Questo, da un certo punto di vista, significa che le applicazioni vanno a posizionarsi sul perimetro esterno della rete MEF e quindi, se non correttamente sviluppate potrebbero costituire una breccia.
È quindi opportuno descrivere alcune linee guida di programmazione che, indipendentemente dall'implementazione Java o .NET, siano d'ausilio allo sviluppo ed eliminare, per lo meno, quelle che sono le vulnerabilità più comuni e note delle applicazioni web.
Queste linee guida di programmazione sono comprensive anche di aspetti infrastrutturali già presenti o in corso di realizzazione nel MEF che sono di supporto e completamento alla sicurezza delle applicazioni.

Le indicazioni descritte nel presente documento devono considerarsi di riferimento sia per lo sviluppo di applicazioni WEB "critiche" che "non critiche".
La criticità di un'applicazione Web viene stabilita, in fase di definizione dei requisiti, con il supporto dell'Unità di Sicurezza Informatica. Viene compilato un questionario per la valutazione della criticità dell'applicazione, in base a quanto descritto nel documento di "Linee guida per la definizione della criticità di applicazioni Web e per lo sviluppo di applicazioni critiche". Qualora si ritenga l'applicazione "critica", l'Unità di Sicurezza Informatica compila il documento dei "Requisiti di Sicurezza" da implementare per l'applicazione.

Allo scopo di descrivere il processo legato alla sicurezza di applicazioni web è importante definire termini e processi della metodologia di sviluppo, infatti sia che il processo di sviluppo utilizzi il modello "extreme programming" (XP), quello iterativo o quello "waterfall", di fatto, questi modelli sono tutti, in maniera più o meno marcata, riconducibili alle fasi rappresentate in figura.



Il processo di messa in sicurezza di un'applicazione web ha inizio già durante la fase di raccolta dei requisiti con l'individuazione dei requisiti utente o "Use Case di business", descrittivi dei processi funzionali di un sistema. Ogni "Use Case" deve contenere le informazioni necessarie per rispondere alla domanda: "quale processo assicura che i dati del sistema e le informazioni utente sono sicure?". Ne deriva che la sicurezza deve essere vista come un processo specifico e modellata opportunamente.
Per esempio, la necessità di provvedere a funzionalità di autenticazione degli utenti oppure di dover garantire determinati livelli di riservatezza sui dati, sono requisiti di sicurezza dell'applicazione che devono essere stati opportunamente individuati e descritti nel documento di specifica dei requisiti dell'applicazione, come aspetti particolari dei Requisiti Funzionali o Requisiti non Funzionali e, per applicazioni ritenute "critiche", nel documento "Requisiti di Sicurezza" compilato dall'Unità Organizzativa di Sicurezza.

Accanto alla necessità di prevedere la modellazione dei processi funzionali legati alla sicurezza, devono essere descritte, per poi essere sviluppate, le regole che descrivono aspetti funzionali che non possono essere rappresentati come processi ad esempio la disattivazione di un'utenza dopo un periodo d'inattività.

La sicurezza delle applicazioni web è completata poi dai requisiti tecnici o non funzionali; tali requisiti non derivano da necessità utente ma definiscono le caratteristiche delle infrastrutture sottese alle applicazioni. Un esempio di requisito non funzionale o requisito di sicurezza è quello di garantire la protezione delle applicazioni da attacchi esterni (es. "Apertura delle sole porte 80 e 443 sulla web farm di produzione o il time out delle sessioni dopo 10 minuti d'inattività").

In modo analogo, nella fase di analisi e disegno, quando vengono sviluppati i diagrammi delle classi e delle interazioni, è necessario che vengano inseriti "Security Object" che contengano attributi e metodi atti a prevenire attacchi al sistema. Un esempio potrebbe essere il blocco di dati di input che non rispettino il formato previsto.

La fase dello sviluppo è la più complessa, infatti, spesso gli standard di programmazione sono legati a linguaggi e piattaforme ma, in ogni caso, è sempre opportuno aggiungere alcuni standard o meglio "best practices" che si accompagnino alle regole di nomenclatura ed alla documentazione dell'applicazione, quali ad esempio:

- applicare sempre controlli ai dati di input;
- non memorizzare sul client nessun dato per banale che sia;
- gli errori devono generare messaggi generici e comprensibili per l'utente, ma messaggi ben documentati per il personale di supporto;
- l'ereditarietà, polimorfismo ed incapsulamento devono essere utilizzati quanto più possibile;
- le variabili di ambiente devono essere utilizzate con moderazione controllandone sempre formato e buffer.

L'utilizzo di standard di programmazione può non essere sufficiente, qualora non sia prevista una fase di Software Quality Assurance che permetta di individuare difformità prima di passare alla fase di test delle applicazioni stesse. Tale fase, realizzata a campione e per applicazioni particolarmente critiche, prevede un'ispezione preventiva del codice prodotto.
In fase di test è necessario che l'ambiente in cui l'applicazione è testata (ambiente di collaudo) abbia tutte le caratteristiche di sicurezza dell'ambiente di esercizio questo al fine di evitare la segnalazione di vulnerabilità legate all'ambiente e non a bachi dell'applicazione.

La sicurezza dell'ambiente di esercizio è un elemento fondamentale, un'applicazione può aver superato la verifica della Software Quality Assurance ma diventare vulnerabile qualora l'ambiente d'esercizio fosse insicuro.
È necessario quindi che i server di questo ambiente siano installati e configurati in maniera sicura con procedure documentate e periodicamente controllate.

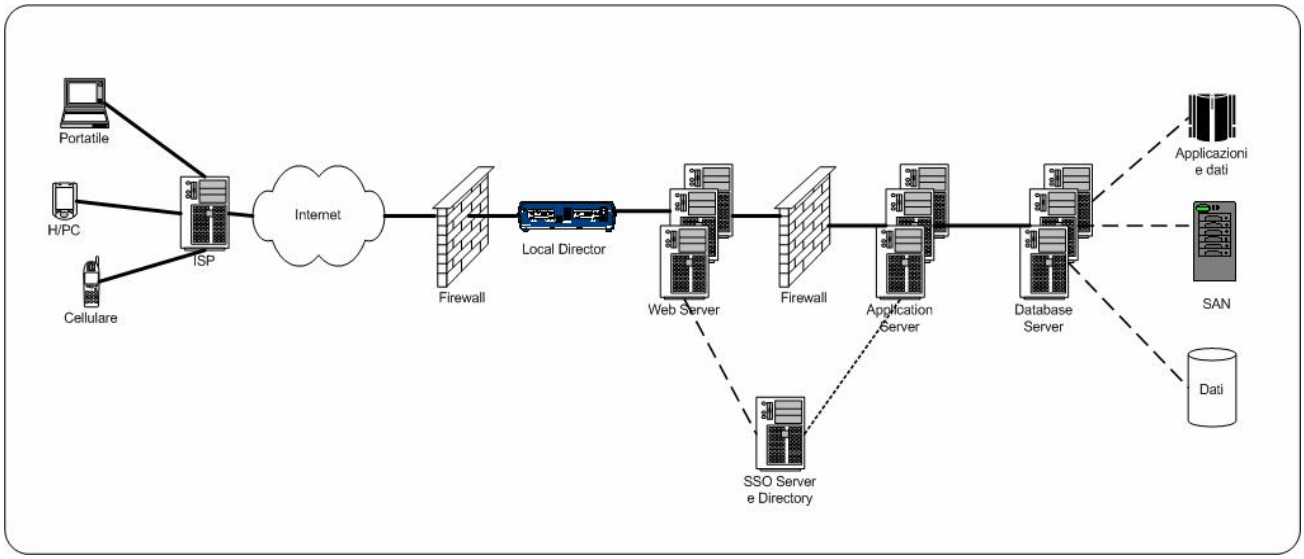
Tra le procedure indispensabili si individuano:

- il piano di backup e recovery;
- la limitazione degli accessi al server ed il cambio periodico delle password di amministratore;
- la chiusura delle porte dei Web server ad esclusione di quelle necessarie alle applicazioni (80 e 443);
- la corretta definizione e segmentazione delle connessioni di rete;
- l'applicazione costante delle patch ai sistemi operativi;
- eliminazione di tutti i file obsoleti dai web server;
- eliminazione di tutti i servizi non necessari dai web server.

Infine è bene considerare che le applicazioni non sono statiche, sarà quindi necessario instaurare dei processi periodici che verifichino che le applicazioni in produzione, non sviluppino vulnerabilità a causa di modifiche anche semplici del codice.

2 Schema architetturale di riferimento

La figura che segue riporta quella che è oggi nel MEF l'infrastruttura di riferimento a supporto della maggior parte delle applicazioni a tre livelli e quindi dei siti ad esse sottesi. Per semplicità di lettura l'infrastruttura rappresentata è quella esposta verso Internet ma i siti interni sono dotati di una topologia del tutto simile.



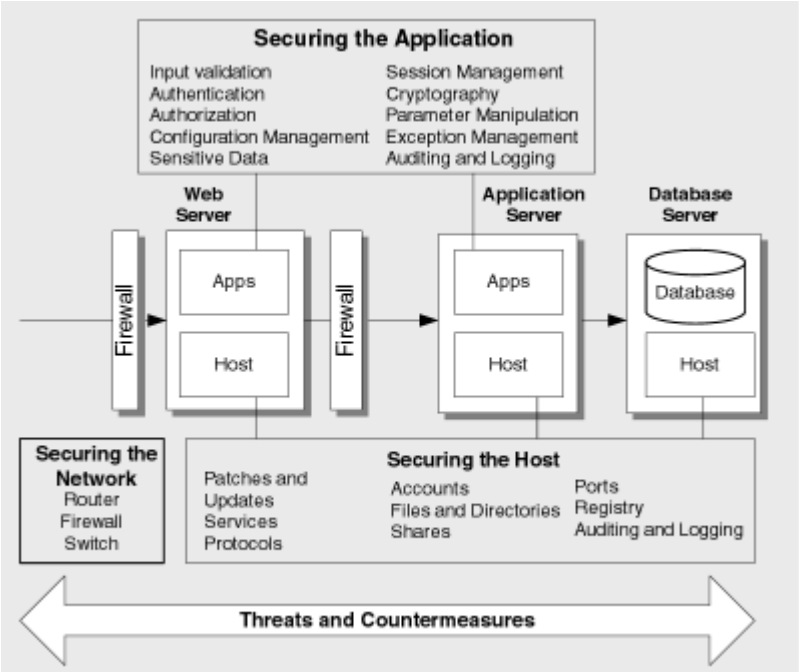
I web server sono organizzati in "web farm" gestite da un dispositivo di distribuzione del traffico HTTP il "Local Director". Ciascuna farm è costituita da più macchine fisiche ma si presenta con un unico indirizzo IP. Alla data di stesura del documento sono state realizzate le farm per IIS, per il listener HTTP Apache di Websphere, .NET e per il Oracle.

L'impianto prevede che non sia presente codice applicativo a questo livello, fa eccezione per caratteristiche architetturali la farm .NET che prevede la presenza di parte del framework ASP.NET nella componente listener.

Gli Application Server, così come i Database Server (o altre fonti di dati), sono protetti da un ulteriore livello di firewall e non sono accessibili direttamente dall'utenza.

3 Linee Guida di programmazione

La figura, che segue, descrive la sicurezza di un'applicazione web come il risultato delle misure di sicurezza applicate su ogni livello fisico e logico.



In questo capitolo vengono descritte alcune raccomandazioni di programmazione che, indipendentemente dall'implementazione di un'applicazione web a tre livelli, Java o .NET, possano essere di supporto ad una programmazione ed utilizzo sicuro delle applicazioni.
Per comprendere pienamente l'obiettivo di ciascuna raccomandazione e renderla più comprensibile, i paragrafi che seguono sono articolati in una breve descrizione delle diverse tipologie divulnerabilità possibili, sintetizzate nella tabella in figura, con indicazione delle azioni da mettere in atto come

contromisura

Vulnerability Category	Potential Problem Due to Bad Design
Input Validation	Attacks performed by embedding malicious strings in query strings, form fields, cookies, and HTTP headers. These include command execution, cross-site scripting (XSS), SQL injection, and buffer overflow attacks.
Authentication	Identity spoofing, password cracking, elevation of privileges, and unauthorized access.
Authorization	Access to confidential or restricted data, tampering, and execution of unauthorized operations.
Configuration Management	Unauthorized access to administration interfaces, ability to update configuration data, and unauthorized access to user accounts and account profiles.
Sensitive Data	Confidential information disclosure and data tampering.
Session Management	Capture of session identifiers resulting in session hijacking and identity spoofing.
Cryptography	Access to confidential data or account credentials, or both.
Parameter Manipulation	Path traversal attacks, command execution, and bypass of access control mechanisms among others, leading to information disclosure, elevation of privileges, and denial of service.
Exception Management	Denial of service and disclosure of sensitive system level details.
Auditing and Logging	Failure to spot the signs of intrusion, inability to prove a user's actions, and difficulties in problem diagnosis.

3.1 Autenticazione

L'autenticazione (Authentication) è il processo volto a determinare se un utente o un'entità sia chi dichiara di essere. In un'applicazione web è facile confondere l'autenticazione con la gestione delle sessioni.
Gli utenti sono tipicamente autenticati tramite userid e password o strumenti simili. Quando un utente è autenticato, viene posto, nel browser utente in un token di sessione (o cookie),; questo permette al browser di inviare il token ogni qualvolta invia una richiesta effettuando, quindi, un'autenticazione di entità.
L'autenticazione di un utente avviene una volta per sessione mentre l'autenticazione d'entità avviene con ogni richiesta.

Userid e password costituiscono nell'ambito del MEF lo strumento più utilizzato per l'autenticazione. Per garantire un adeguato livello di sicurezza è necessario che alle password siano applicate politiche che ne aumentino la robustezza e che esistano strumenti che permettano la disattivazione di un'utenza qualora non sia utilizzata.

A questo scopo in ambito MEF è stata adottata un'infrastruttura di Access Management e Single-Sign-On per applicazioni Web multilivello di impostazione molto moderna, che presuppone un modello di accesso alle applicazioni basato sul Ruolo ("RBAC", Role Based Access Control). Per un dettaglio maggiore, consultare i documenti di linee guida, inseriti nel raccoglitore degli standard aziendali Consip paragrafo "Standard di programmazione", relativi alle modalità d'integrazione delle applicazioni Java e DOT.NET con Oracle Login Server.
Secondo questo paradigma le applicazioni non gestiscono più in alcun modo le informazioni di anagrafica utenti, che vengono portate fuori dai rispettivi database e gestite in un unico repository LDAP centralizzato.

Al momento dell'autenticazione, l'Access Manager fornisce alle applicazioni la Userid e le credenziali dell'utente che chiede di accedere (ovvero il ruolo per quella applicazione). Alle applicazioni rimane solo la gestione dell'associazione tra le credenziali utente e le funzioni o le viste sui dati che quel particolare ruolo ha diritto/facoltà di esercitare.

L'SSO Server (Single Sign-on Server della Oracle) è il prodotto adottato in ambito MEF e costituisce la naturale estensione del prodotto "Login Server" adottato nel corso del 2001.

Il prodotto SSO mantiene tutte le funzionalità del Login server introducendo la nuova e potente funzionalità di registrazione diretta dei listener HTTP.

I più importanti vantaggi sono:

1. l'integrazione con le applicazioni diventa ancora più immediata, in quanto l'interazione con l'Access Manager è mediata dal Web Server; l'applicazione non deve più verificare continuamente che l'utente sia autorizzato (nella vecchia modalità di integrazione con il sistema di SSO tutte le maschere devono verificare l'esistenza e la validità del cookie di sessione, nella nuova questo è un compito del Web Server). Il processo di autenticazione si svolge nei seguenti passi:
   - installazione di un "plug-in" sul Web Server (attualmente disponibile per Apache e IIS);
   - registrazione del listener sul sistema di SSO;
   - configurazione delle URL da "proteggere" (ad esempio :http://applicazioneX/*)
   - quando arriva una richiesta di accesso per una URL protetta, il plug-in la intercetta, verifica che il client richiedente sia autenticato (in caso contrario esegue la redirect sull'Access Manager per il riconoscimento) e quindi concede l'accesso impostando l'informazione nella variabile di enviroment del browser `http_sso_user`;
   - l'applicazione tramite il metodo get_cgi_env acquisisce l'informazione relativa all'utente;
   - con questa informazione, via protocollo LDAP o via API, l'applicazione reperisce sull'OID le informazioni sul profilo dell'utente.
2. Diviene possibile "proteggere" l'accesso diretto a URL sul Web Server e quindi a risorse fisiche.

Per le interazioni con l'access manager sono stati sviluppati due package, uno per le applicazioni in tecnologia Java (j-sso_sdk), uno per quelle in tecnologia Microsoft (ms_sso_sdk), contententi tutte le API necessarie.

3.1.1 Linee guida per l'integrazione delle applicazioni con SSO.

Le applicazioni, per l'integrazione con il sistema di Single Sign On, utilizzeranno come metodo preferenziale la registrazione del web server anche se sarà mantenuto anche il metodo precedente sia per compatibilità con le applicazioni esistenti sia per gestire l'assenza di plug-in specifici.

Il plug-in necessario per la registrazione del Web Server è oggi disponibile per:

- Apache
- IIS
- Iplanet

Il sistema è quindi compatibile con tutte le applicazioni Oracle (comprese quelle Java) e con quelle su server Microsoft. Le applicazioni già esistenti potranno continuare ad usare il Login Server senza problemi, prevedendo eventuali interventi di manutenzione evolutiva qualora si decida di sostituire i gruppi attuali per la profilazione degli utenti con gli objectclass di SSO.. Ma non si ritiene necessario chiedere di stravolgere l'impianto per registrare il Web Server.

L'interazione delle applicazioni con l'SSO dovrà avvenire esclusivamente via API secondo lo schema rappresentato in figura:

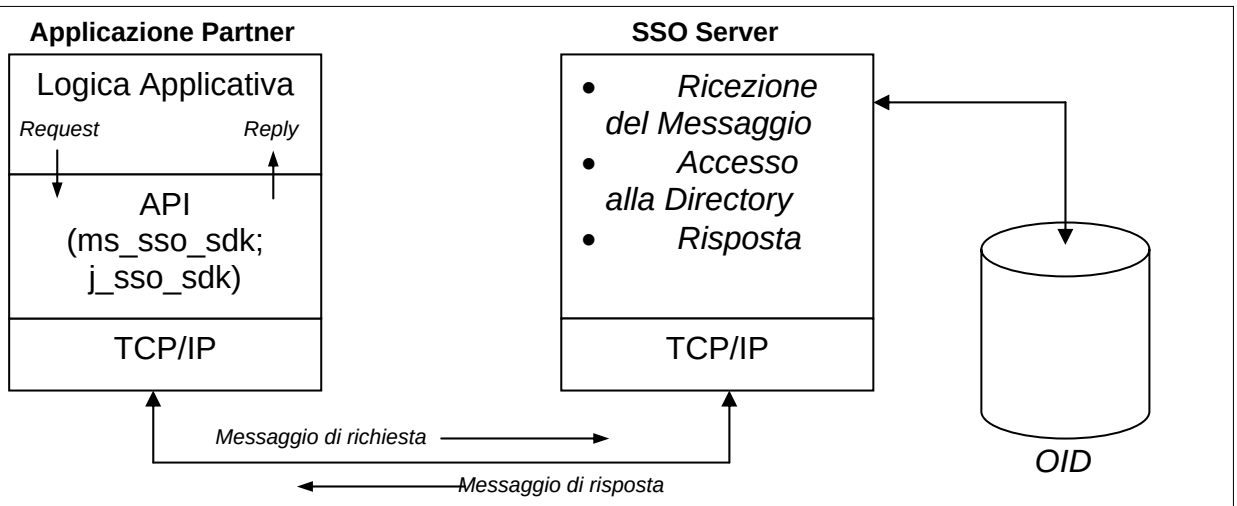


Figura 1 - L'interazione tra applicazioni e SSO Server

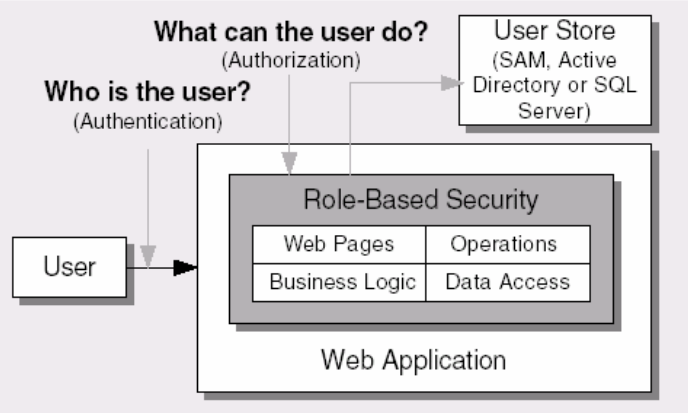
L'accesso diretto via LDAP all'OID, pur essendo in linea teorica possibile, è sconsigliato per due motivi, uno di ordine prestazionale, il secondo, più importante, per non "cablare" la struttura DIT dell'LDAP in alcun modo nel codice dell'applicazione (ad esempio, per ottenere via LDAP i valori degli attributi contenuti in un objectclass di un utente è necessario conoscere la collocazione della Entry dell'utente nell'alberatura).

3.2 Controllo dell'accesso ed autorizzazione

I meccanismi di controllo degli accessi (Authorization) costituiscono un elemento cruciale nel disegno della sicurezza delle applicazioni. In generale un'applicazione web dovrebbe proteggere i

dati ed i sistemi di front-end e back-end implementando restrizioni su cosa un utente può fare, a quali risorse può accedere e quali funzioni eseguire sui dati.

I termini "autorizzazione" e "controllo degli accessi" sono spesso confusi. Autorizzazione è la verifica che l'utente abbia gli opportuni permessi per accedere a dati e funzioni, l'autorizzazione è legata alle credenziali dell'utente e determina l'appartenenza dello spesso a specifici gruppi preimpostati. E' evidente come qualsiasi controllo degli accessi sia dipendente dal sistema di autenticazione che definisce le autorizzazioni.



Il termine "controllo degli accessi" si riferisce più genericamente al modo con cui si controlla l'accesso alle risorse web includendo, tra questi, restrizioni basate su cose quali: l'ora del giorno, l'indirizzo IP di provenienza, il dominio http dell'utente, il possesso di un qualsiasi tipo di token hardware/software.

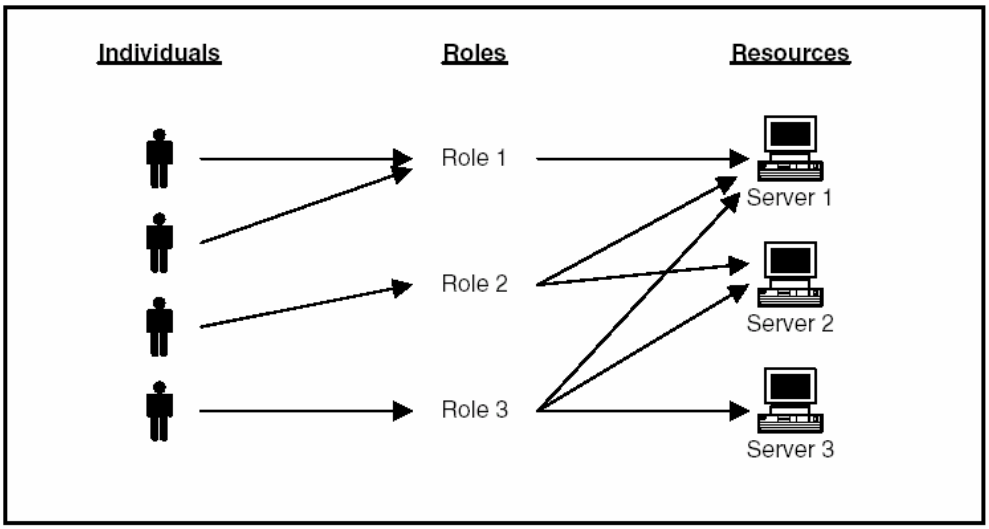
Esistono diversi modelli/tecnologie di controllo degli accessi ma quelli principali sono:
- DAC (Discretionary Access Control)
- MAC (Mandatory Access Control)
- RBAC (Role Based Access Control)

Il concetto base del DAC è che chi possiede i dati è in grado di controllare l'accesso agli stessi.
Le ACL (Access Control List) possono essere considerate un'implementazione del DAC. In questo modello l'accesso alle risorse è basato sull'identità dell'utente e su regole che specificano quale utente abbia accesso a quali dati.

Il modello MAC invece rende sicuri i dati associando a ciascuno di essi una "etichetta" che ne descrive il livello di sicurezza (per es. pubblico, riservato, segreto etc.) e verificando questa "etichetta" con il livello di sicurezza in cui sta operando l'utente. Per determinare se un utente abbia o meno l'accesso a determinati dati, sono utilizzati due principi:
- l'utente può leggere documenti con livello di sicurezza inferiore;
- l'utente può scrivere documenti con un livello di sicurezza superiore.

Nel modello RBAC il controllo degli accessi è basato sul ruolo e responsabilità dell'utente nell'ambito di un'organizzazione. Questo modello è stato disegnato per gestire centralmente i permessi definendo dei livelli di astrazione (ruoli) che sono mappati one-to-any con gli utenti, le

funzioni e le risorse. Quest'approccio riduce la complessità, in questo modello i permessi sono associati ai ruoli e gli utenti sono associati ai ruoli acquisendo quindi i permessi legati ad essi.
I ruoli sono creati e gestiti centralmente e questo rende possibile riassegnare un utente da un ruolo all'altro.



Il modello RBAC è il modello che viene utilizzato,come detto, dalle applicazioni web del MEF. Questa scelta deriva sia dalla facilità gestionale ma soprattutto dalla robustezza insita nel modello. L'utente avrà attribuito solo il livello minimo di permessi necessari per compiere le sue attività, e questo si traduce nell'impossibilità di accedere a dati e funzioni al di fuori della autorizzazione assegnata.

3.3 Gestione delle sessioni utente

L'HTTP è il protocollo basato su TCP, utilizzato nelle comunicazioni tra client e server in ambito web. Questo protocollo definisce una semplice interazione di tipo request-response che convenzionalmente viene definita "transazione web". L'HTTP è definito come protocollo "stateless" poiché non include il concetto di sessione o interazione che superi la risposta alla richiesta formulata. È quindi necessario, applicare un "meccanismo di stato" affinchè tutte le richieste, provenienti da un unico utente, siano associate tra loro in una "sessione" ("Session Management").
Questo meccanismo viene realizzato a livello di applicazione ed una sua implementazione non corretta può determinare severi rischi per l'applicazione stessa.

La maggior parte delle implementazioni di questo meccanismo si basa sull'utilizzo dei "cookies". I "cookies" furono introdotti da Netscape ed oggi sono descritti nel RFC2965. I "cookies" non sono stati disegnati per contenere alcun tipo d'informazione sensibile, questo, insieme ad alcuni accorgimenti, può essere utile per comprendere come utilizzarli correttamente. I cookies

tipicamente contengono un token , ossia un numero unico, non predicibile, in grado di resistere ad operazioni di "reverse engineering", che rappresenta la sessione[1].

Tra gli accorgimenti che devono essere rispettati nel caso di utilizzo di cookies i principali sono:

- Session Time-out: i cookies devono prevedere una scadenza, per limitare le possibilità di riuso, spesso alcuni sistemi prevedono la rigenerazione automatica dopo un numero definito di iterazioni;
- Ri-autenticazione: prima di eseguire operazioni critiche è opportuno chiedere all'utente di fornire la propria password;
- Trasmissione dei cookies: i cookies devono essere trasferiti esclusivamente su canale crittografato;
- Logout: qundo un utente esegue il logoff devono essere cancellati in modo definitivo tutti i cookie

Si sottolinea che opzioni quali "Remember me" sono assolutamente da evitare.

3.4 Registrazione degli eventi (Logging)

La registrazione degli eventi (logging) è essenziale per fornire elementi chiave relativi ad un'applicazione ed alle operazioni ad essa associate.

Questo log è destinato a registrare principalmente le operazioni di modifica alle basi dati e quelle relativi agli errori che, intercettati dall'applicazione, devono qui essere riportati per poter intervenire a loro correzione.

Questi log:
- possono rivelarsi come uno strumento per individuare comportamenti sospetti;
- forniscono una traccia delle azioni degli utenti;
- permettno di ricostruire gli eventi dopo che un problema si è verificato, facilitando il processo di recovery;
- possono, in alcuni casi, essere utilizzati in processi legali come testimonianza di utilizzo doloso, si noti che in questo ultimo caso l'infrastruttura deve garantire l'inalterabilità dei dati di log.

Gli eventi devono essere registrati accompagnati dall'indicazione della data ed ora dell'evento e dell'utente o processo che ha effettuato l'operazione.

Si suggerisce un'ipotesi di nomenclatura quale:
XXX_Logging.*

Dove XXX è l'acronimo dell'applicazione e * l 'estensione del file dipendente dalla piattaforma.

[1] Per implementare ulteriori livelli di sicurezza potrebbe essere necessario vincolare un token di sessione ad una specifica istanza di un client http, per fare ciò un metodo è la generazione di token di pagina e mantenere questa associazione sul server.

Questi log, da includere nelle procedure giornaliere di backup, devono essere periodicamente (almeno settimanalmente) chiusi, archiviati e ricreati.

Questi log accompagnati dai dati di Audit dei sistemi operativi, DB etc. permettono una completa traccia delle attività in un sistema informativo.

Si ricorda che la scrittura di un file sequenziale è un'operazione onerosa e che quindi tale file non deve essere utilizzato, specie in ambiente di esercizio, per "trace" dettagliate dell'attività dell'applicazione.

3.5 Gestione degli errori

Una non corretta gestione degli errori determina rischi potenziali per un'applicazione. Infatti, senza un'opportuna gestione informazioni quali, struttura dei database, versione del sistema operativo, stack etc possono essere visualizzati all'utente finale.
Questi sono dettagli che, di fatto, disturbano gli utenti e che possono trasmettere, ad un utente malevolo, informazioni utili ad individuare potenziali vulnerabilità.

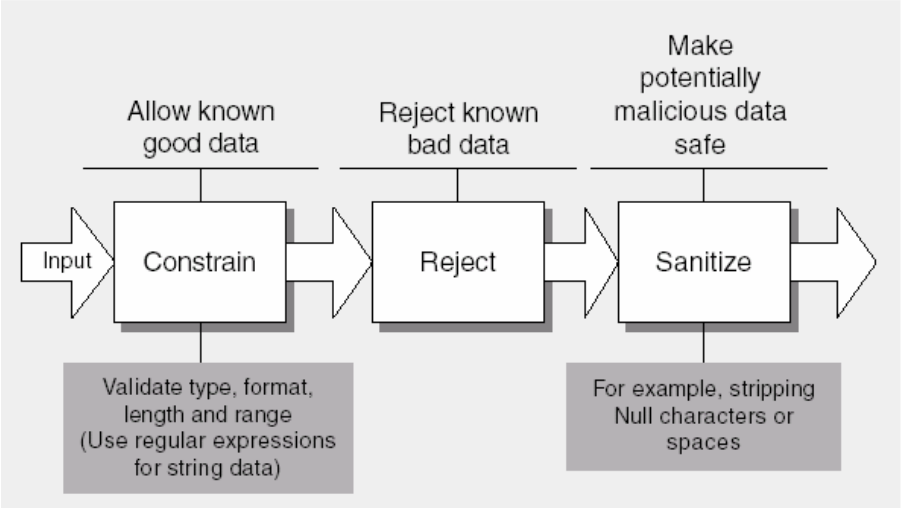
È quindi necessario applicare alcune best practices nella gestione degli errori:
- presentare all'utente esclusivamente messaggi di errore chiari e comprensibili e comunque non contenenti informazioni o diagnostici non utili a chi legge il messaggio;
- registrare dettagliate informazioni di errore su log su filesystem;
- inserire codice per la gestione e la cattura di eccezioni non previste. Quest'ultima indicazione fa sì che l'applicazione non rimanga mai in uno stato inconsistente e che l'evento sia registrato sul log per una successiva analisi.

Si rammenta, inoltre, che in ambiente di produzione/esercizio devono essere disattivate tutte le opzioni relative a funzionalità di DEBUG.

3.6 Validazione dei dati di input

La validazione dei dati di input è una delle contromisure più efficaci nella prevenzione di attacchi quali Cross-Site-Scripting, SQL injection, buffer overflows e altri attacchi basati sulla manipolazione dei dati di input.

Per creare una efficace strategia di validazione dell'input esistono tre approcci:
- accettare solo i dati validi;
- rifiutare i dati malevoli;
- bonificare i dati



Il primo approccio è quello da privilegiare: le applicazioni devono filtrare ogni input basandosi su caratteristiche quali tipo di dato, lunghezza, formato, caratteri ammissibili etc. A queste verifiche realizzate con script dal lato client è però necessario integrare anche verifiche dal lato server.

Infatti se la validazione dei dati lato client è accettabile in termini di rapidità e facilità d'uso, per contro, non può essere considerata esaustiva perché esiste la possibilità di disattivare gli script di controllo lato client. È quindi necessario, per una sicurezza più effettiva, che sul server siano presenti routine di validazione dell'input anche se apparentemente questo può essere ritenuto ridondante.

L'approccio che rifiuta dati malevoli richiede che l'applicazione sia in grado di riconoscere i "pattern" /variazione di tali dati. È un approccio meno efficace e soprattutto meno robusto infatti mentre i dati validi rimangono costanti nel tempo questo non è vero per i "pattern" dei dati malevoli che cambiano costantemente.

La bonifica dei dati è l'ultimo approccio e prevede di rendere inoffensivi dati potenzialmente dannosi. È un approccio che si rivela utile quando il range dei dati permessi non garantisce che questi siano inoffensivi. Un esempio può essere l'assunzione che nessun input venga considerato come eseguibile ma trattato come semplice testo.

È comunque possibile sommare i tre approcci ma risulta imprescindibile che il primo metodo sia sempre applicato accompagnato da una verifica lato server.

Si sottolinea che tutte le pagine che trattino dati dinamici non devono essere poste in cache e devono prevedere le opportune istruzioni perché siano scartate dai meccanismi di caching.

3.7 Crittografia

3.7.1 Cifratura dati su HTTP

Nei frequenti casi in cui è necessario proteggere integrità e riservatezza dei dati trasferiti è necessario prevedere l'utilizzo di HTTPS (SSL/TLS su HTTP). Devono essere evitate situazioni in cui vi sia trasferimento di informazioni fra cliente e server con protocolli diversi da HTTP.

Per ragioni prestazionali si raccomanda, nel disegno dell'applicazione, di separare, per quanto possibile i dati sensibili da quelli pubblici in modo da limitare l'impiego del protocollo HTTPS alle sole pagine con dati sensibili.

Qualora questo fosse assolutamente necessario, l'invio di informazioni critiche deve essere comunque protetto o mediante l'applicazione di SSL/TLS ad altri protocolli, oppure mediante altri meccanismi di protezione (es. SSH o simili).

L'invio di password non dinamiche deve essere sempre protetto mediante cifratura.

3.7.2 Meccanismi di non ripudio mediante PIN

Per alcune operazioni è previsto che l'utente debba fornire espressa conferma all'input di informazioni e/o all'attivazione di processi elaborativi o di trasferimento delle medesime. Il meccanismo previsto per impedire che dette operazioni possano essere eseguite senza una effettiva volontà da parte dell'operatore consiste nel codificare il blocco del tasto "Invio". Lo sblocco di detto tasto, e di conseguenza la possibilità di effettuare l'operazione, avviene solo se l'utente inserisce un corretto PIN.

3.7.3 Firma digitale

Secondo gli standard consolidati e la normativa vigente, la firma digitale di informazioni (firma qualificata avanzata realizzata secondo la normativa Italiana precedente al D. Lgs. 10/2002) è realizzata con strumenti esterni alle applicazioni che si interfacciano con l'infrastruttura a chiave pubblica e l'utente e producono documenti firmati in forma di file.

Per quanto riguarda le applicazioni di particolare criticità, deve essere eseguito un controllo sulla corrispondenza del Codice Fiscale indicato nel certificato utilizzato per la firma e quello memorizzato nel database utenti relativo all'utente che ha eseguito l'operazione di firma ed invio delle informazioni firmate.

Se l'applicazione deve gestire informazioni con associata una firma digitale, è necessario, in fase di progettazione e realizzazione dell'applicazione, prevedere quanto segue:

Immissione informazioni firmate

1. L'applicazione deve riconoscere di trovarsi in uno stato in cui è previsto l'input di informazioni firmate ed evidenziare ciò all'utente. L'applicazione deve permettere all'utente di indicare la locazione del file firmato e di avviare il trasferimento del file mediante un meccanismo di FTP o FTP su HTTP.

2. L'applicazione deve procedere alla verifica del formato del file immesso in modo da garantire che i file dichiarati ricevibili siano effettivamente firmati digitalmente.

Gestione informazioni firmate

1. La definizione delle modalità di memorizzazione, elaborazione trasferimento delle informazioni firmate deve tenere conto delle caratteristiche dell'informazione firmata. In particolare deve essere sempre mantenuta l'associazione fra l'informazione e la relativa firma.

Verifica della firma

1. L'applicazione deve poter riconoscere uno stato in cui è necessaria o possibile una verifica di firma di informazioni e fornire all'utente evidenza di ciò.

2. L'applicazione deve poter trasferire al sistema di verifica esterno all'applicazione l'informazione nel formato previsto dal sistema di verifica.

3.8 Firma del codice applicativo

Le applicazioni WEB, vista l'evoluzione e la sempre crescente complessità che raggiungono, richiedono sempre più spesso l'uso di codice residente sul client e quindi di applet e ActiveX [2].

Con il termine 'applet' viene comunemente indicato del codice scaricabile da server a client unitamente al contenuto di una pagina web HTML. Tale "embedded object" Java è però soggetto a particolari condizioni di sicurezza che ne restringono le potenzialità.
Generalmente si dice che un 'applet' gira dentro una 'sandbox', ovvero un contenitore di sicurezza, dall'interno di tale contenitore, l'applet, e' impossibilitata ad accedere a risorse locali del client, ad aprire connessioni di rete che non siano verso lo stesso host dal quale e' stata scaricata, e così via..

Per ovviare a questa serie di limitazioni e' comunque possibile far garantire i permessi necessari ad un applet per poter operare in maniera corretta. A tal fine è necessario disporre di un certificato, e quindi di una coppia di chiavi, con cui 'firmare' elettronicamente un archivio in cui risiede l'applet compressa.

Per gli ActiveX, non esistono le stesse restrizioni, ed è quindi molto più pericoloso scaricarli in locale, in quanto il codice potrebbe contenere virus, dialer o comunque possono essere vettori di

[2] Quanto segue è applicabile anche ad altro tipo di codice destinato a risiedere sul client quali driver, macro Office etc.

codice malevolo. Microsoft ha riconosciuto la la debolezza di tale tecnologia, per cui consente agli utenti di bloccare il download di ActiveX non firmati dal browser.

Nell'ambito del MEF, per conciliare la necessità di sicurezza con quella di non impedire l'utilizzo di codice residente sul client, si è scelta la strada della firma di tale codice.

Sono stati acquisiti dalla società Verisign due "Digital ID" ossia una coppia di certificati, uno per oggetti Java ed uno per oggetti Microsoft, che permettono di firmare il codice.

Quest'operazione garantisce l'utente che scarica il codice di applicazioni MEF sotto due punti di vista:

- che il codice effettivamente proviene da una fonte nota (Consip per il MEF);
- che il software non sia stato alterato dal momento in cui viene firmato.

3.9 Ambiente di esercizio

In questo paragrafo si vogliono accennare alcune caratteristiche che devono essere presenti nell'ambiente d'esercizio di un'applicazione. Si sottolinea che non vuole essere una trattazione esaustiva ma piuttosto un'indicazione di quali accortezze sono messe in opera per garantire la sicurezza di un servizio/applicazione. Per maggiori dettagli si rimanda ai documenti di configurazione/installazione esistenti.

3.9.1 Configurazione Sistema Operativo

Un primo elemento di attenzione è riservato all'installazione dei sistemi operativi (Windows/Unix).

In fase di installazione dei server vengono osservate le seguenti regole:

- il sistema operativo viene installato al livello consolidato di patch di sicurezza mantenuto presso il CED di appartenenza, ed inserito nel processo di aggiornamento periodico delle stesse.
- gli account applicativi o di sistema, non usati, sono rinominati così come gli account di default;
- sono attivati i meccanismi di auditing in modo da registrare il verificarsi di eventi significativi dal punto di vista della sicurezza. Gli eventi registrati includono:
    - log-on e log-off;
    - tentativi di accesso al sistema riusciti e falliti;
    - tentativi di accesso a risorse e dati riusciti e falliti;
    - avvio e arresto del sistema;
    - avvio e arresto delle funzioni di audit;
    - la connessione e disconnessione dei device di input/output.

  La registrazione deve riportare almeno i seguenti dati:
    - identità dell'utente che ha scatenato l'evento;
    - data e ora dell'evento;
    - tipo dell'evento;

- oggetti coinvolti dall'evento (file, applicazioni, ecc.).

I dati relativi agli eventi registrati sono conservati per un periodo di tempo sufficiente alla loro analisi e/o utilizzazione a fini statistici, come prove da esibire in caso di dispute, come elementi da considerare nell'identificazione di misure migliorative della sicurezza.

- i servizi ed i protocolli non necessari sono disattivati.
- su alcuni filesystem sono applicate ACL più restrittive rispetto a quelle di default;
- viene installato l'antivirus (se piattaforma Windows) ed il server è inserito nell'infrastruttura Enterprise di aggiornamento.

Una volta installato ogni server diventa oggetto di aggiornamenti periodici delle patch di sicurezza e di controlli di Vulnerability Check

3.9.2 Web Server

Come descritto in precedenza i web server sono organizzati in web farm. La configurazione dei listener (Apache o IIS) prevede, come passi minimali:

- l'attivazione dei log di errore e di accessi nel formato WSC (per questi ultimi ai fini statistici è in corso di realizzazione un'infrastruttura di raccolta e trattamento basata sul prodotto WebTrends);
- la rimozione di tutte le applicazioni/servizi demo, nonché ogni altro servizio, utility di sistema o funzionalità non strettamente necessaria;
- la disattivazione degli account di default;
- la configurazione del controllo d'accesso per gli utenti anonimi in base ai seguenti criteri:
  - abilitare l'accesso sui contenuti statici (.txt, .gif, .jpg, .html) in sola lettura;
  - abilitare l'accesso agli eseguibili, fruibili via web, in sola esecuzione;
  - abilitare l'accesso agli script (.asp, .php, ecc.), fruibili via web, in sola esecuzione;
  - abilitare l'accesso ai file di tipo 'include' (.inc, .shtm, ecc.), fruibili via web, in sola esecuzione.
- l'attivazione, se richiesto, del supporto https con l'installazione dell'apposito certificato.

Nel caso di siti istituzionali a contenuto informativo è poi opportuno:

- che siano definite procedure formali per la pubblicazione di informazioni su siti web che prevedano:
  - controllo dell'attendibilità delle fonti da cui sono state tratte le informazioni;
  - protezione delle informazioni nel caso di memorizzazione su supporti temporanei precedenti alla pubblicazione;
  - workflow formalizzato per la pubblicazione delle pagine web;
  - verifica periodica dell'attendibilità delle pagine pubblicate;
  - eventualmente l'utilizzo di strumenti per la salvaguardia dell'integrità delle informazioni pubblicate (ad esempio, la firma delle pagine web).

3.9.3 Application Server

In modo analogo a quanto previsto per i Web Server anche l'installazione delle piattaforme Application Server richiede le seguenti accortezze:

- l'attivazione dei log di piattaforma/sottosistema per il tracciamento degli errori (non di applicazione);
- la rimozione di tutte le applicazioni/servizi demo, nonché ogni altro servizio, utility o funzionalità non strettamente necessaria;
- la disattivazione degli account di default.

Per queste piattaforme risulta particolarmente importante la restrizione delle autorizzazioni all'interfaccia amministrativa, che deve essere acceduta solo con autenticazione e da un ristretto numero di utenti appartenenti esclusivamente alla struttura di gestione sistemistica e possibilmente solo dalla rete locale.