

UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

The background of the page features a large, faint watermark of the University of Padua seal. The seal is circular and contains a central shield with two figures: a woman on the left holding a wheel and a man on the right holding a staff. The shield is flanked by two stars. The text 'UNIVERSITAS STUDII PADOVAE' is written around the perimeter of the seal, and 'MCCXXII' is at the bottom.

**Integrazione di metodi per
l'accesso sicuro in un sistema
informativo sanitario**

Relatore: prof. Carlo Ferrari

Correlatore: prof. Diego Cecchin

Correlatore: dott. Paolo Mosca

Laureando: Massimiliano Sciacco

12 Marzo 2012

Ringraziamenti

Desidero innanzitutto ringraziare il prof. Carlo Ferrari che ha reso possibile la collaborazione tra il Dipartimento di Ingegneria dell'Informazione e il Policlinico Universitario di Padova, permettendomi di maturare un'esperienza importante e conoscere delle persone splendide.

Ringraziamenti speciali vanno quindi al prof. Diego Cecchin e al dott. Paolo Mosca, non solo per l'incredibile disponibilità che hanno dimostrato nell'aiutarmi a sviluppare questo lavoro di tesi (nonostante tutti i loro impegni di lavoro), ma soprattutto per avermi fatto sentire come un loro collega, prima che uno studente laureando.

Ringrazio Luca e Felice, che se non fosse stato per loro, adesso non avrei di certo l'esperienza e le competenze web che ho.

Un ringraziamento speciale va a Riccardo, con il quale ho condiviso quasi interamente la mia carriera di studi e che anche dopo essersi laureato, ha continuato a sostenermi nei miei studi.

Un mega ringraziamento va ai miei genitori e a mio fratello, che hanno sempre creduto nelle mie capacità, a prescindere dal "voto che portavo a casa".

Infine, ma non meno importanti, ringrazio tutti gli amici e le persone che mi sono state vicine in questi anni di studio, che in modi diversi, ma ugualmente importanti, hanno saputo aiutarmi a credere in me stesso.

Abstract

L'avvento del web ha aperto nuove e illimitate opportunità all'industria dell'informazione. Professionisti di tutto il mondo hanno iniziato ad esercitare le proprie attività sul web, dalle banche alle librerie, dalle aste ai giochi, fino alle applicazioni in ambito sanitario. Inoltre, con la crescente popolarità di smartphone capaci di navigare su internet, molte di queste applicazioni diventano accessibili ovunque e in qualsiasi momento.

La possibilità di aggiornare e mantenere le applicazioni web senza distribuire e installare software su potenzialmente migliaia di client e la crescente diffusione di web browser multiplatforma, sono le ragioni principali dietro l'enorme successo delle applicazioni web. Tuttavia, a causa della loro natura altamente tecnica e complessa, tali applicazioni non sono tuttora pienamente comprese. Di conseguenza, i rischi ad esse associate, sono stati spesso sottovalutati, se non addirittura ignorati.

Solo di recente si sta realizzando l'importanza di adottare adeguate misure di sicurezza per proteggere le proprie applicazioni web dalla pirateria informatica. D'altra parte, in una realtà dove l'innovazione tecnologica è estremamente rapida e la quantità di concetti e competenze richieste aumentano in continuazione, riuscire a realizzare un'applicazione web priva di falle richiede un dispendio di risorse non indifferente.

In particolare inoltre, in ambiente sanitario, bisogna affrontare ulteriori complicazioni: la tutela della privacy, il monitoraggio dell'audit per il tracciamento delle responsabilità utente, l'autenticazione sicura, la cifratura dei dati sensibili. L'iniziativa IHE (Integrating the Healthcare Enterprise) promuove soluzioni basate su standard per investigare e risolvere queste esigenze in ambito sanitario, fornendo le linee guida necessarie allo sviluppo di applicativi conformi alle specifiche.

In questa tesi vengono investigati i principali requisiti di sicurezza che caratterizzano un'applicazione in ambito sanitario: l'autenticazione client-server, l'autorizzazione e il log delle attività. Lo sviluppo di un prototipo sperimentale di sistema, servirà a valutare l'effettiva applicabilità di tali requisiti e le complicazioni ad esse associate. L'implementazione è calata nel contesto del Policlinico Universitario di Padova (Dipartimento di Medicina: Servizio di Radiologia e Medicina Nucleare) e tiene quindi conto dell'attuale sistema informativo già in uso, promuovendo una soluzione modulare, flessibile e pertanto di facile integrazione.

Viene inoltre estesa l'effettiva applicabilità di tali sistemi nell'ambito più generale della sanità, evidenziando possibili miglioramenti e integrazioni con altri sistemi, tenendo conto anche dei vincoli normativi che attualmente la legge prevede.

Sommario

Sommario	vii
INTRODUZIONE	1
1 Le Applicazioni Web	3
1.1 Introduzione	3
1.2 Architettura	4
1.3 Punti deboli delle applicazioni web	7
1.4 Complicazioni in ambito sanitario	8
1.4.1 Politiche di sicurezza e privacy	9
1.5 Profili IHE	11
1.5.1 Technical Framework	11
1.5.2 Profili di integrazione	12
1.5.3 I profili di sicurezza	13
1.6 Policlinico Universitario di Padova	15
1.6.1 Evoluzione del sistema informativo	15
1.6.2 Altri strumenti software	16
1.6.3 Sviluppo del sistema	16
1.7 Aspetti legali	16
1.7.1 Testo unico sulla privacy	17
1.7.2 Privacy e amministratori di sistema	19
1.7.3 Codice dell'amministrazione digitale	21
MATERIALI E METODI	23
2 Autenticazione	25
2.1 Definizione	25
2.1.1 Tecniche di autenticazione	25
2.2 Vulnerabilità comuni ai sistemi di autenticazione con username e password	26
2.2.1 Username enumeration	26
2.2.2 Password guessing	27
2.2.3 Eavesdropping	29
2.3 Autenticazione con username e password basata su form	30
2.3.1 Attaccare i token di sicurezza	31
2.3.2 Cross-site request forgery	36

2.3.3	Input injection.....	37
2.4	Autenticazione con chiavi e certificati	42
2.5	Specifiche del profilo ATNA.....	44
3	Autorizzazione.....	45
3.1	Definizione.....	45
3.2	Analizzare i token di sessione	45
3.3	Attaccare le Access Control List	46
3.3.1	Directory traversal	47
3.3.2	Horizontal Privilege Escalation	47
3.3.3	Vertical privilege escalation.....	48
3.4	Contromisure	49
3.4.1	Autorizzazioni Apache	49
3.4.2	Sicurezza dei token di sessione	50
3.4.3	Log di sicurezza	50
3.5	Specifiche del profilo IHE	51
3.5.1	Principi di progettazione.....	51
3.5.2	Modelli di controllo degli accessi comuni	52
3.5.3	Sistema di controllo degli accessi.....	53
3.5.4	Principio “needs-to-know”.....	53
3.5.5	Esempio con thin client	54
4	Audit Trail e Log.....	57
4.1	Log applicativo	57
4.1.1	Web server log.....	58
4.1.2	Log applicazione.....	59
4.1.3	Strategie per l’individuazione degli attacchi.....	59
4.1.4	Individuare le vulnerabilità	61
4.2	Definizione di audit trail.....	65
4.3	Specifiche del profilo ATNA.....	66
4.3.1	Schema del messaggio di audit.....	69
5	Paradigmi e Strumenti	75
5.1	Paradigmi e metodi di progettazione.....	75
5.1.1	Software design pattern.....	75
5.2	Software e strumenti utilizzati.....	77
	RISULTATI	81

6	Autenticazione e Autorizzazione	83
6.1	Crono.....	83
6.2	Struttura del database per la gestione utenti.....	84
6.3	La classe User	87
6.3.1	Il file di configurazione.....	88
6.3.2	Implementazione della classe User	88
6.3.3	Sicurezza.....	93
6.4	Implementazione dei certificati.....	97
6.4.1	Entità certificanti autorizzate	102
6.4.2	Rivisitazione del database per la gestione utenti	102
6.4.3	Rivisitazione della classe User	103
6.4.4	Vantaggi e svantaggi nell'uso dei certificati	104
6.5	La classe UserManager.....	105
7	Log Applicativo	107
7.1	Struttura del database per la gestione dei log.....	107
7.1.1	Log applicativo	107
7.1.2	Audit trail.....	109
7.2	La classe Log.....	111
7.2.1	L'handler mysql.....	113
7.3	Prestazioni nella registrazione degli audit trail	114
8	Ambiente di Prova.....	115
8.1	Struttura dell'applicazione web	115
8.2	Descrizione del flusso operativo	115
8.3	Pagine di amministrazione.....	121
9	Conclusioni.....	123
9.1	Conclusioni.....	123
9.2	Miglioramenti e considerazioni personali.....	125
	Bibliografia	129
	Appendice.....	133
	Acronimi	133
	Elenco delle figure	135
	Normativa sulla privacy.....	137

PARTE I

INTRODUZIONE

Capitolo 1

Le Applicazioni Web

L'obiettivo di questo primo capitolo è di presentare una visione d'insieme delle applicazioni web, descrivendone l'architettura, i vantaggi e i punti deboli sotto l'aspetto della sicurezza. Vengono quindi descritte le complicazioni che sorgono in ambito sanitario e le iniziative proposte per affrontarle e risolverle. Una panoramica della situazione vigente presso il Policlinico Universitario di Padova, chiuderà quindi il capitolo fornendo un caso reale da analizzare.

1.1 Introduzione

Le applicazioni web si sono evolute notevolmente negli ultimi anni, e con i recenti progressi tecnologici nell'ambito della sicurezza, molti sistemi e applicazioni tradizionali basati su software possono essere migliorati migrandone le funzionalità verso un'applicazione di tipo web. Quelli che seguono sono solo alcuni dei vantaggi apportati da un'applicazione web:

1. Compatibilità multiplatforma

Gran parte delle applicazioni web sono molto più compatibili tra le diverse piattaforme rispetto ai software installati. Solitamente il requisito minimo richiesto è un browser web (Internet Explorer, Firefox, Safari, Chrome, ecc.). Questi browser web sono disponibili per una moltitudine di sistemi operativi e pertanto, eseguire le applicazioni web non è un problema, sia che si utilizzi Windows, che Linux o Mac OS. Questo vantaggio garantisce anche una significativa riduzione dei problemi legati ad hardware specifici o impostazioni del sistema operativo.

2. Maggiore praticità

I sistemi web devono essere installati solamente sul server, permettendo alle workstation degli utenti finali di mantenere i requisiti al minimo. Ciò rende la manutenzione e l'aggiornamento di tali sistemi molto più semplice, poiché può essere realizzata interamente sul server. Inoltre, nello stesso momento, tutti gli utenti hanno accesso sempre all'ultima versione del prodotto.

3. Completa accessibilità

Un'applicazione web ha l'enorme vantaggio di essere accessibile ovunque e in qualsiasi momento. Inoltre la recente diffusione di dispositivi mobili capaci di navigare sul web (smartphone e tablet in primis), ha reso questa tecnologia ancora più conveniente.

4. Elevato grado di distribuzione

Grazie a una maggiore praticità e al supporto multiplatforma, distribuire le applicazioni web all'utente finale è molto semplice. Inoltre sono la soluzione ideale quan-

do la banda a disposizione è limitata e sia il sistema che le informazioni sono accessibili solo in remoto. È sufficiente inviare all'utente l'indirizzo web per permettergli di autenticarsi e accedere al sistema. Ciò ha un impatto enorme, permettendo di allargare l'accesso dei propri sistemi, snellire i processi e migliorare le relazioni aumentando l'accesso a nuovi utenti, fornitori e terze parti. In questo modo è possibile creare un movimento sociale tra gli utenti, i quali possono collaborare tra di loro e condividere i risultati del proprio lavoro.

5. Sicurezza dei dati

Solitamente nei sistemi più complessi, le informazioni sono salvate e spostate in diversi sistemi separati. Nei sistemi web invece, questi processi possono essere consolidati riducendo la necessità di migrare i dati. Le applicazioni web inoltre, forniscono uno strato di sicurezza aggiuntivo rimuovendo la necessità per l'utente di accedere direttamente alla banca dati e al backend del server.

6. Costi ridotti

Le applicazioni web riducono drasticamente i costi dovuti al supporto e alla manutenzione, ai requisiti di sistema per le macchine degli utenti finali e all'architettura semplificata. Inoltre, snellendo ulteriormente le proprie operazioni, spesso si riesce ad ottenere ulteriori risparmi.

Le applicazioni web offrono dunque soluzioni vantaggiose rispetto ai software tradizionali, garantendo un miglior rendimento dei processi e una riduzione dei costi di rilascio e mantenimento.

1.2 Architettura

L'architettura di un'applicazione web ricorda moltissimo quella di un modello di calcolo centralizzato, dove tanti "thin" client, che solitamente non eseguono nulla più che il rendering delle informazioni, si connettono a un server centrale che si occupa del grosso dell'elaborazione. Ciò che rende le architetture web diverse dai tradizionali modelli di calcolo centralizzati è che esse si basano sostanzialmente sulla tecnologia resa famosa dal World Wide Web, l'HTML (HyperText Markup Language), e il suo principale mezzo di trasporto, l'HTTP (HyperText Transfer Protocol).

Anche se HTML e HTTP definiscono la tipica architettura delle applicazioni web, alle spalle c'è molto di più di queste due tecnologie. I principali componenti di un'applicazione web tradizionale, sono illustrati in Figura 1.1.

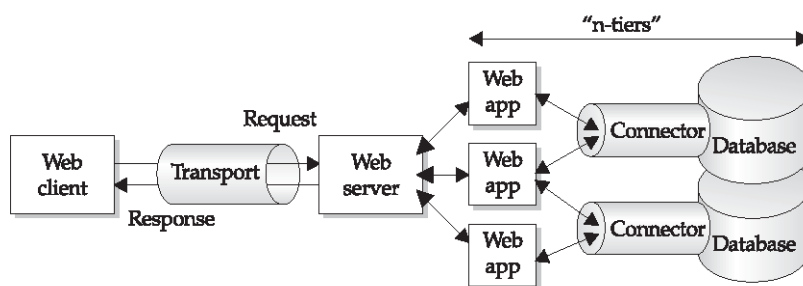


Figura 1.1. I componenti di un'architettura web application.

HTML

Come tutti i linguaggi di markup, l'HTML è descritto da tag, che definiscono la formattazione o le funzionalità degli elementi del documento. Quindi l'HTML è principalmente il motore di rendering dei dati di un'applicazione web.

Alcuni dei tag più utilizzati sono quelli connessi all'input utente. Un esempio di input è quello "hidden" (nascosto), il cui valore specificato non viene visualizzato dal browser, ma viene comunque inviato assieme agli altri dati della stessa form. Anche se non visualizzati direttamente dal browser, i campi nascosti possono comunque essere facilmente alterati lato cliente con un editor di testo e inviati al server.

Ironicamente, nonostante l'eleganza e l'iniziale influenza dell'HTML, questo è stato soppiantato da altre tecnologie. Ciò è principalmente dovuto al fatto che l'HTML è un formato statico che non può essere alterato al volo per soddisfare le mutevoli richieste dell'utente finale. Gran parte dei siti web oggi utilizzano tecnologie di scripting per generare contenuti al volo.

Infine, l'avvento in un nuovo linguaggio di markup, l'XML (eXtensible Markup Language), sta segnando il declino dell'HTML. Nonostante l'XML sia molto simile all'HTML, il suo successo è da attribuire alla sua capacità di rappresentare informazioni di qualsiasi tipo.

HTTP

HTTP versione 1.0 è un semplice protocollo, stateless, basato sull'ASCII. Solitamente opera su TCP, porta 80, ma può applicarsi a qualsiasi porta inutilizzata.

La semplicità dell'HTTP deriva dal suo limitato insieme di funzioni: request e response. HTTP definisce un meccanismo di richiesta di una risorsa, che il server restituisce quando tale risorsa diventa disponibile. Le risorse sono chiamate URI (Uniform Resource Identifiers) e possono spaziare da pagine statiche a streaming video dinamici.

Inoltre HTTP è senza stato (stateless): nessun tipo di stato di sessione viene mantenuto dal protocollo. Non mantiene nessun tipo di sessione, né cerca di mantenere l'integrità di una connessione con il client.

Infine HTTP opera su una porta TCP ben nota. Quasi tutti i web browser cercano automaticamente di connettersi alla porta 80, pertanto ogni web server è in ascolto su quella porta. Firewall e altri dispositivi di sicurezza diventano in sostanza inutili contro attacchi web se configurati per permettere l'accesso alla porta 80. D'altra parte, se si vuole pubblicare il proprio sito web, bisogna renderlo accessibile.

SSL

Per far fronte alle vulnerabilità appena citate, molte applicazioni web oggi veicolano l'HTTP verso un altro protocollo chiamato SSL (Secure Sockets Layer). SSL fornisce un meccanismo di cifratura del canale di trasporto, rendendo illeggibili le informazioni HTTP tra client e server. Tuttavia, a parte questo, SSL non altera in nessun modo il meccanismo di request/response dell'HTTP. Semplicemente rende più difficile intercettare il traffico tra client e server. Se però viene implementata una caratteristica opzionale del protocollo SSL, i certificati client-side, allora si può realizzare una mutua autenticazione client-server.

Gestione degli stati

Si è detto che l'HTTP di per sé è stateless, ma esistono dei meccanismi in grado di renderlo stateful. Il meccanismo più usato sono i cookie, che possono essere scambiati come parte del dialogo request/response dell'HTTP, per simulare un circuito virtuale tra il client e il server. I cookie vengono rilasciati dal server e salvati sulla memoria del client temporaneamente o permanentemente. Tuttavia i cookie possono diventare un grosso rischio per la sicurezza e la privacy se implementati scorrettamente.

Autenticazione

Legato allo stato della connessione, vi è il concetto di autenticazione. HTTP fornisce diversi tipi di protocolli di autenticazione, tra i quali:

- Base: nome utente e password inviati in chiaro.
- Digest: come sopra ma la password viene cifrata.
- Basata su form: viene utilizzata una form personalizzata per l'invio del nome utente e password e viene processata dal backend dell'applicazione.
- NTLM: protocollo di autenticazione proprietario Microsoft, basato sugli header HTTP.
- Negotiate: un nuovo protocollo Microsoft che permette un qualsiasi tipo di autenticazione specificato sopra, aggiungendo il supporto Kerberos per i browser Microsoft Internet Explorer 5 e superiori.
- Certificati client-side: SSL fornisce l'opportunità di controllare l'autenticità di un certificato digitale presentato dal client, trattandolo essenzialmente come un token identificativo.
- Microsoft passport: un servizio di single-sign-in (SSI) gestito da Microsoft che permette a certi siti web (chiamati "Passport Partner") di autenticare gli utenti basandosi sulla loro partecipazione al servizio Passport. Il meccanismo utilizza una chiave condivisa tra Microsoft e il Passport Partner, per creare un cookie che identifica univocamente l'utente.

Questi protocolli di autenticazione si appoggiano direttamente all'HTTP (o SSL), con le credenziali direttamente inserite nel traffico request/response.

L'applicazione web

Il cuore di un moderno sito web risiede nella sua logica server-side. Questa architettura, detta a "n-livelli", estende la normale esecuzione HTTP in un motore dinamico di funzionalità che rendono l'applicazione apparentemente stateful e con cui l'utente può interagire in tempo reale.

Un'applicazione web può essere composta di diversi livelli. La rappresentazione più diffusa per descrivere le applicazioni web è quella di un'architettura a 3-livelli, composta dagli strati di presentazione, applicazione e dati, come illustrato in Figura 1.2.

Lo strato di presentazione, fornisce i mezzi per ricevere l'input e mostrare i risultati. Il livello applicazione riceve l'input dal livello di presentazione ed esegue del lavoro su di essi (a volte

richiedendo l'assistenza del livello dati), quindi restituisce il risultato allo strato di presentazione. Infine, il livello dati fornisce un deposito di informazioni non volatili che può essere interrogato o aggiornato dal livello applicazione, realizzando un'astrazione dei dati, in modo da non dover codificare le informazioni direttamente nel livello applicazione e poterle facilmente aggiornare.

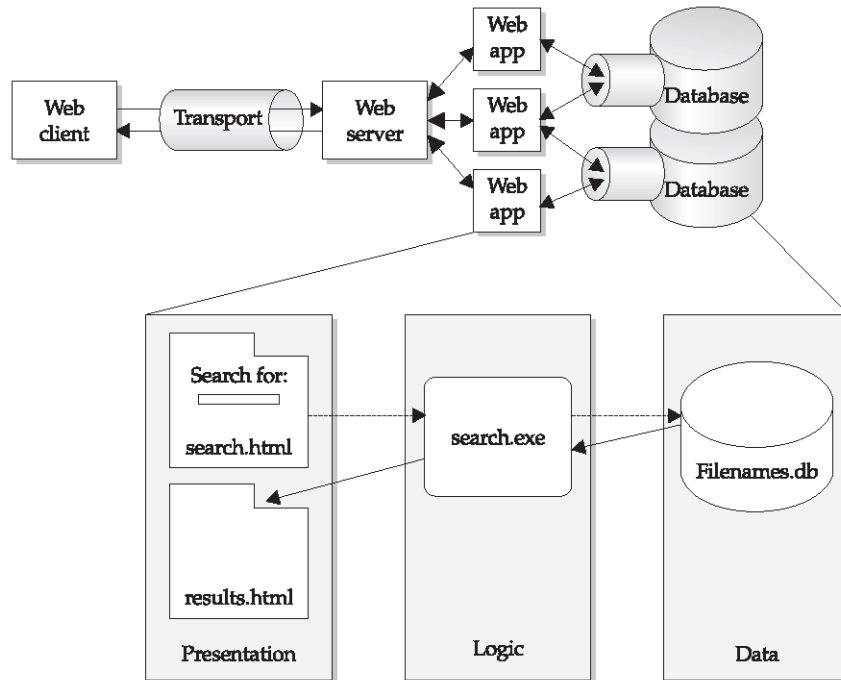


Figura 1.2. L'architettura a 3-livelli delle applicazioni web.

Molte delle tecnologie utilizzate oggi per realizzare applicazioni, integrano le funzionalità di uno o più di questi strati, rendendo quindi difficile distinguerli l'uno dall'altro [1].

1.3 Punti deboli delle applicazioni web

Nonostante i loro vantaggi, le applicazioni web sollevano un certo numero di perplessità riguardo la sicurezza. Vulnerabilità e falle nel codice, permettono agli utenti malevoli di ottenere l'accesso diretto al database. Molti di questi database contengono informazioni preziose da rivendere alla concorrenza, per esempio. Inoltre, il fascino delle applicazioni web è dovuto alla loro natura di "costante apertura al pubblico", che le rende particolarmente esposte a continui rischi.

Holz, Marechal e Raynal osservano: "Dal punto di vista di un attaccante, l'applicazione web è interessante per diverse ragioni. Primo, la qualità del codice di sicurezza è spesso piuttosto scadente, come mostrano numerosi report. Un altro fattore è la complessa struttura delle applicazioni [2]. Molte vulnerabilità diverse sono state scoperte nelle web application: applicazioni come i Content Management System (CMS), Wiki, portali, bacheche e forum, vengono utilizzate da piccole e grandi imprese. Ogni settimana centinaia di vulnerabilità vengono segnalate. Il numero di tentativi di attacchi portati a segno ogni giorno a grandi web farm spaziano dalle centinaia di migliaia fino a milioni [3]."

Quelli che seguono sono una panoramica dei principali tipi di attacco che si possono tipicamente compiere contro i diversi elementi dell'applicazione [4]:

- **Piattaforma web:** vulnerabilità legate al software della piattaforma web, incluse le infrastrutture sottostanti (per esempio Apache) o i framework di sviluppo (per esempio ASP.NET o PHP).
- **Applicazione web:** attacchi verso i meccanismi di autenticazione, l'autorizzazione, la struttura del sito, la validazione dell'input o la logica applicativa.
- **Database:** esecuzione di comandi privilegiati sul database, manipolazione delle query per restituire dataset ingenti e inserimento di codice maligno (SQL Injection).
- **Web client:** inserimento di contenuto eseguibile, sfruttamento delle vulnerabilità del software client, errori di cross-site scripting e truffe phishing.
- **Trasporto:** intercettazione delle comunicazioni client-server e reindirizzamento dell'SSL.
- **Disponibilità:** negazione del servizio (DoS, Denial of Service).

Ricerche recenti hanno dimostrato che il 75% della pirateria informatica viene eseguita a livello applicazione [5].

1.4 Complicazioni in ambito sanitario

Gli ospedali usano diversi sistemi standard per attività di contabilità, agenda e simili, ma hanno anche un certo numero di requisiti di sicurezza interessanti, gran parte dei quali hanno a che fare con la sicurezza e la privacy del paziente [6].

Con la crescente adozione delle tecnologie web negli ospedali, sorgono nuove problematiche di garanzia. Per esempio, mano a mano che le liste di riferimento, come quella relativa ai farmaci, vengono messe online, i medici devono essere sicuri che le informazioni critiche (come il dosaggio) siano esattamente quelle pubblicate dalle autorità competenti e non siano state manomesse in qualche modo, volontariamente o accidentalmente che sia. Un altro esempio è l'accesso alle pagine contenenti dati sensibili sui pazienti, da casa o altri ambienti non controllati. In questo caso deve essere implementato un adeguato sistema di autenticazione e cifratura.

Le banche dati contenenti le informazioni sui pazienti non dovrebbero consentire l'accesso a qualsiasi membro dello staff. A questi sistemi spesso viene richiesto di risolvere regole complesse come, "gli infermieri possono accedere solamente ai record dei pazienti che hanno avuto in cura nel loro dipartimento negli ultimi 90 giorni". Ciò può risultare estremamente complesso da risolvere tramite i tradizionali meccanismi di sicurezza informatici, poiché i ruoli possono cambiare (gli infermieri si muovono da un dipartimento all'altro) ed esistono delle dipendenze tra sistemi. Per questo si sta cercando di adottare soluzioni basate sui ruoli.

Nuove tecnologie possono introdurre rischi ancora incompresi. Gli amministratori delle aziende ospedaliere realizzano l'importanza di avere procedure di backup con cui affrontare interruzioni di corrente elettrica, servizi telefonici e via dicendo, tuttavia le pratiche mediche

stanno sensibilmente iniziando a dipendere dalla rete. Per esempio, i singoli dipartimenti clinici possono utilizzare una base di dati online per i farmaci, interrompendo l'utilizzo di stampe cartacee per l'elenco, senza avere delle appropriate misure di emergenza in mancanza del servizio di rete (dovuti per esempio a virus o attacchi di tipo DoS, Denial of Service).

1.4.1 Politiche di sicurezza e privacy

Nell'ambito della sanità, le complicazioni derivanti dallo scambio di dati sensibili, deve fare riferimento ai principi di sicurezza e tutela della privacy delle politiche HIE (Health Information Exchange) [7]. HIE promuove la mobilitazione delle informazioni elettroniche sanitarie tra le diverse organizzazioni all'interno della stessa regione, comunità o azienda ospedaliera. Essa fornisce le capacità di condividere le informazioni cliniche tra i diversi sistemi informativi sanitari, mantenendo al contempo facile e veloce l'accesso e il reperimento delle stesse.

Queste politiche sono strutturate su diversi livelli, il cui complesso è detto Policy Environment, come illustrato in Figura 1.3. Al livello più alto troviamo le politiche internazionali, come gli International Data Protection Principles. Ogni stato o regione ha poi le proprie politiche, per esempio la norma sulla sicurezza e la privacy americana HIPAA, Health Insurance Portability and Accountability Act, che variano di stato in stato [8]. Ci sono quindi le politiche orizzontali, comuni tra le specifiche attività, come quella medica. Infine, all'interno delle singole aziende ospedaliere, vigono politiche specifiche.

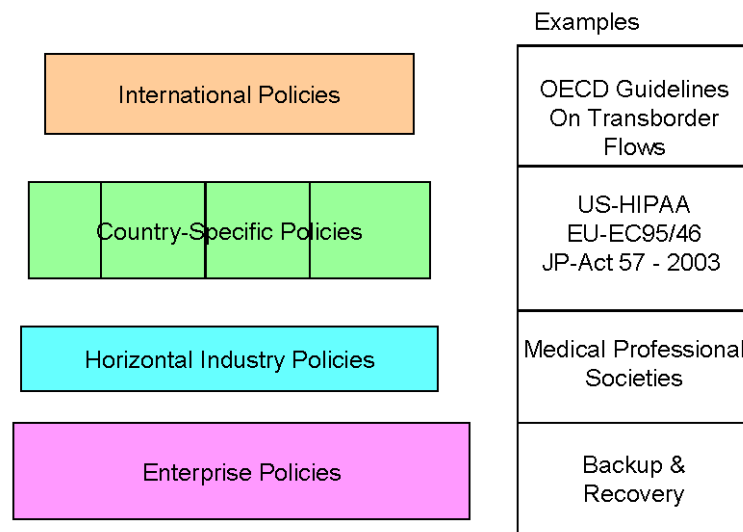


Figura 1.3. Policy environment.

Nel 1980, l'organizzazione per la cooperazione e lo sviluppo economico (OECD, Organization for Economic Cooperation and Development) [9], sviluppò delle linee guida sulla protezione della privacy e la circolazione dei dati personali. Queste regole hanno l'obiettivo di armonizzare le leggi sulla privacy, sostenere i diritti umani e promuovere la circolazione delle informazioni tra i suoi stati membri. Complessivamente, questi principi e leggi forniscono una solida ossatura per lo sviluppo dei requisiti di protezione delle informazioni dei sistemi informativi sanitari.

Tra le varie politiche abbiamo:

- Politiche che disciplinano chi ha accesso ai documenti e di che tipo di documenti si tratta.
- Politiche che stabiliscono che tipo di documenti possono essere pubblicati nell’HIE e chi ha i permessi per la pubblicazione.
- Politiche di mantenimento dei documenti nell’HIE (durata).
- Politiche sulla gestione delle operazioni in caso di emergenza.
- Politiche sull’uso della rete.
- Politiche sui metodi di autenticazione accettabili.
- Politiche di backup e recupero.
- Politiche sull’accesso di terze parti.
- Politiche che stabiliscono la durata delle operazioni di manutenzione.

Spesso le politiche s’intrecciano tra di loro con un effetto a cascata: a livello comunitario le politiche tendono a essere più generiche. Via via che si scende nella gerarchia, queste generalità possono venire ulteriormente rifinite.

Un insieme importante di politiche è quello riguardante le situazioni di emergenza. Esistono numerosi casi che possono definirsi tali:

- Catastrofi causate da eventi naturali o umani.
- Interruzione di un servizio (per esempio la corrente elettrica).
- Guasto all’infrastruttura informatica (per esempio la rottura di un disco rigido).
- Necessità di fornire privilegi speciali in caso di emergenza.
- Necessità di bypassare una restrizione imposta dal paziente per motivi legati alla salute del paziente stesso.

Lo sviluppo di politiche spesso crea dei conflitti. Tuttavia la maggior parte di essi può essere risolta facilmente una volta compresi i dettagli che le caratterizzano. Per esempio in Europa esiste l’esplicito divieto di salvare l’etnia nel profilo del paziente, anche se questa spesso è utile a fini medici. Questo conflitto può essere risolto salvando degli specifici marker genetici al posto dell’etnia.

L’insieme dei controllori dedicati alla sicurezza e alla privacy deve fornire le seguenti funzionalità:

1. **Riconoscimento della responsabilità:** è il controllo che monitora le attività in accordo con le politiche. Include sistemi di monitoraggio dell’audit, report e avvisi.
2. **Identificazione e autenticazione:** è il controllo che identifica il sistema o la persona in esame e certifica che sia effettivamente chi dice di essere. Esempi di tali controllori sono i certificati digitali o Kerberos.
3. **Controllo degli accessi:** limita l’accesso alle informazioni e alle funzioni alle entità autenticate che dispongono delle appropriate autorizzazioni d’accesso. Solitamente vengono implementati tramite controlli basati sui ruoli.
4. **Confidenzialità:** proteggono le informazioni dall’esposizione attraverso meccanismi di cifratura o controllo degli accessi.

5. **Integrità dei dati:** assicura che i dati non siano manomessi. Alcuni esempi sono le firme digitali, gli algoritmi di hash, il controllo a ridondanza ciclica (CRC, Cyclic Redundancy Check) e il checksum.
6. **Non-ripudiabilità:** garantisce la non ripudiabilità di entità coinvolte in determinate azioni. L'esempio più classico è la firma digitale.
7. **Privacy dei pazienti:** definisce delle specifiche istruzioni nel trattamento dei dati sensibili.
8. **Disponibilità:** garantisce la disponibilità delle risorse. Per esempio: backup, replica, tolleranza ai guasti, sistemi RAID (Redundant Array of Independent Disks), meccanismi di ripristino, gruppi di continuità, ecc.

Questi controllori ricevono gli input adeguati dalle diverse politiche e riflettono i loro effetti nei singoli sistemi.

1.5 Profili IHE

Integrating the Healthcare Enterprise (IHE) [10] è un'iniziativa nata da professionisti dell'ambiente sanitario e dall'industria a essa legata, con lo scopo di migliorare i metodi con cui i sistemi informatici condividono le informazioni medicali. IHE promuove l'uso coordinato di standard ben definiti come DICOM [11] e HL7 [12] per risolvere le specifiche necessità cliniche di supporto alla cura del paziente. I sistemi sviluppati secondo le specifiche IHE comunicano tra loro meglio, sono più semplici da implementare e permettono di utilizzare le informazioni in modo più efficiente.

IHE ha creato un forum, dove operatori sanitari, esperti HIT e altre parti interessate, provenienti dalle diverse cliniche nel mondo, possono accordarsi su soluzioni standard relative agli aspetti critici dell'interoperabilità dei sistemi informativi sanitari. Le linee guida implementative pubblicate da IHE (chiamati profili IHE [13]) servono inizialmente a raccogliere l'opinione degli utenti e successivamente a fornire un supporto all'implementazione di prova presso rivenditori HIT e altri sviluppatori. IHE fornisce un processo dettagliato d'implementazione e collaudo delle applicazioni per promuovere l'interoperabilità basata su standard da parte di rivenditori e utenti di sistemi informativi sanitari. Questo processo culmina nel *Connectathon*, un evento di test svolto regolarmente con lo scopo di valutare il livello d'interoperabilità delle applicazioni proposte.

Dopo che un comitato scelto ha stabilito che un profilo ha superato un adeguato numero di test ed è stato applicato con successo in un ambiente sanitario reale, esso viene incorporato nell'appropriato Technical Framework IHE. Il Technical Framework è quindi una risorsa unica per lo sviluppo e l'utilizzo di sistemi HIT: è un insieme di soluzioni testate e basate su standard, atte a far fronte alle comuni esigenze d'interoperabilità e supporto per l'utilizzo sicuro delle cartelle cliniche elettroniche (EHR, Electronic Healthcare Records).

1.5.1 Technical Framework

Il Technical Framework IHE [14] ha l'obiettivo di promuovere i mezzi necessari a sviluppare applicazioni integrabili capaci di realizzare un'appropriata condivisione delle informazioni

sanitarie. Viene aggiornato ogni anno, dopo un periodo di revisione pubblica, e mantenuto regolarmente attraverso l'identificazione e la correzione degli errori.

Il Technical Framework IHE identifica un sottoinsieme dei componenti funzionali delle aziende ospedaliere, chiamati Attori IHE, e specifica le loro interazioni in termini di transazioni. Attualmente sono presenti i seguenti Technical Framework:

- Anatomic Pathology
- Cardiology
- Eye Care
- IT Infrastructure
- Laboratory
- Patient Care Coordination
- Patient Care Device
- Pharmacy
- Quality, Research and Public Health
- Radiation Oncology
- Radiology Technical Framework

1.5.2 Profili di integrazione

I profili di integrazione dell'infrastruttura IHE [15], illustrati in Figura 1.4, offrono un linguaggio comune che, professionisti e rivenditori nell'ambito della sanità, possono utilizzare per discutere nel dettaglio le diverse esigenze d'interoperabilità delle aziende sanitarie e delle capacità di integrazione dei sistemi informativi. I profili di integrazione specificano le implementazioni di standard specifiche per il raggiungimento di determinati obiettivi clinici. Essi permettono a utenti e rivenditori di indicare quali funzionalità IHE sono richieste o fornite.

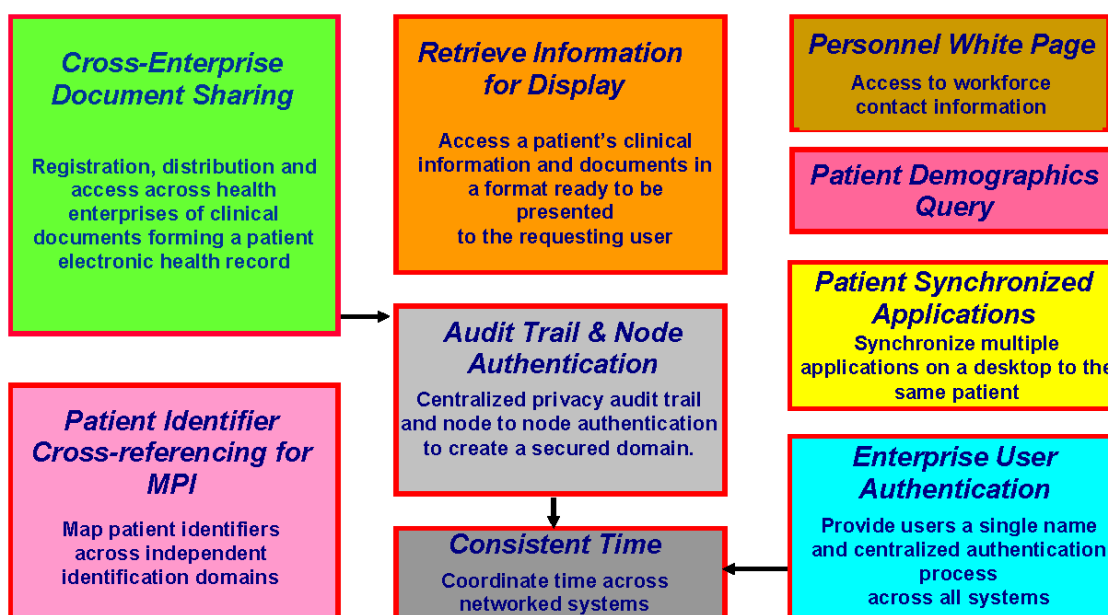


Figura 1.4. Profili di integrazione dell'infrastruttura IHE.

I profili di integrazione sono definiti in termini di Attori IHE e transazioni. Gli attori sono sistemi informatici, o componenti degli stessi, che producono, gestiscono o agiscono sulle informazioni associate alle attività operazionali e cliniche dell'azienda. Le transazioni sono interazioni tra attori che comunicano le informazioni richieste tramite messaggi basati su standard.

Le applicazioni supportano i profili di integrazione attraverso l'implementazione degli attori e delle transazioni previste da quel profilo. Un determinato prodotto può implementare più di un attore o profilo di integrazione alla volta.

I profili si possono suddividere in tre classi:

- Content Profiles – si occupano della gestione di un particolare tipo di contenuto: come viene creato, immagazzinato, richiesto e recuperato.
- Workflow Profiles – affrontano la gestione dei processi dei flussi di lavoro, i quali tipicamente riguardano le worklist e il monitoraggio dei loro progressi.
- Infrastructure Profiles – gestiscono questioni generali riguardanti il dipartimento, come per esempio l'audit trail di radiologia o l'accesso al sistema informativo radiologico.

In generale i profili di integrazione IHE non operano indipendentemente. Oggetti che fungono da input di un profilo, possono a loro volta essere il risultato di un altro profilo. In alcuni casi questo tipo di dipendenze è strettamente necessario per il corretto funzionamento del profilo. In generale comunque, ogni profilo ha poco valore se a esso non è associato un relativo profilo di contenuto.

1.5.3 I profili di sicurezza

Le transazioni IHE spesso contengono informazioni che devono essere protette in conformità con le leggi sulla privacy e le normative vigenti nelle diverse regioni. IHE include alcuni profili incentrati sulla sicurezza e la privacy, mentre la maggior parte non specifica alcun tipo di protezione, ma vanno raggruppate e integrate adeguatamente con i precedenti. In particolare IHE definisce i seguenti profili di sicurezza:

- Il profilo ATNA (Audit Trail and Node Authentication) specifica i mezzi attraverso i quali autenticare i singoli nodi della rete.
- Il profilo ATNA definisce, inoltre, i messaggi di audit per il monitoraggio degli eventi attinenti questioni di sicurezza e privacy.
- Il profilo EUA (Enterprise User Authentication) descrive i mezzi per autenticare gli utenti di sistema e condividere, tra le applicazioni, le informazioni a essi relativi.
- Il profilo PWP (Personnel White Pages) fornisce una base di dati, dove immagazzinare le informazioni identificative degli utenti.

Gli sviluppatori possono seguire questi profili per soddisfare le loro esigenze di sicurezza.

Audit Trail and Node Authentication

Il profilo di integrazione ATNA stabilisce le misure di sicurezza che, assieme alle procedure e politiche di sicurezza, garantiscono la confidenzialità dei dati sensibili, l'integrità delle infor-

mazioni e la responsabilità utente. L'ambiente così definito, è detto dominio di sicurezza e può spaziare da un singolo dipartimento, fino all'intera azienda ospedaliera. Il modello ATNA tiene in considerazione che all'interno del dominio di sicurezza le seguenti regole siano rispettate:

1. Tutte le macchine sono host autenticati. Questa autenticazione identifica la macchina come un client noto al sistema di sicurezza dell'ospedale. Macchine sconosciute possono comunque accedere al sistema, ma con la clausola che abbiano accesso solamente a informazioni di carattere pubblico.
2. L'identificazione dell'host è richiesta per determinare che tipo di accesso (se presente) garantire all'host e/o al personale che gestisce la macchina in questione.
3. Il nodo sicuro è responsabile di fornire un adeguato controllo degli accessi. Ciò tipicamente include l'autenticazione utente e l'autorizzazione. Questa autenticazione deve saper bilanciare i possibili impatti sulla sicurezza e salute del paziente dovuti a ritardi causati da eccessive e prolungate fasi di autenticazione.
4. Il nodo sicuro è anche responsabile del log degli audit per tracciare gli eventi di sicurezza. Nell'ambito sanitario, l'audit log è spesso più utile di un controllo degli accessi stringente e ci si dovrebbe basare su di esso in caso di emergenze.

Questo modello è parzialmente motivato dall'implicita assunzione che ci saranno situazioni in cui dei documenti saranno scambiati tra client e salvati nelle macchine destinatarie. Questa caratteristica è richiesta principalmente per garantire ai sistemi sanitari di operare anche in caso di disastro o sovraccarico, quando l'operatività della rete è limitata o assente. Non si può assumere che i client operino solamente in visualizzazione, pertanto deve risiedere una copia semi-permanente di gran parte delle informazioni chiave. Possono esistere dei thin client con solo funzioni di visualizzazione, ma sono limitati a utilizzi che non introducono rischi alla salute del paziente in caso di funzionamento limitato o assente.

Inoltre, ATNA prevede che:

- Tutti i sistemi partecipanti a un dominio di sicurezza implementino un attore nodo sicuro per il profilo ATNA. Il profilo ATNA definisce le transazioni tra i nodi sicuri per realizzare un dominio di sicurezza sotto l'amministrazione di un ente autorizzato.
- Tutte le applicazioni su un nodo sicuro soddisfano i requisiti ATNA, indipendentemente dal fatto che siano o meno attori IHE. Ciò include tutte le attività IT che creano, accedono, aggiornano o cancellano direttamente dati di tipo PHI (Protected Healthcare Information), e non solo quelli specificati da IHE ed eseguiti dai relativi attori.
- IHE affronta solo i requisiti di sicurezza legati ai sistemi coinvolti con l'ambito delle applicazioni sanitarie IHE. Non si occupa di altri requisiti di sicurezza come la difesa contro attacchi di rete, virus informatici, ecc. Il principale obiettivo del meccanismo di Audit Trail è di tenere traccia dell'accesso ai dati PHI, non delle transazioni.
- Dispositivi mobili possono far parte del profilo di integrazione ATNA, ma questioni specifiche legate a tali dispositivi non vengono trattate da questo profilo.

- ATNA assume che il controllo degli accessi fisici, le politiche di gestione del personale e altre considerazioni riguardanti la sicurezza organizzativa, necessarie a rendere l'azienda aderente alle norme sulla privacy e sicurezza, siano correttamente applicate.

1.6 Policlinico Universitario di Padova

In tutte le grandi strutture, dove si richiede un'organizzazione ordinata ed efficiente, l'utilizzo di sistemi informatici è diventata prassi comune. Le aziende ospedaliere, in particolare, hanno a che fare con enormi quantità di dati, spesso conservata in registri cartacei complessi da consultare e immagazzinare. Considerata poi l'importanza dell'efficienza dell'attività operativa, si intuisce chiaramente come l'informatica si stia diffondendo velocemente in questi ambienti. Padova, in particolare, con la sezione di Radiologia, si è distinta per essere una delle prime ad aver informatizzato il proprio reparto.

1.6.1 Evoluzione del sistema informativo

La sezione di Radiologia del Policlinico Universitario di Padova è stata una delle prime ad adottare strumenti informatici per la gestione delle attività operative e delle cartelle cliniche, fino ad allora ancora in forma cartacea o fisica (refertazione su nastro, conservazione delle immagini su pellicola).

Il punto di partenza per il nuovo sistema informatico, fu l'adozione di un RIS (Radiology Information System) commerciale, che però si rivelò ben presto insufficiente a risolvere le esigenze del reparto. Fu da questa necessità che nacque un'unità informatica all'interno dell'allora Istituto di Radiologia, con l'obiettivo di sviluppare un software che rispondesse adeguatamente alle esigenze interne.

Nello stesso periodo, l'avvento di standard come DICOM e HL7, ha creato lo stimolo a ridurre ulteriormente l'utilizzo di supporti cartacei. Venne quindi affiancato un sistema PACS, il CTN della RSNA, per l'archiviazione e distribuzione di immagini DICOM. Nel 2001 intanto nasceva IHE, un'organizzazione per la promozione di standard condivisi tra le aziende ospedaliere.

Nacque quindi ESO, un progetto a basso costo basato su architettura client-server per l'inserimento di referti, realizzato con tecnologie proprietarie. La prima versione lavorava in ambiente Microsoft (Windows 95 e Windows NT 4 Server), con Delphi 3.0 lato client e Paradox come banca dati. Tuttavia il sistema non fu in grado di reggere l'ingente mole di dati, portando ben presto all'adozione di un nuovo server database: SQL Server. Purtroppo anche questo fu insufficiente, per cui si scelse di tentare la strada della base di dati Oracle. Oracle tuttavia, mostrava segni di incompatibilità con Windows NT.

Nel frattempo, la necessità di distribuire le informazioni di radiologia ai reparti richiedenti e rispondere alle esigenze dell'azienda ospedaliera, diede vita al progetto WebESO, un'applicazione web sviluppata con tecnologia Microsoft Active Server Pages su sistema Internet Information Server.

L'architettura client-server presentava però continui problemi legati alla compatibilità software, ai costi degli strumenti proprietari e allo scarso controllo delle applicazioni. Si decise

quindi di passare a soluzioni open source, partendo dapprima con la conversione del sistema in una web application PHP, quindi l'adozione di un server Linux e infine di un software open-source per la distribuzione e visualizzazione delle immagini DICOM.

L'introduzione di Linux ha spinto verso la sperimentazione di sistemi di gestione di basi di dati open source, mettendo dapprima alla prova il sistema Interbase (che per le esigenze interne si è dimostrato equivalente a Oracle) e in seguito MySQL, noto sistema open source.

Il passaggio all'architettura web si è dimostrata talmente superiore in termini di efficienza da convincere a realizzare una conversione totale del sistema. Nacque il progetto MARiS, internamente noto col nome di Crono. Fu l'occasione giusta per iniziare a implementare i profili di integrazione previsti dalle linee guida IHE.

MARiS è attualmente utilizzato presso la Sezione di Radiologia del Dipartimento di Scienze Medico-Diagnostiche e Terapie Speciali dell'Università di Padova, e vanta un insieme di funzionalità che vanno oltre la semplice separazione fra RIS e PACS, divenendo a tutti gli effetti un sistema gestionale di dati radiologici. Tra le varie funzionalità citiamo la gestione dei ticket, la refertazione vocale e la firma digitale.

Intanto, l'Università di Padova ha favorito la nascita di una collaborazione con il Dipartimento di Ingegneria dell'Informazione, per la realizzazione di nuove funzionalità e l'innovazione del sistema con paradigmi all'avanguardia. Rimane forte il legame con i continui aggiornamenti forniti da IHE, e con aziende terze per supportare l'adozione del sistema MARiS anche presso altre aziende ospedaliere.

1.6.2 Altri strumenti software

Oltre al già citato MARiS, vengono utilizzati altri strumenti open-source basati su una robusta implementazione degli standard DICOM e HL7. Tra i vari citiamo dcm4che [16] e mirth [17]. Il primo definisce delle applicazioni per l'interoperabilità DICOM, mentre il secondo realizza soluzioni per lo scambio di informazioni HL7.

1.6.3 Sviluppo del sistema

Una delle peculiarità che caratterizza lo sviluppo del sistema MARiS, è lo stretto contatto con figure professionalmente legate al mondo della sanità. Questa interazione diretta tra lo sviluppatore e l'utilizzatore finale permette di ridurre i tempi che intercorrono tra una specifica richiesta e la soddisfazione della stessa; di assicurare all'utilizzatore finale lo sviluppo di quanto specificatamente richiesto, e allo sviluppatore la possibilità di realizzare esattamente quanto richiesto; e di creare soluzioni ad hoc per la realizzazione di funzioni specifiche.

1.7 Aspetti legali

Come già accennato nel paragrafo 1.4, l'ambiente sanitario è sottoposto a diverse problematiche relative alla privacy e alla sicurezza dei dati. In particolare, l'avvento di sistemi di archiviazione elettronici richiedeva l'intervento di precise norme, in grado di disciplinare questi nuovi metodi di gestione dei dati sensibili. Con la legge 196/2003, meglio conosciuta come *Co-*

dice della privacy o *Testo unico sulla privacy*, la precedente legge sulla tutela dei dati personali, 675/1996, è stata abrogata.

1.7.1 Testo unico sulla privacy

In Italia la produzione normativa riferentesi alla privacy è al momento presente, primariamente, nella Costituzione (articoli 15 e 21), nel Codice penale (Capo III - Sezione IV) e, ovviamente, nel Decreto legislativo 30 giugno 2003, n. 196, intitolato *Codice in materia di protezione dei dati personali*, noto ai più anche come *Codice della privacy* o *Testo unico sulla privacy* [18]. La precedente legge in tale materia, legge 675/96, *Tutela delle persone e di altri soggetti rispetto al trattamento dei dati personali*, fu a suo tempo introdotta nell'ordinamento italiano, al fine di rispettare e far applicare gli Accordi di Schengen, ed era entrata in vigore nel maggio 1997. Con il tempo, data la tipica stratificazione normativa che si produce nei sistemi a Civil Law (caratteristica ancora più spiccata in Italia), a tale norma si erano affiancate numerose altre disposizioni, concernenti specifici aspetti del trattamento dei dati, che sono state riassunte nel Testo Unico vigente (il quale, per definizione, è un "contenitore", non una fonte del diritto), entrato in vigore il 1° gennaio 2004.

Il Testo Unico sulla privacy si compone di tre parti e tre allegati, secondo il seguente schema:

1. disposizioni generali (artt. 1-45) relative alle regole "sostanziali" della disciplina del trattamento dei dati personali, applicabili a tutti i trattamenti, salvo eventuali regole specifiche per i trattamenti effettuati da soggetti pubblici o privati (art. 6);
2. disposizioni particolari per specifici trattamenti (artt. 46-140) a integrazione o eccezione alle disposizioni generali della parte di cui sopra;
3. le disposizioni relative alle azioni di tutela dell'interessato e al sistema sanzionatorio (artt. 141-186).

Di seguito citiamo alcuni degli articoli maggiormente connessi all'aspetto informatico (per la versione completa, consultare l'Appendice).

Articolo 34, comma 1:

Art. 34. Trattamenti con strumenti elettronici

1. Il trattamento di dati personali effettuato con strumenti elettronici è consentito solo se sono adottate, nei modi previsti dal disciplinare tecnico contenuto nell'allegato B), le seguenti misure minime:

- a) autenticazione informatica;
- b) adozione di procedure di gestione delle credenziali di autenticazione;
- c) utilizzazione di un sistema di autorizzazione;
- d) aggiornamento periodico dell'individuazione dell'ambito del trattamento consentito ai singoli incaricati e addetti alla gestione o alla manutenzione degli strumenti elettronici;
- e) protezione degli strumenti elettronici e dei dati rispetto a trattamenti illeciti di dati, ad accessi non consentiti e a determinati programmi informatici;
- f) adozione di procedure per la custodia di copie di sicurezza, il ripristino della disponibilità dei dati e dei sistemi;
- g) tenuta di un aggiornato documento programmatico sulla sicurezza;
- h) adozione di tecniche di cifratura o di codici identificativi per determinati trattamenti di dati idonei a rivelare lo stato di salute o la vita sessuale effettuati da organismi sanitari.

L'articolo 76, comma 1:

Art. 76. Esercenti professioni sanitarie e organismi sanitari pubblici

1. Gli esercenti le professioni sanitarie e gli organismi sanitari pubblici, anche nell'ambito di un'attività di rilevante interesse pubblico ai sensi dell'articolo 85, trattano i dati personali idonei a rivelare lo stato di salute:

- a) con il consenso dell'interessato e anche senza l'autorizzazione del Garante, se il trattamento riguarda dati e operazioni indispensabili per perseguire una finalità di tutela della salute o dell'incolumità fisica dell'interessato;
- b) anche senza il consenso dell'interessato e previa autorizzazione del Garante, se la finalità di cui alla lettera a) riguarda un terzo o la collettività.

Allegato B, disciplinare tecnico in materia di misure minime di sicurezza, comma 1, 2, 5, 7, 13, 25:

Sistema di autenticazione informatica

1. Il trattamento di dati personali con strumenti elettronici è consentito agli incaricati dotati di credenziali di autenticazione che consentano il superamento di una procedura di autenticazione relativa a uno specifico trattamento o a un insieme di trattamenti.

2. Le credenziali di autenticazione consistono in un codice per l'identificazione dell'incaricato associato a una parola chiave riservata conosciuta solamente dal medesimo oppure in un dispositivo di autenticazione in possesso e uso esclusivo dell'incaricato, eventualmente associato a un codice identificativo o a una parola chiave, oppure in una caratteristica biometrica dell'incaricato, eventualmente associata a un codice identificativo o a una parola chiave.

5. La parola chiave, quando è prevista dal sistema di autenticazione, è composta da almeno otto caratteri oppure, nel caso in cui lo strumento elettronico non lo permetta, da un numero di caratteri pari al massimo consentito; essa non contiene riferimenti agevolmente riconducibili all'incaricato ed è modificata da quest'ultimo al primo utilizzo e, successivamente, almeno ogni sei mesi. In caso di trattamento di dati sensibili e di dati giudiziari la parola chiave è modificata almeno ogni tre mesi.

7. Le credenziali di autenticazione non utilizzate da almeno sei mesi sono disattivate, salvo quelle preventivamente autorizzate per soli scopi di gestione tecnica.

13. I profili di autorizzazione, per ciascun incaricato o per classi omogenee di incaricati, sono individuati e configurati anteriormente all'inizio del trattamento, in modo da limitare l'accesso ai soli dati necessari per effettuare le operazioni di trattamento.

25. Il titolare che adotta misure minime di sicurezza avvalendosi di soggetti esterni alla propria struttura, per provvedere alla esecuzione riceve dall'installatore una descrizione scritta dell'intervento effettuato che ne attesta la conformità alle disposizioni del presente disciplinare tecnico.

In particolare è stabilito che:

- le credenziali di autenticazione debbano basarsi su:
 - a) password;
 - b) security token, smartcard e analoghi;
 - c) caratteristica biometrica;
 - d) una combinazione della a) con b) o c);
- se è presente una password nel sistema di autenticazione, anche in combinazione con altri sistemi di autenticazione, questa deve essere composta da almeno otto caratteri (se possibile) e deve essere modificata ogni tre mesi in caso di trattamento di dati sensibili;
- gli utenti inattivi da almeno sei mesi devono essere bloccati;
- esistano degli adeguati sistemi di permessi, divisi per classi di utenti o per singoli utenti.

1.7.2 Privacy e amministratori di sistema

Sulla Gazzetta Ufficiale del 24 dicembre 2008 sono stati pubblicati i provvedimenti sulle misure e gli accorgimenti prescritti ai titolari dei trattamenti effettuati con strumenti elettronici a proposito delle attribuzioni delle funzioni di amministratore di sistema, da parte del Garante per la protezione dei dati personali.

Con questo provvedimento, l'Autorità intende richiamare tutti i titolari di trattamenti effettuati, anche in parte, mediante strumenti elettronici alla necessità di prestare massima attenzione ai rischi e alle criticità implicite nell'affidamento degli incarichi di amministratore di sistema.

Questo nuovo adempimento non si esaurisce nella mera predisposizione di una nuova lettera di incarico o nella modifica di quella già esistente ma richiede al titolare una serie di "misure e accorgimenti" e, non ultimi, di adempimenti in ordine all'esercizio dei doveri di controllo da parte del titolare sulle attività dell'amministratore.

Allo scopo di facilitare il compito per quelle realtà nelle quali taluni servizi informatici sono svolti da società esterne, il Garante ha consentito che non solo i titolari, ma anche i responsabili possano effettuare gli adempimenti connessi all'individuazione degli AdS (Amministratori di Sistema) e alla tenuta dei relativi elenchi. L'applicazione di questo criterio fa sì che, in caso di outsourcing, l'onere di tali operazioni ricada sul responsabile esterno del trattamento. Il titolare, cui spetta sempre il compito di controllare, sarà pertanto libero da adempimenti che altrimenti sarebbe difficile, se non impossibile, attuare.

Ai sensi del provvedimento, l'AdS è la figura professionale dedicata alla gestione e alla manutenzione di impianti di elaborazione con cui vengono effettuati trattamenti di dati personali. Può essere comunque solo una persona fisica.

Le funzioni principali dell'AdS sono richiamate nell'allegato B del Testo Unico sulla privacy. Infatti, la maggior parte dei compiti previsti nel medesimo allegato spettano tipicamente all'AdS: dalla realizzazione di copie di sicurezza (operazioni di backup e recovery dei dati) alla custodia delle credenziali, alla gestione dei sistemi di autenticazione e di autorizzazione. Tali operazioni possono comportare un'effettiva capacità di azione su informazioni da considerarsi a tutti gli effetti alla stregua di un trattamento di dati personali; ciò, anche quando l'amministratore non consulti "in chiaro" le informazioni medesime.

Da notare poi che, secondo il Codice penale, buona parte delle funzioni tecniche attribuite all'AdS, possono rappresentare una circostanza aggravante, se svolte da chi commette un reato. È il caso ad esempio dell'accesso abusivo a sistema informatico o telematico (art. 615-ter) e di frode informatica (art. 640-ter), nonché per le fattispecie di danneggiamento di informazioni, dati e programmi informatici (articoli 635-bis e ter) e di danneggiamento di sistemi informatici e telematici (articoli 635-quater e quinquies).

Il provvedimento si rivolge a tutti i soggetti pubblici e privati che trattano dati personali con sistemi di elaborazione elettronica. Sono esclusi i trattamenti effettuati in ambito pubblico e privato a fini amministrativo-contabili che, ponendo minori rischi per gli interessati, sono stati oggetto delle misure di semplificazione introdotte nel corso del 2008 per legge [19].

In sostanza, non debbono adeguarsi al provvedimento coloro che utilizzano dati personali non sensibili, oppure che trattano, come unici dati sensibili, quelli riferiti ai propri dipenden-

ti e collaboratori (anche a progetto), quelli costituiti dallo stato di salute o malattia e dall'adesione a organizzazioni sindacali o a carattere sindacale, oppure, ancora, quei soggetti che trattano dati personali unicamente per correnti finalità amministrative e contabili. Si tratta, generalmente, di piccole e medie imprese, sia pubbliche che private, di artigiani, di commercianti e di liberi professionisti.

Adempimenti

Oltre agli adempimenti noti, il Garante fissa alcuni punti che il titolare del trattamento deve rispettare nei confronti dell'amministratore di sistema. E precisamente:

1. Valutazione delle caratteristiche soggettive;
2. Designazioni individuali;
3. Elenco degli amministratori di sistema;
4. Servizi in outsourcing;
5. Verifica delle attività;
6. Registrazione degli accessi.

Senza andare ad approfondire ambiti non pertinenti allo scopo di questa tesi, vediamo solamente il punto 6 (l'intero provvedimento è reperibile presso il sito del Garante per la protezione dei dati personali [20]). In particolare al comma 2 lett. f, viene stabilito:

Devono essere adottati sistemi idonei alla registrazione degli accessi logici (autenticazione informatica) ai sistemi di elaborazione e agli archivi elettronici da parte degli amministratori di sistema. Le registrazioni (access log) devono avere caratteristiche di completezza, inalterabilità e possibilità di verifica della loro integrità adeguate al raggiungimento dello scopo di verifica per cui sono richieste.

Le registrazioni devono comprendere i riferimenti temporali e la descrizione dell'evento che le ha generate e devono essere conservate per un congruo periodo, non inferiore a sei mesi.

In seguito alle crescenti perplessità che tale provvedimento ha suscitato, il Garante ha pubblicato una serie di FAQ. Tra le varie, meritano particolare attenzione le seguenti (la lista completa delle domande e risposte relative ai log è consultabile in Appendice):

Il provvedimento non chiede in alcun modo che vengano registrati dati sull'attività interattiva (comandi impartiti, transazioni effettuate) degli amministratori di sistema.

La raccolta dei log serve per verificare anomalie nella frequenza degli accessi e nelle loro modalità (orari, durata, sistemi cui si è fatto accesso...). L'analisi dei log può essere compresa tra i criteri di valutazione dell'operato degli amministratori di sistema.

Tra gli accessi logici a sistemi e archivi elettronici sono comprese le autenticazioni nei confronti dei data base management systems (DBMS), che vanno registrate.

Si afferma quindi, che l'analisi dei log raccolti può essere compresa tra i criteri di valutazione dell'operato dell'amministratore di sistema, in quanto consente di verificare eventuali anomalie nella frequenza e nelle modalità degli accessi.

Riguardo la durata di conservazione di sei mesi dei log, Confidustria fa notare che:

[...] tale previsione appare eccessivamente ampia e non perfettamente in linea con il principio di conservazione sancito dal Codice all'art. 11, co. 1, lett. e), confermato in diversi Provvedimenti e decisioni del Garante (es. videosorveglianza), secondo cui i dati oggetto di trattamento devono essere conservati per il periodo di tempo strettamente necessario agli scopi per i quali sono stati raccolti o successivamente trattati. L'obbligo in esame richiederebbe inoltre un prevedibile notevole impegno di risorse di memorizzazione, complesse procedure di copia dei log e

file di log enormi, risultando in questo modo inutilmente costoso per le imprese titolari, oltre che tecnicamente complicato [21].

Per la precisione, l'articolo 11, comma 1, lettera e) cita testualmente:

1. I dati personali oggetto di trattamento sono:
e) conservati in una forma che consenta l'identificazione dell'interessato per un periodo di tempo non superiore a quello necessario agli scopi per i quali essi sono stati raccolti o successivamente trattati.

Riguardo la registrazione di filmati di videosorveglianza, nel provvedimento generale sulla videosorveglianza, al punto 3.4 si cita:

In applicazione del principio di proporzionalità (v. anche art. 11, comma 1, lett. e), del Codice), anche l'eventuale conservazione temporanea dei dati deve essere commisurata al grado di indispensabilità e per il solo tempo necessario - e predeterminato - a raggiungere la finalità perseguita.

La conservazione deve essere limitata a poche ore o, al massimo, alle ventiquattro ore successive alla rilevazione, fatte salve speciali esigenze di ulteriore conservazione in relazione a festività o chiusura di uffici o esercizi, nonché nel caso in cui si deve aderire ad una specifica richiesta investigativa dell'autorità giudiziaria o di polizia giudiziaria.

Solo in alcuni specifici casi, per peculiari esigenze tecniche (mezzi di trasporto) o per la particolare rischiosità dell'attività svolta dal titolare del trattamento (ad esempio, per alcuni luoghi come le banche può risultare giustificata l'esigenza di identificare gli autori di un sopralluogo nei giorni precedenti una rapina), è ammesso un tempo più ampio di conservazione dei dati, che non può comunque superare la settimana.

Un eventuale allungamento dei tempi di conservazione deve essere valutato come eccezionale e comunque in relazione alla necessità derivante da un evento già accaduto o realmente imminente, oppure alla necessità di custodire o consegnare una copia specificamente richiesta dall'autorità giudiziaria o di polizia giudiziaria in relazione ad un'attività investigativa in corso.

Il sistema impiegato deve essere programmato in modo da operare al momento prefissato - ove tecnicamente possibile - la cancellazione automatica da ogni supporto, anche mediante sovraregistrazione, con modalità tali da rendere non riutilizzabili i dati cancellati.

Tuttavia da nessuna parte si specifica la durata di conservazione dei log di audit e di sistema.

1.7.3 Codice dell'amministrazione digitale

Il decreto legislativo del 7 marzo 2005 n. 82, *Codice dell'amministrazione digitale* [22], stabilisce, al Capo VI, *Sviluppo, acquisizione e riuso di sistemi informatici nelle pubbliche amministrazioni*, l'obbligo di comparare le diverse soluzioni disponibili sul mercato che consentano la rappresentazione dei dati e documenti in più formati, di cui almeno uno di tipo aperto. In particolare, l'articolo 68, comma 2 afferma:

2. Le pubbliche amministrazioni nella predisposizione o nell'acquisizione dei programmi informatici, adottano soluzioni informatiche che assicurino l'interoperabilità e la cooperazione applicativa, secondo quanto previsto dal decreto legislativo 28 febbraio 2005, n. 42, e che consentano la rappresentazione dei dati e documenti in più formati, di cui almeno uno di tipo aperto, salvo che ricorrano peculiari ed eccezionali esigenze.

E ancora, all'articolo 69, comma 1 e 2, viene stabilito:

1. Le pubbliche amministrazioni che siano titolari di programmi applicativi realizzati su specifiche indicazioni del committente pubblico, hanno l'obbligo di darli in formato sorgente, completi della documentazione disponibile, in uso gratuito ad altre pubbliche amministrazioni che li richiedono e che intendano adattarli alle proprie esigenze, salvo motivate ragioni.

2. Al fine di favorire il riuso dei programmi informatici di proprietà delle pubbliche amministrazioni, ai sensi del comma 1, nei capitolati o nelle specifiche di progetto è previsto ove possi-

bile, che i programmi appositamente sviluppati per conto e a spese dell'amministrazione siano facilmente portabili su altre piattaforme.

La versione completa degli articoli 67-70 al Capo VI, è reperibile in Appendice.

PARTE II

MATERIALI E METODI

Capitolo 2

Autenticazione

In questo capitolo sono descritte le principali tecniche di autenticazione e gli attacchi a cui sono vulnerabili. Una breve panoramica dei profili IHE illustra le specifiche richieste per una corretta applicazione in ambito sanitario.

2.1 Definizione

L'autenticazione è il processo con il quale si determina se qualcuno o qualcosa è, effettivamente, chi dichiara di essere. Pertanto possono coesistere sistemi di autenticazione dell'utente, del client o del server:

- **Autenticazione utente:** garantisce che l'utente, nella maggior parte dei casi una persona, è l'effettivo proprietario delle credenziali inserite.
- **Autenticazione del client:** garantisce che la macchina o il terminale con il quale si accede al sistema, sia un dispositivo noto alla rete e pertanto affidabile. Con l'avvento di connessioni wifi e dispositivi mobili in grado di accedere alla rete, questo tipo di autenticazione sta diventando sempre più importante per la realizzazione di un sistema affidabile e sicuro.
- **Autenticazione del server:** oltre all'autenticazione dei client è importante che lo stesso server sia in grado di dimostrare la propria identità, per evitare che utenti malevoli possano reindirizzare gli utenti al proprio server, emulando il portale originale (tipico degli attacchi phishing).

L'unione dell'autenticazione client e di quella server è detta complessivamente autenticazione della connessione, o mutua autenticazione client-server, in quanto entrambe le parti verificano le proprie credenziali a vicenda.

Perché un'applicazione possa garantire un buon sistema di autenticazione, deve essere in grado di fornire tutte e tre i tipi di autenticazione sopra elencati. In questa tesi saranno studiati principalmente i meccanismi di autenticazione utente.

L'autenticazione gioca un ruolo fondamentale nella sicurezza di un'applicazione web poiché tutte le successive decisioni sono tipicamente compiute sulla base dell'identità stabilita dalle credenziali fornite.

2.1.1 Tecniche di autenticazione

Le tecniche di autenticazione sono estremamente differenziate, sia come metodo che come efficacia, in funzione di diversi fattori. In termini semplicistici, si può dire che una tecnica è efficace se garantisce con elevata probabilità (tendente a 1) che il soggetto che ha richiesto l'accesso sia effettivamente chi dichiara di essere.

La complessità di una tecnica di autenticazione dipende essenzialmente da quanto critica deve essere l'identificazione: quali fattori possono limitarla, che valore hanno le informazioni garantite in seguito all'accesso e quali sono le implicazioni in caso di fuga di tali informazioni.

Tipicamente le tecniche di autenticazione si distinguono sulla base delle entità alle quali fanno riferimento:

- Qualcosa che si conosce: autenticazione con username/password.
- Qualcosa che si possiede: autenticazione con chiavi o certificati.
- Tratti unici: dispositivi biometrici.

Più tecniche possono combinarsi per garantire un maggior livello di sicurezza. Va specificato comunque che nessun sistema di autenticazione può dirsi realmente efficace se non vi è la stretta collaborazione dell'utente.

2.2 Vulnerabilità comuni ai sistemi di autenticazione con username e password

I sistemi di autenticazione con username e password sono i più diffusi grazie alla loro semplicità d'implementazione. I più comuni attacchi di cui sono vittima questi sistemi sono: *username enumeration*, *password guessing* ed *eavesdropping*.

2.2.1 Username enumeration

L'*username enumeration* (letteralmente "elencare i nomi utente"), consiste nello sfruttare le informazioni date dal sistema per identificare la lista degli username esistenti nel sistema. Questo permette di velocizzare nettamente il successivo processo d'individuazione della password.

Di seguito sono elencati alcuni metodi per determinare l'esistenza di uno username:

1. Messaggi di errore nel login

Accedendo con credenziali invalide, solitamente i sistemi restituiscono un messaggio di errore. Quando questo messaggio è troppo specifico, è possibile risalire all'esistenza dell'utente. Per esempio se un sistema restituisce i seguenti messaggi:

- a) Nome utente errato
- b) Password errata
- c) Nome utente o password errata

l'attaccante può sapere se un utente esiste (b).

2. Messaggi di errore nei meccanismi di recupero password

Simile al precedente, ma la vulnerabilità risiede nel meccanismo di recupero password (il classico "Hai dimenticato la password?" delle più comuni applicazioni web). Di solito questi meccanismi richiedono l'inserimento del nome utente o dell'indirizzo e-mail associato a quell'account. Questo metodo "autentica" l'utente, dando per scontato che solo lui può accedere al contenuto dell'e-mail associata all'account. Tuttavia, una cattiva implementazione di tale meccanismo spesso riferisce se il nome utente o l'indirizzo e-mail inserito è valido. Da queste informazioni si può risalire all'esistenza dell'username. Inoltre, se il sistema im-

plementa un meccanismo di SSPR (Self-Service Password Reset), è possibile, tramite un attacco DoS, sommergere l'account degli utenti il cui username è stato scoperto con continue e-mail contenenti nuove password.

3. Registrazione

Molte applicazioni permettono agli utenti di registrarsi scegliendo il proprio username. Durante la procedura di registrazione, se viene selezionato un nome utente già in uso da un altro account, viene visualizzato un messaggio di errore che specifica di inserire un nome utente diverso.

4. Attacchi temporizzati

Se nessuna delle tecniche precedenti fornisce risultati accettabili, l'ultima risorsa per gli attaccanti è temporizzare i tempi di risposta dei messaggi di errore. A seconda dell'algoritmo implementato per controllare gli accessi, è possibile che un messaggio di errore dovuto ad un nome utente errato, venga generato più velocemente di uno dovuto all'inserimento sbagliato della password (ciò capita spesso perché la password attraversa delle procedure di cifratura prima di essere controllata). Osservando le differenze di tempo intercorse tra un tentativo e l'altro, è possibile ottenere indizi sull'esistenza dell'username. Tuttavia perché questa tecnica sia efficace, le differenze di tempo devono essere sufficientemente grandi da compensare le fluttuazioni dovute al carico e alla latenza della rete.

Per proteggersi da questo tipo di attacco è sufficiente prestare attenzione alle informazioni che vengono restituite in fase di accesso. Purtroppo però, nelle applicazioni che prevedono sistemi di registrazione degli utenti automatici, le contromisure non sono facilmente realizzabili. Per questo motivo molti web master hanno accettato i rischi legati alla possibilità di indovinare gli username, in cambio di una maggiore libertà e flessibilità delle loro applicazioni.

2.2.2 Password guessing

Solitamente, le tecniche di *password guessing* (letteralmente “tirare a indovinare la password”) possono essere eseguite a prescindere dal protocollo di autenticazione in uso.

1. Password guessing manuale

Indovinare manualmente la password è un compito noioso, ma i vantaggi dell'intuito umano lo rendono spesso più efficiente e veloce di uno strumento automatico, soprattutto quando i messaggi di errore sono particolarmente dettagliati. Questa tecnica si appoggia a dizionari di password comunemente utilizzati e facilmente reperibili in rete. Attraverso strumenti appositi, è possibile inviare un intero dizionario di coppie username/password a un'applicazione. Inoltre, se si è a conoscenza delle politiche di creazione delle password adottate dall'applicazione, si può ridurre l'insieme delle possibili password da inserire nel dizionario. Per esempio, se la politica permette l'utilizzo solo di password con almeno un numero al loro interno, è inutile includere nel dizionario parole che non ne hanno.

2. Password guessing automatico

Esistono principalmente due approcci al *password guessing* automatico: *depth first* e *breadth first*. Il primo prova tutte le combinazioni possibili di password per un determinato username. Questo può portare facilmente al blocco dell'account, se tale caratteristica è stata implementata nell'applicazione in cui accedere. Il secondo invece lavora al contrario: la stes-

sa password viene provata per diversi username. In questo modo si evita il rischio di blocco dell'account in applicazioni dotate di questa caratteristica.

Alcuni dei principali software per scovare le password sono: THC-Hydra [23], WebCracker [24] e Brutus [25].

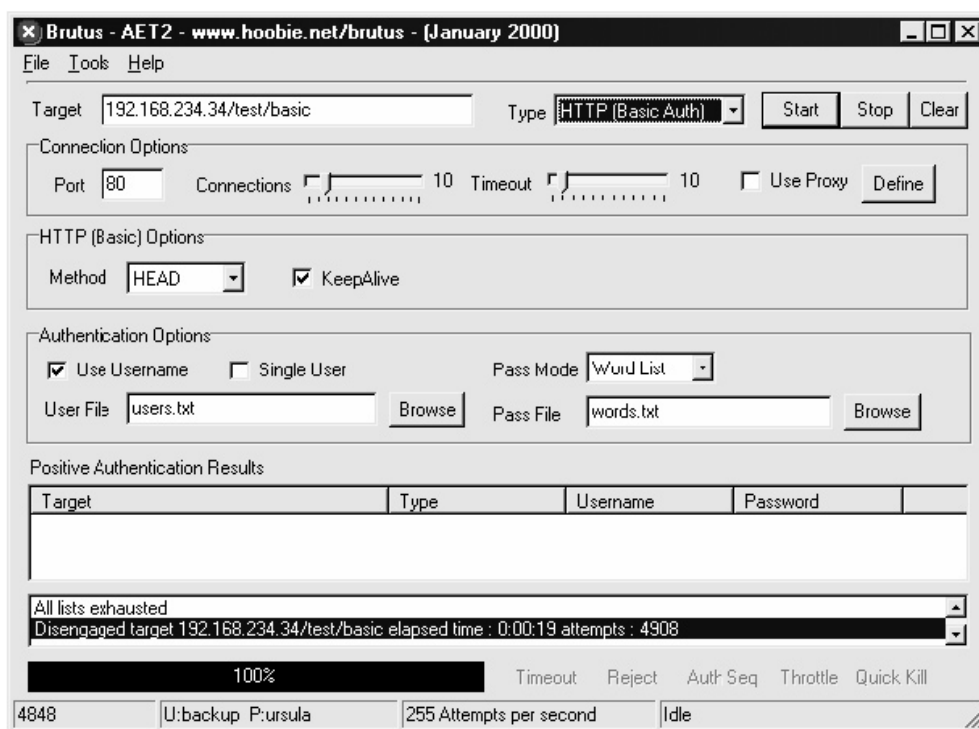


Figura 2.1. Un'esecuzione di Brutus per un sistema di autenticazione base HTTP.

Le contromisure più efficienti contro attacchi di questo tipo consistono nella combinazione di rigide politiche di creazione delle password e di blocco degli account. Va tenuto conto però, che un'applicazione che implementa il blocco degli account è vulnerabile ad attacchi di tipo DoS. Un utente malevolo potrebbe tentare di bloccare tutti gli account del sistema con tentativi di autenticazione ripetuti. Un buon compromesso consiste nel bloccare solo temporaneamente l'account utente. Questo rallenterebbe notevolmente la velocità richiesta per indovinare le password e quindi ridurre l'efficienza dell'attacco. Una politica severa di creazione delle password, riduce il rischio di indovinare casualmente le password.

Recentemente, alcuni siti hanno iniziato a tracciare l'indirizzo IP dei client che accedono al sistema, legandoli all'account. L'accesso da un IP insolito fa scattare dei meccanismi di autenticazione aggiuntivi, come i CAPTCHA (Completely Automated Public Turing Tests to Tell Computers and Humans Apart) [26], dei test composti da una o più domande (solitamente un'immagine con caratteri alfanumerici da riconoscere e inserire in un apposito campo) capaci di determinare dalle risposte se l'utente è un essere umano o un bot. Queste tecniche permettono di prevenire gli attacchi distribuiti o di *password guessing*.

Infine, è buona norma salvare in un log di sistema tutti gli eventi relativi ad autenticazioni fallite e monitorare tale log periodicamente per individuare segni di possibili attacchi a forza bruta.

2.2.3 Eavesdropping

Qualsiasi protocollo di autenticazione che espone le credenziali in transito sulla rete, è potenzialmente vulnerabile ad attacchi *eavesdropping* (letteralmente “origliare”), detti anche attacchi di *sniffing* (letteralmente “fiutare”). L’*eavesdropping* è alla base degli attacchi di tipo *replay*, il quale utilizza le credenziali “origliate” per simulare, in risposta, l’identità dell’utente originale.

L’autenticazione base viene inizializzata nel momento in cui un client invia una richiesta di una risorsa protetta a un web server. A questa richiesta, il server risponde con un HTTP 401 Unauthorized, un messaggio che specifica la richiesta di autenticazione causando la comparsa di una finestra nel browser, simile a quella in Figura 2.2.

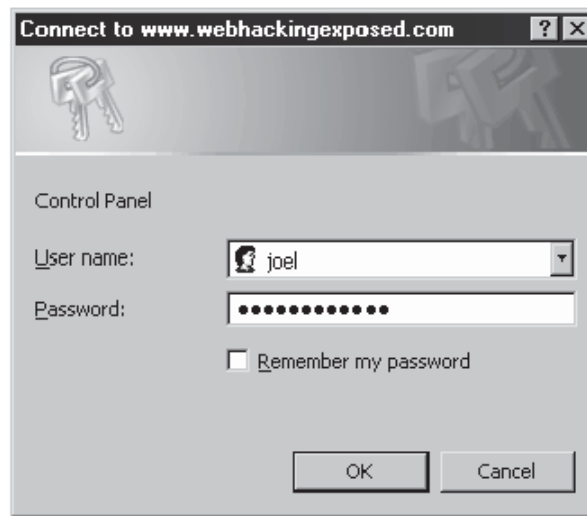


Figura 2.2. La finestra di autenticazione base di un browser.

L’utente inserisce le sue credenziali e un messaggio simile al seguente viene inviato via HTTP:

```
GET /test/secure HTTP/1.0
Authorization: Basic dGVzdDp0ZXN0
```

Username e password sono incapsulate nella sezione *Authorization* dell’header, codificate con l’algoritmo Base64 [27]. L’intero flusso è riassunto in Figura 2.3. Trattandosi di un sistema di numerazione posizionale, il Base64 è facilmente decodificabile e pertanto l’autenticazione base è vulnerabile agli attacchi di tipo *eavesdropping*. L’utilizzo dell’HTTPS (HyperText Transfer Protocol over Secure socket layer), che applica un protocollo di cifratura asimmetrica al protocollo standard HTTP, permette di mitigare questa limitazione.

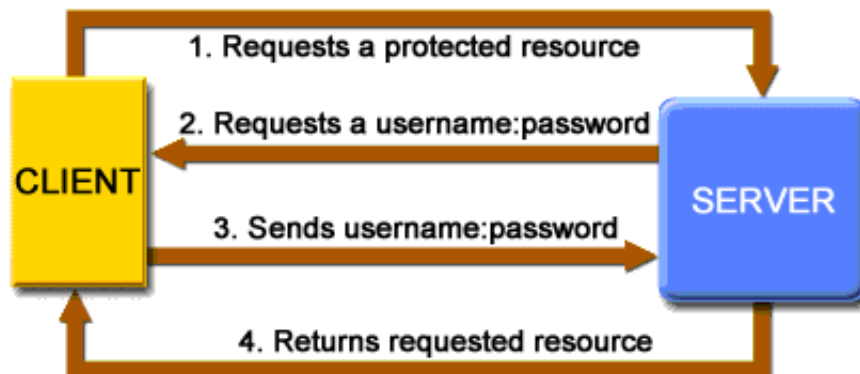


Figura 2.3. Flusso esecutivo di una richiesta di autenticazione HTTP base.

L'autenticazione tramite digest è stata appositamente pensata per fornire un maggiore livello di sicurezza rispetto a quella di base. L'autenticazione con digest si basa sul modello di autenticazione *challenge-response*, in cui una delle parti pone una domanda ("challenge") e l'altra deve fornire una valida risposta per autenticarsi ("response"). Il meccanismo è illustrato in Figura 2.4. In seguito ad una richiesta di accesso da parte del client, il server risponde inviando un valore casuale detto *nonce* (contrazione di *number used once*), un numero che viene utilizzato una sola volta e poi scartato. Il browser usa quindi una funzione di cifratura asimmetrica per generare un messaggio di *digest* con la richiesta, il nome utente, la password e il nonce stesso. Il server quindi confronta l'hash e se coincide la richiesta viene soddisfatta.

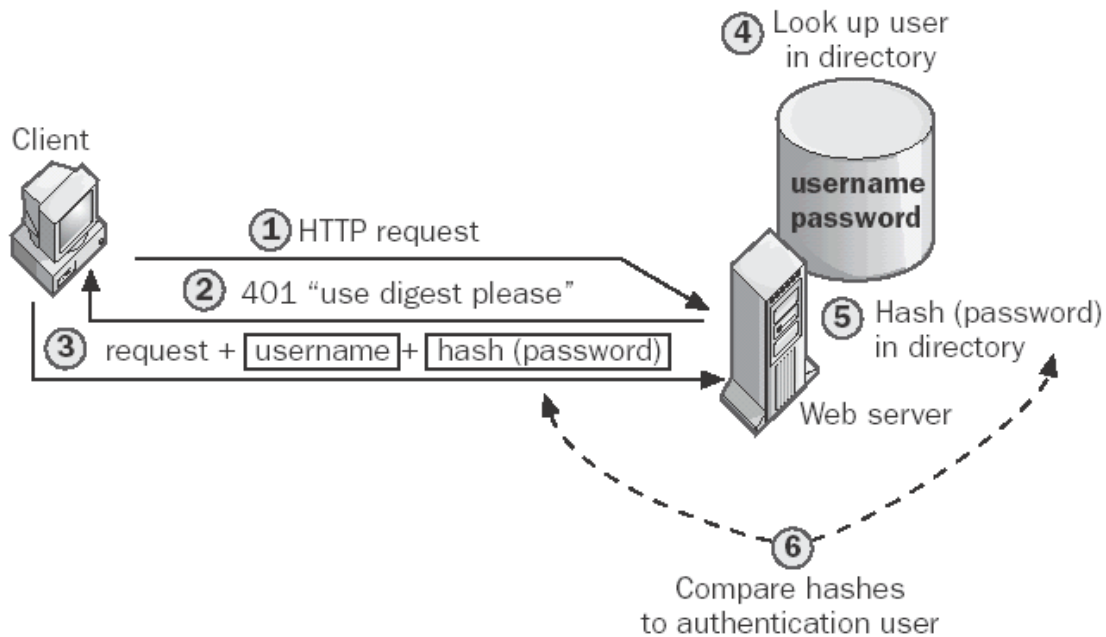


Figura 2.4. Funzionamento dell'autenticazione con digest.

L'autenticazione con digest è un notevole passo avanti rispetto all'autenticazione di base, in quanto le credenziali utente non sono inviate in chiaro sulla rete.

2.3 Autenticazione con username e password basata su form

Diversamente da quanto descritto finora, l'autenticazione basata su form non dipende dalle caratteristiche supportate dai protocolli di base come l'HTTP, ma utilizza dei meccanismi di autenticazione altamente personalizzabili chiamati form.

Dopo aver inviato le credenziali tramite HTTP o HTTPS, esse vengono controllate da una logica server-side e se valide, viene restituito al client uno speciale token casuale unico. Questo token verrà utilizzato dal client per tutte le successive richieste. Grazie alla sua elevata flessibilità e le numerose tecnologie a disposizione per realizzarla, l'autenticazione su form è probabilmente la tecnica di autenticazione più adottata su Internet.

Le form, solitamente presentano un tasto "Login", che, una volta premuto, invia i dati al server tramite uno dei due metodi GET e POST (solitamente è preferibile il secondo per evitare

l'esposizione delle credenziali utente o altre informazioni riservate). Va notato che a meno di aver implementato un canale sicuro SSL, sia che vengano inviate in GET che in POST, le credenziali attraversano la rete in chiaro. Se le credenziali corrispondono a quelle presenti nella banca dati del server, esso restituisce il token di autenticazione (*cookie*) che il client potrà successivamente utilizzare per richiedere le risorse. L'intero flusso è illustrato in Figura 2.5.

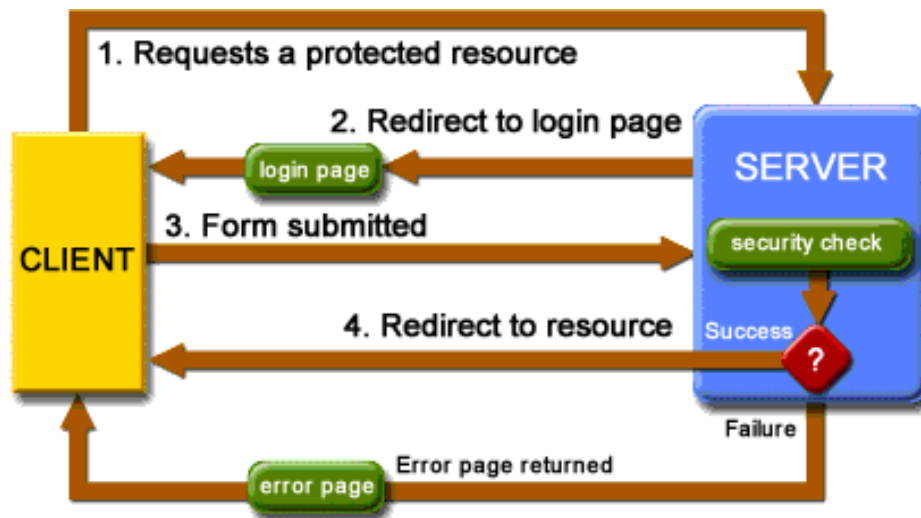


Figura 2.5. Flusso esecutivo di un'autenticazione basata su form.

Come gli altri metodi di autenticazione discussi finora, anche quelli basati su form sono vulnerabili agli attacchi di *password guessing* e di *eavesdropping* (se il canale non è cifrato con HTTPS o altri protocolli). Inoltre, l'ampia gamma di tecnologie per la gestione degli strati applicativi e dati, aggiunge nuove vulnerabilità a questo tipo di autenticazione.

2.3.1 Attaccare i token di sicurezza

I token di sicurezza sono dei meccanismi comunemente utilizzati dalle applicazioni web per ricordare le credenziali dell'utente durante la navigazione. Sfortunatamente, questo meccanismo permette di bypassare l'autenticazione semplicemente replicando un token catturato. Questo fenomeno viene chiamato *session hijacking*. Tipicamente, le applicazioni web tengono traccia delle sessioni utente tramite l'utilizzo di ID salvati in cookie del browser.

I cookie solitamente conservano informazioni importanti associate all'autenticazione. Se i cookie contengono password o identificatori di sessione, rubarli può essere un ottimo sistema per violare la sicurezza di un sito web.

Lo stesso reverse engineering dei cookie offline può garantire degli ottimi risultati. L'approccio migliore è quello di raccogliere un certo numero di campioni di cookie utilizzando diversi input per vedere come essi cambiano. Attacchi di *bit-flipping* adottano un approccio a forza bruta, modificando metodicamente i bit, per vedere se il nuovo cookie è ancora valido e se fornisce risultati diversi (in particolare accessi con maggiori privilegi). In particolare, bisogna prestare particolare attenzione alla codifica utilizzata per il cookie. Uno degli errori più comuni fatto dagli sviluppatori è quello di utilizzare una codifica simmetrica facilmente decodificabile, come il Base64.

L'eavesdropping è il metodo più semplice per rubare i token di sicurezza. Canali sicuri come SSL o altre tecniche di cifratura della sessione, sono sufficienti a prevenire questo tipo di attacco. In generale comunque, l'approccio migliore è di utilizzare un identificativo di sessione non predicibile, im-

plementare dei controlli di integrità dei cookie per identificare la manomissione (in particolare si possono utilizzare dei codici hash di autenticazione dei messaggi, HMAC) e in linea di principio, non salvare mai informazioni sensibili su token client-side.

Esistono tre famiglie di attacchi ai token di sessione:

- Predizione (manuale o automatizzata)
- Capture / Replay
- Fixation

2.3.1.1 Predizione manuale

La predizione manuale è spesso efficiente in quelle situazioni in cui il valore del token di accesso è costruito in una sintassi o formato umanamente leggibile. I più comuni meccanismi utilizzati per tracciare lo stato della sessione sono le stringhe di query, i dati inviati in POST, gli headers HTTP e i cookie.

Query string

La string di query passata dal client può contenere diverse coppie attributo-valore passate nell'URI dopo il carattere di punto esclamativo (?). Per esempio, il seguente URI contiene il nome utente della mail da controllare, passati in GET:

```
http://www.mail.com/mail.aspx?mailbox=joe&company=acme
```

L'utilizzo del metodo POST per l'invio dei dati è preferibile in quanto non appare visibile nell'URI. Tuttavia, il fatto che il client non “veda” questi dati, non significa che essi siano illeggibili o impossibili da manipolare. Oscurare i dati garantisce solamente una blanda confidenzialità dei dati. Per i dati sensibili è meglio utilizzare meccanismi di sicurezza ben più sofisticati.

Dati POST

Le informazioni inviate in POST spesso contengono informazioni legate alla sessione e autorizzazione. Di seguito viene presentato un esempio di pagina HTML per il login di un'applicazione web:

```
<form name="login_frm" action="https://www.victim.com/login" method="post">
  <input name="Tries" type="hidden" />
  <input value="us" name="I8N" type="hidden" />
  <input name="Bypass" type="hidden" />
  <input value="64mbvjoubpd06" name="U" type="hidden" />
  <input value="pVjsXMKjKD8rlggZTYDLW" name="Challenge" type="hidden" />
  User Name: <input name="Login" />
  Password: <input type="password" maxlength="32" name="Passwd" />
</form>
```

Quando l'utente invia il proprio username e password, in realtà sta inviando sette informazioni. I campi di tipo “hidden”, infatti, anche se non visibili dal browser, sono in realtà facilmente accessibili e modificabili. Alcuni di questi campi, come `Bypass` o `U`, potrebbero svelare vulnerabilità non previste.

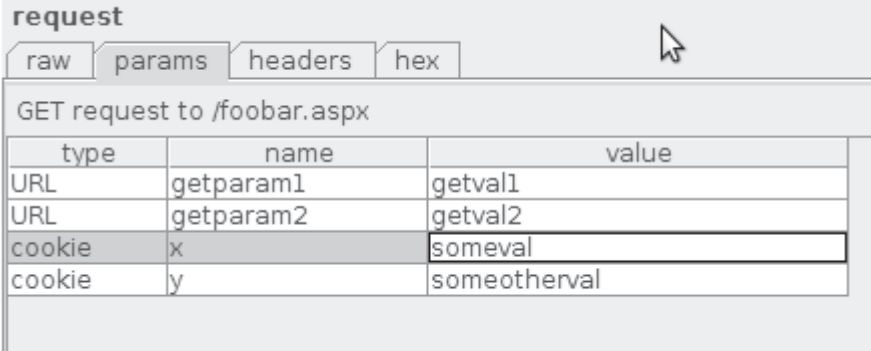
Header HTTP

Gli header HTTP vengono inviati come parte del protocollo stesso e in alcuni casi contengono informazioni da passare ai meccanismi di autorizzazione o alla sessione. I cookie sono gli header più

conosciuti dell'HTTP, e vengono spesso utilizzati per il tracciamento dello stato della sessione e dell'autorizzazione. Alcuni controlli vengono effettuati sulla base di valori indicati dall'header (come per esempio il campo `User-Agent` che determina il browser utilizzato dal client). Esistono diverse applicazioni, come *cURL*, capaci di modificare gli header, quindi l'applicazione di tali sistemi di controllo è completamente inutile.

Cookie

I cookie vengono spesso utilizzati per conservare qualsiasi tipo di dato. Purtroppo però tali campi sono facilmente modificabili utilizzando strumenti di analisi HTTP, reperibili sul web. L'illustrazione in Figura 2.6 mostra uno di tali strumenti modificare il valore di un cookie.



The screenshot shows a web proxy tool interface with tabs for 'raw', 'params', 'headers', and 'hex'. The 'params' tab is selected, displaying a table of request parameters for a GET request to /foobar.aspx. The table has three columns: 'type', 'name', and 'value'. One of the rows, representing a cookie, is highlighted with a mouse cursor.

type	name	value
URL	getparam1	getval1
URL	getparam2	getval2
cookie	x	someval
cookie	y	someotherval

Figura 2.6. Modifica di un cookie con Burp.

Un altro esempio di cookie è il seguente, che descrive un'implementazione della funzionalità “remember me”, tipica delle applicazioni web:

```
Set-Cookie: autolog=bWlrZTpteXMzY3IzdA%3D%3D; expires=Sat, 01-Jan-2012 00:00:00 GMT; path=/; domain=victim.com
```

Anche senza decodificare il codice, pure un attaccante inesperto intuisce che gli è sufficiente replicare tale cookie per impersonare l'utente corrispondente. Se poi si esamina in dettaglio il campo “autolog”, è facile rendersi conto che il valore utilizza una codifica Base64. Una veloce decodifica del campo restituisce la stringa ASCII `mike:mys3cr3t`, che sono chiaramente il nome utente e la password dell'utente in questione. Non solo, è possibile modificare il campo “expires” per prolungare la durata di scadenza del cookie.

2.3.1.2 Predizione automatica

Se i cookie non sono scritti in un formato umanamente leggibile, un'analisi automatica può aiutare a identificare le potenziali falle di sicurezza. I passi fondamentali per l'analisi automatica sono i seguenti:

1. Raccogliere un adeguato numero di campioni da analizzare. Questo può essere svolto manualmente o, più velocemente, attraverso script e software specializzati.
2. Analizzare la casualità degli ID di sessione. Si possono utilizzare strumenti di analisi non lineare o altre tecniche simili. Ciò servirà a stabilire il grado di difficoltà di predizione dei valori e i limiti del vocabolario utilizzato.
3. Eseguire l'analisi automatica con tecniche a forza bruta o di bit flipping. Solitamente il valore da indovinare è il risultato di una combinazione di diversi singoli campi, come la con-

catenazione della password più il nome utente, oppure il timestamp più la password, ecc. In questi casi è sufficiente individuare eventuali ripetizioni nei valori ed eseguire gli algoritmi di ricerca sui caratteri che cambiano.

4. Provare i nuovi cookie generati e individuare eventuali cambiamenti nelle risposte del sistema per stabilire le correlazioni tra porzioni di codice e i comportamenti dell'applicazione.

Non approfondiamo i dettagli di questa tecnica in quanto possono essere realizzati in molteplici modi. In generale, è sufficiente sapere che le metodologie sono simili a quelle adottabili per decodificare la tecniche di cifrature.

2.3.1.3 Capture / Replay

Le tecniche di predizione sono spesso o facili da applicare a causa di distrazioni da parte dello sviluppatore o impossibili. Ecco perché è preferibile evitare la complessità di analizzare i token, semplicemente replicando il token di un altro utente. Se la replicazione riesce, l'attaccante diventa a tutti gli effetti quell'utente.

L'attacco *capture/replay* consiste nell'acquistare il token di un altro utente attraverso tecniche di eavesdropping, man-in-the-middle e ingegneria sociale. Dell'eavesdropping si è già discusso nel paragrafo 2.2.3.

Un attacco di tipo *man-in-the-middle*, è una forma di eavesdropping attivo in cui l'attaccante crea delle connessioni indipendenti tra le vittime e trasmette i messaggi tra di loro, facendo credere che siano in comunicazione diretta su una connessione privata, quando in realtà l'intera conversazione è controllata dall'attaccante. L'attaccante deve essere in grado di intercettare tutti i messaggi tra le due vittime. Le tecniche per realizzare questo tipo di attacco sono molteplici e dipendono dalla situazione, tuttavia, senza scendere nel dettaglio, un attacco di questo tipo può addirittura compromettere una sessione cifrata o ingannare un utente remoto ad accettare un certificato SSL scaduto.

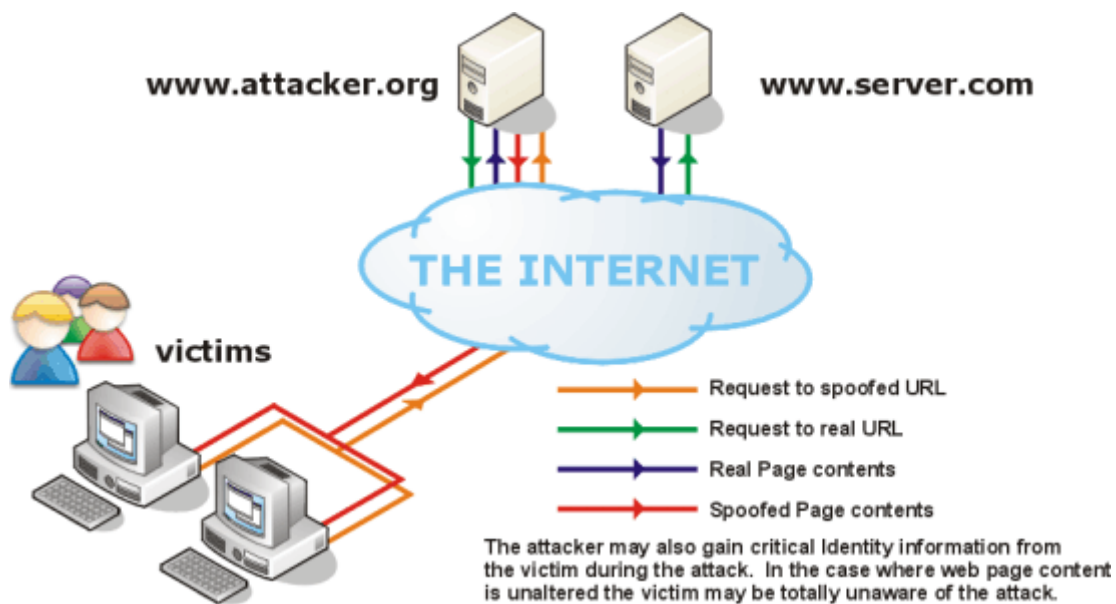


Figura 2.7. Esempio di attacco man-in-the-middle.

Infine, un metodo semplice ma spesso efficiente, è quello di chiedere, con l'inganno, l'ID di sessione direttamente alla vittima. Questa tecnica è detta ingegneria sociale, e verrà discussa in maggior dettaglio nel paragrafo 2.4.

2.3.1.4 Attaccare l'ID di sessione

Le applicazioni web solitamente tracciano le sessioni degli utenti attraverso ID di sessione salvate in appositi cookie. Esistono due tecniche per ottenere gli ID di sessione: predizione o forza bruta.

Le applicazioni più vecchie spesso utilizzano identificativi di sessione predicibili o addirittura sequenziali. ID non sequenziali generati usando algoritmi non sicuri o generatori di numeri pseudo-casuali con entropia insufficiente, possono essere facilmente prevedibili con l'ausilio di tecniche matematiche di previsione statistica.

La tecnica a forza bruta consiste invece, nell'effettuare migliaia di richieste utilizzando tutte le possibili combinazioni di ID di sessione, nella speranza di indovinarne una corretta. Il numero di richieste dipende dalla grandezza dello spazio delle chiavi utilizzato per l'ID e quindi la probabilità di successo può essere calcolata basandosi sulla dimensione e il vocabolario utilizzato. Fortunatamente la dimensione degli ID di sessione generati dalle più comuni piattaforme di applicazioni web, come Java, PHP, ASP.net, ecc., sono sufficientemente grandi da garantire un'adeguata sicurezza a questo tipo di attacco.

Proprio per questa ragione, si è ampiamente diffusa una nuova famiglia di attacchi detta *session fixation*. Lo scopo di questi attacchi è di fissare in anticipo l'ID di sessione che un'applicazione utilizzerà in una successiva autenticazione utente. Siccome è l'attaccante a fissare il valore dell'ID, un utente che accede utilizzando questo ID prefissato, sarà immediatamente esposto ad attacchi di *session hijacking*.

Un tipico attacco di session fixation funziona nel seguente modo:

1. L'attaccante accede ad un'applicazione vulnerabile, impostando un ID di sessione valido, che verrà utilizzato per "intrappolare" la vittima.
2. Quindi convince la vittima ad accedere alla stessa applicazione utilizzando lo stesso ID di sessione (la crittoanalisi dell'ingegneria sociale è uno dei metodi utilizzabili [28]).
3. Una volta che la vittima si è autenticata nell'applicazione, l'attaccante replica lo stesso ID di sessione, manipolando la stessa sessione della vittima (in un certo senso si può dire che la vittima si è autenticata con la sessione dell'attaccante).

Questo attacco, apparentemente semplice, presenta fortunatamente due inconvenienti:

- L'attaccante deve convincere la vittima a lanciare l'URI che lo autenticerà nell'applicazione usando l'ID "trappola". Va specificato che se si riesce a convincere qualcuno ad eseguire tale azione, ci sono allora cose ben peggiori che si può riuscire a fargli fare.
- L'attaccante deve autenticarsi utilizzando lo stesso ID di sessione prima che la vittima si disconnetta e la sessione scada (ovviamente se l'applicazione non gestisce adeguatamente la chiusura delle sessioni, queste possono diventare delle finestre perennemente aperte).

Esistono inoltre delle contromisure molto semplici da applicare per prevenire completamente questo tipo di attacco: generare un nuovo ID di sessione ad ogni autenticazione riuscita, permettere

solamente al server di scegliere tale ID e assicurare che le sessioni scadano basandosi su una logica server-side e con tempi di scadenza assoluti.

2.3.2 Cross-site request forgery

Il cross-site request forgery (CSRF o XSRF) è un attacco che sfrutta la relazione di fiducia tra l'applicazione web e l'autenticazione utente per forzare tale utente ad eseguire transazioni arbitrarie per conto dell'attaccante.

In cosa consiste quest'attacco? Prendiamo in considerazione un'applicazione bancaria che permette a un utente correttamente autenticato, di trasferire dei fondi da un conto corrente a un altro usando delle semplici richieste GET. Assumiamo che l'azione di trasferimento sia la seguente:

```
http://xsrft.vulnerablebank.com/transferFunds.aspx?toaccount=12345&funds=1000.00&currency=dollars
```

Ora supponiamo che un'attaccante realizzi una pagina HTML contenente il seguente codice JavaScript su un sistema sotto il suo controllo:

```
<script type="text/javascript">
  var i = document.createElement("image");
  i.src =
    "http://xsrft.vulnerablebank.com/transferFunds.aspx?toaccount=EVIL_ATTACKER_ACCOUNT_NUMBER&funds=1000.00&currency=dollars"
</script>
```

L'effetto di questo JavaScript è di creare un'immagine dinamica HTML, che ha impostato come sorgente la pagina che gestisce il trasferimento di fondi sull'applicazione bancaria di prima. Il browser delle vittime autenticate con il sito della banca che vengono attirati sulla pagina dell'attaccante, eseguiranno inconsapevolmente il codice JavaScript come se l'avessero eseguito di loro spontanea volontà. Il risultato è che l'attaccante è riuscito a forzare l'utente di una banca a trasferire i fondi dal conto dell'utente al proprio. Va notato che anche se l'esempio illustrato fa uso di una richiesta GET, anche le richieste POST sono vulnerabili.

Esistono principalmente tre metodi per prevenire attacchi XSRF:

1. **Double-posted cookie** – con questa tecnica, ogni form utilizzata per eseguire una transazione di dati delicati, è generata con un campo di input nascosto che contiene il valore del corrente ID di sessione utente o un altro valore casuale sicuro salvato in un cookie client-side. Quando il form viene inviato, il server verifica se il valore del cookie nella form corrisponde al valore ricevuto nell'header HTTP. Se i valori non corrispondono, la richiesta viene rifiutata e viene generato un messaggio di audit per segnalare un possibile attacco. Questo metodo si basa sull'assunzione che l'attaccante non conosce il valore di sessione del client.
2. **Unique form nonce** – nella strategia basata su un unico nonce, ogni form è realizzato in modo che ad ogni richiesta un certo campo di input nascosto contenga un nonce casuale sicuro. Il nonce deve essere generato con un generatore di numeri pseudocasuali criptograficamente sicuro per non essere vulnerabile agli attacchi. Quando il server riceve la richiesta di POST, confronta il valore del nonce con il valore salvato in memoria e rigetta le richieste che espongono un nonce diverso o scaduto. Questo metodo può essere difficile da realizzare, soprattutto se l'applicazione richiede di associare la generazione e scadenza del nonce a ogni richiesta che contiene dati importanti.

3. **Richiedere credenziali di autenticazione** – questo metodo richiede all'utente di reinserire le proprie credenziali come ulteriore verifica per ogni transazione in cui sono coinvolti dati sensibili. È importante considerare soluzioni di audit e blocco dell'account in questo tipo di pagine per prevenire possibili attacchi a forza bruta per indovinare la password.

Come esempio, consideriamo come il form della banca di prima può prevenire questo attacco usando un nonce unico:

```
<form id="fundsTransfer" method="post" action="transferFunds.PHP">
  <input type="text" name="funds" value="0.00" />
  <input type="text" name="toaccount" value="" />
  <input type="hidden" name="xsrftoken" value="eozMKoWO6g3cIUa13y5wLw" />
</form>
```

Il parametro aggiuntivo nascosto `xsrftoken` è generato casualmente ogni volta che la pagina viene richiesta. In questo modo l'attaccante non può conoscere questo valore e sarà pertanto incapace di creare una form analoga che trasferisca i fondi al suo conto.

2.3.3 Input injection

La validazione degli input serve come prima linea difensiva di un'applicazione web. Molte vulnerabilità, come SQL injection, HTML injection (e i vari tipi di cross-site scripting) e messaggi di errore eccessivamente dettagliati, permettono a un attaccante di iniettare un qualche tipo di input inaspettato o dannoso nell'applicazione. Se implementata a dovere, le routine di validazione dell'input assicura che i dati siano di un formato, tipo, lunghezza e valori utili all'applicazione. Senza questi controlli, la confidenzialità, integrità e disponibilità di un'applicazione e delle sue informazioni sono a rischio.

2.3.3.1 SQL Injection

Uno degli attacchi più pericolosi per un'applicazione web è sicuramente SQL injection. Questo attacco consiste nell'iniettare del codice SQL dentro una query costruita dinamicamente. Iniettando sintassi SQL, la logica della query viene modificata in modo che la sua esecuzione produca un risultato diverso. Nella sintassi di SQL, l'apice singola, delimita l'inizio o la fine di una stringa. Quindi, quando si inietta un apice in una query vulnerabile, si può potenzialmente interrompere l'accoppiamento tra delimitatori di stringa e generare un errore applicativo.

Le vulnerabilità a SQL injection possono trovarsi in qualsiasi parametro che influenzi una query al database. I punti di attacco includono parametri passati nell'URL, dati POST, cookie, input form, ecc. Il modo migliore per identificare una vulnerabilità a SQL injection è provare diversi caratteri o valori speciali e valutare le risposte dell'applicazione.

Dipartimento di Ingegneria dell'Informazione - Università degli studi di Padova

Login

Accesso tramite username e password.

Username:

Password:

Progetto di tesi sviluppato da Massimiliano Sciacco (max.sciacco@gmail.com), presso l'università degli studi di Padova, in collaborazione con il Policlinico universitario.

Figura 2.8. Una form di login con controllo delle credenziali basato su database.

L'esempio classico di vulnerabilità SQL injection è nel form di login, come quello in Figura 2.8. Un'applicazione basata su database, può utilizzare una query simile alla seguente per convalidare il nome utente e la password di un utente:

```
SELECT * FROM userTable WHERE username = '$username' AND password = '$password'1
```

Se l'utente fornisce le credenziali corrette, la query restituirà una riga e l'applicazione consentirà l'accesso all'utente, altrimenti il risultato della query sarà un insieme vuoto a indicare che una o entrambe le credenziali sono errate. Immaginiamo che non ci sia nessun controllo dell'input e proviamo a inserire negli appositi campi, dei valori che permettano l'accesso solamente inserendo il nome utente. In questo caso è sufficiente usare la stringa "admin'%20--%20" (%20 è il carattere spazio nella codifica url esadecimale [29]) nel campo username e lasciare vuoto il campo password. Vediamo cosa succede alla query:

```
SELECT * FROM users WHERE username = 'admin' -- AND password = ''
```

La parte di SQL inserita è quella sottolineata. Il doppio carattere di trattino -- viene interpretato come "commenta il resto della query". Questo significa che quest'ultima query garantirà l'accesso semplicemente verificando l'esistenza di un username con valore admin.

Vediamo un altro esempio. Nel campo password viene inserito il valore "' OR 1='1'" e la query diventa:

```
SELECT * FROM users WHERE username = '' AND password = '' OR 1='1'
```

Come prima, la parte sottolineata è la porzione di codice iniettato. Con questo piccolo accorgimento è possibile creare un'identità che rende la concatenazione di clausole where sempre vere, consentendo l'ingresso anche senza conoscere un username valido.

Spesso però, per evitare il trasferimento in chiaro sulla rete, il campo password attraversa dei meccanismi di cifratura che trasformano la stringa inserita, in qualcosa di incomprensibile. Per esempio se il nostro sistema di autenticazione cifra la password con un algoritmo crittografico di hashing MD5 (Message Digest algorithm 5) [30], la stringa "' OR 1='1'" si trasforma in un ammasso incomprensibile di caratteri: "b3a5e686a8b1c32f6f0a236defd19fc7". Inutile dire che l'iniezione di tale stringa crea un errore di sintassi nel motore di elaborazione delle query. Come si può fare allora a bypassare l'autenticazione se non si conosce un username valido e il campo password è inutilizzabile perché tutto ciò che viene inserito al suo interno (compresa la stringa vuota) viene cifrata? Un metodo semplice è ricorrere nuovamente ai doppi trattini. Un input tipo "' OR 1=1 --" nel campo utente dovrebbe bastare:

```
SELECT * FROM users WHERE username = '' OR 1=1 -- ' AND password = ''
```

Oppure, si può concatenare, in maniera adeguata, una serie di condizionali e identità che renda l'intera clausola sempre vera. Per esempio si potrebbe inserire la stringa "' OR 1=1 OR 1='1'" ottenendo la seguente query (il valore dentro il campo password, è l'equivalente MD5 della stringa vuota):

```
SELECT * FROM users WHERE username = '' OR 1=1 OR 1='1' AND password = 'd41d8cd98f00b204e9800998ecf8427e'
```

¹ In questo e nei successivi esempi si fa uso della sintassi tipica di MySQL per la query e PHP per le variabili.

In assenza di parentesi, l'operatore AND ha sempre la precedenza sull'operatore OR. La seguente tabella riassume i risultati delle clausole:

Clausola	Alias	Risultato
username = ''	a	F
1=1	b	T
1='1'	c	T
password = 'd41d8cd98f00b204e9800998ecf8427e'	d	F

L'ordine di esecuzione dell'intera clausola va quindi interpretata in questo modo:

```
a OR b OR (c AND d) = F OR T OR (T AND F) = F OR T OR F = T OR F = T
```

Tecniche avanzate di SQL Injection

Gli esempi proposti sono solo una piccola parte delle possibilità che la vulnerabilità a SQL injection concede. Utilizzando query innestate (o sottoquery) è possibile risalire a informazioni che normalmente non sarebbero accessibili. Prendiamo per esempio una query che restituisca il prezzo di un oggetto filtrato per id. La query sarà simile alla seguente:

```
SELECT price FROM products WHERE id = $id
```

Proviamo a eseguire un injection inserendo nel campo predisposto all'id, la porzione di stringa sottolineata nella seguente query:

```
SELECT price FROM products WHERE id = (SELECT count(*) FROM users WHERE username = 'admin')
```

Questa query restituisce nome e cognome dell'utente il cui id è pari al numero di utenti che hanno come username admin. A prima vista questa informazioni non sembra molto utile, ma a un occhio più esperto, si intuisce subito un primo utilizzo. Se supponiamo che il campo username sia univoco (tipicamente vero per la maggior parte delle applicazioni), se l'username inserito esiste, il count(*) della query innestata varrà 1, altrimenti 0. Questo valore sarà quindi utilizzato come id del prodotto di cui si vuole conoscere il prezzo. In breve cosa succede? Per ogni username esistente nel sistema, verrà visualizzato sempre lo stesso prezzo, ossia quello del prodotto con id=1. Se l'username non esiste, allora la query restituisce il prezzo del prodotto con id=0 (tipicamente l'id è un valore positivo, quindi questo risultato restituisce un insieme vuoto). Utilizzando dei meccanismi a forza bruta, è possibile in questo modo eseguire un attacco di tipo username enumeration.

Un altro operatore utile negli attacchi di SQL injection è UNION, il quale permette di combinare i risultati di due SELECT separate, con l'unica eccezione che il numero di colonne restituite da ciascuna SELECT siano uguali, restrizione facilmente aggirabile con un po' di tentativi. Infatti, se la seconda tabella contiene meno colonne della prima, è sufficiente aggiungere dei campi null nella select; se invece ne ha di più, basta concatenare più campi in uno unico usando il comando CONCAT(a, b, c, d, ...). Per esempio, consideriamo le seguenti due tabelle, in cui la seconda è quella che vogliamo aggiungere alla query tramite un UNION:

tabella_1			tabella_2	
a1	b1	c1	a2	b2

La query complessiva deve essere simile alla seguente:

```
SELECT * FROM tabella_1 UNION SELECT *, null FROM tabella_2
```

Il campo null permette di equiparare il numero di colonne nelle due tabelle.

Nel caso contrario (la tabella da aggiungere è la prima) basta scrivere:

```
SELECT * FROM tabella_2 UNION SELECT CONCAT(a1, b1), c1 FROM tabella_1
```

A questo punto non resta che eseguire l'attacco e godersi lo spettacolo. In Figura 2.9, è illustrata una pagina che restituisce i log applicativi del sistema, opportunamente filtrati dagli appositi campi. O per lo meno, questo è quello che dovrebbe fare.



Figura 2.9. Una pagina di monitoraggio dei log vulnerabile a SQL injection.

Analizzando la form, è facile intuire che i singoli campi aggiungano nuovi filtri da aggiungere alla clausola WHERE della query. Supponendo che le clausole siano, ragionevolmente, aggiunte nello stesso ordine in cui compaiono i campi della form, proviamo a manipolare l'ultimo campo, quello che imposta un upper bound sulla data. Dopo vari tentativi, alla fine siamo riusciti ad ottenere il valore giusto da inserire nel campo:

```
2002-01-07' UNION SELECT id, username, password, name, surname, email, phone FROM users --
```

Il risultato è illustrato in Figura 2.10. La tabella elenca in ordine, l'id, il nome utente, la password, il nome, il cognome, l'indirizzo email e il numero di telefono di ogni utente. Sfortunatamente il campo password è cifrato, rendendo difficile accedere agli altri account, ma le altre informazioni sono tutte in chiaro. Provando diversi nomi di campi, è possibile risalire ad altri dati sensibili, come l'indirizzo di residenza, per esempio.

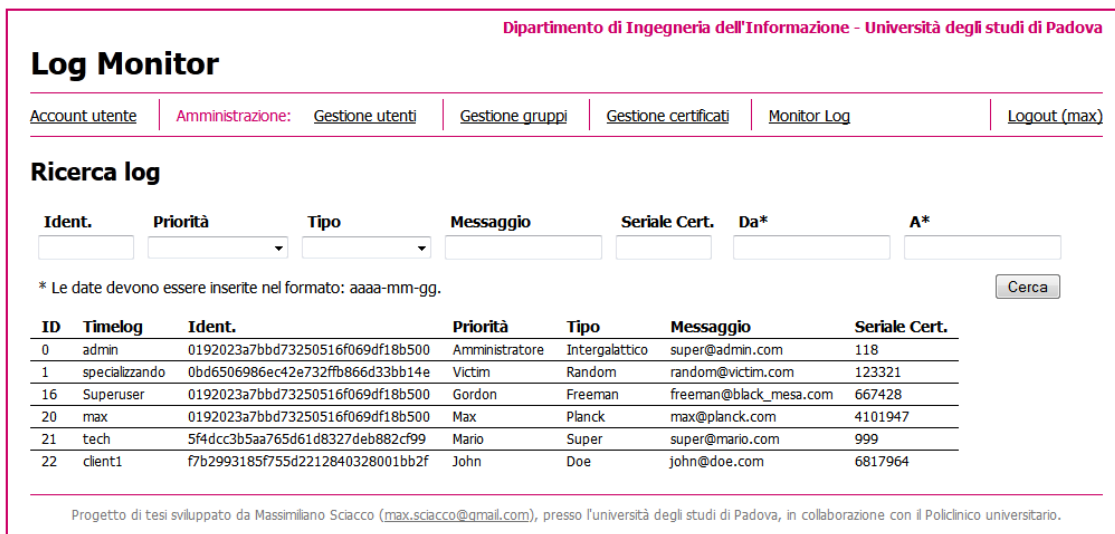


Figura 2.10. Un attacco di SQL injection eseguito con l'operation UNION.

L'esecuzione di questo attacco è stata piuttosto semplice. Il numero dei campi da bilanciare era facilmente intuibile dall'intestazione della tabella, la quale presenta sette campi; mentre per quanto riguarda i nomi delle colonne della tabella `users`, è stato sufficiente analizzare i campi presenti nella pagina "Account Utente" (visualizzabile in alto a sinistra in Figura 2.10) e tentare varie combinazioni di nomi comunemente utilizzati per ogni singolo parametro che ivi compariva.

Contromisure a SQL Injection

Fortunatamente, proteggersi dagli attacchi di SQL injection è facile. È sufficiente eseguire un adeguato controllo di convalida degli input, impedendo l'inserimento di caratteri "pericolosi" e facendo l'escape di altri necessari, come l'apice singola. PHP fornisce una funzione che aggiunge automaticamente il carattere di escape (il backslash, `\`) davanti ai caratteri critici. Dalla versione 5, PHP ha esteso la precedente libreria per la gestione del database con una nuova orientata agli oggetti, chiamata `mysqli`. In aggiunta ad altri vantaggi, `mysqli` permette di creare delle query parametriche (o *prepared statements*), che oltre ad essere nettamente più performanti, aggiungono dei controlli automatici ai valori inseriti e altri meccanismi di sicurezza.

2.3.3.2 XPATH Injection

Molte applicazioni web salvano le informazioni in file XML. XPATH è un linguaggio che permette di analizzare ed estrarre i dati da un documento XML. Il meccanismo è simile a quello SQL: iniettando una porzione di codice nella query XPATH, è possibile alterare la logica della query. Prendiamo per esempio il seguente documento XML:

```
<?xml version="1.0" encoding="utf-8" ?>
<users>
  <admins>
    <user>admin</user>
    <pass>admin123</pass>
  </admins>
  <basic>
    <user>guest</user>
    <pass>guest123</pass>
  </basic>
</users>
```

Supponiamo di avere un oggetto che legga gli XML ed eseguiamo il seguente codice:

```
xmlDoc.selectNodes("/users/admins/pass/text()")
```

La porzione di query è `/users/admins/pass` che restituisce `admin123`.

Un esempio di XPATH injection per bypassare l'autenticazione può basarsi su un codice simile al seguente:

```
String(//users/admins/[user/text()=' ' + txtUser.Text + " \' and pass/text() = \' " + txtPass.Text + " \'])
```

I campi `txtUser.Text` e `txtPass.Text` rappresentano genericamente gli input per il nome utente e la password. In questo esempio non è assunto un linguaggio di programmazione particolare. Se nel campo `txtUser.Text` viene inserito un input del tipo "admin' or 1=1 or 'a'='b", la query diventa:

```
String(//users/admins/[user/text()='admin' or 1=1 or 'a'='b' and pass/text() = ''])
```

In maniera analoga a quanto descritto per le clausole di SQL, l'operatore logico AND ha priorità maggiore rispetto ai precedenti OR, quindi indipendentemente dal risultato delle ultime due clau-

sole, se viene iniettato un operatore che causa l'intero primo blocco ad essere vero, allora l'intera query diventa vera.

Come per SQL injection, anche l'XPath injection è facilmente prevenibile con un'adeguata convalida dell'input.

2.4 Autenticazione con chiavi e certificati

Le password sono qualcosa che l'utente "sa" e pertanto, come tali, sono una risorsa replicabile. Domandarsi se un sistema di autenticazione basato su password è adeguatamente sicuro, è l'equivalente di porsi le seguenti questioni psicologiche:

1. L'utente violerà la sicurezza di sistema rivelando la password a terze parti accidentalmente, volontariamente o in seguito a un inganno?
2. L'utente inserirà la password correttamente con una probabilità sufficientemente alta?
3. L'utente sarà in grado di ricordare la password oppure sarà costretto ad annotarla da qualche parte o adottarne una più semplice e quindi facile da indovinare?

Uno dei rischi più elevati per la confidenzialità delle informazioni è che l'attaccante può ricavarle direttamente dalle persone tramite una qualche elaborata menzogna. Questo metodo di crittoanalisi è meglio conosciuto come ingegneria sociale [28]. In uno studio sperimentale, 336 studenti di informatica dell'Università di Sydney, hanno ricevuto una e-mail che chiedeva di fornire la loro password col pretesto di convalidare il database a seguito di una violazione dello stesso. Ben 138 di essi (più del 40%) ha inviato la password. Alcuni erano sospettosi: 30 hanno inviato una password simile ma non corretta, oltre 200 hanno modificato la loro password senza un sollecito ufficiale. Solo pochi di essi hanno inoltrato la mail alle autorità di sicurezza [31].

Un altro problema è legato alla lunghezza o complessità della password. Una lunga password casuale può confondere la persona che deve inserirla, e se l'operazione che sta cercando di eseguire è urgente, questo può avere ripercussioni sulla sicurezza (soprattutto in ambito sanitario).

Pertanto, è nata l'esigenza di affidarsi a sistemi di autenticazioni più robusti e semplici da utilizzare: i certificati digitali. L'autenticazione con certificati utilizza la crittografia asimmetrica (detta anche crittografia a chiave pubblica) e un certificato digitale per autenticare l'utente. L'autenticazione con certificati spesso è accompagnata da meccanismi di autenticazione basati su password per garantire un maggiore grado di sicurezza. Ciò viene definita autenticazione a due fattori perché oltre a richiedere qualcosa che solo l'utente dovrebbe conoscere (password/pin), viene richiesto anche qualcosa che solo egli possiede (il certificato). Il certificato può essere salvato in un dispositivo hardware resistente alla manomissione (come smart card o security token) o direttamente sulla macchina.

L'autenticazione con certificati utilizza il protocollo HTTP su SSL, in cui il client e il server si autenticano presentando i rispettivi certificati. Il certificato fornito dal client solitamente è in formato X.509, uno standard ITU-T (International Telecommunication Union - Telecommunication Standardization Bureau) per le infrastrutture a chiave pubblica (PKI, Public Key Infrastructure) [32].

La Figura 2.11, illustra il funzionamento di un meccanismo di mutua autenticazione.

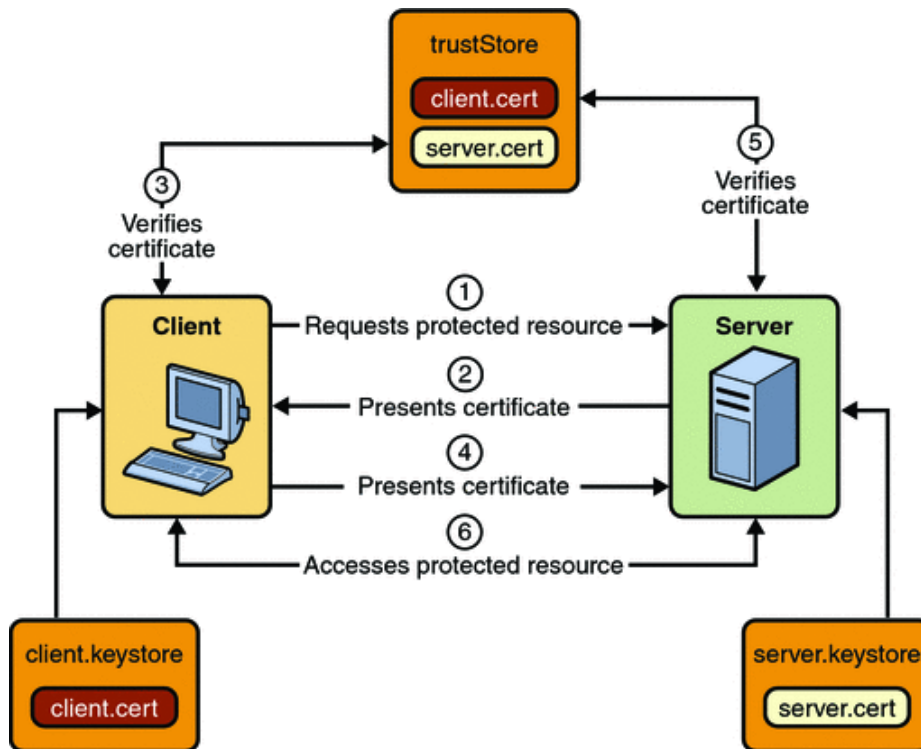


Figura 2.11. Funzionamento della mutua autenticazione con certificati.

1. Il client richiede una risorsa protetta.
2. Il server presenta il suo certificato al client.
3. Il client verifica la validità del certificato del server.
4. Se la convalida ha successo, il client invia il proprio certificato al server.
5. Il server verifica la validità delle credenziali del client.
6. Se la verifica va a buon fine, il server garantisce l'accesso alla risorsa protetta richiesta.

La maggior sicurezza dovuta ai certificati ha tuttavia un costo: la difficoltà di ottenere, distribuire e gestire i certificati, rende questo metodo eccessivamente costoso per siti con un grande bacino di utenza. Tuttavia, siti che ospitano dati sensibili o hanno una base di utenti limitata, come le applicazioni business-to-business (B2B), possono beneficiare enormemente dall'uso dei certificati.

Al momento non si conosce nessun tipo di attacco contro l'autenticazione basata su certificati, a patto che la chiave privata resti al sicuro. Tuttavia, va specificato che i sistemi che non riescono a controllare la validità dei certificati, in base alla lista di revoca dei certificati (CRL, Certificate Revocation List), potrebbero impropriamente permettere l'utilizzo di certificati revocati perché rubati, persi o scaduti.

Certificati di ruolo

In molti casi, l'esigenza di gestire politiche complicate, richiede l'utilizzo di meccanismi di controllo degli accessi basati su ruoli. In alcuni casi è possibile che allo stesso utente vengano assegnati più ruoli ed è quindi importante distinguere l'incarico con cui ha svolto determinate azioni sul sistema informativo. I certificati di ruolo permettono di soddisfare questa necessità includendo direttamente al loro interno il ruolo associato all'utente e gestendo quindi le autorizzazioni in base ad esso.

2.5 Specifiche del profilo ATNA

ATNA contribuisce al controllo degli accessi ai nodi limitandolo solo agli utenti autorizzati. Le comunicazioni di rete tra nodi sicuri in un dominio di sicurezza sono riservate solamente agli altri nodi sicuri presenti in quel dominio. I nodi sicuri limitano l'accesso agli utenti autorizzati secondo le specifiche politiche di autenticazione locale e controllo degli accessi.

Autenticazione utente

Il profilo di integrazione ATNA richiede solamente l'autenticazione utente locale. Il profilo permette a ogni nodo sicuro di adottare qualsiasi tipo di tecnologia di controllo degli accessi per autenticare gli utenti.

Autenticazione della connessione

È richiesto l'uso di un'autenticazione bidirezionale basata su certificati per le connessioni da e verso ogni nodo. I protocolli DICOM, HL7 e HTML definiscono già dei meccanismi di autenticazione basata su certificati. Questi, tuttavia, identificano i nodi piuttosto che l'utente. Connessioni a queste macchine che non siano autenticate in modo bidirezionale, dovrebbero essere proibite o progettate per prevenire l'accesso ai dati PHI.

Questo requisito può essere soddisfatto anche utilizzando configurazioni di rete fisica che assicurino la completa sicurezza degli accessi. Questo significa che nessuna macchina *untrusted*, può ottenere l'accesso fisico ad alcuna porzione della rete. Rendere l'autenticazione della connessione configurabile migliora le prestazioni in reti fisicamente sicure.

IHE non obbliga a utilizzare alcun tipo di cifratura durante le trasmissioni. Tuttavia, consiglia l'utilizzo di meccanismi di negoziazione basati su canale sicuro (SSL, TLS) per tutte le comunicazioni che coinvolgano nodi sicuri.

Capitolo 3

Autorizzazione

I sistemi di autorizzazione possono essere realizzati con numerose tecniche implementate sui diversi livelli dell'architettura. Vista la varietà che caratterizza quest'ambito, più che soffermarsi sui vari meccanismi di autorizzazione, questo capitolo illustra le principali tecniche di attacco e le contromisure adottabili. Inoltre sono descritti i modelli di controllo degli accessi definiti da IHE.

3.1 Definizione

L'autorizzazione determina quali parti dell'applicazioni un client autenticato può accedere e che tipo di azioni può intraprendere all'interno della stessa.

Tipicamente l'autorizzazione viene implementata fornendo alla sessione dell'utente un token di accesso che lo identifica univocamente. L'applicazione, quindi, determina se concedere o negare l'accesso a certe sezioni, confrontando tale token con una lista di controllo degli accessi (ACL, *access control list*). Se vi è un riscontro tra l'identificatore e la configurazione dei permessi, l'accesso viene garantito; altrimenti viene negato. Il token quindi, funge da meccanismo di ri-autenticazione persistente, ovviando alla necessità di reinserire continuamente e manualmente le credenziali di accesso. Al logout o timeout della sessione, il token viene cancellato o invalidato.

Spesso l'identificatore usato per distinguere le sessioni, è comunemente chiamato *session ID*, e coincide con l'access token di cui sopra. Solitamente, è salvato all'interno di un cookie di sessione.

Come abbiamo già visto nel capitolo 2, indovinando, rubando o semplicemente replicando il token altrui, un attaccante può impersonare un altro utente ottenendo gli stessi suoi privilegi. Molti attacchi si concentrano su manomettere i token di sessione usati dall'applicazione per determinare i permessi, o bypassare le ACL.

3.2 Analizzare i token di sessione

Di solito i token di sessione non sono immediatamente decifrabili, molto spesso però, alcuni componenti dell'ID di sessione tendono ad essere piuttosto scontati. Per esempio, un timestamp può essere facilmente identificato da valori nel token che incrementano continuamente. La seguente tabella illustra alcuni dei più comuni elementi che compongono un token di sessione.

Componente	Caratteristiche identificative
Timestamp	Cambia costantemente.
Contatore	Cambia monotonicamente.
Profilo utente	Forma codificata di valori noti come nome, cognome, mail, ...
Indirizzo IP del server	Quattro byte, per es. 192.168.0.1

Indirizzo IP del client	Come sopra
Valore salt	Può cambiare a ogni richiesta, sessione o restare statico.

Se il token appare come un ammasso indecifrabile di caratteri ASCII, le possibili spiegazioni sono due: sono presenti meccanismi di codifica o cifratura. Nel primo caso è sufficiente scoprire il tipo di codifica utilizzato e decodificarne il contenuto. Uno degli algoritmi di codifica più usati è il Base64, facilmente decodificabile con strumenti gratuiti reperibili sul web. Nel secondo caso, invece, le cose si complicano poiché l'asimmetria di questi algoritmi impedisce di risalire al vero contenuto della stringa. Tuttavia, anche se cifrati, i token sono comunque soggetti ad attacchi di replay e fixation.

Analizzare i limiti numerici all'interno degli ID di sessione, permette di identificare il numero di bit utilizzati dai meccanismi di generazione dei token. Per esempio se i valori presentano un range compreso tra -32768 e 32767, significa che il token utilizza un intero con segno a 16 bit. Questa informazione può essere cruciale per eseguire attacchi a forza bruta.

Un'altra tecnica utile a individuare pattern comuni nell'ID è detta *analisi differenziale*. Nonostante il termine faccia pensare a una tecnica particolarmente sofisticata, in realtà è molto semplice: si naviga il sito con due account diversi e si annotano le differenze di comportamento dell'applicazione. Il seguente esempio illustra un'applicazione dell'analisi differenziale.

Esempio di analisi differenziale

Supponiamo di essere ingaggiati da una società per valutare la sicurezza del loro sistema informativo. Ci vengono dati due utenti, uno con privilegi di amministratore e uno "standard". Si naviga il sito con entrambi gli account, registrando tutte le pagine e i dati inviati. Dai dati ricavati si vede che per entrambi gli utenti, i cookie inviati dall'applicazioni sono nella stessa quantità. Rimuovendoli uno per volta si risale al token responsabile dell'autorizzazione:

```
STANDARD ACCNT: jonafid=833219244.213a72e5767c1c7a6860e199e2f2bfaa.0092.783823921
ADMIN ACCNT:   jonafid=833208193.dd5d520617fb26aeb18b8570324c0fcc.0092.836100218
```

Il lettore attento avrà subito notato la segmentazione del cookie in quattro parti di egual lunghezza (separati da un punto ciascuno). In particolare, sono state sottolineate le parti che coincidono nei due token. Nonostante le somiglianze, queste informazioni suggeriscono granché. Si inizia quindi a modificare i singoli valori e vedere come risponde l'applicazione a tali modifiche. Dalle varie risposte si scopre che solo i primi cinque caratteri del cookie sono legati all'autorizzazione. Osservando i due cookie, ci si accorge che dei primi cinque numeri del primo segmento di cookie, solo il quinto carattere differisce tra i due. Svolgendo delle manipolazioni più approfondite su quella porzione di cookie, si scopre che i primi cinque caratteri determinano il numero dell'account e cambiandoli è possibile quindi, accedere alle sessioni di altri utenti.

3.3 Attaccare le Access Control List

Tutte le applicazioni web, seppur a livelli diversi, si basano su liste di controllo degli accessi per la protezione dei dati. Attaccare le ACL è spesso più facile che compromettere i token di autenticazione e accesso (di cui abbiamo già ampiamente discusso nel paragrafo 2.3.1).

3.3.1 Directory traversal

Letteralmente “attraversamento della directory”, la directory traversal è un attacco comunemente utilizzato per bypassare le ACL e ottenere l’accesso non autorizzato a directory riservate. Questo attacco prende di mira le pagine che fanno uso di template o fanno riferimento ad altri file inclusi nel web server. Usando i caratteri “../” (dot-dot-slash), usati nella navigazione dei sistemi, si può infatti, risalire da una sottodirectory ad una directory superiore. Un tipico esempio di vulnerabilità in un’applicazione PHP è la seguente:

```
1. $template = content.PHP';
2. if (isset($_COOKIE['TEMPLATE']))
3.     $template = $_COOKIE['TEMPLATE'];
4. include ("/home/users/PHPguru/templates/" . $template);
```

Un attacco contro un sistema del genere può essere svolto con la seguente richiesta HTTP:

```
GET /vulnerable.PHP HTTP/1.0
Cookie: TEMPLATE=../../../../../../../../../../../../../../../../etc/passwd
```

I ripetuti “../” inseriti alla fine della stringa di inclusione hanno fatto attraversare le directory fino al file Unix che contiene le password.

Alcuni sistemi effettuano dei controlli di validazione sul file incluso o aggiungono pezzi di stringhe all’input finale (tipicamente estensioni) per evitare l’inclusione di file sospetti. Per esempio, l’applicazione potrebbe posporre alla stringa il suffisso di un file immagine (.jpg, .gif, ecc.) per cui l’inclusione risulta in qualcosa del tipo:

```
include ("/home/users/PHPguru/templates/" . $template . ".jpg");
```

Questo tipo di protezione è facilmente aggirabile aggiungendo il carattere %00 alla fine del file che vogliamo includere. %00 è la rappresentazione in codifica URL del carattere nullo, il quale rappresenta il carattere di fine stringa in linguaggi come il C. Linguaggi come Perl o PHP non interpretano %00 come delimitatore di stringa, ma il sistema operativo sì, essendo scritto in C/C++. L’inclusione fa accesso al file system, quindi l’applicazione web interagisce con una qualche funzione del sistema operativo. Pertanto la stringa apparentemente innocua:

```
../../../../../../../../../../../../../../../../etc/passwd%00.jpg
```

In realtà dà accesso al file delle password.

Fortunatamente, i server più utilizzati, come Apache, forniscono dei sistemi di protezione a questo tipo di attacco. Altre contromisure adottabili consistono nella validazione massiccia degli input che includono file. In particolare è fondamentale filtrare i caratteri punto e slash rappresentati sia in Unicode che in esadecimale, adottare degli appropriati sistemi di controllo degli accessi alle directory sensibili e se possibile, evitare il più possibile inclusioni di file dinamiche.

3.3.2 Horizontal Privilege Escalation

L’horizontal privilege escalation è lo sfruttamento di una vulnerabilità di autorizzazione per guadagnare i privilegi di un utente con uguali o minori privilegi. Consideriamo per esempio un’applicazione commerciale.

In seguito alla registrazione del nostro nuovo account, il server restituisce la seguente risposta:

```
HTTP/1.x 302 Object moved
Set-Cookie: BIGipServerSecure2.TEAM.WebHosting=1852316332.20480.0000; path=/
```

```
Set-Cookie: UserID=2366239; path=/
Set-Cookie: ShopperID=193096346; path=/
Set-Cookie: 20214200UserName=foo@foo.com; path=/
Date: Wen, 12 Oct 2010 18:13:23 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Location: https://secure2.site.com/secure/MyAcctBillingSuccess.asp?r=1
Content-Length: 185
Content-Type: text/html
Cache-Control: private
```

Tra i cookies, spiccano i due ShopperID e UserID. Questi cookie potrebbero servire per eseguire dei controlli di autorizzazione, pertanto si rimuovono individualmente uno alla volta e si vede come risponde il server. Immaginiamo che la rimozione del cookie ShopperID causi un HTTP 302, ossia un messaggio di redirect che indica che quel token era necessario all'autorizzazione. A questo punto è sufficiente modificare i valori e replicare il cookie al server, osservando le risposte del sito. Per esempio si guarda come cambia la pagina con i dettagli del nostro account. Data la natura del valore, possiamo immaginare che si tratti di un contatore che indica il numero di account, ed essendo il nostro account l'ultimo creato, proviamo a decrementarlo:

```
Set-Cookie: ShopperID=193096345;
```

Inviando la richiesta al server, e il risultato è che stiamo impersonando un altro utente, con tutti i suoi privilegi e le sue informazioni personali. Provando diversi valori, un attaccante può risalire a ogni account, raccoglierne i dati personali e impersonare il suo profilo liberamente.

3.3.3 Vertical privilege escalation

Il vertical privilege escalation è la capacità di guadagnare accesso a un account di livello superiore o con maggiori permessi. Esistono quattro scenari perché questo accada:

- **Ruoli modificabili** – l'applicazione permette impropriamente di modificare il ruolo dell'utente.
- **Manomissione dell'account** – avviene quando un utente non autorizzato riesce a manomettere la sessione di un altro utente con privilegi maggiori.
- **Sfruttare altre falle di sicurezza** – guadagnare l'accesso tramite falle di sicurezza a sezione amministrative del sito che permettono la modifica dei permessi.
- **Funzioni di amministrazione insicure** – funzioni di amministrazione che non hanno degli adeguati controlli di autorizzazione.

Ruoli modificabili

Come già visto più volte, molte applicazioni web salvano le informazioni di autorizzazione, come livello di permessi o ruoli, in locazioni facilmente modificabili dall'utente. Prendiamo per esempio il seguente cookie:

```
Auth=897ec5aefa56fd4fe5a4f65es3af1e9af4e56s4fa6el fa84f9sa4f56e4saf8e9asf89;
x=d1sa56d1a891ds5a6d15sa6;
Y=sad49a489ds1ad591sa89d4;
role=ee11cbb19052e40b07aac0ca060c23ee
```

Il campo `role` è subito evidente, ma apparentemente non suggerisce molto. La cosa migliore da fare in caso di valori poco intellegibili è svolgere un'analisi differenziale (come spiegato nel paragrafo 3.2). Provando un secondo account otteniamo il seguente risultato:

```
Auth=897ec5aefa56fd4fe5a4f65es3af1e9af4e56s4fa6e1fa84f9sa4f56e4saf8e9asf89;
t=0;
v=5sa61d6a1d65a1d56sa1d56;
Y=sad49a489ds1ad591sa89d4;
role=ee11cbb19052e40b07aac0ca060c23ee
```

Il campo `role` non è cambiato. Questo ci fa subito capire che il valore utilizzato per identificare il ruolo è una stringa statica, probabilmente una parola che identifica un ruolo in particolare o qualcosa di simile. Inoltre, si contano 32 caratteri, il che fa pensare ad una possibile codifica MD5. A questo punto non ci resta che prendere un dizionario dei più comuni nomi utilizzati per indicare il ruolo di amministratore, convertirli in MD5 con uno dei tanti software presenti sul web e aspettare una risposta che non sia un HTTP 302. Dopo un po' di tentativi, si scopre che la parola giusta per accedere con privilegi superiori era `09348c20a019be0318387c08df7a783d`, ovvero `supervisor`. Giusto per completezza, si può facilmente intuire a cosa corrispondeva la stringa originale `ee11cbb19052e40b07aac0ca060c23ee`: alla parola `user`.

Manomissione dell'account

Questa tecnica si basa sull'horizontal privilege escalation. Se la numerazione degli account è eseguita sequenzialmente, è facile risalire a un account con poteri amministrativi, poiché solitamente è il primo a essere creato.

Sfruttare altre falle di sicurezza

Accedere a un sistema tramite altre falle di sicurezza come SQL injection, spesso è più che sufficiente a modificare quello che serve per salire di ruolo. Il capitolo 2 definisce molte di queste vulnerabilità.

Funzioni di amministrazione insicure

Molto spesso le funzioni amministrative di un'applicazione web non sono autenticate o autorizzate correttamente. Si consideri per esempio un'applicazione con cui si richiama uno script tramite POST. L'applicazione suppone irragionevolmente che lo script sia accessibile solamente dalla porzione amministrativa del sito, la quale richiede una qualche forma di autenticazione. Naturalmente però, un potenziale intruso potrebbe con un pizzico di fortuna e tanta dedizione, risalire alla directory che contiene quello script ed eseguirlo come utente normale.

3.4 Contromisure

In questa sezione si descrivono alcune pratiche utili a realizzare un sistema di autorizzazione sicuro ed efficiente.

3.4.1 Autorizzazioni Apache

Il web server Apache usa due direttive per il controllo degli accessi utente a specifici URL. La direttiva `Directory` viene usata per eseguire un controllo degli accessi basata su file path. Per esempio, il seguente insieme di direttive limita l'accesso alla cartella `/admin` solamente agli utenti che appartengono al gruppo `admin`.

```
<Directory /var/www/htdocs/admin>
  AuthType Digest
```

```
AuthName "Admin Interface"  
AuthUserFile /etc/apache/passwd/users  
AuthGroupFile /etc/apache/passwd/groups  
Require group admin  
</Directory>
```

Si possono anche limitare gli accessi a certi comandi HTTP, tra i quali GET, POST, PUT, DELETE, CONNECT, ecc.

La direttiva `Location` invece, viene utilizzata quando il controllo degli accessi si basa sull'URL in maniera del tutto simile alla direttiva `Directory`.

3.4.2 Sicurezza dei token di sessione

Riassumiamo alcune delle pratiche di sicurezza già discusse in precedente:

- Usare SSL per prevenire attacchi di tipo eavesdropping.
- Impostare i cookie con il parametro `secure`, così da cifrarlo e renderlo meno vulnerabile.
- Utilizzare sistemi di autenticazione affidabili e testati, come ASP.NET e PHP.
- Non includere dati sensibili nei token.
- Rigenerare l'ID di sessione a ogni accesso per mitigare il rischio di session fixation.
- Forzare il timeout della sessione per ridurre la finestra di tempo per gli attacchi replay.
- Eseguire una stringente validazione degli input, visto che i cookie, i dati inviati in POST o GET e i valori degli header HTTP, sono sotto il completo controllo dell'utente e quindi facilmente modificabili.

3.4.3 Log di sicurezza

Un'altra contromisura di sicurezza spesso sottovalutata è il log. La piattaforma sottostante l'applicazione web dovrebbe già generare dei log per il sistema operativo e il web server. Tuttavia questi log sono spesso inadeguati a identificare attività potenzialmente dannose o eventi sospetti. I seguenti eventi che coinvolgono gli utenti dovrebbero essere tracciati:

- Modifiche al profilo.
- Cambi di password.
- Modifiche ad altri utenti da parte di amministratori o superutenti.
- Aggiunta/rimozione di un utente.

L'applicazione dovrebbe salvare nel log quanti più dettagli possibili. Al minimo devono essere registrati l'IP dell'utente che ha generato l'evento, il nome utente, altri caratteri identificativi che aiutino il riconoscimento e, naturalmente, il timestamp dell'evento.

Inserire nel log dati sensibili, come numeri di carta di credito, informazioni riservate e altri dati sensibili, non è una buona idea poiché la violazione di tali informazioni comprometterebbe l'intera applicazione e i suoi utenti. Inoltre, in alcuni casi, salvare informazioni personali, come indirizzi di residenza, dati finanziari, informazioni sanitarie, e simili, può violare norme e leggi locali o nazionali.

3.5 Specifiche del profilo IHE

Il controllo degli accessi in ambito sanitario ha numerosi requisiti che dipendono dall'infrastruttura, le persone e le risorse coinvolte nell'elaborazione dei dati, e dagli obiettivi di sicurezza stabiliti dall'azienda [33]. La complessità insita nel flusso operativo medico, rende impossibile analizzare l'intero insieme di casi e scenari plausibili. La seguente lista perciò, è rappresentativa di un piccolo sottoinsieme dei vari scenari che coinvolgono il sistema di controllo degli accessi. Si assume che gran parte dei casi reali siano riconducibili ai seguenti casi chiave:

- Sicurezza delle risorse interne: all'interno di un ospedale, l'accesso alla cartella clinica di un paziente dovrebbe essere riservata solamente al personale direttamente coinvolto nel trattamento del paziente, ed eventualmente al personale amministrativo. L'accesso a certi dati sensibili, è ulteriormente limitato a certi ruoli funzionali, in modo da assicurarne la divulgazione solo al personale che ne avesse veramente bisogno.
- Consenso alla privacy del paziente: il paziente deve essere in grado di decidere quali organizzazioni appartenenti alla rete ospedaliera, possono accedere ai propri dati clinici.
- Uso secondario: un paziente può garantire l'accesso a certi dati a scopo di studio, a patto che tutte le informazioni vengano prima anonimizzate.
- Casi di emergenza: in caso di emergenza, le restrizioni previste dalle politiche dei pazienti e le norme di sicurezza interne, devono poter essere temporaneamente annullate da una politica di emergenza, la quale permette l'accesso a tutte le informazioni mediche del paziente. Questa politica di emergenza ha l'obbligo di registrare un record di audit nel database.

3.5.1 Principi di progettazione

La seguente lista riassume alcuni dei più importanti principi di progettazione per le soluzioni di controllo degli accessi.

- Economia dei meccanismi: i sistemi adottati devono essere quanto più semplici possibili. Inoltre devono cercare di mirare a risolvere questioni specifiche piuttosto che generalizzare scenari più ampi.
- Mediazione completa: ogni tentativo di accesso deve essere esplicitamente controllato dai meccanismi di controllo degli accessi. Questo significa che non deve essere possibile bypassare tali controlli, nemmeno da parte di personale facente parte ruoli speciali o amministrativi.
- Design trasparente: tutti gli algoritmi e i meccanismi di sicurezza devono essere disponibili e verificabili.
- Meccanismo "least-common": i meccanismi capaci di garantire diritti di accesso, non devono essere condivisi tra diversi utenti o software. Ognuno di essi deve utilizzare meccanismi diversi o istanze dello stesso meccanismo diverse.
- Politica di default "fail-safe": tutto ciò che non è esplicitamente coperto dalle politiche deve essere negato.
- Separazione dei privilegi: quando possibile, i meccanismi di sicurezza devono poter verificare condizioni multiple e indipendenti, prima di garantire l'accesso a risorse protette (per

esempio, un utente potrebbe dover appartenere a un determinato ruolo e accedere da un sistema trusted).

- Privilegi minimi: ogni identità dovrebbe avere accesso a un insieme minimo di permessi, sufficienti a garantire l'applicabilità dei propri compiti. Se viene richiesto un permesso speciale per un incarico straordinario, questo può essere attivato per la durata dell'incarico, al termine del quale va immediatamente rimosso.
- Accessibilità utente: i meccanismi di sicurezza devono essere facilmente utilizzabili dal personale.
- Riluttanza alla fiducia: ogni sistema, umano o artificiale che sia, dovrebbe essere sempre considerato inaffidabile.
- Isolamento: tutti i meccanismi di controllo e gestione degli accessi devono essere isolati dagli altri sistemi, operare indipendentemente ed essere particolarmente sicuri.

3.5.2 Modelli di controllo degli accessi comuni

I quattro paradigmi fondamentali per il controllo degli accessi, sono i seguenti:

- Discretionary Access Control (DAC): la convalida e il rilascio dei permessi sono eseguiti unicamente sulla base dell'identità del soggetto.
- Mandatory Access Control (MAC): la convalida e il rilascio dei permessi vengono eseguiti utilizzando regole o politiche di controllo. A ogni utente è assegnato un livello di autorizzazione, mentre a ogni risorsa è assegnato un grado di riservatezza. Le regole determinano come i livelli di autorizzazione si relazionino con i diversi gradi di riservatezza.
- Role-Based Access Control (RBAC): a ogni utente viene assegnato un insieme di ruoli, all'interno dei quali sono definiti i permessi. La concreta esecuzione di un diritto di accesso non è quindi direttamente legata all'utente, ma al suo ruolo corrente.
- Context-Aware Access Control (CAAC): il passaggio da DAC e MAC a RBAC è il risultato di un disaccoppiamento tra i soggetti e le risorse. CAAC fa un passo ulteriore rompendo l'assegnazione statica dei ruoli e delle politiche di ruolo. Ciò avviene fornendo ulteriori regole di controllo a tali assegnazioni.

Molte leggi governative considerano il paziente come l'unico proprietario dei propri dati medici. Questo ha portato a un'influenza del modello DAC in numerose soluzioni sanitarie. Questa influenza è particolarmente presente in situazioni in cui si utilizza un consenso firmato dal paziente per trasferire parte dei suoi diritti all'ospedale.

Il modello MAC richiede che le classi di soggetti e di oggetti siano parzialmente ordinati. Se si considera che i ruoli funzionali sono utilizzati per l'assegnazione dei permessi, quest'ordinamento parziale deve essere a granularità grossolana o costruita artificialmente.

Il modello RBAC è il più adatto ad allineare i permessi con l'organizzazione interna di una struttura ospedaliera.

Il modello CAAC è pensato per superare alcune delle rigidità del RBAC, dove il personale si trova spesso a ricoprire più ruoli, e dove i diritti di accesso mutano in base allo stato del paziente o ai metodi operativi dell'azienda (turni notturni, gestione dei disastri, ecc.).

3.5.3 Sistema di controllo degli accessi

Un sistema di controllo degli accessi (ACS, Access Control System) è un meccanismo di controllo (logico) avvolto attorno a tutti i componenti dei sottosistemi di autorizzazione che sono logicamente collegati con un attore.

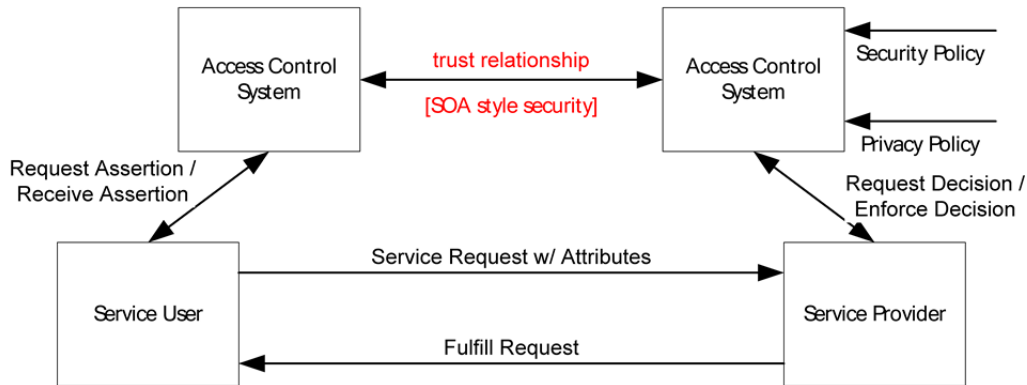


Figura 3.1. Schema di relazione tra attore e ACS.

Come illustrato in Figura 3.1, ogni attore che utilizza o fornisce delle funzionalità operative, è legato al suo sistema di controllo degli accessi. Sebbene il servizio comunichi sempre tramite messaggi di richiesta e risposta, i loro rispettivi ACS potrebbero essere debolmente accoppiati. Per esempio un servizio consumatore potrebbe richiedere la convalida dell'identità e ruolo di un utente dal suo ACS locale e inviarla al provider come parte del messaggio di richiesta. Il provider del servizio chiama l'ACS locale per decidere se accettare la richiesta. Parte di questa chiamata include la convalida ricevuta dal servizio consumatore.

3.5.4 Principio "needs-to-know"

Ogni sistema può essere descritto dai suoi attori e dalle transazioni che scambiano i dati tra di loro. I dati riguardanti il controllo degli accessi, scambiati tra gli attori di un ACS, sono rappresentati principalmente dalle politiche e dagli attributi.

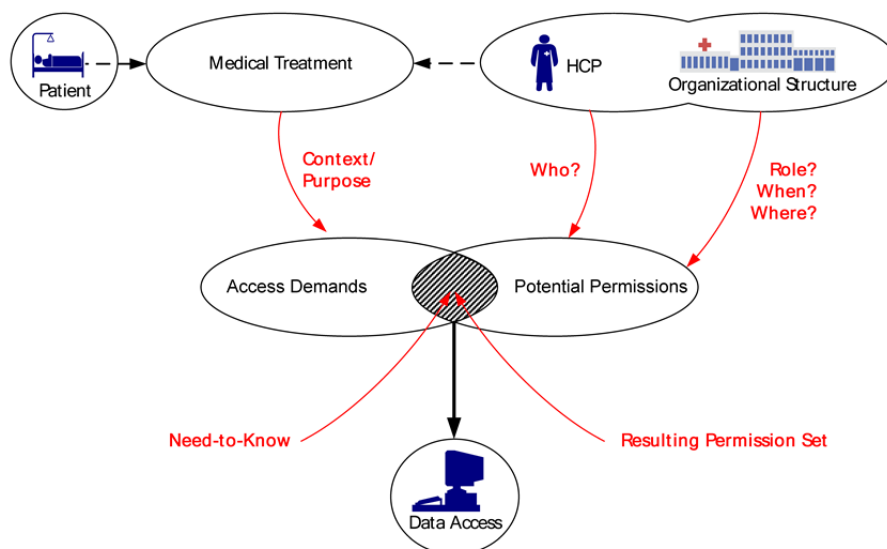


Figura 3.2. Derivazione dei permessi tramite il principio "Needs-to-Know".

Per esempio, l'accesso alle informazioni, derivante da un trattamento medico (Context/Purpose), deve essere in accordo con i permessi garantiti al soggetto (Who). Questi permessi si basano sul ruolo, l'organizzazione, la situazione in corso, ecc. (Role, When, Where). L'intersezione illustrata in Figura 3.2 riflette da una parte il risultato di un insieme di permessi del soggetto, dall'altra le informazioni che tale soggetto deve conoscere (needs-to-know) per svolgere i propri incarichi.

Il principio "Needs-to-Know" è una relazione tra il soggetto (per esempio il medico) e una risorsa protetta. Poiché ogni accesso a una risorsa protetta è mediato dal sistema, questa singola logica relazionale è divisa in due relazioni fisiche: una relazione "Need-to-Use" e una relazione di accesso mediato (vedi Figura 3.3).

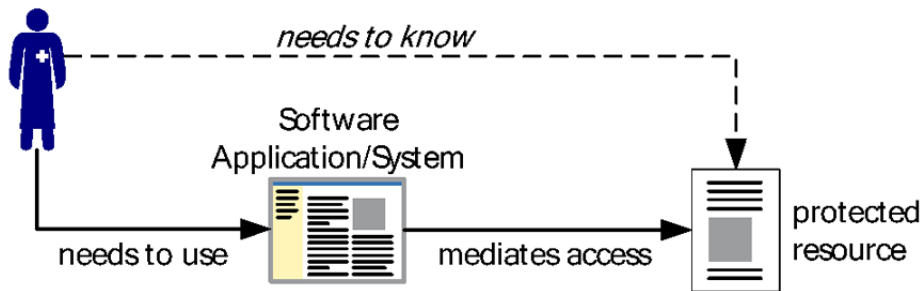


Figura 3.3. Relazioni "Needs-to-Use" e accesso mediato.

Una buona applicazione deve assicurare che tutte le sequenze di relazioni "Needs-to-Use" e accesso mediato, siano semanticamente identiche alla corrispondente relazione "Needs-to-Know" che riflette quello scopo.

3.5.5 Esempio con thin client

Lo schema in Figura 3.4 mostra le relazioni tra i vari domini degli attori in uno scenario di accesso tramite thin client.

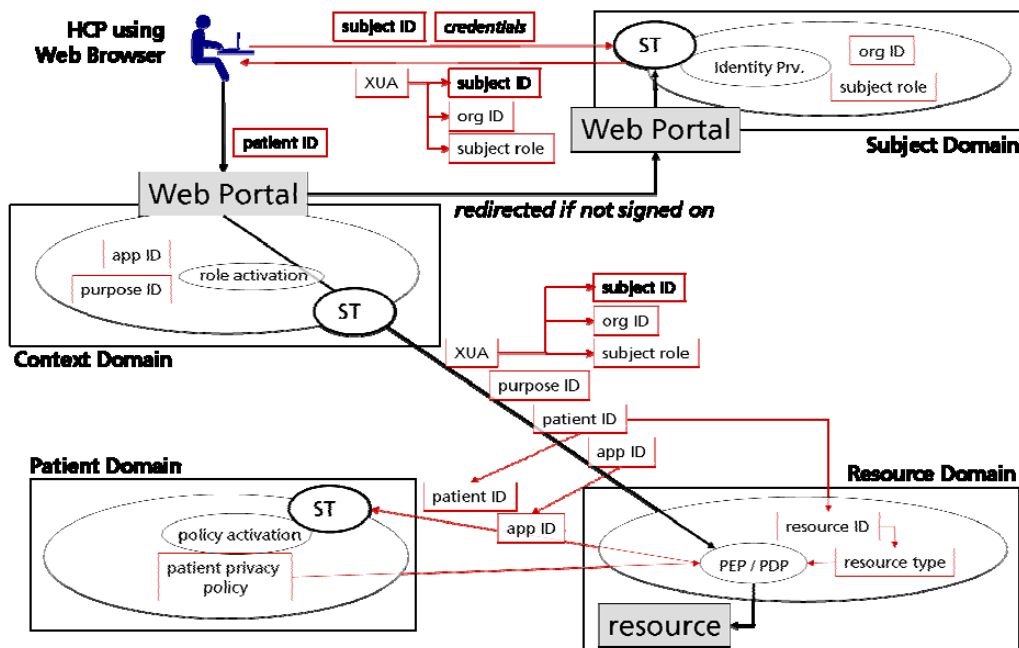


Figura 3.4. Relazioni tra domini in un sistema con thin client.

Il contesto del caso e il suo ACS risiede su un'applicazione web, accessibile da un comune browser. Un Health Care Provider (HCP), tenta di accedere all'applicazione, e viene reindirizzato su una pagina di autenticazione (essa può far parte dell'applicazione principale essere esterna). L'identità del soggetto autenticato vengono trasferite all'applicazione per mezzo di un sistema detto XUA (Cross-Enterprise User Assertion). Questa collaborazione tra i due portali è invisibile all'utente e può essere realizzata con protocolli standard HTTP. Il dominio del soggetto, che tra le altre cose comprende anche il ruolo e l'identificativo, determina un insieme di permessi che vengono confrontati dal PEP/PDP (Policy Enforcement and Policy Decision) con le politiche di accesso della risorsa richiesta e del paziente ad essa collegata (per esempio la sua cartella clinica). Se l'intersezione tra le varie politiche non è vuota, l'HCP ha accesso alla risorsa richiesta.

I vari domini sono collegati tra di loro da vincoli di sicurezza detti ST (Security Token).

Capitolo 4

Audit Trail e Log

In questo capitolo vengono investigati i meccanismi di tracciamento del flusso operativo. Sono definiti i concetti di log e audit trail. Sono descritte le tecniche con cui è possibile individuare anomalie e violazioni di sistema tramite l'utilizzo dei vari tipi di log. A proposito dell'audit trail, sono valutati i requisiti necessari al completo tracciamento dell'attività utente, il flusso dei messaggi di audit e gli standard da adottare, secondo le specifiche IHE.

4.1 Log applicativo

Nella realtà informatica è inevitabile che qualcosa prima o poi funzioni storto, un utente commetta un errore o un attaccante prenda di mira la nostra applicazione. La vita di un'applicazione si compone di tre fasi basilari: l'installazione, la configurazione e il monitoraggio. Che cosa succede se l'applicazione non funziona esattamente come previsto o smette di funzionare completamente? Il logging è uno strumento fondamentale per diagnosticare e isolare la causa di tali malfunzionamenti.

Tra le altre cose, il logging permette:

- **Amministrazione e debugging** – il logging permette all'amministratore di sistema di identificare attività insolite e assicurarsi che il sistema funzioni come previsto.
- **Stato del sistema** – fornendo al sistema un audit di base, si può stabilire il corretto funzionamento del proprio sistema, identificando che tipo di record sono "normali", che traffico di rete viene generato in situazioni standard e come il sistema reagisce a determinati eventi. Facendo un confronto tra questi comportamenti di base con l'attuale log, è possibile individuare anomalie nel flusso operativo dell'applicazione.
- **Individuazione delle intrusioni** – come nel caso precedente, è possibile identificare tentativi di intrusione confrontando il log con un audit di base. In particolare, i tentativi di intrusione sono accomunati da uno stesso obiettivo: mantenere l'anonimità. Valutando le registrazioni del log è possibile quindi individuare questi attacchi.
- **Contenimento degli incidenti** – se è stata individuata una manomissione, è richiesta un'investigazione strutturata per risalire all'estensione e gravità del danno. Va specificato che il log è solo una piccola parte dell'intera fonte di informazioni utilizzate nell'investigazione. Log di sistema, del firewall, del router, e molti altri, vengono utilizzati dal team di investigazione per risalire al quadro completo dell'incidente.

4.1.1 Web server log

I web server più diffusi come Apache e IIS generano automaticamente messaggi di logging nel formato CLF (Common Log Format). Il file di log CLF contiene una linea separata per ogni richiesta HTTP. Ogni linea è composta a sua volta da diversi segmenti separati da spazi:

- Host: il dominio o il suo indirizzo IP.
- Ident: se la direttiva `IdentityCheck` è attiva e il client usa `identd`, allora qui viene inserita l'informazione di identità riportata dal client.
- Authuser: se l'URL richiede un'autenticazione base HTTP, allora in questo campo viene salvato il nome utente del richiedente.
- Date: data e ora della richiesta.
- Status: un codice a tre cifre che identifica il tipo di risposta HTTP restituita al client.
- Bytes: il numero di byte restituiti al client, esclusi gli header HTTP.

Di seguito è mostrato un esempio di log che coinvolge il login e immediato logout da una pagina con autenticazione basata su certificato su canale sicuro SSL:

```

1. 127.0.0.1 - - [04/Feb/2012:12:15:50 +0100] "GET /atna HTTP/1.1" 301 231
2. 127.0.0.1 - - [04/Feb/2012:12:15:50 +0100] "GET /atna/ HTTP/1.1" 302 311
3. 127.0.0.1 - - [04/Feb/2012:12:15:50 +0100] "GET /atna/www/login.PHP HTTP/1.1" 200
  1410
4. 127.0.0.1 - - [04/Feb/2012:12:15:50 +0100] "GET /atna/style/style.css HTTP/1.1"
  200 2017
5. 127.0.0.1 - - [04/Feb/2012:12:15:50 +0100] "GET /favicon.ico HTTP/1.1" 404 209
6. 127.0.0.1 - - [04/Feb/2012:12:15:50 +0100] "GET /favicon.ico HTTP/1.1" 404 209
7. 127.0.0.1 - - [04/Feb/2012:12:15:50 +0100] "GET /favicon.ico HTTP/1.1" 404 209
8. 127.0.0.1 - - [04/Feb/2012:12:15:51 +0100] "POST /atna/www/login.PHP HTTP/1.1"
  302 1410
9. 127.0.0.1 - - [04/Feb/2012:12:15:52 +0100] "GET /atna/www/user account.PHP
  HTTP/1.1" 200 9362
10. 127.0.0.1 - - [04/Feb/2012:12:15:53 +0100] "GET /atna/www/user_account.PHP?logout
  HTTP/1.1" 302 648
11. 127.0.0.1 - - [04/Feb/2012:12:15:54 +0100] "GET /atna/www/login.PHP HTTP/1.1" 200
  1410

```

Una rapida analisi delle risposte HTTP ci fa capire il flusso esecutivo svolto dal client:

- Alla prima riga, la risposta 301 alla richiesta HTTP illustra un redirect forzato. Ciò non ci sorprende in quanto nelle configurazioni di Apache è stato impostato un redirect automatico da HTTP ad HTTPS.
- La risposta 302 della seconda riga è un redirect svolto dall'applicazione per rimandare gli utenti non autenticati alla pagina di login.
- La risposta 200 alla terza riga è, appunto, il risultato del redirect precedente.
- L'ottava riga presenta una richiesta POST con risposta 302. Ancora una volta questo risultato non ci sorprende: l'utente ha inserito il codice segreto associato al suo certificato e, dopo aver convalidato le sue credenziali, è stato reindirizzato alla pagina con il suo account.
- La nona riga è esattamente il risultato della richiesta precedente.
- Infine la decima e undicesima richiesta sono il risultato di un redirect in seguito al logout.

Vediamo un altro esempio di log (i puntini di sospensione indicano il ripetersi delle stesse righe):

```

1. ...
2. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293
3. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293
4. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293

```

```

5. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293
6. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293
7. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293
8. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 302 293
9. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "GET /mysite/main.PHP HTTP/1.1" 200 27
10. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293
11. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293
12. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293
13. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293
14. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293
15. 127.0.0.1 - - [04/Feb/2012:16:25:00 +0100] "POST /mysite/ HTTP/1.1" 200 293
16. ...

```

Sebbene a prima vista questo log non sembri molto utile, in realtà nasconde delle informazioni molto interessanti. Prima di tutto le richieste sono state eseguite tutte nello stesso istante (16:25:00), il che fa subito pensare a una procedura di attacco automatizzata. L'altra informazione importante è quella alle righe 8 e 9, dove in seguito ad un POST, è stato fatto un redirect alla pagina `main.php`. Questo dovrebbe metterci subito in allarme, perché significa che l'attacco è riuscito ad ottenere delle credenziali valide.

4.1.2 Log applicazione

Come abbiamo visto, il log del web server registra tutte le richieste HTTP e le relative risposte. Quello che però non salva sono gli header HTTP, i quali spesso contengono informazioni importanti, come i parametri passati in POST. Ecco perché è richiesto un ulteriore livello di log in grado di registrare queste nuove informazioni. In particolare, l'applicazione, avendo accesso all'intero flusso esecutivo, dovrebbe tenere traccia di ogni azione eseguita dall'utente (dall'autenticazione, fino alla sua disconnessione). Un log completo ed esaustivo a livello applicativo permette di individuare utilizzi scorretti o fraudolenti del sistema e ricostruire le azioni svolte dall'utente.

Una delle questioni più difficili da affrontare è stabilire la quantità di dati che un log deve contenere. Un'eccessiva quantità di informazione può risultare difficile da gestire, sia in fase di salvataggio, che in fase di ricerca, per questo i dati raccolti andrebbero salvati in strutture dati più sofisticate di un semplice file di testo (per esempio database, XML, ecc.). Inoltre, se il traffico è molto elevato, vanno presi in considerazione meccanismi di archiviazione periodica per evitare il sovraccarico dei supporti di memoria fisica.

4.1.3 Strategie per l'individuazione degli attacchi

Gli attacchi possono essere rilevati in diverse zone e dispositivi dell'infrastruttura di rete, ognuno dei quali vede diversi aspetti del traffico (vedi Figura 4.1). I principali punti di individuazione sono il firewall, il web server e la web application. I firewall tradizionali lavorano sui livelli di rete e trasporto del modello standard ISO/OSI [34], quindi non raccolgono informazioni sul livello applicazione e pertanto non sono di grande utilità nell'individuare comportamenti anomali degli strati più alti. Sui log del web server e dell'applicazione si è già discusso nei paragrafi precedenti. Di seguito vengono discusse due principali strategie per l'individuazione degli attacchi: le regole statiche (o *rule-based strategy*) e quelle dinamiche (o *anomaly-based strategy*).

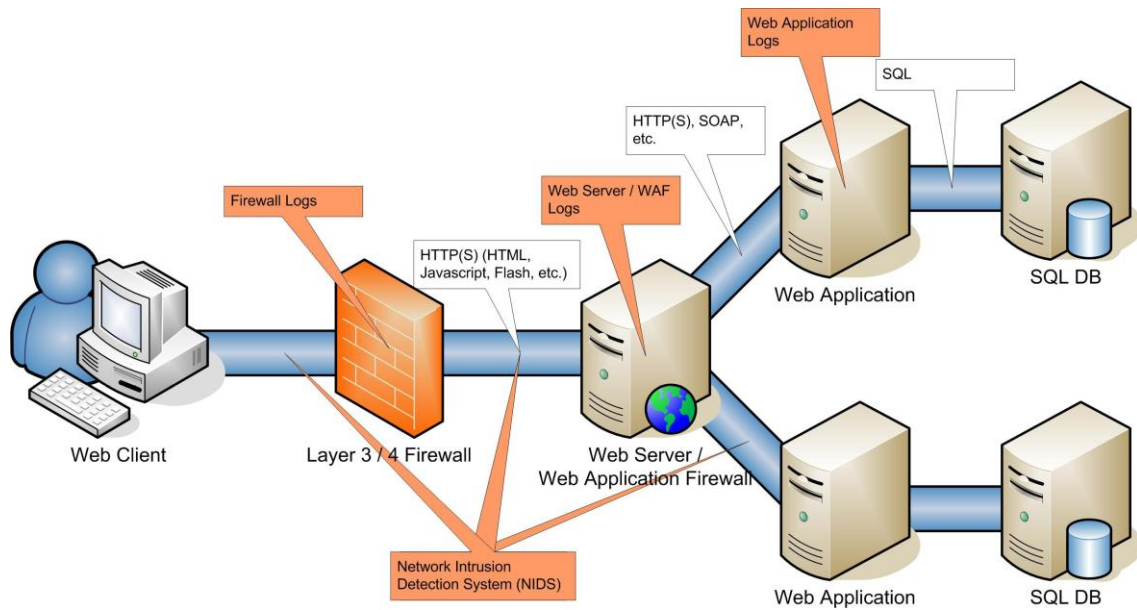


Figura 4.1. Individuare gli attacchi nella rete.

Regole statiche

Questo tipo di strategia definisce un insieme di regole statiche che vanno formalizzate prima di iniziare l'analisi. Queste possono essere semplici regole come l'individuazione di caratteri speciali, o ben più complesse come quelle in grado di distinguere attacchi di session fixation. Le regole statiche sono definite in modo definitivo e permangono per la durata della fase di individuazione. Devono essere stabilite con chiarezza e specificate in dettaglio in ogni diversa applicazione. Le regole statiche hanno senso quando si conoscono a priori i valori, la lunghezza o altre caratteristiche di certi input.

Le regole statiche possono a loro volta essere suddivise in due sottocategorie: il modello di sicurezza negativo e quello positivo. Il primo, detto anche approccio blacklist, "lista nera", assume, come politica di default, l'esecuzione di qualsiasi azione. Ciò significa che ogni cosa è considerata "normale" e quindi accettabile. La blacklist definisce le regole per ciò che non è consentito o, in termini di Intrusion Detection System (IDS), ciò che viene considerato un attacco. Questo modello è più facile da implementare, ma non è un buon approccio alla sicurezza. Il grosso svantaggio di questo modello, infatti, è che l'individuazione è limitata dall'efficienza della blacklist. Ogni nuovo tipo di attacco deve essere riconosciuto e aggiunto alle politiche. In compenso, l'aspetto positivo del modello è che produce pochi falsi positivi, in quanto le regole controllano comportamenti o attacchi ben noti.

Il modello di sicurezza positivo è il contrario del precedente. La politica di default è di negare tutto, le regole quindi stabiliranno cosa invece è consentito. Queste regole, o politiche, sono dette anche whitelist, "lista bianca", in contrapposizione alla blacklist, e stabiliscono cosa è considerato "normale" e quindi accettabile. Questa whitelist può essere definita automaticamente o manualmente in fase di apprendimento. È importante che la fase di apprendimento, in questo caso, consista solamente di traffico valido e comportamenti legittimi. Questo modello è preferibile dal punto di vista della sicurezza. I falsi negativi vengono ridotti al minimo e i falsi positivi possono aiutare a migliorare le politiche della whitelist. I firewall, per esempio, sono impostati di default a questo modello, negando ogni tipo di accesso se non diversamente impostato.

Regole dinamiche

Come implica il nome stesso, queste regole non vengono definite manualmente, ma durante la fase di apprendimento. In questa fase il traffico accettabile viene salvato come traffico “buono”, ed è pertanto di fondamentale importanza che esso sia pulito e libero da ogni tipo di attacco o azione che possa compromettere la sicurezza. Lo scopo della fase di apprendimento è appunto quello di definire cosa è “normale” e definire “l’aspetto” del traffico accettabile, in modo da poter, successivamente, individuare comportamenti anomali e segnalare il sistema della violazione.

4.1.4 Individuare le vulnerabilità

Il SANS Institute, ha individuato alcune tecniche per l’individuazione di possibili vulnerabilità tramite le informazioni ricavate dai log [35]. Di seguito illustriamo degli esempi di rilevamento di alcuni dei principali attacchi.

4.1.4.1 Cross-Site Scripting (XSS)

Questo tipo di attacco è molto simile al XSRF, descritto nel paragrafo 2.3.2. Gli attacchi Cross-site scripting inseriscono del codice dannoso, solitamente JavaScript, in posti dove altri utenti possono vederlo, come per esempio form, forum di discussione, ecc. Il codice ha l’obiettivo di rubare i cookie, che permetteranno all’attaccante di impersonare la vittima o convincerlo a divulgare la password tramite tecniche di ingegneria sociale. Questi attacchi sfruttano la fiducia tra la vittima e l’applicazione e il fatto che non esistono metodi di validazione dell’input per rifiutare i caratteri JavaScript o altro codice. Un semplice metodo per individuare i tag malevoli è utilizzare la seguente espressione regolare:

```
/((\%3C)|<)((\%2|\\)*[a-z0-9\%]+((\%3E)|>))/ix
```

Senza addentrarci troppo in spiegazioni tecniche sulla struttura delle espressioni regolari (facilmente reperibile online [36]), diciamo che questa espressione individua tutte le stringhe che iniziano con il carattere < (o equivalente esadecimale), contengono uno \ (o equivalente esadecimale) opzionale come secondo carattere (per i tag di chiusura), contengono un arbitrario numero di caratteri alfanumerici, e infine terminano la stringa con il carattere > (o equivalente esadecimale). I caratteri finali i e x, indicano rispettivamente il case insensitive e di ignorare gli spazi bianchi. In breve, tutte le stringhe nella forma <something> o <\something>.

Purtroppo questa espressione regolare riconosce anche eventuali tag HTML e XML legittimi, utilizzati per esempio nei forum per formattare il testo, segnalando quindi eventuali falsi positivi. Si potrebbe filtrare solo per i tag che contengono la parola `script`, ma purtroppo anche questo non basterebbe. Infatti JavaScript può essere incluso in diversi tag. Uno dei più popolari è il tag `img`, usato per impostare le immagini. Il parametro `src` del tag `img` lavora molto bene in sincronia con JavaScript. Di seguito sono elencati alcuni esempi di codice JavaScript inclusi in altri tag HTML:

- ``
- ``
- `link`
- `<body onload="alert(String.fromCharCode(88, 83, 83))">`

Un approccio corretto, dovrebbe controllare tutte le possibili espressioni che possano contenere del codice JavaScript, o altro codice attivo:

```
/(javascript|vbscript|expression|applet|script|embed|object|iframe|frame)/i
```

Di seguito sono elencate due richieste registrate da un log server:

```
1. 217.160.165.173 - - [12/Mar/2004:22:31:12 0500] "GET
/foo.jsp?<SCRIPT>foo</SCRIPT>.jsp HTTP/1.1" 200 578
2. 217.160.165.173 - - [12/Mar/2004:22:37:17 0500] "GET /cgibin/
cvslog.cgi?file=<SCRIPT>window.alert</SCRIPT> HTTP/1.1" 403 302
```

La prima richiesta mostra una risposta positiva del server (HTTP 200) alla risorsa `foo.jsp`. Richiedere manualmente la risorsa potrebbe essere un buon test per valutare se la vulnerabilità esiste davvero o se la richiesta è lecita. La seconda richiesta invece, ha restituito un accesso negato (HTTP 403) alla richiesta di una risorsa `cvslog.cgi`.

4.1.4.2 Vulnerabilità all'injection

L'injection può essere di qualsiasi tipo: SQL, LDAP, XPATH, HTML, XML, comandi del sistema operativo, ecc. XSRF precedentemente descritto è, in effetti, un sottoinsieme di HTML injection. In questa sezione affrontiamo solo SQL injection, fermo restando che l'individuazione di altri tipi di injection è facilmente realizzabile sulla base della stessa tecnica.

Come già visto nel paragrafo 2.3.3.1, i caratteri più pericolosi sono l'apice singola e il doppio trattino, i quali, ricordiamo, rappresentano rispettivamente il delimitatore di stringa e l'inizio di un commento. La seguente espressione regolare è sufficiente a individuare tali caratteri:

```
/(\'|(\%27)|(\-\-)|(\#)|(\%23))/ix
```

Il lettore attento avrà riconosciuto i caratteri indicati e le loro equivalenti versioni in esadecimale. In particolare si è aggiunto anche il carattere cancelletto, utilizzato nei database MySQL.

Purtroppo però, come abbiamo già visto, gli attacchi di SQL injection possono anche non fare affatto uso di delimitatori di stringhe o commenti, per esempio innestando le query una nell'altra o concatenandole:

```
SELECT * FROM table WHERE value = $user_input
```

In questo caso, si potrebbe inserire un'altra query concatenata nell'input:

```
$user_input = 7; SELECT * FROM users
```

Facendo diventare la query originale, una concatenazione di due query:

```
SELECT * FROM tabel WHERE value = 7; SELECT * FROM users
```

Fortunatamente (o sfortunatamente, secondo i punti di vista), la libreria di base PHP non permette di eseguire più query concatenate (cosa che invece la nuova classe `mysqli`, sempre di PHP, permette). Abbiamo visto però, che è possibile eseguire delle query innestate utilizzando dei `SELECT` all'interno delle clausole `WHERE`.

Altre stringhe importanti da controllare sono `OR`, `UNION`, `INSERT`, `UPDATE`, `DELETE`, `REPLACE`, `TRUNCATE`. In particolare, si è già vista la potenza dell'operatore `UNION`. La seguente espressione filtra le stringhe `OR` in aggiunta ai caratteri `'` e `--`:

```
/\w*((\%27)|(\'))(\s|\+|\%20)*((\%6F)|o|(\%4F))((\%72)|r|(\%52))/ix
```

Se a questa espressione si aggiungono quelle relative agli altri operatori e funzioni pericolose, abbiamo completato il nostro controllo di sicurezza.

4.1.4.3 Esecuzione di file pericolosi

Le applicazioni che permettono all'utente di inserire filename completi o parziali, sono spesso vulnerabili se l'input non viene adeguatamente controllato. Permettere a un utente di scegliere arbitrariamente il nome del file, può portare l'applicazione a eseguire programmi di sistema o URL esterni. In passato, PHP è stato fortemente criticato per la possibilità di accettare URL nelle dichiarazioni `include` e `require`. Questa vulnerabilità è comunemente definita *remote URL include*. La seguente dichiarazione, per esempio, permette di includere ed eseguire tutto ciò che viene inviato in POST al server:

```
include "PHP://input";
```

Cerchiamo prima di catturare le inclusioni di file remoti. Se un file viene referenziato su una macchina remota, esso sarà caratterizzato da una stringa composta di protocollo e percorso, tipo `http://www.example.com/bad.inc`. Possiamo individuare questi identificatori di protocollo con la seguente espressione regolare:

```
/(https?|ftp|PHP|data):/i
```

Se poi l'applicazione prevede l'upload di file, c'è il rischio che il pericolo provenga direttamente dall'interno dell'applicazione stessa. In questo caso, un accurato controllo dei file caricati, dovrebbe essere sufficiente a garantire la sicurezza del sistema.

4.1.4.4 Insecure direct object reference

Spesso alcuni oggetti interni sono resi disponibili all'esterno tramite parametri. Quando questi oggetti sono esposti, se non sono presenti appropriate misure di controllo degli accessi, un attaccante può manipolare degli oggetti non autorizzati. Tra gli oggetti interni possiamo avere: file o directory, URL, chiavi o tabelle di database.

Alcuni file pericolosi, possono essere facilmente riconosciuti dal loro nome, per esempio `/etc/passwd`, `/etc/shadow`, `cmd.exe`, ecc. Inoltre, come abbiamo già visto, le directory possono essere attraversate usando l'attacco `dot-dot-slash`. La seguente espressione regolare filtra questi ultimi caratteri:

```
/(\.|\(|\)|\%2E)(\.\(|\)|\%2E)(\./|\%2F|\\|\%5C)/i
```

La prima metà controlla i due caratteri punto includendo le controparti codificate in URL, la seconda metà invece verifica l'esistenza dei caratteri `slash` o `backslash`, entrambi utilizzati per delimitare le directory.

4.1.4.5 Cross-Site Request Forgery (XSRF)

XSRF è probabilmente uno degli attacchi più diffusi al momento, poiché sfrutta una delle caratteristiche basilari dell'HTML, i link. Qualsiasi processo che fa uso di richieste come la seguente, è vulnerabile:

```
https://www.example.com/transfer.PHP?amount=100&toAcct=12345
```

Più genericamente, ogni applicazione le cui richieste si basano esclusivamente sulle credenziali di accesso, come i cookie di sessione, sono vulnerabili (i dettagli sono stati descritti al paragrafo 2.3.2).

Per sfruttare una vulnerabilità XSRF, un attaccante può inserire la seguente immagine nel post di un forum:

```
<img src=https://www.example.com/com/transfer.PHP?amount=100&toAcct=99999 width="0" height="0">
```

Il browser cercherà di caricare l'immagine con dimensioni nulle (i.e. invisibile) mandando la richiesta all'URL specificato. Non è importante che l'URL faccia riferimento a un'immagine realmente esistente; la richiesta viene inviata ugualmente.

Come fare a individuare questo subdolo attacco? Si usa il `referer`. Il `referer` è un header HTTP che indica al server l'ultimo URL che linka alla richiesta corrente (i.e. da dove viene la richiesta). Questi vengono automaticamente aggiunti dal browser ad ogni link cliccato, ma non compaiono se vengono scritti manualmente gli URI sulla barra degli indirizzi. Consideriamo il seguente esempio in cui un'attaccante prepara una pagina al seguente URL:

```
http://www.attacker.com/freestuff.PHP
```

L'attaccante convince la vittima, che al momento è connessa e autenticata al proprio home banking (<https://www.bank.com> per esempio), a navigare il suo sito. Nel sito è contenuto un attacco XSRF che esegue un trasferimento di denaro tramite l'applicazione dell'home banking. Nel file di log dell'applicazione bancaria comparirà un record simile al seguente:

```
192.168.4.6 - - [10/Oct/2007:13:55:36 -0700] "GET /transfer.PHP?amount=100&toAcct=99999 HTTP/1.0" 200 4926 "http://www.attacker.com/freestuff.PHP" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322)"
```

I due campi importanti qui sono, l'URL richiesto (`/transfer.PHP?amount=100&toAcct=99999`) e il `referer` (`http://www.attacker.com/freestuff.PHP`). Solitamente il `referer` dovrebbe rappresentare un URL della stessa banca (<https://www.bank.com>), quindi questa riga indica chiaramente un tentativo di XSRF. Con dei semplici script, è possibile individuare questi tentativi di attacco:

```
if (($referer ne '-') && ($referer !~ /^https:\\\\www.bank.com\\(login|overview|transfer)\\.jsp/)) {
    print("XSRF attack: $referer\n");
}
```

Purtroppo, come abbiamo già visto più e più volte nel corso di questo trattato, gli header sono facilmente manipolabili lato client. Per esempio, il seguente script perl modifica il valore del `referer` facendo credere che la richiesta arrivi dal sito della banca:

```
use HTTP::Request::Common qw(POST GET);
use LWP::UserAgent;
$ua = LWP::UserAgent->new();
$req = POST 'https://www.bank.com/opentransfer.PHP';
$req->header(Referer => 'https://www.bank.com/transfer.PHP');
$res = $ua->$request($req);
```

La soluzione migliore quindi, resta un buon sistema di sicurezza, come quelli descritti in 2.3.2.

4.1.4.6 Comunicazioni non sicure

Ogni applicazione HTTP utilizza una qualche forma di autenticazione che deve, necessariamente, transitare sulla rete. Per assicurare che la confidenzialità e la sicurezza della trasmissione su un mezzo insicuro come Internet, tutto il traffico che coinvolge l'intera operazione di autenticazione deve transitare su SSL.

Molti web server forniscono degli strumenti di log per le richieste SSL. Apache, per esempio permette, tramite la direttiva `CustomLog`, di salvare le informazioni sui parametri SSL. Per esempio la seguente direttiva:

```
CustomLog "logs/ssl_request.log" \
"%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
```

Restituisce log di questo tipo:

```
[05/Feb/2012:12:26:28 +0100] 127.0.0.1 TLSv1 DHE-RSA-CAMELLIA256-SHA "GET /atna/
HTTP/1.1" 311
```

L'esempio mostra una connessione TLSv1 con cifratura DHE-RSA-CAMELLIA256-SHA.

Se il web server supporta solamente connessioni SSL su porta TCP 443, allora nel log file dovrebbero essere presenti solo record di richieste cifrate.

4.2 Definizione di audit trail

Al giorno d'oggi, il termine *audit trail* è utilizzato in più di un ambito nelle varie attività. Storicamente l'audit trail serviva a fornire una storia completa di qualunque transazione finanziaria. L'idea era di essere in grado di identificare ogni singola fase del processo, dall'inizio della transazione fino al suo completamento. Tipicamente, questo processo si realizzava con la produzione di documenti che mostravano il progresso della transazione dall'inizio alla fine. Oggi l'audit trail non si limita più alle sole transazioni finanziarie e ha spesso a che fare anche con il tracciamento elettronico dei dati.

Un qualsiasi sistema di audit trail, stabilisce una lista cronologica delle fasi necessarie all'iniziazione della transazione, così come quelle che l'hanno portata al suo completamento. Questo processo può essere molto semplice o estremamente complicato, a seconda del numero di fasi coinvolte nella transazione. In generale comunque, l'audit trail è un meccanismo di controllo di un sistema che permette, per i dati imputati e successivamente processati dal sistema, la tracciabilità a ritroso al dato originale.

Il centro NCSC (National Computer Security Center) ha approvato la seguente definizione di audit trail: *un audit-trail è una registrazione cronologica delle attività di sistema, sufficiente per consentire la ricostruzione, la revisione e l'esame della sequenza di situazioni e di attività che hanno riguardato o che hanno condotto a un'operazione, una procedura o un evento in una transazione dal suo inizio ai suoi risultati finali* [37].

Un audit trail nel contesto degli *Electronic Medical Record* (EMR) è adottato per le seguenti ragioni:

- Come misura di sicurezza per stabilire chi ha avuto accesso alla cartella clinica del paziente.
- Per la fatturazione delle parcelle.
- A fini statistici sulla salute pubblica e le ricerche mediche.

Non solo dovrebbero essere protetti gli EMR dall'accesso non autorizzato, ma è necessario anche tracciarli, in modo da garantirne l'accesso solo in caso di necessità. Leggi come l'HIPAA [8] impongono che l'accesso alle cartelle cliniche siano riservate solo agli utenti autorizzati.

Le leggi stabiliscono con fermezza le linee guida che gli amministratori o lo staff delle aziende sanitarie devono rispettare riguardo alle cartelle cliniche e ai dati sensibili dei pazienti. La violazione di tali regole possono portare a sanzioni pesanti e perfino l'arresto. L'audit trail elettronico per l'accesso agli EMR dovrebbe essere progettato per garantire la conformità all'autenticazione e alla confidenzialità. Inoltre, alle aziende sanitarie viene di tanto in tanto richiesto dagli stati e dalle agenzie federali di pubblica sanità, di raccogliere informazioni su diagnosi e scoppi epidemiologici.

Attraverso l'analisi costante degli archivi che formano il sistema di audit trail, è possibile scoprire le cause di eventuali anomalie. Gli “audit-trail” costituiscono uno dei metodi migliori per individuare le possibili infiltrazioni nel sistema e sono utili non solo dopo che si è verificato un attacco, ma anche durante lo svolgimento dello stesso. Il rovescio della medaglia è la loro vulnerabilità. Data la loro natura, sono spesso scritti e conservati sul sistema stesso, e quindi soggetti a possibili alterazioni. Ecco perché sono richieste severe politiche di controllo degli accessi alla banca dati dei messaggi audit. In alcuni casi si arriva anche a disporre tale database in un server remoto, inaccessibile fisicamente se non da parte degli enti predisposti al monitoraggio degli audit.

Si possono utilizzare i record di audit per individuare i danni provocati oppure i dati compromessi e/o mancanti. La politica di sicurezza aziendale deve includere una regolare manutenzione, analisi e copia degli audit-trail. Ogni sistema, anche i più sicuri, sono vulnerabili a utilizzi errati da parte degli utenti. In questo caso il tracciamento dell'audit rappresenta l'unico modo per rilevare attività abusive attuate da utenti legittimi.

Tuttavia, si deve riconoscere che l'auditing, la manutenzione e l'osservazione degli audit-trail non offre una protezione infallibile contro gli attacchi al sistema. Eseguendo un'operazione di *spoofing* (falsificazione dell'identità) del sistema, ad esempio, questa non è rilevata dall'auditing. Se qualcuno intercetta le trasmissioni di sistema con uno *sniffer*, l'auditing probabilmente non si accorgerà di nulla poiché non si accede ai dati del server, ma semplicemente si osservano i dati in transito. In sostanza l'auditing non può sostituire un sistema di autenticazione efficiente o una robusta politica di sicurezza.

4.3 Specifiche del profilo ATNA

La responsabilità utente viene garantita attraverso l'Audit Trail. L'Audit Trail deve permettere a un ufficiale di sicurezza, autorizzato alla verifica delle attività, di valutare la conformità alle politiche del dominio di sicurezza, individuare istanze di comportamenti non conformi e facilitare l'individuazione di attività di creazione, accesso, modifica e cancellazione di informazioni sanitarie protette (PHI). Queste informazioni possono essere richieste dagli utenti o scambiate tra sistemi. Ciò include le informazioni esportate e importate da e verso ogni nodo sicuro del dominio di sicurezza.

La responsabilità utente è ulteriormente assicurata da un deposito di Audit Record centralizzato (Centralized Audit Record Repository) altamente sicuro. Il trasferimento di un Audit Record da un qualunque attore IHE all'Audit Record Repository, riduce le possibilità di manomissione e rende più semplice monitorare il dipartimento. Nodi disconnessi dalla rete dovrebbero salvare i propri Audit Record localmente e trasferirli all'Audit Record Repository non appena viene ripristinata la connessione al dominio di sicurezza.

L'Audit Trail contiene informazioni capaci di rispondere a domande del tipo:

- Il PHI di che paziente è stato richiesto?
- Che utente l'ha richiesto?
- Quali autenticazioni utente fallite sono state riportate?
- Quali autenticazioni nodo fallite sono state riportate?

L'*Audit Trail and Node Authentication* fornisce gli strumenti necessari a rendere l'azienda conforme alle norme di sicurezza e privacy (HIPAA, European, Japanese, ecc.), ma non la rende automaticamente conforme. Per una guida alla gestione degli audit log, fare riferimento al documento del National Institute of Standards and Technology (NIST) [38].

Flusso dei messaggi di audit

L'utilizzo dell'auditing come parte di un processo di sicurezza e privacy è pensato per quelle situazioni in cui le persone coinvolte sono solitamente attendibili ed esiste la necessità di ampia flessibilità per rispondere rapidamente al mutare delle situazioni. Questo è tipico degli ambienti sanitari. L'auditing tiene traccia di ciò che accade e le persone coinvolte sanno che le loro azioni vengono registrate. Questo significa che l'audit record deve raccogliere le descrizioni degli eventi per l'intera durata del processo, non solamente dei singoli elementi corrispondenti agli attori IHE coinvolti.

L'audit IHE è il primo di diversi profili che corrispondono alle varie forme di controllo degli accessi e autenticazione. L'auditing è richiesto indipendentemente dai metodi scelti per il controllo degli accessi e l'autenticazione.

L'immagine in Figura 4.2 descrive il flusso audit specificato da IHE:

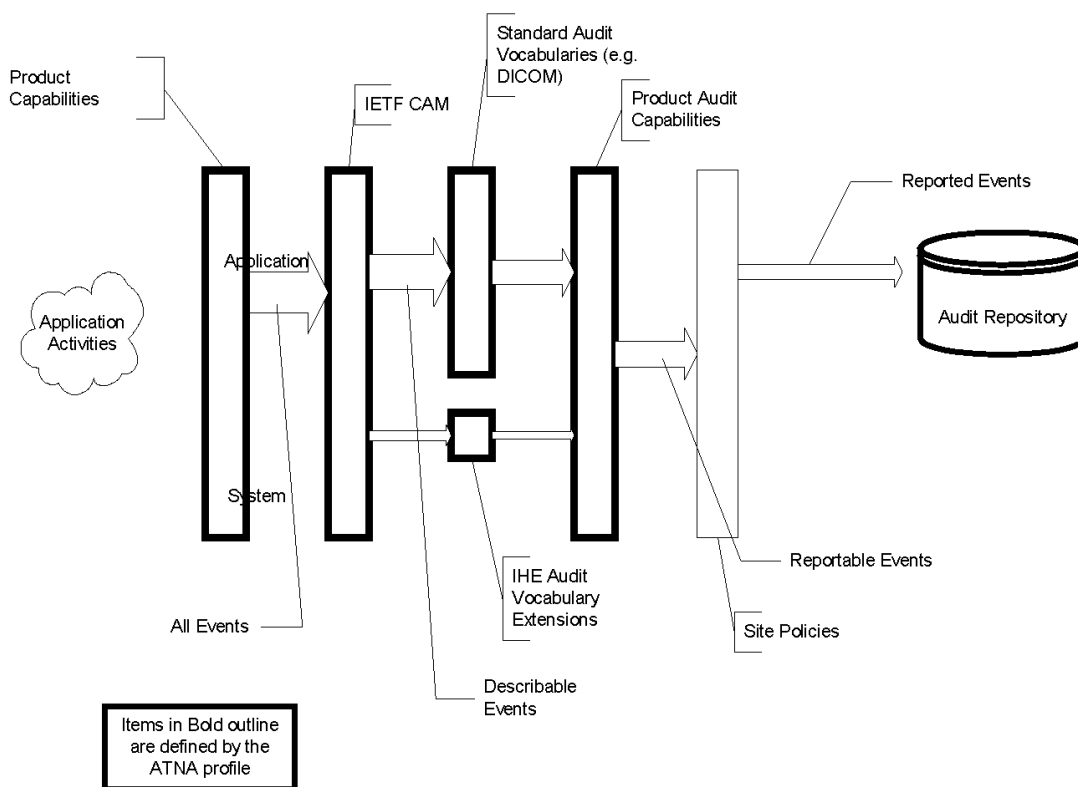


Figura 4.2. Flusso di eventi in un messaggio di audit.

1. Vengono eseguite delle attività nel mondo reale, alcune delle quali comprendono l'utilizzo di dispositivi che includono un supporto ad un qualche profilo IHE. Questo prodotto può avere componenti che corrispondono a specifici attori IHE.
2. Un'ampia varietà di eventi avviene durante questo processo, alcuni dei quali sono direttamente correlati alle attività degli attori IHE. Gli eventi possono spaziare dai dettagli più insignificanti come la pressione di determinati tasti, fino a eventi di alto livello come l'analisi

di un caso diagnostico. Naturalmente solo pochi di questi eventi sono significativi per l'auditing della sicurezza e la privacy.

3. DICOM ha standardizzato il vocabolario per i messaggi audit. Il Vocabolario di Messaggi Audit DICOM estende il vocabolario di base fornito dallo standard RFC-3881 e specifica ulteriormente alcuni degli elementi opzionali dell'RFC-3881. Un esempio di estensione del vocabolario è l'aggiunta di un codice per indicare che un campo contiene un'istanza di Studio DICOM. Un esempio di specifica di elemento opzionale è l'UserID, che dovrebbe corrispondere all'identificativo utente del sistema operativo locale utilizzato, mentre l'AlternateID dovrebbe indicare l'identificativo utente utilizzato dal sistema di autenticazione dell'azienda (se diversi). Il profilo definisce altri eventi che non corrispondono a eventi definiti nel vocabolario DICOM. Questi eventi sono descrivibili tramite RFC-3881.
4. Il sito locale è responsabile delle proprie politiche di reporting. Il profilo IHE specifica le funzioni che dovrebbero essere presenti nell'audit reporting e i dettagli relativi al controllo dei report accessibili dall'autorità predisposta alla sicurezza locale.
5. IHE specifica gli eventi che devono essere riportati nell'audit trail. Esistono altri eventi legati alla sicurezza, che possono essere inclusi nell'audit trail o tramite altri mezzi.

Come meccanismi di trasporto dei messaggi di record verso l'Audit Record Repository, vengono riconosciuti due protocolli:

1. Trasmissione dei messaggi su UDP. Alcune limitazioni riscontrate:
 - non avviene alcun tipo di conferma di messaggio ricevuto dal destinatario;
 - non esiste la possibilità di cifrare i messaggi;
 - non è possibile autenticare il nodo inviante e la banca dati centrale tramite certificati;
 - i messaggi potrebbero essere troncati o persi.
2. Trasmissione dei messaggi su SSL/TLS [39].

Attori e transazioni

L'immagine in Figura 4.3, illustra le transazioni e gli attori direttamente coinvolti nel profilo di integrazione ATNA. Quando un'implementazione sceglie di supportare questo profilo per un attore, quell'attore deve essere raggruppato con l'attore nodo sicuro.

L'attore nodo sicuro dovrebbe includere:

1. Le transazioni di tutte le connessioni dei nodi autenticati che possano esporre dati sensibili.
2. Tutte le funzioni utente protette (login, logout, ecc.) per assicurare l'accesso solamente a utenti autorizzati.
3. L'evento di registrazione dell'audit come specificato dalla transazione ITI-20 [40].

Il nodo sicuro include tutti gli aspetti che si riferiscono all'autenticazione utente, la protezione del file system e la sicurezza della piattaforma operativa. L'applicazione sicura è invece, un prodotto che non include l'ambiente operativo, ma fornisce servizi di sicurezza per le caratteristiche dell'applicazione [15].

L'audit repository dovrebbe supportare:

1. entrambi i meccanismi di trasporto di audit descritti sopra;

2. ogni formato IHE dei messaggio di audit inviato tramite uno dei meccanismi indicati al punto precedente.
3. meccanismi di protezione e controllo dell'accesso utente.

Questo profilo non specifica altre funzioni per l’Audit Repository, ma ci si aspetta che sia in grado di eseguire tutte le funzioni tipiche delle banche dati, come screening, reporting, archiviazione, ecc.

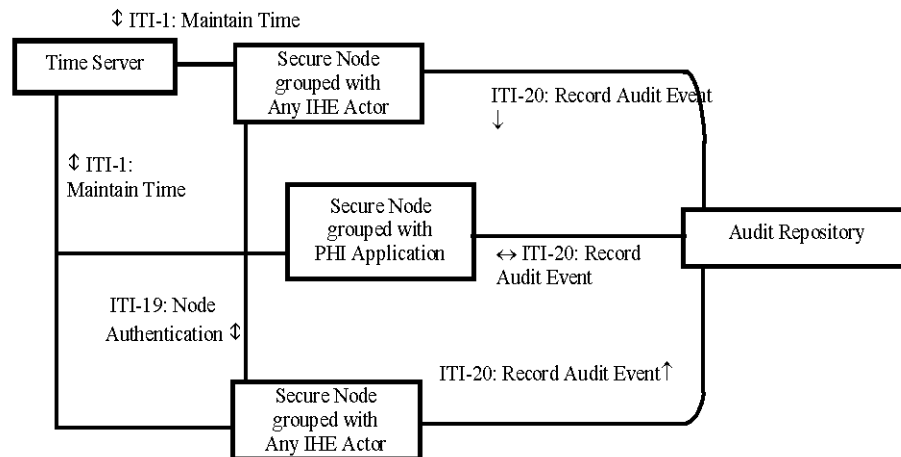


Figura 4.3. Diagramma di funzionamento del profilo ATNA.

4.3.1 Schema del messaggio di audit

Le implementazioni che si dichiarano conformi al profilo ATNA, devono utilizzare il seguente schema XML come formato per i messaggi di audit trail [41]. Questo schema è derivato dallo standard specificato nell’RFC-3881 [42], in accordo alle raccomandazioni W3C sulle strutture XML [43], e include le estensioni DICOM.

```

datatypes xsd = http://www.w3.org/2001/XMLSchema-datatypes

# This defines the coded value type. The comment shows a pattern that can be used to
# further constrain the token to limit it to the format of an OID. Not all schema
# software implementations support the pattern option for tokens. #

other-csd-attributes =
  (attribute codeSystemName {token} |
  attribute codeSystemName {token}),
  # codeSystemName is either an OID or String
  attribute displayName {token}?,
  attribute originalText {token}

CodedValueType =
  attribute csd-code {token},
  other-csd-attributes

# Define the event identification, used later
EventIdentificationContents =
  element EventID {CodedValueType},
  element EventTypeCode {CodedValueType}*,
  attribute EventActionCode {
    "C" | ## Create
    "R" | ## Read
    "U" | ## Update
    "D" | ## Delete
    "E" }?, ## Execute
  attribute EventDateTime {xsd:dateTime},
  attribute EventOutcomeIndicator {
    "0" | ## Nominal Success
    "4" | ## Minor failure
    "8" | ## Serious failure
  }
    
```

```

        "12"}, ## Major failure,
        element EventOutcomeDescription {text}?

# Define AuditSourceIdentification, used later
# Note: This includes one constraint that cannot be represented yet in RNC. The use
of a token other than the specified codes is permitted only if the codeSystemName is
present. #
# Note: This has no elements, only attributes.
AuditSourceIdentificationContents =
    attribute code {
        "1" | ## End-user display device, diagnostic device
        "2" | ## Data acquisition device or instrument
        "3" | ## Web Server 225 process or thread
        "4" | ## Application Server process or thread
        "5" | ## Database Server process or thread
        "6" | ## Security server, e.g., a domain controller
        "7" | ## ISO level 1-3 network component
        "8" | ## ISO level 4-6 operating software
        "9" | ## other
        token }, ## other values are allowed
    other-csd-attributes?, ## they define the meaning of code
    attribute AuditEnterpriseSiteID {token}?,
    attribute AuditSourceID {token},
    element AuditSourceTypeCode {token}*

# Define ActiveParticipantType, used later

    ActiveParticipantContents =
        element RoleIDCode {CodedValueType}*,
        element MediaIdentifier {
            element MediaType {CodedValueType}}?,
        attribute UserID {text},
        attribute AlternativeUserID {text}?,
        attribute UserName {text}?,
        attribute UserIsRequestor {xsd:boolean},
        attribute NetworkAccessPointID {token}?,
        attribute NetworkAccessPointTypeCode {
            "1" | ## Machine Name, including DNS name
            "2" | ## IP Address
            "3" | ## Telephone Number
            "4" | ## Email address
            "5"? ## URI (user directory, HTTP-PUT, ftp, etc.)

# The BinaryValuePair is used in ParticipantObject descriptions to capture param-
eters. All values (even those that are normally plain text) are encoded as
xsd:base64Binary. This is to preserve details of encoding (e.g., nulls) and to pro-
tect against text contents that contain XML fragments. These are known attack points
against applications, so security logs can be expected to need to capture them with-
out modification by the audit encoding process. #

        ValuePair = # clarify the name
            attribute type {token},
            attribute value {xsd:base64Binary}
            # used to encode potentially binary, mal-formed XML text, etc.

# Define ParticipantObjectIdentification, used later

# Participant Object Description, used later
DICOMObjectDescriptionContents =
    element MPPS {
        attribute UID {token}}*,
        # OID pattern="[0-2](\\.0|\\. [1-9][0-9]*)*"
    element Accession {
        attribute Number {token}}*,
    element SOPClass { # SOP class for one study
        element Instance {
            attribute UID {token}}*,
            # OID pattern="[0-2](\\.0|\\. [1-9][0-9]*)*"
            attribute UID {token}?,
            # OID pattern="[0-2](\\.0|\\. [1-9][0-9]*)*"
            attribute NumberOfInstances {xsd:integer}
        },
    element ParticipantObjectContainsStudy {
        element StudyIDs {
            attribute UID {token}}*

```



```

    },
    element Encrypted {xsd:boolean}?,
    element Anonymized {xsd:boolean}?

ParticipantObjectIdentificationContents =
    element ParticipantObjectTypeCode {CodedValueType},
    (element ParticipantObjectName {token} | # either a name or
    element ParticipantObjectQuery {xsd:base64Binary}), # a query ID field,
    element ParticipantObjectDetail {ValuePair}*,
    # optional details, these can be extensive and large
    element ParticipantObjectDescription {token}*,
    # optional descriptive text DICOMObjectDescriptionContents,
    attribute ParticipantObjectID {token}, #mandatory ID
    attribute ParticipantObjectTypeCode {( # optional type
        "1" | #3 Person
        "2" | #3 System object
        "3" | #3 Organization
        "4")}? , ## Other
    attribute ParticipantObjectTypeCodeRole {( ## optional role
        "1" | ## Patient
        "2" | ## Location
        "3" | ## Report
        "4" | ## Resource
        "5" | ## Master File
        "6" | ## User
        "7" | ## List
        "8" | ## Doctor
        "9" | ## Subscriber
        "10" | ## guarantor
        "11" | ## Security User Entity
        "12" | ## Security User Group
        "13" | ## Security Resource
        "14" | ## Security Granulativity Definition
        "15" | ## Provider
        "16" | ## Report Destination
        "17" | ## Report Library
        "18" | ## Schedule
        "19" | ## Customer
        "20" | ## Job
        "21" | ## Job Stream
        "22" | ## Table
        "23" | ## Routing Criteria
        "24")}? , ## Query?,
    attribute ParticipantObjectDataLifeCycle {( # optional life cycle stage
        "1" | ## Origination, Creation
        "2" | ## Import/ Copy
        "3" | ## Amendment
        "4" | ## Verification
        "5" | ## Translation
        "6" | ## Access/Use
        "7" | ## De-identification
        "8" | ## Aggregation, summarization, derivation
        "9" | ## Report
        "10" | ## Export
        "11" | ## Disclosure
        "12" | ## Receipt of Disclosure
        "13" | ## Archiving
        "14" | ## Logical deletion
        "15")}? , ## Permanent erasure, physical destruction
    attribute ParticipantObjectSensistity {token}?

# The basic message
message = element AuditMessage {
    element EventIdentification {EventIdentificationContents},
    # The event must be identified
    element ActiveParticipant {ActiveParticipantContents}+,
    # It has one or more active participants
    element AuditSourceIdentification {AuditSourceIdentificationContents},
    # It is reported by one source
    element ParticipantObjectIdentification {ParticipantObjectIdentificationCon-
    tents}*
    # It may have other objects involved
}
# And finally the magic statement that message is the root of everything.
start=message

```

Di seguito vengono descritti i campi dello schema e i relativi attributi. Nella prima colonna vengono elencati i principali elementi che compongono il messaggio di audit (evento, partecipanti, sorgente e oggetti coinvolti), nella seconda colonna gli attributi per l'elemento in questione, nella terza se l'attributo è obbligatorio o no e nella quarta una breve descrizione dell'attributo. Per la lista completa dei vari identificativi, fare riferimento ai Technical Framework e supplementi [40, 41].

	Nome del campo	Obbl.	Descrizione
Event	EventID	S	Identificatore di uno specifico evento di audit.
	EventActionCode	N	Identifica il tipo di azione eseguita durante l'evento che ha generato l'audit.
	EventDateTime	S	L'ora e la data in cui è avvenuto l'evento, in un formato universale UTC.
	EventOutcomeIndicator	S	Indica se l'evento ha avuto successo o no.
	EventTypeCode	N	Identifica la categoria dell'evento.
Active Participant (multivalore)	UserID	S	Identificatore univoco dell'utente attivamente partecipe all'evento.
	AlternativeUserID	N	Identificativo univoco alternativo dell'utente.
	UserName	N	Nome utente interpretabile.
	UserIsRequestor	S	Indica se l'utente è il richiedente o l'iniziatore dell'evento.
	RoleIDCode	N	Specifica del ruolo interpretato dall'utente al momento dell'evento.
	NetworkAccessPointTypeCode	N	Identificativo del tipo di punto di accesso alla rete.
	NetworkAccessPointID	N	Identificativo del dispositivo utilizzato dall'utente come punto di accesso alla rete.
Audit Source	AuditEnterpriseSiteID	N	Locazione logica all'interno della rete dell'azienda sanitaria. Qualifica ulteriormente l'AuditSourceID.
	AuditSourceID	S	Identifica la sorgente dell'evento.
	AuditSourceTypeCode	N	Specifica il tipo di sorgente.
Participant Object (multivalore)	ParticipantObjectTypeCode	N	Codice per il tipo di oggetto.
	ParticipantObjectTypeCodeRole	N	Codice che rappresenta il ruolo funzionale dell'oggetto.
	ParticipantObjectDataLifeCycle	N	Identificativo per il life-cycle stage.
	ParticipantObjectIDTypeCode	S	Descrive l'identificativo contenuto nel ParticipantObjectID.
	ParticipantObjectSensitivity	N	Denota specifiche politiche per l'oggetto.
	ParticipantObjectID	S	Identifica l'istanza specifica dell'oggetto.
	ParticipantObjectName	N	Descrittore della specifica istanza dell'oggetto.
	ParticipantObjectQuery	N	La query per un oggetto tipo query.
	ParticipantObjectDetail	N	Dettagli relativi ad informazioni specifiche dell'oggetto utilizzato.
	SOPClass	S/N	Richiesto se il ParticipantObjectIDTypeCode è di tipo 110180.
	Accession	N	Accession Number associato all'oggetto.
	MPPS	N	L'ID dell'istanza MPPS associata.
	NumberOfInstances	N	Numero di istanze SOP riferite all'oggetto.
	Instance	N	Valori ID delle istanze SOP
Encrypted	N	Indica se le informazioni sono cifrate.	

	Anonymized	N	Indica se tutte le informazioni identificative del paziente sono state rimosse.
	ParticipantObjectContainsStudy	N	Un'istanza di studio, presente se il ParticipantObjectIDTypeCode non è 110180.

Lo schema XSD di riferimento è reperibile presso il sito dell'RFC-3881 [44].

Capitolo 5

Paradigmi e Strumenti

Prima di introdurre i sistemi implementati, vengono elencati i principali metodi di progettazione adottati e i software per realizzarli. Tra gli altri, sono citati anche alcuni strumenti per l'analisi e la manipolazione delle richieste HTTP/HTTPS.

5.1 Paradigmi e metodi di progettazione

L'intera realizzazione è stata implementata con una libreria ad oggetti PHP. Uno dei principali vantaggi della programmazione a oggetti è la possibilità di creare moduli che non abbiano bisogno di modifiche all'aggiunta di nuovi oggetti o classi. Un programmatore può semplicemente creare un nuovo oggetto che eredita alcune delle caratteristiche degli oggetti esistenti. Tra gli altri vantaggi, la programmazione a oggetti fornisce un supporto naturale alla modellazione software di concetti reali o dei modelli astratti da riprodurre; permette una più facile gestione e manutenzione di progetti complessi; organizza il codice in classi favorendone la modularità e il riuso; implementa i meccanismi di incapsulamento, ereditarietà e polimorfismo.

A partire dalla versione 5.0, del 13 luglio 2004, PHP ha introdotto il supporto nativo alla programmazione a oggetti. Questo ha permesso l'adozione, anche nel web, di tecniche di software design pattern moderne, ampiamente utilizzate nel resto del mondo informatico.

Per lo sviluppo dei meccanismi di autenticazione sono state implementate le soluzioni con username e password basata su form e di mutua autenticazione client-server con certificati, entrambe ampiamente descritte nel capitolo 2.

Per quanto riguarda invece, la realizzazione dei sistemi di audit trail e log, si è fatto uso di due design pattern, il *factory method pattern* e il *singleton pattern*.

5.1.1 Software design pattern

I design pattern furono inizialmente introdotti alla comunità software da Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides (comunemente detti "la gang dei quattro"), nel loro libro *Design Patterns* [45]. Il concetto chiave dietro i design pattern, presentato nell'introduzione, era semplice. Durante i loro anni di sviluppo software, Gamme et al. si accorsero che alcuni schemi di progettazione emergevano con maggiore frequenza, allo stesso modo in cui un architetto progetta dei template su dove dovrebbe essere collocato il bagno o come dovrebbe essere configurata la cucina. Avere questi template, o *design pattern*, significa poter progettare edifici migliori in minor tempo. Lo stesso si applica al software.

Un design pattern è una soluzione progettuale riutilizzabile e generale a un problema ricorrente. Esso non è una libreria o un componente di software riusabile, quanto piuttosto una descrizione o

un modello da applicare per risolvere un problema che può presentarsi in diverse situazioni durante la progettazione e lo sviluppo del software. I design pattern orientati agli oggetti tipicamente mostrano relazioni e interazioni tra classi o oggetti, senza specificare le classi applicative finali coinvolte. I design pattern non sono solamente un mezzo per sviluppare software robusto più velocemente, ma forniscono anche delle tecniche per incapsulare concetti più ampi in termini più semplici. Per esempio, come vedremo in seguito nel capitolo 7.2, si può immaginare un sistema di log astratto, il quale può in seguito, essere implementato in più modi per ottenere diversi meccanismi di logging.

Factory pattern

Gran parte dei design pattern descritti da Gamma et al., incoraggiano a sviluppare sistemi con un basso grado di accoppiamento o dipendenza (*loose coupling*). Immaginiamo di compiere una modifica a un pezzo di codice e successivamente trovarsi davanti ad una cascata di errori in altre parti del sistema, parti che non si credeva fossero correlate al pezzo di codice modificato. Questo fenomeno viene detto accoppiamento forte (*tight coupling*). Funzioni e classi di una parte del sistema si affidano troppo ai comportamenti e alle strutture di altre funzioni e classi di un'altra parte del sistema. È necessario implementare una serie di schemi che permettano a queste classi di interagire tra loro senza però che siano eccessivamente legati da incastrarsi a vicenda.

In grandi sistemi, molta parte del codice si basa su poche classi chiave. Le difficoltà possono nascere qualora diventi necessario modificare tali classi. Per esempio, supponiamo di avere una classe `User` che legge dei dati da un file. Si vuole modificarla per farla leggere da un database, ma tutto il codice fa riferimento alla classe originale che legge i dati da un file. Qui entra in gioco lo schema di factory pattern, illustrato in Figura 5.1.

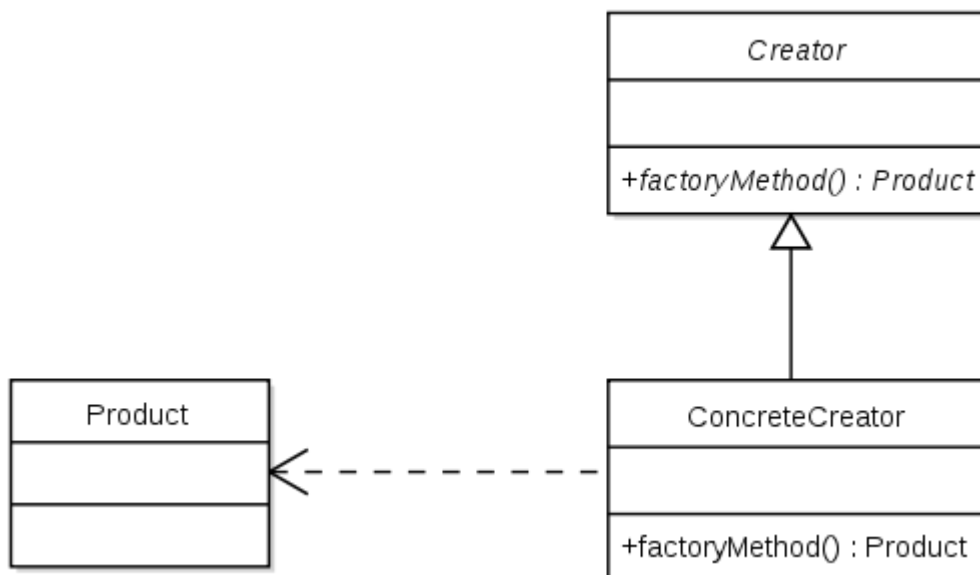


Figura 5.1. Schema UML di un factory method.

Come gli altri creational pattern (che hanno a che fare con i meccanismi di creazione degli oggetti), il factory pattern affronta il problema di creare oggetti senza specificare la classe esatta da creare. Siccome la creazione di un oggetto può comportare una significativa replicazione del codice, oltre ai problemi già citati, il factory design pattern risolve questi problemi definendo un metodo separato per la creazione di oggetti, le cui sottoclassi possono sovrascrivere per specificare il tipo derivato

dell'oggetto creato. Quindi, il factory pattern è una classe che ha alcuni metodi che creano gli oggetti. Invece di creare un nuovo oggetto direttamente, si utilizza la classe factory per creare gli oggetti. In questa maniera se si vuole cambiare il tipo di oggetto creato, basta farlo semplicemente nella classe factory. Tutto il codice cambia automaticamente.

Il factory pattern può essere utilizzato in quelle situazioni in cui:

- La creazione di un oggetto preclude il suo riutilizzo a meno di una significativa replicazione del codice.
- La creazione di un oggetto richiede accesso a informazioni o risorse che non dovrebbero essere contenuti all'interno della classe composta.
- La durata della vita degli oggetti generati deve essere centralizzata per assicurare un comportamento consistente nell'applicazione.

Singleton pattern

Alcune risorse sono esclusive, nel senso che ne esiste una e solamente una di quel tipo. Per esempio, la connessione a un database attraverso un gestore di database è esclusiva, quindi è desiderabile poterlo condividere in tutta l'applicazione per ridurre il carico dovuto alla continua apertura e chiusura della connessione.

Il singleton pattern risolve questa necessità. Esso implementa il concetto matematico di singleton (singoletto), restringendo l'allocazione di una classe a un unico oggetto. Un oggetto quindi, si dice singleton se l'applicazione può includere un'unica copia di quell'oggetto nello stesso momento.

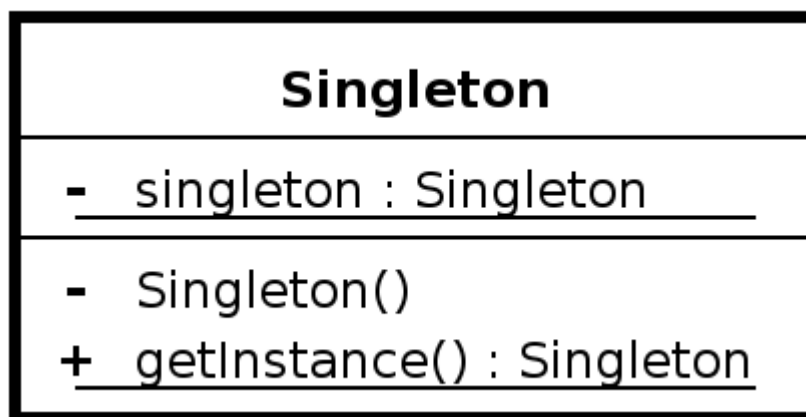


Figura 5.2. Diagramma UML di un singleton.

Spesso i singleton vengono preferiti alle variabili globali perché non inquinano il global space name con variabili non utilizzate e permettono di applicare tecniche di lazy allocation e initiation (ritardare l'allocazione e inizializzazione di un'istanza finché non è strettamente necessario).

5.2 Software e strumenti utilizzati

Tutto il sistema è stato sviluppato su server Apache 2.2.21, con pagine HTML e strato applicativo PHP. Per lo strato dati si è fatto uso di un database MySQL. Per garantire il massimo livello di sicurezza, l'intero progetto non fa uso di tecnologie client-side, come Javascript, Flash o Silverlight. L'intera applicazione è stata testata sia su piattaforma Windows, che Linux, a dimostrare la compa-

tibilità multiplatforma della soluzione. Gli IDE di sviluppo utilizzati sono Adobe Dreamweaver CS5 e Netbeans 7.0.1, col supporto di Subversion 1.6.0 per la gestione del versioning del codice.

Per la creazione e gestione dei certificati sono state utilizzate le funzioni della libreria open source OpenSSL 1.0.0d. Il progetto OpenSSL nasce con lo scopo di sviluppare uno strumento robusto, completo e open source per l'implementazione dei protocolli SSL e TLS, oltre che fornire una libreria di funzioni di crittografia. OpenSSL si fonda sulla libreria SSLeay sviluppata da Eric A. Young e Tim J. Hudson, ed è licenziata sotto licenza Apache, che ne permette l'utilizzo sia per scopi commerciali che non [46]. Il modulo OpenSSL per PHP supporta, alcune delle funzioni per la generazione e verifica delle firme e per la codifica e decodifica dei dati. Tuttavia, molte caratteristiche non sono ancora supportate, come la revoca dei certificati, la generazione di liste di revoca, ecc.

Per la valutazione dei rischi di sicurezza, sono state usate le seguenti estensioni per Firefox per l'analisi dell'HTTP: Live HTTP Headers, Tamper Data, HTTP Fox e FireForce.

Live HTTP Headers è un plugin realizzato da Daniel Savard e Nikolas Coukouma, che scarica il traffico grezzo HTTP e HTTPS in una barra laterale all'interno dell'interfaccia browser. L'estensione visualizza ogni richiesta/risposta HTTP e HTTPS comprensiva degli header. Tramite una funzionalità di replay è possibile anche modificare l'intera richiesta (per esempio, in Figura 5.3 è stato fatto credere all'applicazione che si sta utilizzando un browser non compatibile).

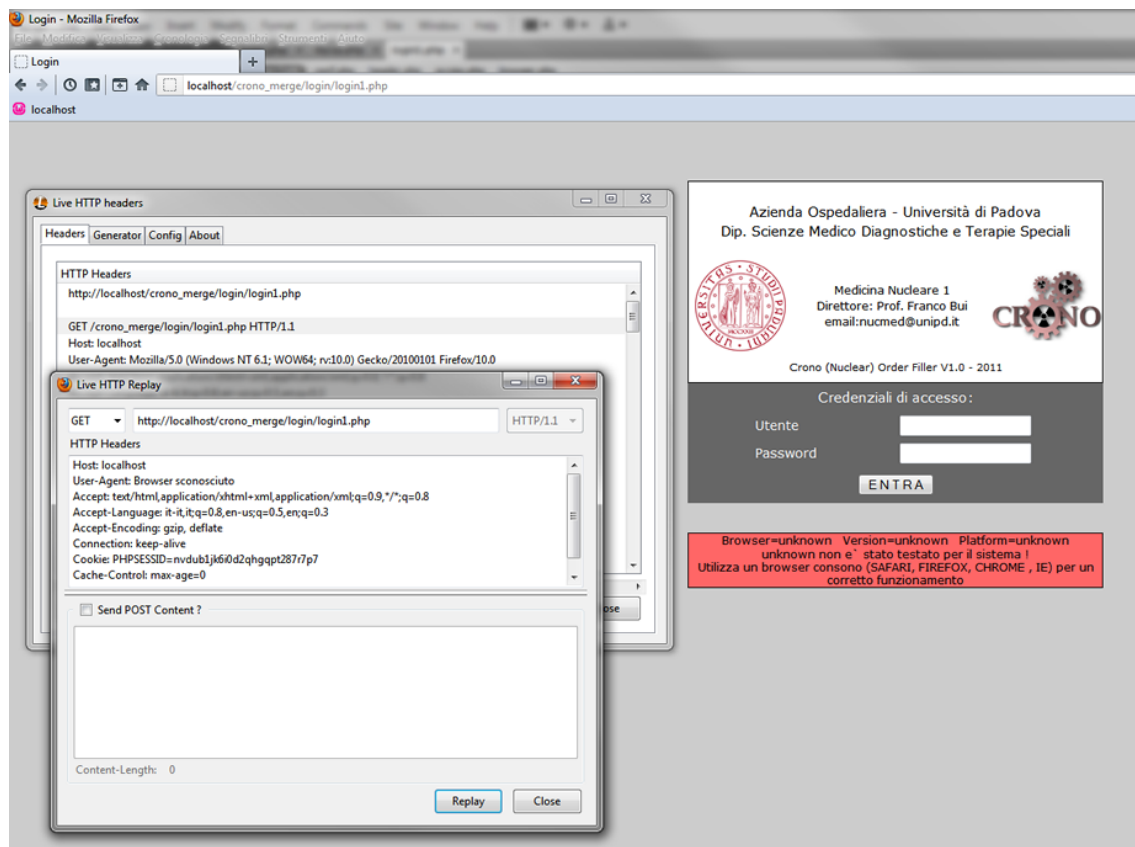


Figura 5.3. Manipolazione dell'header tramite Live HTTP Headers.

Tamper Data, scritto da Adam Judson, permette di modifica le richieste HTTP e HTTPS, inclusi gli header e i parametri in POST, direttamente a ogni richiesta. Una volta attivata l'opzione di manipolazione, il browser presenta una finestra di dialogo in cui chiede se modificare la richiesta, inviarla senza modifiche o annullarla.

HTTP Fox, di Martin Theimer, è simile ai precedenti, in quanto monitora e analizza tutto il traffico HTTP in entrata e uscita tra il browser e il web server. Le informazioni includono: header di richiesta e risposta, cookies inviati e ricevuti, parametri query e POST, corpo della risposta (più comunemente la pagina HTML o il foglio di stili CSS).

FireForce, della SCRT, è un addon che permette di eseguire attacchi a forza bruta su sistemi di autenticazione basati su form, sia in GET, che POST. Può utilizzare sia dizionari che password generate su diversi alfabeti. Tra le varie (poche) opzioni, è possibile impostare anche il numero di richieste al secondo. Per la sua semplicità, FireForce non è adatto ad attacchi sofisticati, per i quali si consiglia invece, il più completo THC-Hydra, già citato al paragrafo 2.2.2.

Infine, per il debugging dell'applicazione, è stata utilizzata un'altra estensione per Firefox, Firebug, che tra le altre cose consente di ispezionare e modificare il codice HTML, correggere in tempo reale il foglio di stile CSS, monitorare l'attività della rete, debuggare il codice Javascript, navigare il DOM, ecc.

PARTE III

RISULTATI

Capitolo 6

Autenticazione e Autorizzazione

Prima di descrivere i sistemi sviluppati, viene fatta una breve panoramica dei meccanismi attualmente in uso nel sistema informativo utilizzato presso il Policlinico Universitario di Padova, in modo da evidenziare i punti deboli di tale sistema e giustificare lo sviluppo di nuovi strumenti di autenticazione e autorizzazione.

6.1 Crono

L'attuale sistema in uso presso il Policlinico Universitario di Padova, Crono (quasi completamente realizzato in PHP), utilizza un meccanismo di autenticazione con username e password basato su form. Tale implementazione appare insufficiente a garantire un adeguato livello di sicurezza, soprattutto in virtù dei seguenti fatti: non vengono eseguiti controlli sulla password, non esistono meccanismi di blocco degli account, né viene gestita la scadenza della password come previsto dalla normativa sulla privacy [18]. Tutto ciò, unito all'assenza di un sistema di log, rende l'intera applicazione facilmente vulnerabile ad attacchi di tipo password guessing e session hijacking. Inoltre, la mancanza di un canale sicuro e di un adeguato controllo degli accessi, permette di eseguire facilmente attacchi eavesdropping.

Dal punto di vista interamente applicativo invece, la gestione degli utenti è realizzata salvando, al momento del login, tutte le variabili associate all'utente in corrispondenti variabili di sessione (inclusi i permessi), come illustrato Figura 6.1.

Sebbene in prima istanza tale sistema possa risultare adeguato, esso presenta numerosi difetti:

- Le variabili sono salvate in maniera non strutturata, rendendo difficile il loro accesso e riutilizzo.
- Le variabili sono salvate nella sessione con nomi arbitrari difficili da ricordare e altrettanto facili da sovrascrivere accidentalmente.
- Non vi è nessun legame tra le variabili nella sessione e i record nel database. Una modifica alle variabili non si rispecchia automaticamente in un aggiornamento della tabella utente.
- Non esiste nessun tipo di scope o controllo sulle variabili salvate. Ciò permette di modificare liberamente variabili utilizzate per il controllo dei permessi (come il gruppo dell'utente).
- Modifiche al database o alla logica di sistema richiedono una manutenzione e riscrittura del codice dispendiosa e impegnativa da realizzare.

- Il debugging è complesso.
- Il codice è meno leggibile.

```

$gruppo = $recx[0]['IDROLE'];
$cookie_security=1;
$_SESSION["idcode"] = $recx[0]['IDUSER'];
$fullname_passed=$recx[0]['FAMILY_NAME'].$recx[0]['GIVEN_NAME'];
$_SESSION["user_group"] = $recx[0]['IDROLE'];
$_SESSION["fullname_passed"] = "$fullname_passed";
$_SESSION["cookie_security"] = "$cookie_security";
$_SESSION["username"] = "$username";
$_SESSION["password"] = "$password_crypt";
$_SESSION["diagnostica"] = $recx[0]['PREF_DIAG'];
$_SESSION["tipoesame"] = $recx[0]['PREF_EX'];
$_SESSION["stato"] = $recx[0]['PREF_STATUS'];

// GESTIONE GRUPPI
if ($_SESSION["user_group"] == 3)
    $_SESSION["specializzando"] = $recx[0]['IDUSER'];
else
    $_SESSION["specializzando"]="";

$id_user=$recx[0][0];
$_SESSION["iduser"] = "$id_user";
$level = $recx[0][3];
$id = $recx[0][0];

// GESTIONE PERMESSI UTENTE
#####
>IDUSER_PERMESSI=$_SESSION["idcode"];
$query_permessi=("select * from PRIVILEGI where (IDUSER = '$IDUSER_PERMESSI') ");
$exec_query_permessi = query_select($query_permessi);
  
```

Figura 6.1. Porzione di codice di login del Crono con parte del popolamento dati utente.

Per tali ragioni si è pensato di rinnovare il sistema di gestione utenti utilizzando una logica applicativa a oggetti, molto più strutturata e controllabile, oltre che facilmente espandibile con l'aggiunta di nuove funzioni o classi ausiliarie.

6.2 Struttura del database per la gestione utenti

Prima di introdurre l'implementazione vera e propria, presentiamo lo schema del database per la parte riguardante la gestione degli utenti (Figura 6.2). I campi contrassegnati dalla sigla PK, si riferiscono alle chiavi primarie (tutte le tabelle utilizzano id numerici auto-incrementanti), quelli con la sigla FK si riferiscono invece alle chiavi esterne, U indica un indice univoco. Tutti i campi in grassetto sono obbligatori. Segue la descrizione delle singole tabelle:

- **users**: qui vengono raccolti tutti i dati relativi all'utente (credenziali, ruolo, dati anagrafici, ecc.). Il campo `username` deve essere univoco per evitare duplicazioni di credenziali. Tra i vari campi citiamo `psw_created` e `psw_expires`. Il primo contiene la data di creazione dell'attuale password utente e servirà a eseguire i controlli di scadenza della stessa, come stabilito dall'allegato B relativo al testo unico sulla privacy (vedi paragrafo 1.7.1); il secondo invece è un booleano che stabilisce se la password dell'utente è soggetta a scadenza (il valore di default è impostato a vero).
- **passwords**: in questa tabella vengono salvate le vecchie password (cifrate) di ogni singolo utente e la data di creazione di tale password (i.e. quando è stato effettuato il cambio password). Questa tabella non contiene la password attualmente in uso dall'utente, che invece

è salvata nella tabella `users`. Ciò è stato fatto per semplificare e velocizzare la procedura di autenticazione, evitando un dispendioso `JOIN` tra le due tabelle. Ogni password è legata al suo utente dal campo `id_user`, che fa da chiave esterna con la tabella `users`.

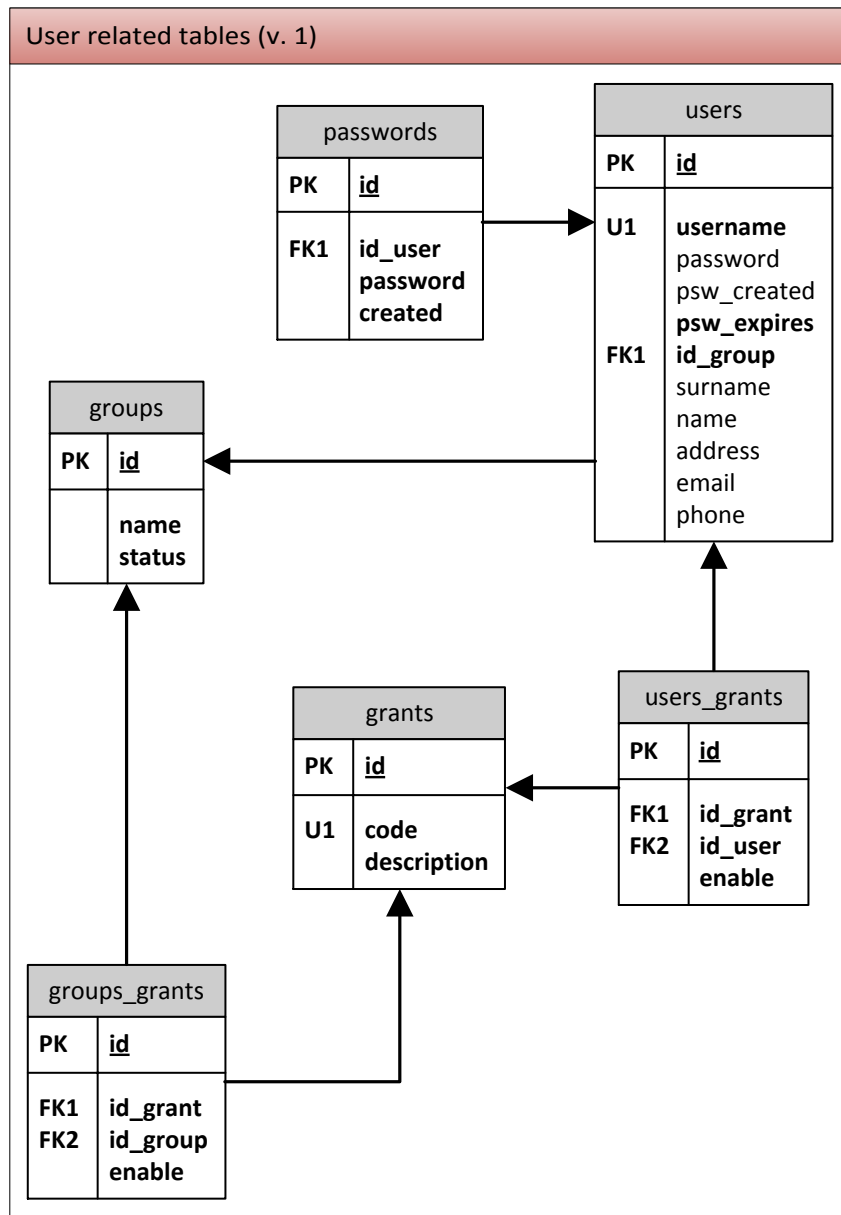


Figura 6.2. Prima versione dello schema delle tabelle utente.

- `groups`: qui sono elencati i vari ruoli (o gruppi) utente. A ogni gruppo sono associati determinati permessi, secondo il modello Role-Based Access Control, descritto al paragrafo 3.5.2. Ogni utente è associato a uno di questi gruppi tramite la chiave esterna `id_group`.
- `grants`: questa tabella elenca i singoli permessi. Ogni permesso è identificato univocamente da un codice facilmente intellegibile e memorizzabile (per esempio INSPRENO, che sta per “inserimento prenotazione”). Il sistema di autorizzazione fa riferimento a tali codici per il controllo degli accessi.
- `groups_grants`: in questa tabella ogni gruppo (chiave esterna FK2, `id_group`) è associato a ogni singolo permesso (chiave esterna FK1, `id_grant`), il quale può essere attivato o disatti-

vato per quel gruppo, tramite il campo `enable`. Questa tabella presenta sempre $N_{gruppi} \times N_{permessi}$ record.

- `users_grants`: questa tabella è analoga alla precedente, ma si riferisce ai permessi utente. Ogni singolo utente può avere degli specifici permessi in più rispetto al gruppo a cui appartiene (privilegi) o in meno (negazioni). A differenza della tabella `groups_grants` che ha sempre $N_{gruppi} \times N_{permessi}$ record, il numero di righe di questa tabella dipende esclusivamente dal numero di privilegi/negazioni assegnati agli utenti.

Tutte le relazioni tra le tabelle sono N-a-1, vale a dire che esistono, per esempio, N password per ogni utente, N utenti per ogni gruppo, N permessi di gruppo per ogni gruppo, N permessi utente per ogni permesso, ecc.

Carico dei volumi e delle operazioni

Segue una stima tecnica dei volumi delle tabelle:

- `users`: un record per utente, quindi approssimativamente un centinaio.
- `passwords`: raccoglie n_u record per utente, dove n_u è il numero di password memorizzate dell'utente u . Si ha quindi un totale di $\sum_u N_{utenti} \times n_u$ record. Ipotizzando che tutti gli utenti cambino la password ogni tre mesi (come previsto dalle norme minime di sicurezza del testo unico sulla privacy, par. 1.7.1), si ottengono $N_{utenti} \times 4$ nuovi record all'anno. Supponendo di avere all'incirca un centinaio di utenti, questo si traduce in circa 400 righe all'anno, cifra più che sostenibile. L'assunzione di avere un cambio password ogni tre mesi non ci appare eccessivamente ottimistica, in quanto l'utente solitamente, evita il più possibile di cambiare password, soprattutto se questo implica doverne inserire una diversa dalle precedenti. Supponendo comunque un cambio di password ogni mese, il valore triplicherebbe a 1200 righe l'anno. In 10 anni di attività si avrebbero quindi "solamente" 120.000 record, valore facilmente sostenibile dai moderni DBMS. Va specificato comunque, che solitamente i controlli sulle password coinvolgono al più le ultime 10, di conseguenza un'eliminazione periodica delle password più vecchie e ormai inutili, ridurrebbe ulteriormente il volume della tabella.
- `groups`: un record per gruppo. Il numero dei gruppi è spesso molto ristretto (inferiore a 10).
- `grants`: un record per permesso. È difficile stimare il numero di permessi da inserire, tuttavia si può stimare con elevata probabilità che difficilmente supererà qualche centinaio di record. Attualmente Crono adotta 31 permessi.
- `groups_grants`: come già specificato, in questa tabella sono sempre presenti $N_{utenti} \times N_{permessi}$ record. Ipotizzando un numero di utenti e di permessi sull'ordine delle centinaia, si ha un totale di 10.000 record.
- `users_grants`: una buona strutturazione dei permessi di gruppo permette di ridurre al minimo la necessità di aggiungere permessi utente, mantenendo questa tabella molto contenuta. In ogni caso, il volume di tale tabella è certamente strettamente inferiore a quella di `groups_grants`.

Segue una tabella con le operazioni più frequenti e una stima qualitativa del carico:

Operazione	Tipo	Frequenza
Reperimento informazioni utente/gruppo	Select	>100 al giorno
Cambio password	Select	≤1 al mese
Reperimento informazioni permessi	Select	>100 al giorno
Inserimento/modifica utente	Update/Insert	Raramente
Inserimento/modifica nuovo gruppo	Update/Insert	Raramente
Inserimento/modifica nuovo permesso	Update/Insert	Raramente

A parte durante la prima fase di popolamento, i successivi inserimenti e modifiche sono piuttosto rari. Se si considera un bacino di utenza di circa un centinaio di utenti, appare evidente che il maggior carico ha a che fare con l'acquisizione dei dati e dei permessi utenti, necessari al corretto funzionamento del sistema. Vedremo nel paragrafo successivo come limitare l'accesso al database utilizzando una struttura a oggetti per la gestione dei dati e dei permessi utente.

Il recupero dei permessi, è forse l'operazione più impegnativa dell'intera struttura dati, coinvolgendo ben quattro tabelle: `users`, `grants`, `groups_grants` e `users_grants`. Va inoltre eseguita in due tempi: prima va recuperato l'array di permessi legati al gruppo, quindi si controlla la tabella `users_grants` per verificare l'esistenza di privilegi o negazioni a livello di utente e, se presenti, si sovrascrivono i vecchi permessi. In particolare le due query sono:

```
SELECT * FROM groups_grants JOIN grants ON (groups_grants.id_grant = grants.id)
WHERE groups_grants.id_group = $id_group

SELECT * FROM users_grants JOIN grants ON (users_grants.id_grant = grants.id) WHERE
users_grants.id_user = $id
```

Dove `$id_group` e `$id` sono rispettivamente l'id del gruppo a cui appartiene l'utente e l'id dell'utente (entrambi dei quali sono contenuti nel record utente).

6.3 La classe User

La classe `User` rappresenta il singolo utente dall'accesso al sito, fino alla sua disconnessione. Esso è l'immagine speculare dell'entità "utente" presente nel database ed è pertanto perfettamente coerente con le informazioni in esso salvate. Per uniformarsi al vecchio sistema, si è scelto di realizzarla in linguaggio PHP.

Lo scopo della classe `User` è di fornire uno strumento in grado di accedere e manipolare i dati utente, implementando tutte le routine di accesso al database, controllo degli accessi e aggiornamento delle informazioni in maniera sicura e strutturata, e di implementare i meccanismi di autenticazione e autorizzazione richiesti. In particolare la classe implementa le seguenti funzionalità:

- Reperimento unico e strutturato delle informazioni da database (i dati vengono estratti una sola volta dal database e salvati in apposite strutture dati) e restituzione degli stessi in un formato facilmente accessibile.
- Modifica dei campi in maniera sicura, controllata e coerente con il database.
- Accesso al database tramite funzioni sicure e protette da SQL injection.
- Sistema di autenticazione sicura con username e password o certificati.
- Controllo dei permessi associati al gruppo di appartenenza dell'utente o all'utente stesso.
- Controllo della scadenza della password e sistema di modifica della stessa in accordo con le misure di sicurezza minime previste dal testo unico sulla privacy (par. 1.7.1).

- Cifratura asimmetrica della password.

6.3.1 Il file di configurazione

Per rendere quanto più flessibile e modulare l'integrazione del sistema nell'ambiente operativo del Crono, si è deciso di includere al pacchetto un file di configurazione. Tale file permette, per esempio, di impostare i parametri fondamentali di connessione al database, i nomi delle tabelle principali e molte altre opzioni di configurazione utilizzate dal sistema, come la durata di scadenza delle password, il numero di password da memorizzare, percorsi a file (certificati, chiavi), ecc. La modifica degli appropriati campi del file di configurazione, garantisce la rapida installazione dell'applicativo in qualsiasi sistema.

6.3.2 Implementazione della classe User

Nella classe `User` sono adottate le seguenti variabili di classe (che in PHP vengono dette "proprietà"):

- `private array $fields`: contiene i nomi dei campi della tabella `users`.
- `private array $data`: è l'array di dati contenente le informazioni dell'utente ricavati dalla tabella `users`. È indicizzato secondo i nomi dei campi forniti dall'array `$fields`.
- `private array $permissions`: contiene degli oggetti di tipo `Permission`, che descrivono i permessi associati all'utente (una descrizione della classe `Permission` è fornita in seguito). Gli array `$data` e `$permissions` sono la rappresentazione speculare dell'entità utente salvata nel database e, nel corso della presentazione, saranno riferiti genericamente come "dati utente".
- `private bool $logged`: si tratta di un flag che assicura il successo dell'autenticazione. Viene impostato a vero solamente da un metodo di autenticazione riuscito.

Tutte le proprietà sono impostate a `private` per garantire l'incapsulamento e la sicurezza dei dati in esse contenuti.

Di seguito sono descritte le principali funzioni della classe. L'ordine con cui vengono presentate è funzionale alla comprensione del flusso esecutivo.

public function login(\$usr, \$psw)

La funzione di `login` verifica il nome utente e la password e in caso di risultato positivo, popola l'oggetto `User` con i dati ricavati dal database, altrimenti interrompe l'esecuzione e restituisce il valore `false` (o zero). Le informazioni vengono salvati nell'array `$data` i cui indici corrispondono ai nomi dei campi della tabella utente sul database (array `$fields`). Questo permette non solo di svincolarsi da una precisa nomenclatura, ma di poter accedere ai campi della classe semplicemente conoscendo quelli della tabella. Inoltre, salvando i dati nella memoria server, si riduce notevolmente la quantità di richieste al database per accedere alle informazioni.

```
1. $row = mysql_fetch_array($result);
2. $row_size = mysql_num_fields($result);
3. $fields = array();
4. $data = array();
5. for ($i = 0; $i < $row_size; $i++) {
6.     $field_name = mysql_field_name($result, $i);
7.     array_push($fields, $field_name);
```

```

8.     $data[$field_name] = $row[$i];
9. }
10. // this call populate both data and permissions
11. $this->populateUser($fields, $data);

```

In seguito al popolamento dei dati utente (che comprende anche i permessi), viene eseguito un controllo sulla scadenza della password. In caso di successo, l'autenticazione prosegue correttamente impostando la variabile di stato `$logged` a `true` e restituendo il valore `1`. In caso contrario, l'esecuzione è interrotta e viene restituito il valore `-1`, che indica appunto, che la password è scaduta.

```

1. // check for password expiration
2. if ($this->isPasswordExpired())
3.     return -1;
4. else {
5.     $this->logged = true;
6.     return 1;
7. }

```

Per verificare quindi, se l'utente si è autenticato con successo, è sufficiente richiamare la funzione `isLoggedIn()` che restituisce il valore della flag `$logged`. L'incapsulamento e il controllo dell'accesso privato della variabile, assicura l'autenticità del valore restituito da quel'ultimo metodo.

public function populateUser(\$fields, \$data)

Viene chiamato dai metodi di autenticazione per assegnare le informazioni ottenute dal database agli array di dati utente. È anche responsabile di chiamare il metodo `populatePermissions()`, che, in maniera analoga ai dati, assegna i permessi associati a quell'utente. Il metodo è stato deliberatamente impostato a `public` per permettere ad altre classi (come `UserManager`, di cui si parlerà nel paragrafo 6.5) di creare e popolare liberamente oggetti di tipo `User`.

```

1. public function populateUser($fields, $data) {
2.     $this->fields = $fields;
3.     $this->data = $data;
4.     $this->populatePermissions();
5.     $this->logged = false;
6.     return $this;
7. }

```

Non a caso la flag `$logged` è mantenuta comunque a `false` (riga 5) per garantire l'accesso solo tramite i metodi di autenticazione.

Grazie a questa funzione, i dati utente vengono salvati nell'oggetto stesso, pronti per essere letti senza dover accedere continuamente al database.

private function populatePermissions()

Come già anticipato, lo scopo di questa funzione è di assegnare all'array di permessi utente i valori restituiti dal database. L'array `$permissions` non rappresenta una semplice stringa di bit, bensì racchiude al suo interno istanze di oggetti `Permission` (vedi schema in Figura 6.3).

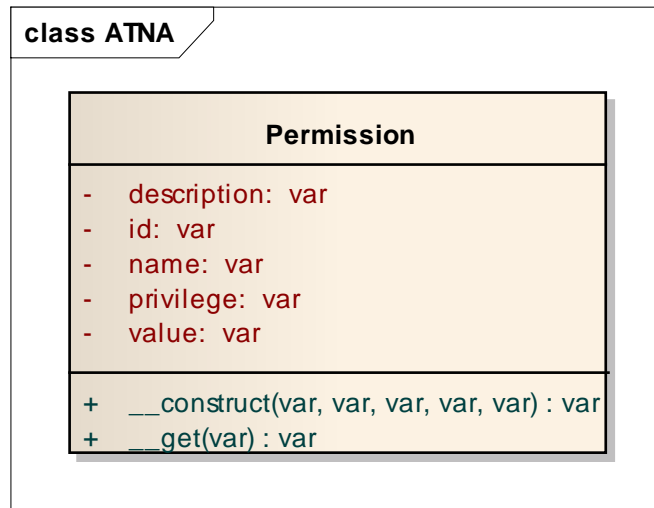


Figura 6.3. Schema UML della classe Permission.

La classe `Permission` rappresenta il concetto astratto di permesso e svolge il compito di descrittore del permesso. Essa, infatti, implementa solamente due metodi: il costruttore e il magic method `__get`. Il principale vantaggio di adottare questa classe è dato dalla strutturazione dei dati relativi ai permessi. In essa sono memorizzati cinque attributi principali: l'id del permesso (un intero positivo utilizzato per l'indicizzazione nel database), il suo nome univoco, il valore associato all'utente (vero o falso), una breve descrizione del permesso e se il valore del permesso in questione è legato al ruolo a cui appartiene l'utente o se si tratta di un privilegio/negazione specifica dell'utente (per maggiori dettagli fare riferimento al paragrafo 6.2). Il metodo, infatti, è in grado di distinguere i permessi utente, da quelli di gruppo ed eseguire le necessarie modifiche all'array di permessi.

```

1. // overwrite existing permission only if they are different
2. if ($this->permissions[$row[1]]->value != $row[3])
3.     $this->permissions[$row[1]] = new Permission($row[0], $row[1],
        $row[2], $row[3], true);
    
```

Si fa notare che, nel caso venga trovato un permesso a livello utente, la sovrascrittura del relativo permesso a livello di gruppo avviene solamente in caso di variazione dell'attuale permesso (riga 2). Questo per evitare inutili sprechi di memoria. Infatti, la sovrascrittura avviene ricreando l'oggetto `Permission` da modificare. La scelta di ricreare l'oggetto piuttosto che modificarlo è dettata da un'unica ragione: la sicurezza. L'oggetto `Permission`, come già specificato, non implementa nessun metodo di modifica dei suoi campi, che sono tutti ad accesso privato. Pertanto, l'unico modo per modificare un oggetto `Permission` è rifarlo.

Considerato lo scarso numero di privilegi/negazioni utente (solitamente non più di 2-3 per utente), l'incremento di sicurezza giustifica il compromesso.

È possibile determinare se un permesso è assegnato a livello di gruppo o di utente utilizzando il metodo pubblico `isUserPermission($perm)`, con parametro il nome univoco del permesso.

```

1. public function isUserPermission($perm) {
2.     return $this->permissions[$perm]->privilege;
3. }
    
```

Per il semplice controllo del permesso invece, si utilizza un metodo analogo: `hasPermission($perm)` che restituisce vero o falso a seconda del valore del permesso richiesto. Il metodo `getPermissions()` invece, restituisce triplette con i valori (permesso, descrizione, valore) associati ai permessi utente.

```

1. public function getPermissions() {
2.     $ret['permission'] = array();
3.     $ret['description'] = array();
4.     $ret['value'] = array();
5.     foreach ($this->permissions as $perm) {
6.         array_push($ret['permission'], $perm->name);
7.         array_push($ret['description'], $perm->description);
8.         array_push($ret['value'], $perm->value);
9.     }
10.    return $ret;
11. }

```

Per garantire l'incapsulamento, in nessun caso viene restituito l'oggetto `Permission`, ma solamente i suoi attributi.

private function isPasswordExpired()

Questo metodo, chiamato dalla funzione di login, verifica, in seguito ad un'autenticazione riuscita, se la password inserita è ancora valida. La durata di validità della password è impostabile dal file di configurazione e quindi facilmente adattabile a ogni esigenza. Per esempio, impostando la durata di validità a zero, le password non scadranno mai. Inoltre, la scadenza della password può essere ulteriormente impostabile a livello di utente singolo tramite l'apposito campo della tabella utente, `psw_expires` (vedi paragrafo 6.2).

public function changePassword(\$newPsw)

Il cambio di password è gestito in modo tale da fornire un controllo sulle n precedenti password utilizzate dall'utente. Il valore di n è comodamente impostabile nel file di configurazione. Questo accorgimento impedisce all'utente di utilizzare come nuova password una delle ultime n già utilizzate.

```

1. $psws = $this->getPasswords();
2. $newPsw = $this->encrypt($newPsw);
3. // new password equals one of the previous
4. if (in_array($newPsw, $psws))
5.     return -1;

```

I valori restituiti dalla funzione sono -1 se la password scelta corrisponde a una delle precedenti, 0 in caso di errore, 1 se il cambio password è andato a buon fine.

Le precedenti password utilizzate dall'utente sono immagazzinate nel database dopo ogni cambio password e restituite, per scopi di controllo, dal metodo privato `getPasswords()`. In particolare, quest'ultimo metodo, restituisce solamente le ultime n password, dove n è il valore impostato nel file di configurazione.

```

1. $psws[0] = $this->password;
2. // store the last n-1 password of the user
3. $query = "SELECT password FROM ".CFG_PASSWORDS_TABLE." WHERE id_user =
    $this->id ORDER BY created DESC LIMIT ".(CFG_DIFFERENT_PSW-1).".";";
4. if($result = query($query)) {
5.     while ($row = mysql_fetch_array($result))
6.         array_push($psws, $row[0]);
7. }

```

Si fa notare, a riga 3, che la query limita la ricerca a $n-1$. Questo perché nella tabella `passwords` vengono salvate solamente le precedenti password utilizzata dall'utente, mentre quella corrente è salvata nella tabella `users`. Non a caso, a riga 1, il primo indice dell'array delle password viene popo-

lato con la password utente attualmente in uso. La password attuale va aggiunta all'array per evitare che un utente possa riutilizzare la password corrente. Quindi, quando si parla di n password precedenti, si intende $n-1$ vecchie più la password corrente.

public function __get(\$field)

In PHP, tutti i metodi che iniziano con un doppio underscore vengono definiti magic method. I magic method sono funzioni che hanno un comportamento standard in risposta a particolari eventi PHP. Essi possono essere ridefiniti per ottenere dei comportamenti diversi, adeguati alle proprie esigenze. Uno di questi metodi è appunto `__get(string $name)`, che permette l'accesso a proprietà non ancora note come se fossero state definite fin dall'inizio. Il metodo viene utilizzato anche per caricare proprietà al volo o utilizzare dei campi, ricavati in seguito a delle operazioni, come proprietà.

```

1. public function __get($field) {
2.     if (!array_key_exists($field, $this->data))
3.         return null;
4.     return $this->data[$field];
5. }
```

In questa classe, il metodo `__get($field)` permette di accedere a uno qualunque dei campi dell'array `$data`, popolato in fase di autenticazione, semplicemente posponendo all'oggetto `User`, il nome del campo. Come visto in precedenza, i dati utente salvati nell'oggetto, sono indicizzati usando esattamente i nomi dei campi estratti dalla tabella utente del database, quindi per accedere, per esempio, al campo `username`, è sufficiente scrivere `$user->username`, assumendo che `$user` sia un oggetto di tipo `User`. In questo modo esiste una reciproca correlazione tra l'oggetto utente e il record salvato nel database, facilitando la migrazione del codice ad altri database. Il seguente codice, per esempio, assegna alla variabile `$name` la concatenazione di nome e cognome dell'utente rappresentato dall'oggetto `$user` (supponendo che `$user` sia un oggetto `User` che si è autenticato con successo e pertanto popolato i suoi dati):

```
$name = $user->name . " " . $user->surname;
```

Alternativamente è possibile ottenere l'intero array `$data` utilizzando il metodo pubblico `getData()`. Questa funzione restituisce le informazioni in un array bidimensionale in cui a ogni colonna è associata una coppia (campo, valore):

CAMPO	CAMPO	CAMPO	CAMPO	...
VALORE	VALORE	VALORE	VALORE	...

Poiché il sistema crea dinamicamente l'array dei dati in base alla tabella utente utilizzata nel database, è anche possibile ottenere il numero di campi tramite il metodo pubblico `getNoFields()`.

public function __set(\$field, \$value)

Analogamente al magic method `__get($field)`, ne esiste uno anche per impostare il valore di una proprietà: il metodo `__set($field, $value)`. Ridefinendo il metodo, si può quindi decidere come modificare degli attributi che normalmente non sono accessibili o non noti a priori. Il metodo `__set($field, $value)` permette di modificare i valori dell'array di dati (e solo l'array di dati) accessibile con il magic method `__get($field)` semplicemente creando un'assegnazione, come la seguente (come sempre si assume che la variabile `$user` rappresenti un'istanza della class `User`):

```
$user->email = "me@email.com";
```

Come il metodo `__get($field)`, il principale vantaggio è quello di poter modificare campi che non conosciamo a priori; vantaggio necessario all'implementazione dinamica adottata, che non permette di conoscere in partenza i nomi dei campi. Tuttavia presenta un notevole svantaggio: permette la modifica di qualsiasi campo dell'array di dati. Ci sono dei valori che non conviene poter modificare liberamente, come per esempio l'id utente, l'id del gruppo, il nome utente, la password (bisogna passare attraverso i controlli e la cifratura). In questo caso è stato predisposto, nel file di configurazione, un'impostazione che permette di specificare quali campi (nomi) rendere accessibili solamente in lettura. La seguente routine, utilizza il metodo `isEditable($field)` per bloccare tutti i tentativi di modifica di un campo in sola lettura:

```
1. if (!$this->isEditable($field))
2.     throw new Exception("Attempt to modify a read-only field.");
```

Il metodo `isEditable($field)`, controlla le impostazioni del file di configurazione per verificare se il campo inserito fa parte dei campi in sola lettura e restituisce vero o falso a seconda del risultato:

```
1. if (in_array($field, unserialize(CFG_READ_ONLY)))
2.     return false;
3. return true;
```

Se tutti i controlli sono andati a buon fine e il campo selezionato è modificabile, viene aggiornato non solo l'array di dati dell'oggetto utente, ma il record corrispondente nel database. In questa maniera vi è una continua corrispondenza tra l'oggetto e il record utente.

Cosa succede se viene assegnato un valore ad un campo inesistente?

```
$user->not_existing_field = "some_data";
```

Il metodo implementa una routine che permette di assegnare nuovi campi all'utente. Questi campi vengono aggiunti alla normale collezione di campi presenti nell'array `$data`, ma, a differenza di essi, che sono creati e popolati in seguito all'autenticazione, questi ultimi sono aggiunti "manualmente" con assegnazioni simili alla precedente. Quando non viene trovato il campo, il metodo `__set($field, $value)` chiama un altro metodo ausiliario `newField($field, $value)` che post-pone un nuovo campo:

```
1. if (!array_key_exists($field, $this->data)) {
2.     $this->newField($field, $value);
3.     return;
4. }
```

Dall'assegnazione in poi, questo nuovo campo si comporta a tutti gli effetti come un qualsiasi altro campo utente, ma la sua vita si limita alla durata della sessione. Infatti, a differenza dei campi popolati dal database, in fase di autenticazione, quelli aggiunti con il metodo `__set($field, $value)` non trovano corrispondenza nel database e pertanto non vanno a modificare la struttura del record in esso contenuto. Essi possono quindi essere utilizzati per tenere traccia di variabili temporanee il cui utilizzo (e vita) si esaurisce con la disconnessione dell'utente dall'applicazione.

6.3.3 Sicurezza

Durante la spiegazione delle varie funzioni, si sono fatti spesso riferimenti alla sicurezza. Ma che aspetti della sicurezza sono a rischio e quali sono i possibili punti deboli del sistema? Sicuramente il primo elemento a rischio è l'autenticazione. Prima di tutto bisogna chiedersi a che tipi di attacco

può essere soggetta la funzione di autenticazione, quindi analizzare e valutare le contromisure adottate.

SQL injection

Prima fra tutte, va menzionata la vulnerabilità SQL injection. Il metodo `login($usr, $psw)` riceve due parametri in ingresso, il nome utente e la password. Queste probabilmente saranno passate dall'utente attraverso una form. La password viene cifrata dal metodo prima di essere confrontata con i risultati della query al database, quindi il campo password è sufficientemente sicuro dagli attacchi. Cosa si può dire del campo username invece? Abbiamo già visto nel paragrafo 2.3.3.1 come sia facile bypassare l'autenticazione anche con un campo cifrato, semplicemente concatenando operatori booleani o usando i trattini doppi per commentare. Fortunatamente, PHP fornisce dei metodi per eseguire l'escape automatico dei caratteri potenzialmente pericolosi, come appunto gli apici singoli o doppi. Uno di questi metodi è il `mysql_real_escape_string(string $unescapeped_string)`, che riceve una stringa in ingresso e ne restituisce la controparte "escaped". Tutte le funzioni che ricevono dati utente in ingresso e operano sul database, contengono una porzione di codice simile al seguente, per evitare il SQL injection:

```
1. if (CFG_ADD_SLASHES) {
2.     $usr = mysql_real_escape_string($usr);
3.     $psw = mysql_real_escape_string($psw);
4. }
```

Il condizionale di riga 1, controlla un'impostazione del file di configurazione che, se impostato a vero, esegue l'escape delle variabili "pericolose". La necessità di adottare questo accorgimento è legato al fatto che in molte applicazioni vecchie si fa uso di una speciale procedura interna a PHP, detta *Magic Quotes*. Quando la direttiva Magic Quotes è attiva, tutti i dati in arrivo agli script PHP, subiscono automaticamente l'escape. Questo significa che un'applicazione con il Magic Quotes attivo, che fa uso del metodo `mysql_real_escape_string()`, vedrà applicare due volte l'escape ai propri input. Ecco perché è sorta la necessità di rendere possibile disattivare l'escaping da file di configurazione. In realtà, la comunità di PHP, sconsiglia l'utilizzo dei Magic Quotes, preferendo l'escaping mirato a ogni necessità. Non a caso, questa caratteristica è stata deprecata con la versione 5.3 di PHP e sarà completamente rimossa con la versione 6.

Session fixation

Ora che abbiamo scongiurato il pericolo più grande, vediamo un altro attacco altrettanto pericoloso: il session fixation. Come abbiamo ampiamente discusso nel paragrafo 2.3.1.4, il session fixation si basa sulla fondamentale debolezza del sistema di non rigenerare a ogni autenticazione, i token di sessione. Ancora una volta PHP ci viene incontro fornendo una comoda funzione che rigenera l'id di sessione: `session_regenerate_id($delete_old_session)`. Non solo, passandogli `true` come parametro si può anche eliminare del tutto la vecchia sessione, cancellando ogni dato memorizzato. È quindi sufficiente inserire il comando prima di ogni tentativo di autenticazione. La Figura 6.4 illustra un tentativo fallito di session fixation: notare come i due id di sessione evidenziati nella finestra del programma Live HTTP headers differiscano (in quella in alto vale `PHPSESSID=u8hd8qo497kms4rbain9hu08v2`, mentre in quella in basso vale `PHPSESSID=np13e7s254b6codos2iu2n1387`) rendendo inutile ogni tentativo di attacco. L'attaccante cerca di fissare l'id di sessione (figura in alto) inserendo delle credenziali a caso nella form (anche vuote). La

vittima accede correttamente al sistema con le sue credenziali, ma il suo id di sessione è stato rigenerato, vanificando il tentativo di intrusione dell'attaccante.

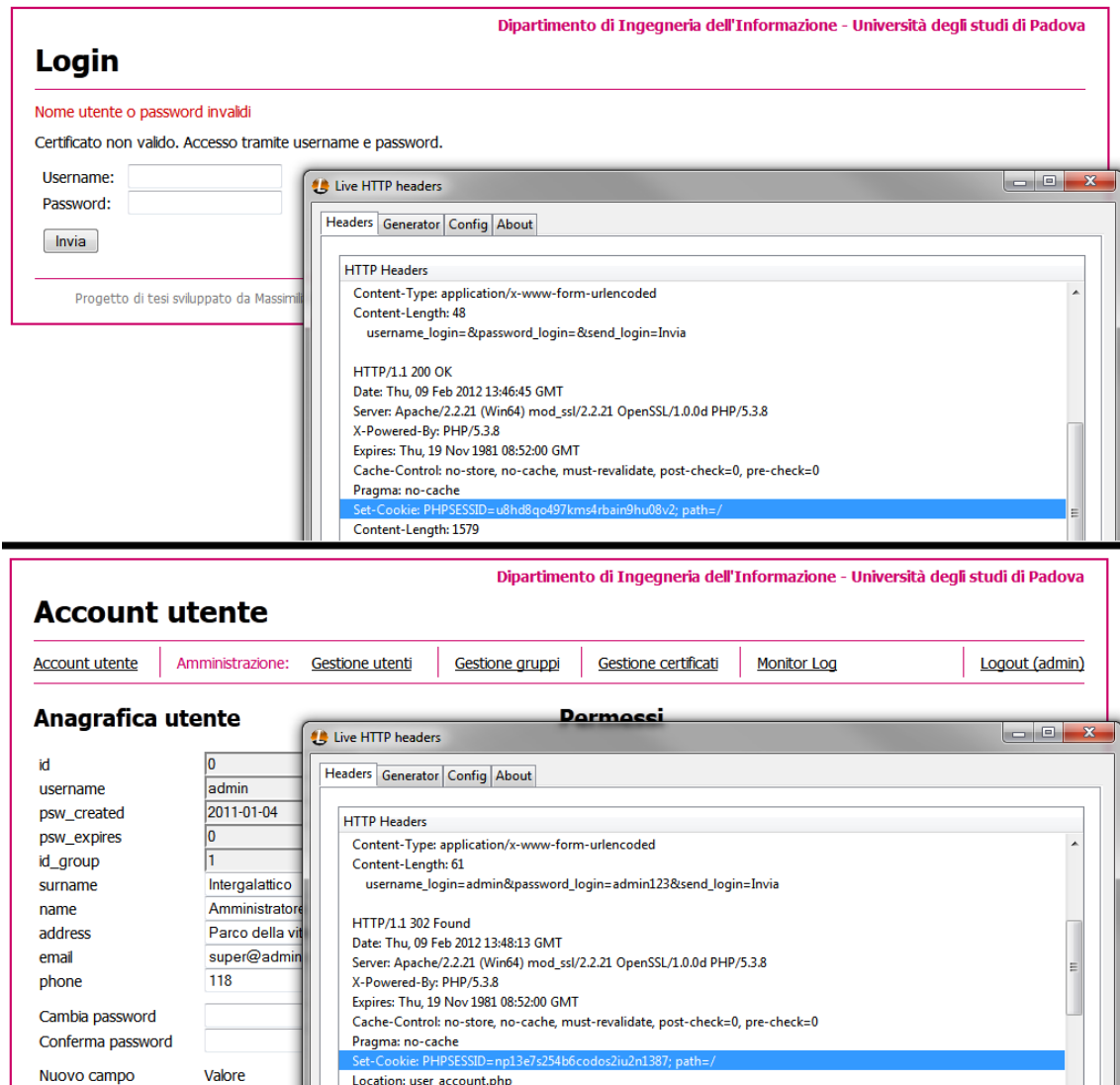


Figura 6.4. Tentativo di session fixation fallito.

Eavesdropping

Bisogna chiedersi se le informazioni di autenticazione viaggiano in chiaro anche sulla rete. In sostanza, il sistema è al sicuro da attacchi di eavesdropping? Sempre in Figura 6.4 è possibile riconoscere la seguente riga:

```
Server: Apache/2.2.21 (Win64) mod_ssl/2.2.21 OpenSSL/1.0.0d PHP/5.3.8
```

La parte importante è quella centrale che fa riferimento al modulo di cifratura SSL di Apache fornito dalla libreria OpenSSL (in particolare versione 1.0.0d). Questa informazione tuttavia, ci dice solamente che sono presenti tali moduli sul server, ma non ci garantisce che i dati inviati siano stati cifrati. Per verificarlo è sufficiente visionare il log del web server, come descritto al paragrafo 4.1.1. Se sono presenti dei record nel log SSL e non in quello HTTP, allora significa che i nostri dati sono al sicuro. Il nostro log SSL si presenta così:

```
1. 127.0.0.1 - - [09/Feb/2012:14:40:31 +0100] "GET /atna/www/login.PHP HTTP/1.1"
200 1496
```

```
2. 127.0.0.1 - - [09/Feb/2012:14:40:45 +0100] "POST /atna/www/login.PHP
HTTP/1.1" 200 1579
3. 127.0.0.1 - - [09/Feb/2012:14:47:31 +0100] "GET /atna/www/login.PHP HTTP/1.1"
200 1496
4. 127.0.0.1 - - [09/Feb/2012:14:48:13 +0100] "POST /atna/www/login.PHP
HTTP/1.1" 302 1496
5. 127.0.0.1 - - [09/Feb/2012:14:48:14 +0100] "GET /atna/www/user_account.PHP
HTTP/1.1" 200 9378
6. 127.0.0.1 - - [09/Feb/2012:15:13:18 +0100] "GET
/atna/www/user_account.PHP?logout HTTP/1.1" 200 1643
```

Mentre quello HTTP non presenta alcun riferimento, quindi le trasmissioni sono cifrate correttamente. In particolare, analizzando il log possiamo ripercorrere i passi del tentato attacco (riga per riga):

1. Alle ore 14:40:31 l'attaccante accede alla pagina di login.
2. Alle ore 14:40:45 l'attaccante invia dei dati in POST che restituiscono una risposta 200. La risposta 200 non è un redirect, quindi significa che l'utente è stato rispedito alla pagina di login invece di essere reindirizzato alla pagina di account. In breve, si tratta di un tentativo fallito di autenticazione.
3. Alle ore 14:47:31 la vittima accede alla pagina di accesso login, dopo aver ricevuto dall'attaccante l'URL del sito, con il suo id di sessione fissato.
4. Alle ore 14:48:13 la vittima, ignara del tentativo di attacco, accede al sito con le sue credenziali. Siamo sicuri del successo, dalla risposta 302 del server, ossia un redirect alla pagina account.
5. Questa riga ci conferma esattamente quanto detto sopra: è stata restituita la pagina di account.
6. Dopo circa mezzora di attività, la vittima si disconnette dal sito attivando la procedura di logout.

Username enumeration e password guessing

Ora che abbiamo dimostrato la sicurezza ai tre attacchi potenzialmente più pericolosi, vediamo come se la cava il sistema contro altri due tipi di attacchi: username enumeration e password guessing. Sul primo siamo piuttosto tranquilli: in caso di autenticazione fallita il sistema fornisce sempre lo stesso messaggio di errore, "Nome utente o password invalidi"; non è presente nessuna procedura di registrazione in quanto gli utenti vengono creati dall'amministratore e non esiste nessuna procedura di recupero della password. Si potrebbero tentare degli attacchi temporizzati, ma anche in questo caso non si riscontrano differenze tali da suggerire l'esistenza di un nome utente.

Riguardo al password guessing, siamo altrettanto confidenti. La normativa sulla privacy impone, infatti, una password composta di almeno otto caratteri alfanumerici. Questo significa che un attacco a forza bruta limitato a password di esattamente otto caratteri, richiederebbe più di 2.000 miliardi di tentativi (per la precisione 2.821.109.907.456). Supponendo che un server sia in grado di servire 2000 richieste al secondo (valore decisamente ottimistico per la maggior parte delle applicazioni), ci vorrebbero comunque più di 40 anni. Utilizzando dei dizionari si può ridurre drasticamente questo numero, ma si limiterebbe a password comunemente usate e facilmente indovinabili (assunzione spesso vera, purtroppo). Forzare gli utenti a inserire password più elaborate può aiutare in questo senso. Naturalmente esagerare tale sofisticazione finisce col causare l'effetto contrario: l'utente, frustrato dalla complessità della password, finirà col scriverla su un post-it appeso alla parete dell'ufficio o in qualche altro posto accessibile.

Data la natura dell'ambiente ospedaliero, dove le emergenze sono all'ordine del giorno e la necessità di agire in tempi rapidi è fondamentale, non riteniamo opportuno implementare un sistema di blocco degli account che possa causare l'interruzione del flusso operativo sanitario.

Privilege escalation

La sicurezza dell'autorizzazione è strettamente legata a quella dell'autenticazione: se si impedisce a un attaccante di rubare le credenziali di un altro utente (magari con poteri amministrativi), si riduce il rischio di privilege escalation. Inoltre, essendo i permessi salvati in una struttura dati memorizzata nella memoria del server (la sessione), la modifica di tali informazioni non è realizzabile, diversamente da quanto avviene coi cookie di sessione (vedi paragrafo 3.2).

6.4 Implementazione dei certificati

La sicurezza data da due fattori (qualcosa che si possiede e qualcosa che si conosce) permette di ridurre il rischio associato all'uso di password troppo semplici o a tecniche di ingegneria sociale.

La procedura completa per realizzare un sistema di mutua autenticazione client-server si può riassumere nei seguenti passi:

- Impostare OpenSSL per svolgere la funzione di autorità certificante (CA, Certificate Authority).
- Creare una chiave privata per l'autorità certificante.
- Creare un certificato server firmato con la chiave del punto precedente.
- Configurare Apache.
- Creare i certificati client firmati con la chiave CA.
- Installare i certificati client sugli host.

Vediamo passo dopo passo i singoli punti.

Impostare OpenSSL

OpenSSL utilizza un file di configurazione, chiamato `openssl.cnf`, in cui vengono definiti i parametri standard per l'utilizzo delle funzioni fornite.

```
[ CA default ]
dir           = C:/wamp/bin/apache/Apache2.2.21/ssl/private
certs        = $dir/certs
crl_dir      = $dir/crl
database     = $dir/index.txt
new certs dir = $dir/newcerts
certificate   = $dir/ca.crt
serial       = $dir/serial
crlnumber    = $dir/crl/crlnumber
crl          = $dir/crl/crl.pem
private_key   = $dir/ca.key
```

In questa sezione di configurazione, vanno inserite le directory in cui salvare i certificati, le chiavi private, la CRL (Certificate Revocation List) e altri file utili alle funzioni OpenSSL.

Dopo aver impostato le varie directory, bisogna definire le politiche di accettazione:

```
[ policy match ]
countryName   = match
stateOrProvinceName = match
```

```
organizationName      = match
organizationalUnitName = optional
commonName            = supplied
emailAddress          = optional
```

Questa sezione definisce quali campi del certificato devono corrispondere affinché sia considerato valido. Nel nostro caso devono combaciare lo stato, la provincia e l'organizzazione. Vedremo più avanti che l'unità organizzativa (`organizationalUnitName`) deve restare opzionale in quanto verrà utilizzata per definire il ruolo associato all'utente. Il `commonName` definisce il nome del proprietario del certificato. L'email è facoltativa.

In aggiunta a quanto detto, `openssl.cnf` fornisce altri parametri di configurazione tra cui la durata di default di un certificato, di una CRL, il numero di bit da utilizzare per la cifratura, i valori standard per il DN (Distinguished Name), ecc.

Creare la chiave per il CA

Ora che OpenSSL sa dove si trova il CA, lo creiamo per poterlo in seguito usare nella firma dei certificati client e server. Quindi ci spostiamo nella cartella `private` dentro Apache (dove andiamo a salvare tutto) e digitiamo:

```
1. openssl genrsa -out ca.key
2. openssl req -new -key ca.key -out ca.csr
3. openssl x509 -req -days 365 -in ca.csr -signkey ca.key -out ca.crt
4. echo FACE > serial
5. touch index.txt
6. openssl ca -gencrl -out crl/ca.crl -crldays 365
```

Vediamo in dettaglio i singoli comandi:

1. Genera la chiave privata per il CA.
2. Genera il CSR (Certificate Signing Request) del CA utilizzando la chiave creata in precedenza.
3. Si firma il CSR con la chiave privata, restituendo il certificato. Il parametro `x509` determina il tipo di formato del certificato, mentre `-days 365` indica la durata del certificato (un anno).
4. Si crea un file che verrà utilizzato come primo numero seriale per i certificati client. Ogni successivo certificato avrà un seriale incrementale. Questo seriale può essere un qualsiasi numero esadecimale di quattro cifre. Noi abbiamo scelto il valore FACE, che in decimale vale 64206.
5. Si crea un file di testo che verrà utilizzato dalla libreria OpenSSL per tracciare tutti i certificati creati e revocati dal sistema. Non a caso questo file viene detto database del CA.
6. Si crea una lista di revoca dei certificati (CRL), utile in futuro per revocare certificati persi, rubati o manomessi. Il parametro `-crldays` determina la durata di validità della lista.

Tutti questi file vanno salvati in un luogo sicuro, non accessibile e possibilmente protetto da password, visto che una qualsiasi manomissione vanificherebbe l'intero sistema.

Creare il certificato server

A questo punto la procedura di creazione del certificato server è semplice e veloce:

```
1. openssl genrsa -out apache.key
2. openssl req -new -key apache.key -out apache.csr
3. openssl ca -in apache.csr -cert ca.crt -keyfile ca.key -out apache.crt
```

1. Si crea la chiave privata per il certificato server.
2. Si crea il CSR del server utilizzando la chiave creata in precedenza.
3. Si firma il CSR con il certificato e la chiave privata del CA, ottenendo il .crt server.

Ancora una volta si suggerisce di salvare chiave e certificato in un luogo sicuro.

Configurare Apache

Adesso che abbiamo tutto, va configurato Apache perché riconosca il nuovo metodo di autenticazione. Apriamo il file httpd-ssl.conf e andiamo a modificare il VirtualHost:

```
<VirtualHost _default_:443>
1.  General setup for the virtual host
DocumentRoot "C:/wamp/www"
ServerName localhost:443
ServerAdmin server@mail.com
ErrorLog "logs/ssl_error.log"
TransferLog "logs/ssl_access.log"

2.  SSL Engine Switch:
SSLEngine on

...

3.  Server Certificate:
SSLCertificateFile "ssl/private/apache.crt"

4.  Server Private Key:
SSLCertificateKeyFile "ssl/private/apache.key"

...

5.  Certificate Authority (CA):
SSLCACertificateFile "ssl/private/ca.crt"

6.  Certificate Revocation Lists (CRL):
SSLCARevocationFile "ssl/private/crl/ca.crl"

7.  Client Authentication (Type):
SSLVerifyClient require
SSLVerifyDepth 1

8.  Access Control:
<Location />
SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
              and %{SSL_CLIENT_S_DN_O} eq "University of Padua" )
</Location>

...

9.  Per-Server Logging:
CustomLog "logs/ssl_request.log" \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>
```

Senza scendere troppo nel dettaglio, descriviamo i vari parametri:

1. Qui viene definita la directory in cui applicare il canale sicuro, il nome del server, la mail dell'amministratore e soprattutto i file di log SSL.
2. Attiva il motore SSL. Senza di questo il canale non viene cifrato.
3. Specifica la directory del certificato server.
4. Specifica la directory della chiave privata del server.
5. Specifica la directory del CA.
6. Specifica la directory del CRL.

7. Indica se l'autenticazione client con certificato è opzionale o necessaria. Se impostata a necessaria, un client senza certificato valido, che cerchi di accedere alla document root definita al punto 1, si vedrà comparire una pagina di errore (vedi Figura 6.5). Il parametro `SSLVerifyDepth` invece, determina la profondità della catena di certificati firmati. Nel nostro caso di certificato auto-firmato, il valore può essere lasciato a 1. Nel caso di certificati acquistati, questo deve essere aumentato adeguatamente.
8. Permette di filtrare i certificati da accettare in base a diversi parametri, combinabili tra loro con operatori booleani. I filtri possono essere espressioni regolari sul certificato (per esempio non vuoto), controlli sul nome dell'organizzazione o dell'unità organizzativa, giorno o orario, indirizzo IP, ecc. I campi del certificato che non vengono filtrati da questa direttiva, passano alle politiche di accettazione OpenSSL descritte in precedenza. Sarebbe buona norma, tuttavia, impostare delle rigide regole di controllo degli accessi.
9. Descrive il tipo di informazioni da salvare nel log SSL.

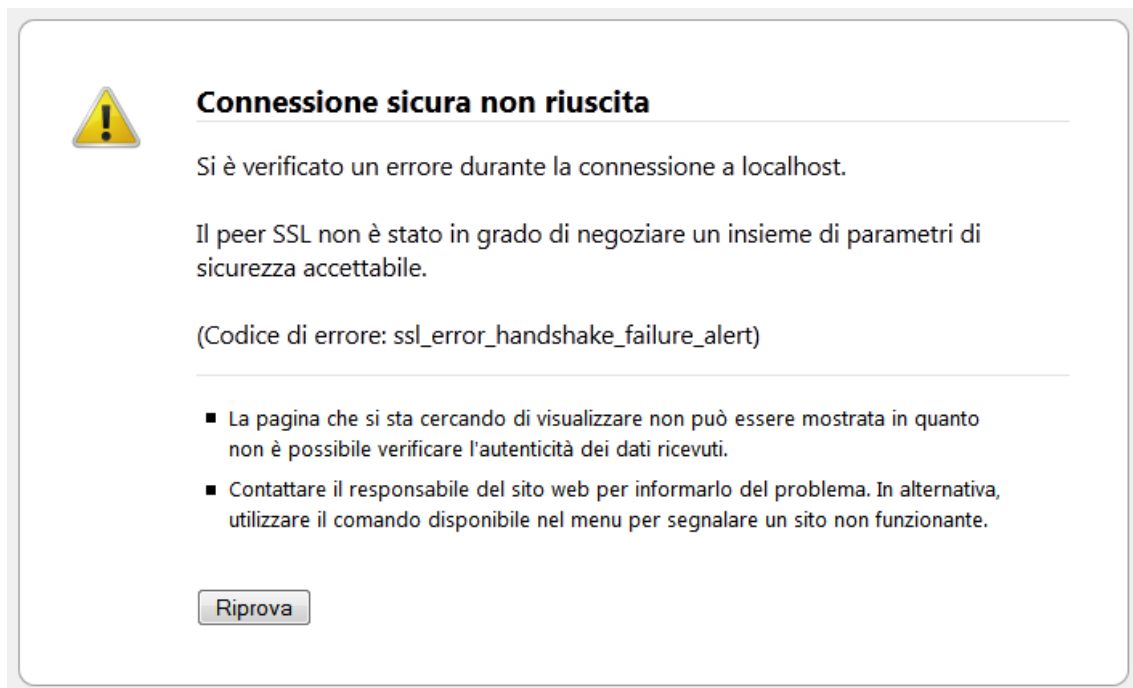


Figura 6.5. Risposta del server a un tentativo di connessione senza certificato.

A questo punto si riavvia il server Apache e, se tutto è stato impostato correttamente e non vi sono errori di sintassi nel file, il server vieterà ogni richiesta priva di certificato (Figura 6.5).

Creare i certificati client

Non ci resta quindi che creare i certificati per gli utenti che dovranno utilizzare la nostra applicazione. Torniamo nella nostra cartella `ssl`, apriamo il terminale e digitiamo i seguenti comandi:

```
1. mkdir -p users/%commonName%/
2. openssl genrsa -des3 -out users/%commonName%/commonName%.key 1024
3. openssl req -new -key users/%commonName%/commonName%.key -out
   users/%commonName%/commonName%.csr
4. openssl ca -in users/%commonName%/commonName%.csr -cert ca.crt -keyfile ca.key
   -out users/%commonName%/commonName%.crt
5. openssl pkcs12 -export -clcerts -in users/%commonName%/commonName%.crt -inkey
   users/%commonName%/commonName%.key -out users/%commonName%/commonName%.p12
```

Segue il significato dei comandi:

1. Creiamo una struttura di cartelle `users/%nome_utente%` dove salvare le coppie (chiave, certificato) di ogni singolo utente. È importante creare un'organizzazione ordinata dei file quando si hanno centinaia di utenti.
2. Si genera la chiave client. Il parametro `-des3` indica il tipo di cifratura, mentre il valore numerico `1024` determina il numero di bit utilizzati.
3. Si crea la CSR come già descritto nei precedenti casi.
4. Si firma il CSR con il CA.
5. Questa procedura serve a convertire il certificato client in un formato browser compatibile, PKCS12 per la precisione. Il formato PKCS12 contiene in un unico file sia il certificato, che la chiave privata. Per tale ragione, dopo aver eseguito il comando, viene chiesta una password di esportazione. Tale password servirà all'utente per importare il certificato nel proprio browser.

Il nuovo certificato `.p12` va quindi consegnato all'utente (tramite un mezzo sicuro) il quale dovrà conservarlo accuratamente.

L'operazione inversa alla creazione di un nuovo certificato client, è la revoca, che permette di negare prematuramente l'accesso a un certificato non ancora scaduto, aggiungendolo alla lista di revoca CRL. Per farlo è sufficiente andare nella solita cartella "private" e digitare i comandi:

```
1. openssl -revoke users/%commonName%/%commonName%.pem
2. openssl ca -gencrl -out crl/ca.crl -crl days 7
```

La prima revoca un certificato aggiornando il database `index.txt`, citato a inizio sezione. Il certificato da revocare può essere passato sia come `.pem` che come `.crt`. La tabella del database si occuperà automaticamente di associare al certificato `.crt`, l'equivalente `.pem` leggendo il seriale.

Il secondo comando rigenera la CRL con il nuovo certificato revocato. La CRL legge i certificati proprio dall'`index.txt` aggiornato di volta in volta dai comandi di creazione e revoca dei certificati.

```
1. ...
2. V 130103092153Z                FAD3 /C=IT/ST=PD/O=University of Pad-
   ua/OU=Tecnico/CN=Mario/emailAddress=super@mario.com
3. V 130103095811Z                FAD4 /C=IT/ST=PD/O=University of Pad-
   ua/OU=superuser/CN=Luigi/emailAddress=super@luigi.com
4. R 120413093212Z 120126130013Z  FAD8 /C=IT/ST=PD/L=Padua/O=University of Pad-
   ua/OU=Registrato/CN=Bowser/emailAddress=bowser@evil.com
5. ...
```

Le righe qui sopra sono un esempio di `index.txt`. A ogni riga è inserito un ben preciso certificato. Il primo campo indica lo stato del certificato: V per valido, R per revocato. Il secondo campo è la data di creazione del certificato in formato ISO 8601 [47], mentre il terzo è la data di revoca (presente solo nei certificati contrassegnati dalla R, come quello di riga 4). Il quarto campo è il seriale del certificato. Infine il quinto è il DN del certificato.

Installare il certificato

Tutti i browser offrono delle procedure semplici per installare i certificati. In questa fase di importazione, viene richiesta la password di esportazione descritta al punto precedente. Quando si visita il sito, verrà richiesto che certificato utilizzare. Selezionando quello che abbiamo appena creato, sarà consentito l'accesso alla document root.

6.4.1 Entità certificanti autorizzate

La procedura finora descritta, fa uso di un CA auto-firmato. Tali certificati non sono riconosciuti come certificati autorizzati dal browser e quindi considerati insicuri dallo stesso (vedi Figura 6.6). I certificati auto-firmati sono facilmente manomettibili creando un nuovo certificato, essendo entrambi considerati invalidi.

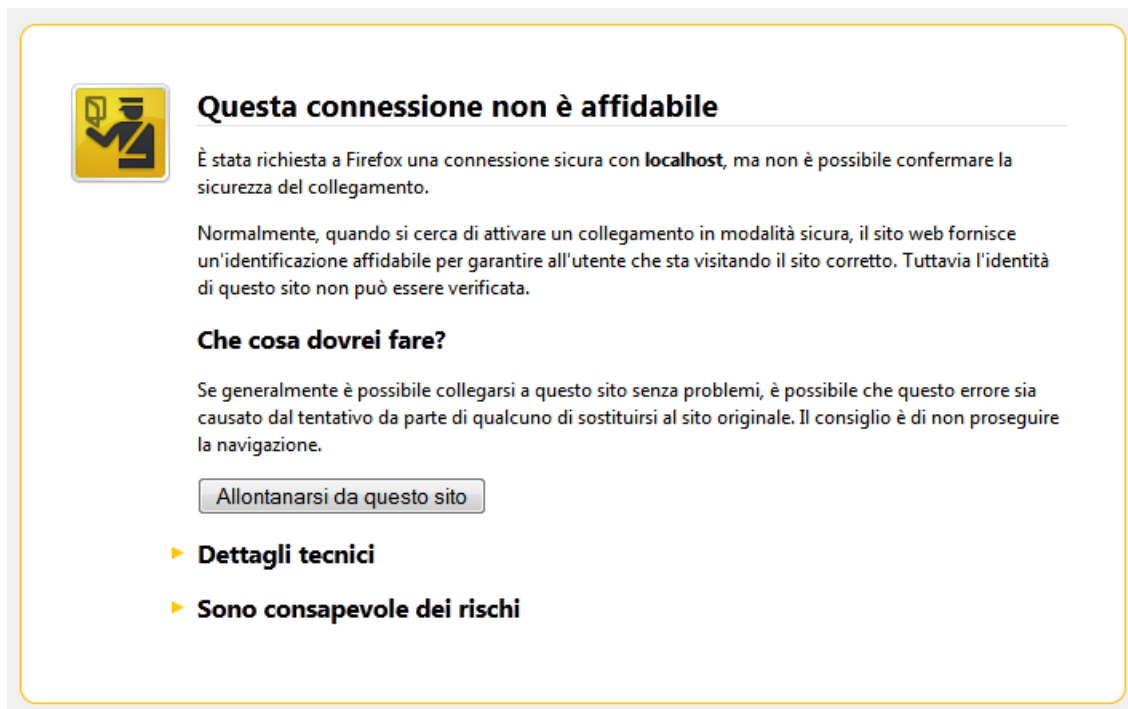


Figura 6.6. Certificato server auto-firmato non affidabile.

Facendo affidamento a un ente certificante, è possibile ottenere dei certificati sicuri e affidabili, che garantiscano l'attendibilità del sito. Purtroppo però, i costi associati possono risultare spesso onerosi, soprattutto quando il bacino di utenza è particolarmente ampio, visto che le autorità certificanti raramente permettono di firmare altri certificati con la stessa chiave CA acquistata.

6.4.2 Rivisitazione del database per la gestione utenti

Vista l'implementazione di un nuovo sistema di autenticazione, si è reso necessario aggiungere nuove funzionalità. La struttura del database è stata modificata per accogliere le nuove modifiche (Figura 6.7).

Come si può notare, l'unica aggiunta consiste nella nuova tabella `client_certs`. Lo scopo di tale tabella è di immagazzinare i dati riguardanti i certificati dei singoli utenti, come il numero seriale, il DN, il CN (Common Name) e la data di registrazione del certificato al sistema. Il certificato è legato, con una relazione 1-a-1, alla tabella `users` tramite la chiave esterna `id_user`, il che significa che a ogni utente è associato uno e un solo certificato.

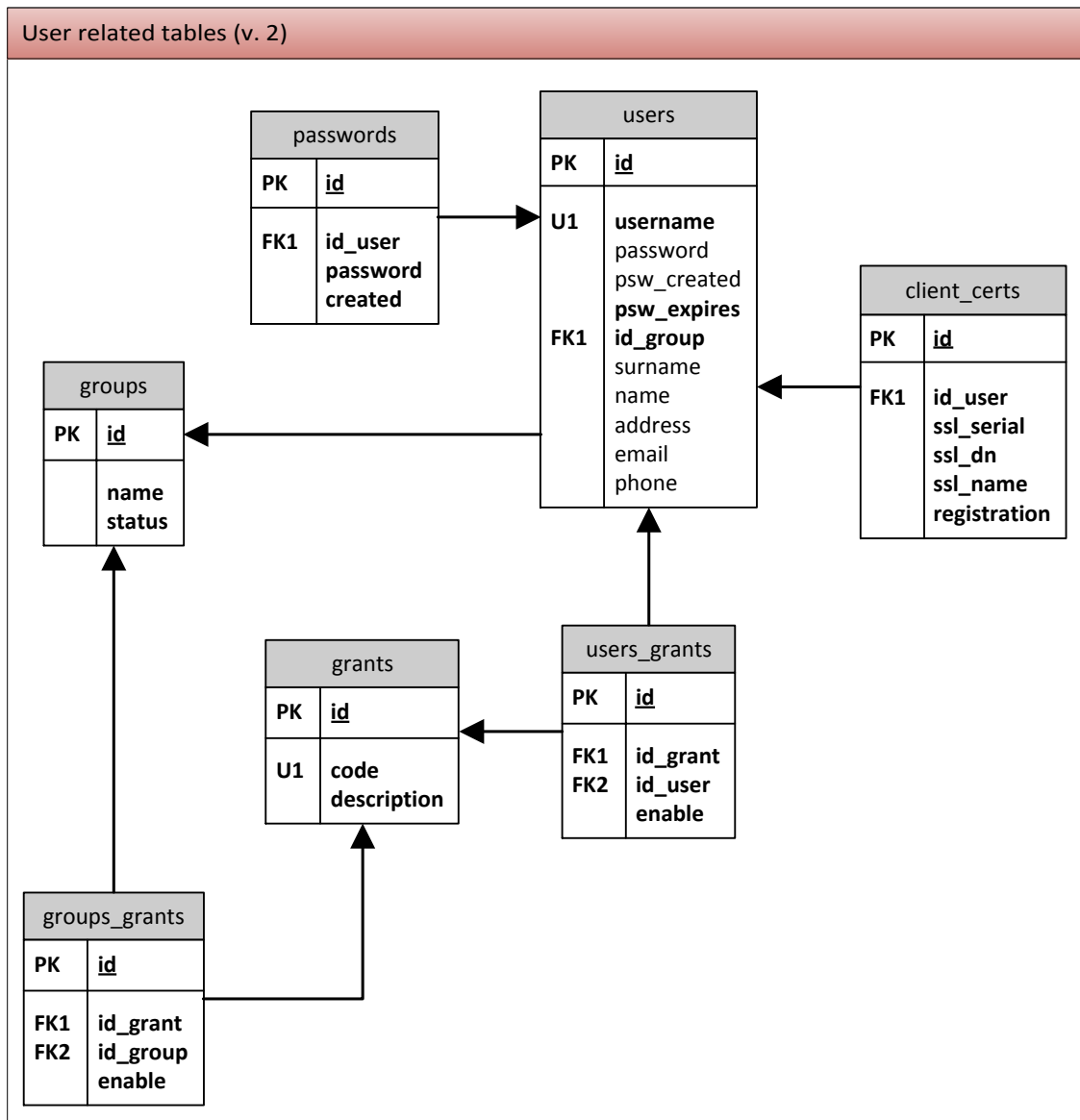


Figura 6.7. Seconda versione dello schema delle tabelle utente.

6.4.3 Rivisitazione della classe User

A una modifica del database è naturalmente seguita un'aggiunta di funzioni alla classe `User`. Oltre a queste, è stato aggiunto un array ad accesso privato `$certificate`, che immagazzina i dati del certificato. Tali informazioni naturalmente, vengono salvate solo se l'utente fornisce un certificato valido.

public function hasValidCert()

Il metodo verifica se l'utente ha fornito un certificato valido. Questa condizione può essere facilmente verificata accedendo alle variabili superglobali dell'array `$_SERVER`, come `$_SERVER['SSL_CLIENT_M_SERIAL']` o `$_SERVER['SSL_CLIENT_VERIFY']`. Se la condizione restituisce un risultato positivo, l'array `$certificate` viene popolato con le informazioni del certificato (in particolare seriale, DN, CN e unità occupazionale se presente) e viene restituito `true` dalla funzione.

public function isCertRegistered()

Viene verificato se il sistema di gestione utenti presenta un record associato al certificato fornito dall'utente. Ciò si realizza eseguendo una semplice query alla tabella `client_certs` controllando l'esistenza di un record con il seriale del certificato fornito. Se esiste, la funzione restituisce vero, altrimenti falso.

Vedremo nel capitolo 8 che la registrazione del certificato nel sistema, farà parte del flusso esecutivo necessario alla creazione dei nuovi utenti.

public function registerCert(\$psw)

Se il certificato non è stato ancora registrato, è possibile farlo tramite questa funzione. Con questo metodo, non viene solamente aggiunto un record alla tabella `client_certs`, ma viene anche creato un utente associato a tale certificato (comprensivo di password di accesso) e il ruolo a esso associato. In particolare, viene letto il campo "Occupational Unit" del certificato per risalire al ruolo associato all'utente:

```
1. $query = "SELECT id FROM ".CFG_GROUPS_TABLE." WHERE name = '". $this->certificate['ou']. "' LIMIT 1;";
2. $result = query($query);
3. if ($result && mysql_num_rows($result))
4.     $group_id = mysql_result($result, 0);
```

La query controlla dalla tabella `groups`, se esiste un id associato all'occupational unit fornita nel certificato. Se esiste, assegna alla variabile `$group_id` (utilizzata in seguito per assegnare il ruolo al nuovo utente), il risultato trovato, altrimenti viene assegnato un gruppo di default (solitamente privo di alcun permesso). La funzione crea quindi un nuovo utente con username il CN del certificato; password, la versione cifrata di quella passata in parametro (viene salvata anche la data di creazione di tale password), e come gruppo, quello restituito dalla query di prima (o il gruppo di default). L'esito dell'intera operazione è restituito da un booleano.

public function checkPsw(\$psw)

Il metodo funziona in maniera del tutto analoga al login, con la differenza che l'username, in questo caso, è provvisto dal certificato stesso. Come per il caso con username e password, la funzione rigenera l'id di sessione, convalida l'input assicurando che non vengano inseriti caratteri pericolosi, esegue il controllo sul database e, in caso di risultato positivo, popola l'utente con i relativi dati e permessi. Infine imposta la flag `$logged` a true e restituisce vero.

public function getCertData(\$field)

Lo scopo di questa funzione è di restituire i valori del certificato salvati nell'array `$certificate`. I campi accettati sono: 'serial', 'dn', 'cn' e 'ou'. Parametri diversi dai precedenti non sono riconosciuti e restituiscono sempre null.

6.4.4 Vantaggi e svantaggi nell'uso dei certificati

Oltre agli evidenti benefici di sicurezza, l'uso dei certificati fornisce due ulteriori vantaggi: delega la registrazione dell'utente all'utente stesso e fornisce un meccanismo di RBAC automatico (certificato di ruolo). Il fatto che la registrazione del certificato nel sistema, crei automaticamente l'utente, svincola l'amministratore dal dover creare singolarmente i diversi utenti, delegando l'incarico all'utente

stesso. Inoltre, l'assegnazione automatica del ruolo, permette all'utente di svolgere pienamente la propria attività già dal momento della registrazione. Inoltre, impostando la scadenza del certificato ogni tre mesi, si va incontro alle norme di sicurezza minime imposte dalla legge sulla privacy: la scadenza del certificato, infatti, prevede la revoca dello stesso e la creazione di un nuovo certificato, che andrà quindi registrato nuovamente dall'utente utilizzando una nuova password.

L'evidente svantaggio legato a questo sistema (oltre a quelli già citati), è la gestione dei certificati stessi. Generare e rinnovare centinaia di certificati è un incarico gravoso e delicato (un certificato sbagliato deve essere revocato e quindi rifatto, non può semplicemente essere sovrascritto). L'adozione di script in grado di automatizzare queste procedure, può essere cruciale per migliorare l'efficienza e la correttezza di tali operazioni. Va inoltre detto che la lista di revoca dei certificati (CRL) ha anch'essa un periodo di scadenza, terminato il quale tratta tutti i certificati (senza alcuna distinzione) come scaduti. L'aggiornamento della CRL (perché scaduta o perché sono stati aggiunti nuovi certificati revocati alla lista), richiede il riavvio del server Apache per essere applicato. Questo può rappresentare un problema serio in un ambiente dove l'accesso rapido deve essere garantito in ogni momento. Apache fornisce un metodo di riavvio "gentile" (*graceful restart*) per alleviare in parte questo problema [48]. La comunità di ASF BugZilla, ha segnalato il caso in cerca di soluzioni che rendano più flessibile l'utilizzo della CRL [49].

6.5 La classe UserManager

Come funzione ausiliaria, è stata creata una classe `UserManager`, per la gestione degli oggetti di tipo `User`. Inizialmente lo scopo della classe era di fornire all'amministratore o a un superutente delle funzioni che gli consentissero di accedere, aggiungere, modificare e rimuovere utenti, gruppi e permessi. `UserManager` si appoggia alla classe `User` per le funzioni di modifica dei singoli utenti (il metodo `set`, per la precisione) riutilizzando lo stesso codice. Successivamente, l'aggiunta dei certificati ha reso necessario introdurre nuovi metodi che permettessero di generare in maniera programmatica nuovi certificati utente. In particolare sono stati sviluppati i seguenti metodi:

- `public function createClientKey($cn, $overwrite = false)`: genera una chiave privata per l'utente il cui CN è passato in parametro. Se il flag `$overwrite` viene impostato a `true` e viene trovata una chiave con lo stesso CN, quest'ultima viene sovrascritta dalla nuova appena creata.
- `public function createCSR($cn, $dn, $privateKey)`: riceve in ingresso tre parametri, il CN, il DN (sotto forma di array) e la chiave privata con cui creare il CSR (restituita dal metodo precedente). Alla fine viene restituito una risorsa di tipo CSR.
- `public function signCSR($cn, $csr, $expiration = 90)`: firma il CSR passata in argomento con la chiave del client il cui nome corrisponde al CN in parametro. Se non viene specificato il campo `$expiration`, il certificato ha durata valida di 90 giorni. La funzione ha anche la responsabilità di aggiornare il file che contiene il seriale dei certificati (che non viene eseguito in automatico). Alla fine il metodo restituisce una risorsa di tipo certificato in formato X509.
- `public function exportToP12($cn, $cert, $privKey, $expPsw = "")`: esporta il certificato passato in argomento nel formato PKCS12. Restituisce vero se l'operazione va a buon

fine, falso altrimenti. La password di esportazione viene impostata col parametro `$expPsw`, che di default è impostata a stringa vuota.

Queste funzioni fanno uso della libreria per PHP di OpenSSL. Purtroppo, trattandosi di una libreria ancora in fase sperimentale, molte funzioni non sono presenti (per esempio non si può revocare un certificato, né creare la CRL) e altre sono ancora incomplete (per esempio alla creazione di un nuovo certificato, non solo non viene aggiornato il seriale, ma neanche il database dei certificati, `index.txt`). Pertanto, fino a nuovi aggiornamenti, si sconsiglia l'uso di tale libreria.

Capitolo 7

Log Applicativo

Questo capitolo descrive le implementazioni adottate per i meccanismi di log a livello applicativo. È incluso anche il sistema di audit trail secondo le specifiche IHE. Infine sono presentati i risultati e le considerazioni sulle diverse strutture dati adottate per l'audit trail.

7.1 Struttura del database per la gestione dei log

I log applicativi e gli audit trail si appoggiano su strutture dati diversi. In particolare, per quest'ultima sono state previste due strutture diverse: database e XML (vedi paragrafo 4.3.1). Le convenzioni degli schemi database sono le stesse descritte al capitolo 6.

7.1.1 Log applicativo

Di seguito è illustrato lo schema del database utilizzato per memorizzare i log di applicazione.

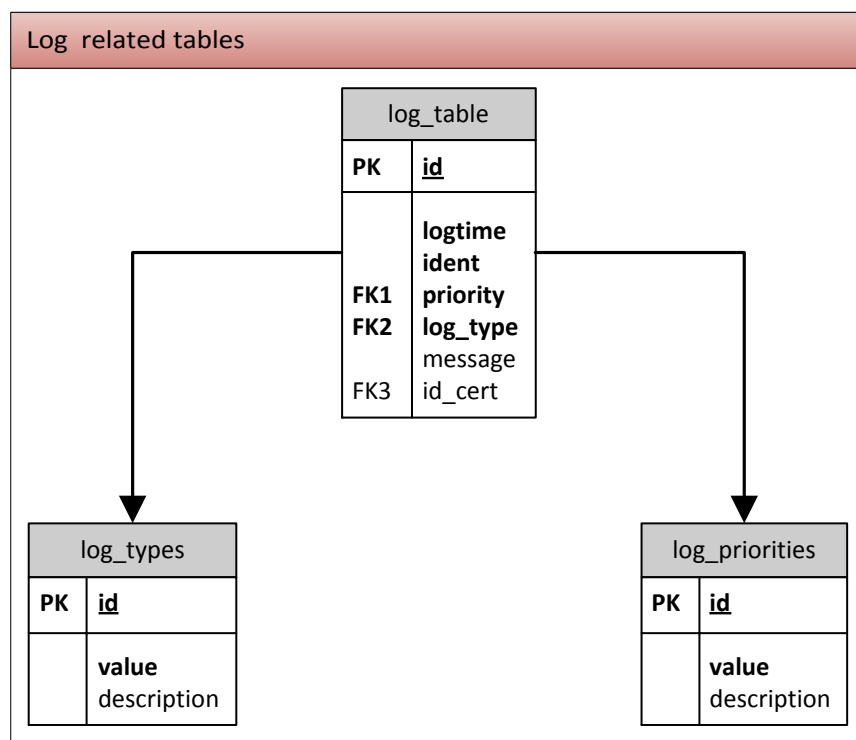


Figura 7.1. Schema delle tabelle di log.

Lo schema è molto semplice: ogni evento di log viene salvato in un record della tabella `log_table`. Le tabelle `log_types` e `log_priorities` contengono rispettivamente gli identificatori per il tipo di log e il livello di priorità.

- `log_table`: contiene i singoli log, riga per riga. Ogni log è caratterizzato da un timestamp (`logtime`), un identificativo (`ident`), un livello di priorità (chiave esterna `priority`), un tipo di log (chiave esterna `log_type`), il messaggio da contenere e, se l'evento è legato all'utente e questi ha presentato un certificato valido, il seriale del certificato (chiave esterna `id_cert`).
- `log_types`: contiene i vari tipi di log con la relativa descrizione. Attualmente sono previste le seguenti tipologie: `system`, `error`, `event`, `user`, `statistic`, `alert`, `connection`, `audit`, `other`.
- `log_priorities`: contiene i vari gradi di priorità e la descrizione ad essa associata. Sono presenti i seguenti livelli:
 - `log_emerg`: il sistema è inutilizzabile;
 - `log_alert`: è richiesta un'azione immediata;
 - `log_crit`: condizioni critiche che possono causare problemi seri al funzionamento complessivo del sistema;
 - `log_err`: condizioni di errore che possono interrompere il flusso operativo e creare inconsistenze in una o più parti del sistema;
 - `log_warning`: condizioni di allerta che indicano un problema di medio livello;
 - `log_notice`: presenza di anomalie che non compromettono la funzionalità del sistema, ma andrebbero corrette;
 - `log_info`: informazioni;
 - `log_debug`: messaggi di debug.

Carico dei volumi e delle operazioni

Segue una stima tecnica dei volumi delle tabelle:

- `log_table`: considerata la sua natura, questa tabella può facilmente raggiungere dimensioni critiche. A seconda del sistema e del grado di dettaglio delle informazioni, si possono avere da poche decine di record per utente, fino a diverse migliaia. Supponendo che per ogni utente vengano registrati un centinaio di log al giorno (cifra verosimile in un ambiente ospedaliero), e considerando un centinaio di utenti attivi al giorno, si ottengono più di 3 milioni di record all'anno, cifra importante da gestire, ma comunque sostenibile con le ultime versioni di MySQL e backup regolari. Si può ridurre il carico registrando solo gli eventi con priorità più alta, ma ciò minerebbe la sicurezza complessiva del sistema.
- `log_types`: solitamente non contiene più di una decina di tipologie.
- `log_priorities`: come sopra, non sono previste più di una decina di righe.

Segue una tabella con le operazioni più frequenti e una stima qualitativa del carico:

Operazione	Tipo	Frequenza
Ricerca di log filtrati per campo	Select	≤1 al giorno
Inserimento nuovo log	Insert	>1000 al giorno

Le ricerche solitamente avvengono solo in caso di necessità e per funzioni di controllo (si può ipotizzare una ricerca al giorno), ma si basano molto spesso su testo libero. Dalla versione 3.23.23, MySQL ha introdotto la ricerca FULL-TEXT che non solo consente di eseguire ricerche su stringhe

molto veloci, ma anche di valutare e ordinare il grado di attinenza dei risultati trovati. Sfortunatamente questa caratteristica è utilizzabile solamente su tabelle con motore MyISAM.

Gli inserimenti sono evidentemente l'operazione più pesante. Fortunatamente viene coinvolta una sola tabella, quindi l'operazione risulta piuttosto semplice dal punto di vista operativo e tecnico.

7.1.2 Audit trail

In questa sezione presentiamo le due strutture dati per i record di audit.

XML

Lo schema XML utilizzato per i record di audit è reperibile presso il sito dell'RCF-3881 [44]. Di seguito è illustrato un esempio di audit per un evento di autenticazione:

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <AuditMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3.     <EventIdentification EventActionCode="E" EventDateTime="2012-02-11
4.     11:39:20" EventOutcomeIndicator="1">
5.         <EventID code="110114" codeSystemName="DCM" displayName="User
6.         Authentication"/>
7.     </EventIdentification>
8.     <ActiveParticipant UserID="FAD9" AlternativeUserID="20"
9.     UserIsRequestor="1" NetworkAccessPointID="127.0.0.1"
10.    NetworkAccessPointTypeCode="2"/>
11.    <ActiveParticipant UserID="/atna/www/user_account.PHP?logout"
12.    UserIsRequestor="0"/>
13.    <AuditSourceIdentification
14.    AuditSourceID="/atna/www/user_account.PHP?logout">
15.        <AuditSourceTypeCode code="4"/>
16.    </AuditSourceIdentification>
17. </AuditMessage>

```

Il significato dei vari parametri è descritto nella sezione 4.3.1. Si noti il valore FAD9 alla riga 6: esso rappresenta il seriale del certificato dell'utente, il cui ID (20) è salvato nel campo `AlternativeUserID`. Altra nota interessante è data dal campo `EventOutcomeIndicator` (riga 3), impostato a 1, che significa che l'evento (in questo caso un tentativo di autenticazione) ha avuto successo (i.e. autenticazione riuscita).

Database

La struttura del database è simile a quella utilizzata per i log di applicazione, con l'unica differenza che presenta più livelli di elementi coinvolti e molte più tabelle con codici identificativi.

Ogni record di audit è salvato in un "master record" cui sono associati i vari record dei sotto-elementi (sorgente, oggetti coinvolti, evento e partecipanti attivi) tramite chiave esterna.

- `audit_trails`: contiene il "master record" a cui fanno riferimento tutti i sotto-elementi che compongono l'audit. Viene salvato solamente il `timestamp`.
- `audit_log_sources`: contiene i dati relativi alla sorgente che ha generato l'evento di audit. Per ogni master record può esserci una e una sola sorgente di audit, quindi la relazione con la tabella `audit_trails` è di 1-a-1.
- `audit_log_objects`: contiene i dati relativi agli oggetti coinvolti nell'evento di audit. Per ogni master record possono essere presenti zero o più oggetti (per esempio gli eventi di login e logout non hanno oggetti).

- `audit_log_events`: contiene i dati relativi all'evento che ha generato l'audit. Per ogni master record esiste uno e un solo record di evento.
- `audit_log_participants`: contiene i dati relativi ai partecipanti attivi all'evento. Ogni master record contiene almeno un partecipante attivo.
- "identificators_codes_tables": si tratta di un insieme di tabelle che descrivono i diversi identificatori utilizzati dalle tabelle appena descritte, come specificato dagli standard IHE. Per questioni di spazio e leggibilità non sono state riportate.

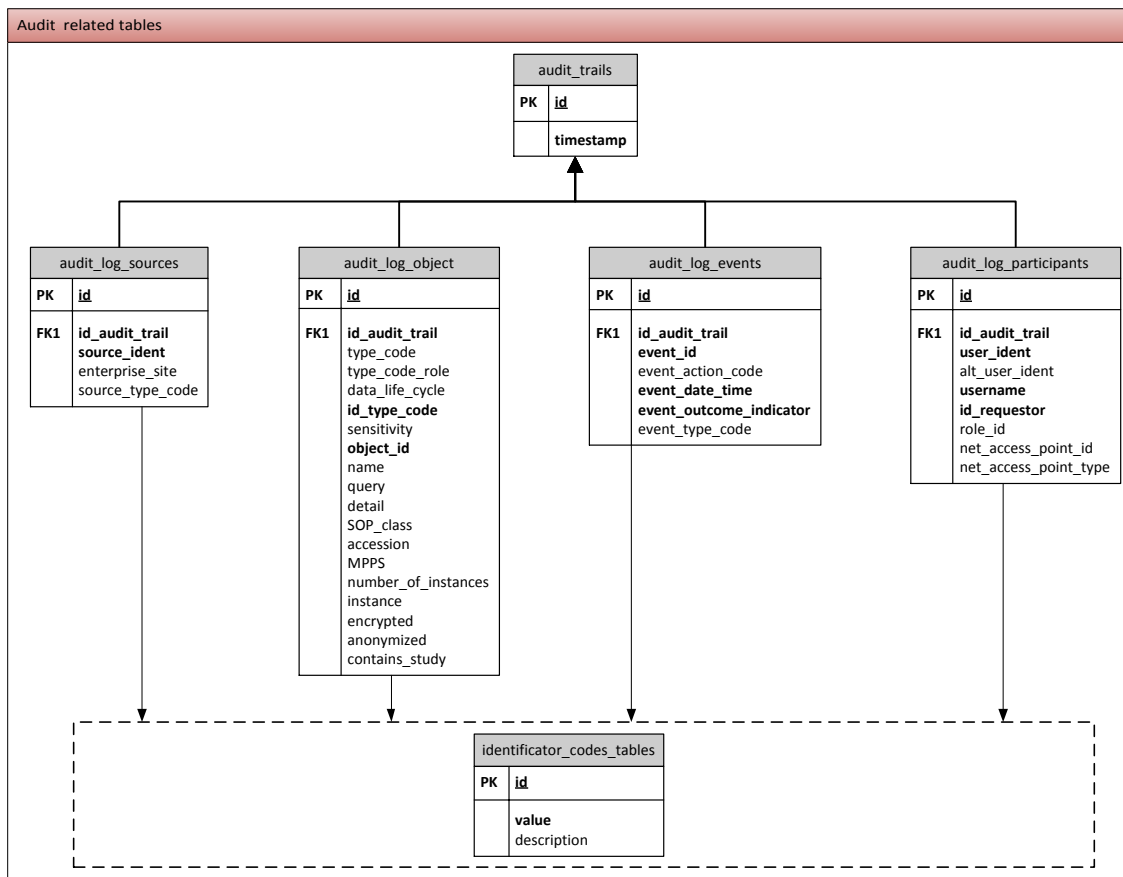


Figura 7.2. Schema delle tabelle di audit trail.

Il significato dei vari campi delle singole tabelle è descritto in sezione 4.3.1.

Carico dei volumi e delle operazioni

Valgono gli stessi discorsi fatti per il log applicativo, con la sostanziale differenza che l'audit trail tiene traccia dell'intero flusso operativo di ogni singolo utente (niente è tralasciabile) e che sono coinvolte cinque tabelle, composte di numerosi campi di testo (VARCHAR). Inoltre ogni record può avere più istanze dello stesso componente (per esempio i partecipanti attivi o gli oggetti sono elementi multi-valore) aumentando ulteriormente il carico. Supponendo gli stessi valori usati per il log applicazione, si possono facilmente superare i 4 milioni di record per tabella all'anno.

Le operazioni di ricerca inoltre, sono molto complesse poiché devono riunire i risultati di tutte le tabelle coinvolte (incluse eventualmente le tabelle con i codici identificativi). Fortunatamente l'accesso all'audit trail, a differenza del log applicazione, viene svolto solo nei casi in cui sono coinvolte delle responsabilità legali, quindi la sua frequenza è molto bassa.

Per questi e altri motivi di sicurezza legati al valore delle informazioni, è consigliabile salvare gli audit trail su un database dedicato dalle prestazioni elevate e meccanismi di caching veloce.

7.2 La classe Log

La classe `Log` implementa il pattern di progettazione factory, descritto al paragrafo 5.1.1. Questo ci ha permesso di riutilizzare lo stesso codice di base per gestire i diversi tipi di log.

La classe `Log` implementa diversi *handler*, gli oggetti derivati dalla classe astratta di base, tra i quali:

- `console`: invia i log direttamente alla console di sistema.
- `display`: stampa i log direttamente sul browser.
- `error_log`: invia i log alle funzione `error_log()` di PHP.
- `file`: stampa i log su file.
- `mail`: aggrega i log di un'unica sessione e li invia a uno specifico indirizzo email tramite la funzione `mail()` di PHP.
- `syslog`: invia i log a un servizio di logging come il `syslog` di Unix o l'Event Log di Windows, utilizzando la funzione `syslog()` di PHP.
- `win`: invia i log in una finestra browser separata.
- `mysql`: salva i log in un database MySQL.

Inoltre, la classe implementa dei metodi per “comporre” i diversi handler in uno unico.

Creare un oggetto Log

La classe definisce tre metodi per creare un oggetto `Log`:

1. metodo `factory`;
2. metodo `singleton`;
3. `direct instantiation`.

Il metodo `factory`, come suggerisce il nome, implementa il Factory Pattern. Esso fornisce un metodo parametrizzato per la costruzione di istanze concrete di oggetti `Log`. Il primo parametro, infatti, determina lo specifico handler da creare. Gli altri parametri dipendono dall'handler che si vuole creare. Per esempio il seguente codice realizza un `Log` che stampa su file:

```
$file = Log::factory('file', 'out.log', 'TEST');
```

Il metodo `singleton` implementa il Singleton Pattern. Il Singleton Pattern assicura che nello stesso momento esista un unico tipo e configurazione di oggetto `Log`. Ciò ha due benefici: il primo è che previene la duplicazione di diverse istanze di `Log`; il secondo è che tutto il codice ha accesso alla stessa istanza di `Log`. La sua costruzione è del tutto analoga alla precedente:

```
$file = Log::singleton('file', 'out.log', 'TEST');
```

È infine possibile istanziare direttamente i diversi handler usando i loro costruttori. Tuttavia questo metodo è sconsigliato proprio perché vanificherebbe i vantaggi dell'utilizzare i diversi design pattern e creerebbe un accoppiamento più stretto del necessario tra l'applicazione e il pacchetto di `Log` usato.

Configurare l'handler

La configurazione degli handler è determinata dagli argomenti passati nel costruttore. Di seguito è data una breve descrizione dei vari parametri:

Parametro	Tipo	Descrizione
\$handler	Stringa	Il tipo di Log da costruire.
\$name	Stringa	Il nome della risorsa di log su cui registrare l'evento. L'utilizzo di questo parametro dipende dalla specifica implementazione dell'handler. Il valore di default è la stringa vuota.
\$ident	Stringa	Una stringa identificativa dello specifico handler. Questo valore è impostato di default a stringa vuota, e può essere modificato in runtime usando il metodo <code>setId()</code> .
\$conf	Array	Array associativo di coppie (chiave, valore) utilizzate dalle impostazioni specifiche dell'handler.
\$level	Integer	Sono loggati solamente i messaggi fino al livello (incluso) indicato da questo parametro. Il valore di default include tutti i livelli.

I livelli di log sono gli stessi elencati come priorità alla sezione 7.1.1, ma utilizzano una nomenclatura diversa:

Priorità	Nome
log_emerg	PEAR_LOG_EMERG
log_alert	PEAR_LOG_ALERT
log_crit	PEAR_LOG_CRIT
log_err	PEAR_LOG_ERR
log_warning	PEAR_LOG_WARNING
log_notice	PEAR_LOG_NOTICE
log_info	PEAR_LOG_INFO
log_debug	PEAR_LOG_DEBUG

La motivazione dietro questa nomenclatura è dovuta all'uso di costanti definite dalla classe `Log`:

```
define('PEAR_LOG_EMERG', 0); /* System is unusable */
define('PEAR_LOG_ALERT', 1); /* Immediate action required */
define('PEAR_LOG_CRIT', 2); /* Critical conditions */
define('PEAR_LOG_ERR', 3); /* Error conditions */
define('PEAR_LOG_WARNING', 4); /* Warning conditions */
define('PEAR_LOG_NOTICE', 5); /* Normal but significant */
define('PEAR_LOG_INFO', 6); /* Informational */
define('PEAR_LOG_DEBUG', 7); /* Debug-level messages */
```

Registrazione un evento

Gli eventi vengono registrati utilizzando il metodo `log()`:

```
$logger->log('Message', PEAR_LOG_NOTICE);
```

Il primo argomento riceve il messaggio da salvare. Anche se il messaggio viene sempre salvato come stringa, esso può ricevere anche oggetti. Se l'oggetto implementa un metodo `toString()` o analogo, `log()` utilizza quel metodo per rappresentare l'oggetto, altrimenti viene utilizzata la versione serializzata dello stesso. Il secondo parametro, opzionale, determina il livello di priorità del messaggio. Il valore di default è impostato a `PEAR_LOG_INFO`.

Il metodo restituisce vero se l'evento è stato registrato con successo, falso altrimenti.

7.2.1 L'handler mysql

L'handler `mysql` si connette a una specifica configurazione di database, grazie al file di configurazione descritto al paragrafo 6.3.1. Il costruttore utilizza il parametro `$name` per determinare il nome della tabella su cui salvare i record. Il metodo `log()` non fa nient'altro che inserire un nuovo record alla tabella `log_table`.

Per quanto riguarda invece l'audit trail, si è preferito fare uso di un metodo diverso per la registrazione dei record. Come abbiamo visto, infatti, l'audit trail necessita di un gran numero di informazioni strutturate, che vanno inserite in maniera logica nel database. Per tale motivo è stato realizzato il metodo `auditRecord($event, $partic, $source, $object = null)`. Tale funzione riceve quattro array in argomento che fanno riferimento rispettivamente all'evento che ha generato l'audit, i partecipanti attivi, la sorgente dell'evento e gli eventuali oggetti coinvolti. Ogni array è indicizzato con i nomi dei parametri indicati dal profilo IHE (vedi tabella 4.3.1), alcuni dei quali sono obbligatori e altri opzionali. Il metodo controlla automaticamente l'assenza dei campi obbligatori e solleva un'eccezione se ne mancano uno o più. In più, gli array `$partic` e `$object`, rappresentando elementi multivalore, possono essere anche multi-dimensionali (per ogni partecipante attivo od oggetto coinvolto è presente una riga diversa nell'array). La struttura degli array è quindi assimilabile alle seguenti tabelle:

\$event

	EventID	EventActionCode	EventDateTime	...
#1	Valore	Valore	Valore	...

\$partic

	UserID	AlternativeUserID	Username	...
#1	Valore	Valore	Valore	...
#2	Valore	Valore	Valore	...
...

\$source

	AuditEnterpriseSiteID	AuditSourceID	AuditSourceTypeCode
#1	Valore	Valore	Valore

\$object

	Part..TypeCode	Part..TypeCodeRole	Part..DataLifeCycle	...
#1	Valore	Valore	Valore	...
#2	Valore	Valore	Valore	...
...

Dopo aver eseguito tutti i controlli necessari, inizia la scrittura del file XML, secondo lo schema previsto. La scrittura utilizza la classe per documenti HTML e XML, `DOMDocument`, fornita dalle librerie PHP. Ogni XML adotta come filename, il timestamp dell'evento che lo ha generato, per evitare sovrascritture accidentali tra audit diversi.

Il metodo `auditRecord()` è anche responsabile di chiamare un altro metodo, `auditStore()`, che ha lo scopo di salvare nel database l'evento di audit.

Il metodo `auditStore($event, $partic, $source, $object)`, riceve in ingresso gli stessi parametri del metodo `auditRecord()`. Ogni singolo valore viene quindi validato prima di essere inserito nel database. L’inserimento crea prima un master record nella tabella `log_tables`, quindi inserisce i vari elementi che compongono l’evento nelle relative tabelle, utilizzando come chiave esterna proprio l’id del master record appena creato. Se durante la procedura di inserimento, qualcosa va storto, per evitare inconsistenze nelle informazioni, l’intera procedura subisce un rollback che elimina il master record e tutti i suoi sotto-elementi. L’audit così perso è facilmente recuperabile dall’XML e reinscrivibile nel database.

7.3 Prestazioni nella registrazione degli audit trail

Siccome l’audit trail deve registrare ogni singola operazione svolta dall’utente, sono stati svolti dei test per valutare l’impatto dovuto alla registrazione dei record di audit sulle prestazioni del sistema¹. In particolare abbiamo misurato i tempi medi richiesti per creare un nuovo record, sia su database che su XML per due tipologie di evento estremamente diverse tra loro. L’evento di tipo “User Authentication” presenta il minimo dei campi richiesti (per esempio non contiene nessun elemento di tipo oggetto), mentre l’evento di tipo “Application Activity” li contiene tutti, anche multivalore (contiene due partecipanti attivi e due oggetti), ed è quindi più “voluminoso”.

Evento	# part.	# ogg.	Tempo DB medio	Tempo XML medio
User Authentication	1	0	176,898 ms	2,173 ms
Application Activity	2	2	258,458 ms	4,009 ms

Si può vedere chiaramente come l’incremento dei dati da salvare influisca sui tempi di scrittura in entrambe le strutture dati, tuttavia è evidente come quella su XML sia nettamente più efficiente di quella su database di quasi 100 volte.

Considerato che il numero di scritture è decisamente più alto del numero di letture degli audit trail e vista la recente diffusione del linguaggio XQuery [50] per interrogare gli XML, è meglio adottare la soluzione basata su XML. Un ritardo di oltre 200 millisecondi per la scrittura su database è chiaramente percettibile durante il flusso esecutivo. Applicato a tutte le operazioni svolte dall’utente, ciò degenera in un calo prestazionale complessivo inaccettabile, soprattutto in vista di un’esecuzione concorrente con gli altri utenti connessi. Va specificato tuttavia, che macchine dedicate potrebbero ottenere risultati decisamente migliori nella scrittura su database.

Un ulteriore vantaggio dell’XML è il suo ampio utilizzo in diversi ambiti, che lo hanno reso il formato standard per lo scambio di informazioni tra sistemi.

¹ I test sono stati compiuti su una macchina con processore Intel Quad Core Q9550 a 2.83 GHz, 4 GB di memoria RAM DDR2 e disco rigido SATA2 con 7200 RPM e 8 MB di cache.

Capitolo 8

Ambiente di Prova

Per dimostrare l'intero sistema implementato, è stato realizzato un ambiente di prova verosimile, dove inserire le diverse funzioni descritte finora. L'ambiente dimostra l'efficacia delle soluzioni di autenticazione, autorizzazione e logging.

8.1 Struttura dell'applicazione web

L'intera mappa del sito è illustrata di seguito:

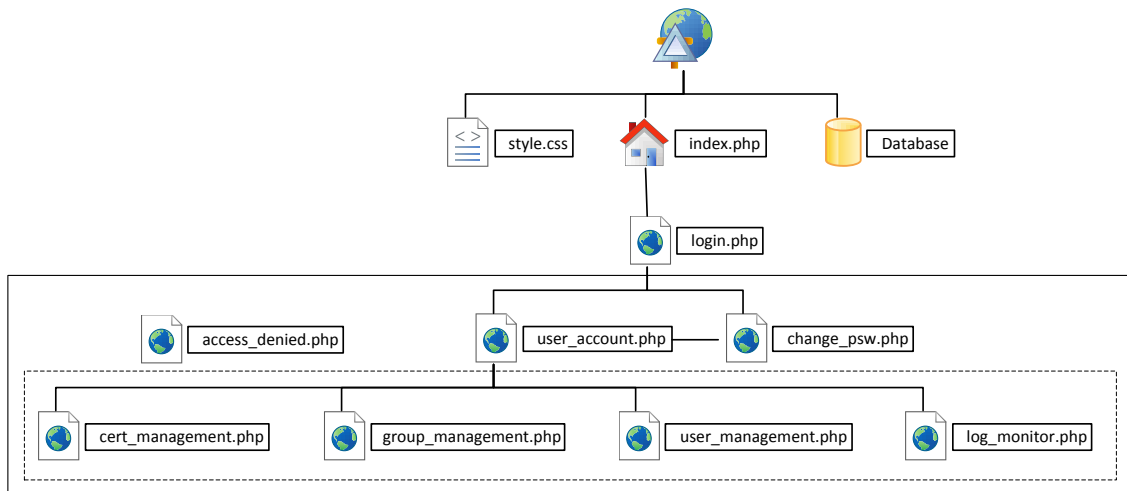


Figura 8.1. Mappa dell'ambiente di prova.

La porzione di sito esterna al riquadro con la linea continua, rappresentano le pagine non protette, accessibili dall'esterno. Quelle interne al riquadro sono invece le pagine protette dal meccanismo di autenticazione: per accedervi bisogna passare attraverso la pagina `login.php` e autenticarsi con successo. Le pagine all'interno del riquadro tratteggiato invece, sono riservate all'amministratore o a superutenti con permessi specifici. Esse implementano i meccanismi di autorizzazione previsti dalla classe `User`. Un tentativo non autorizzato di accedere a una di queste pagine reindirizza alla pagina `access_denied.php`.

L'aspetto grafico del sito è gestito da un foglio di stile, mentre i dati sono salvati in un database.

8.2 Descrizione del flusso operativo

Di seguito viene descritto il flusso operativo per un utente standard (senza permessi speciali) e per un superutente (con permessi di amministrazione).

L'accesso al sito crea un oggetto di tipo `User`, inizialmente vuoto. Tale oggetto viene salvato nella sessione (PHP serializza automaticamente gli oggetti [51]) per essere riutilizzato nelle altre pagine del sito. Un controllo sulla variabile assicura che non venga sovrascritto il precedente oggetto.

```

1. $user = &$_SESSION['user'];
2. if (!isset($user))
3.     $user = new User();
    
```

Il carattere `&`, permette di assegnare un riferimento alla variabile.

Apache è stato configurato per accettare sia client con un certificato valido che senza. La pagina di autenticazione (`login.php`) distingue i casi e risponde in maniera diversa. L'intero flusso di autenticazione è riassunto nel diagramma in Figura 8.2.

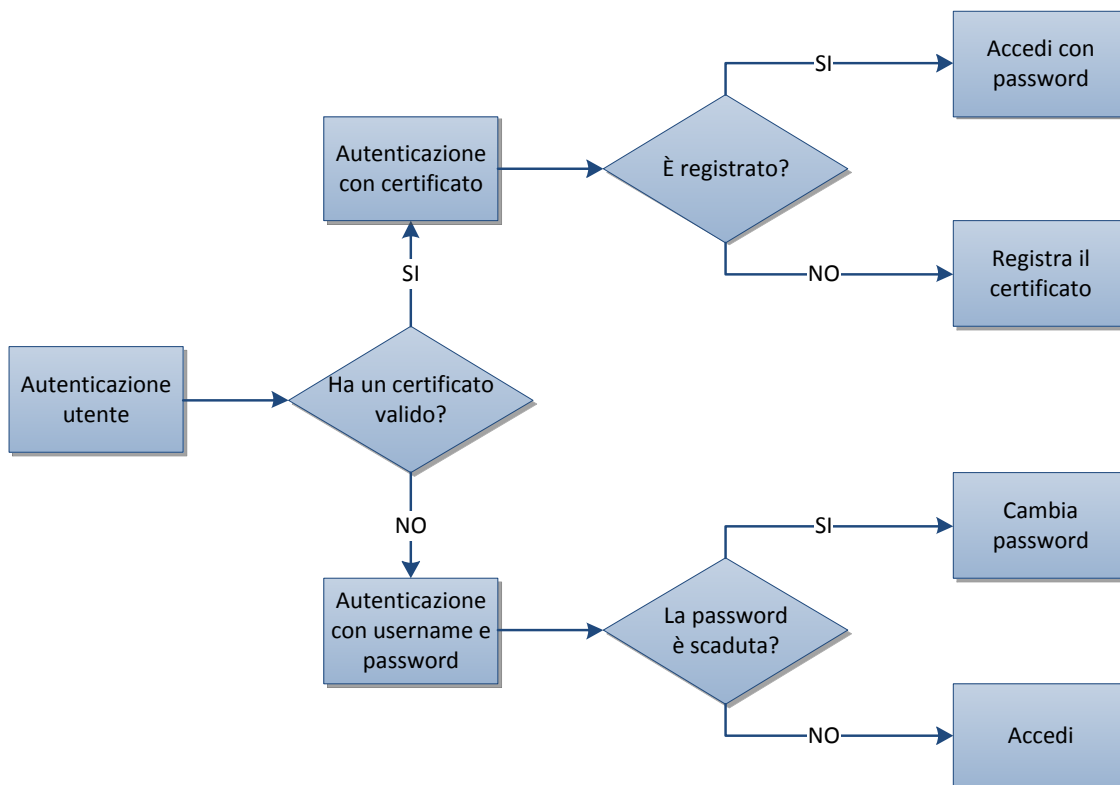


Figura 8.2. Diagramma di flusso dell'autenticazione.

Tramite la funzione della classe `User`, `hasValidCert()`, si verifica se il client ha fornito un certificato valido:

- Se il certificato è valido, la chiamata alla funzione `isCertRegistered()` verifica se il sistema presenta già un certificato con quel seriale.
 - Se esiste, viene proposta una form in cui inserire la password per accedere al sito. Il controllo avviene tramite la funzione `checkPsw($psw)`.
 - Se non esiste, viene proposta una form in cui registrare il certificato selezionando la password da utilizzare come autenticazione. La registrazione avviene tramite il metodo `registerCert($psw)`.
- Se non viene fornito un certificato valido, la pagina propone di autenticarsi con username e password. Dopo essersi autenticati correttamente, con il metodo `login($usr, $psw)`, viene eseguito in automatico un controllo sulla scadenza della password.

- Se la password è scaduta, il sito reindirizza verso la pagina `change_password.php`, in cui si propone all'utente di inserire una nuova password (l'intera operazione è gestita dal metodo `changePassword($newPsw)` della classe `User`). A cambio effettuato, l'utente viene reindirizzato alla pagina di accesso, `user_account.php`.
- Se la password non è ancora scaduta, l'autenticazione reindirizza direttamente alla pagina utente, `user_account.php`.

L'immagine in Figura 8.3, mostra le diverse risposte dell'applicazione alla presenza (in alto) o meno (in basso) di un certificato valido.

The figure shows two screenshots of a login page from the Dipartimento di Ingegneria dell'Informazione - Università degli studi di Padova. The top screenshot shows a message: "Certificato valido. max inserisci la password per accedere al sistema." Below this is a "Password:" input field and an "Invia" button. The bottom screenshot shows a message: "Certificato non valido. Accesso tramite username e password." Below this are "Username:" and "Password:" input fields and an "Invia" button. Both screenshots include a footer: "Progetto di tesi sviluppato da Massimiliano Sciacco (max.sciacco@gmail.com), presso l'università degli studi di Padova, in collaborazione con il Policlinico universitario."

Figura 8.3. Risposte dell'applicazione alla presenza o meno di un certificato valido.

Se si tenta di bypassare l'autenticazione, digitando direttamente nella barra degli indirizzi l'URL delle pagine protette, si viene reindirizzati alla pagina di login. Ciò è realizzato eseguendo un semplice controllo sulla variabile di stato `$logged` della classe `User`, accessibile tramite il metodo `isLoggedIn()`. Viceversa, se si tenta di accedere alla pagina `login.php` dopo essersi già autenticati, si viene automaticamente reindirizzati alla pagina `user_account.php`.

A prescindere dal tipo di autenticazione utilizzata e dal tipo di utente (normale o superutente), dopo essersi correttamente autenticati, si viene reindirizzati sulla pagina `user_account.php`, che contiene tutti i dati relativi all'utente (inclusi i permessi a esso associati). L'immagine in Figura 8.4, per esempio, mostra la pagina di account dell'utente "client1", appartenente al gruppo di default (`id_group = 0`), ossia senza un ruolo ben preciso e quindi privo di alcun permesso. A sinistra sono elencati i dati dell'utente, mentre a destra i permessi.

I permessi sono accessibili tramite il metodo `getPermissions()`, che restituisce un array 3-dimensionale con il codice univoco, la descrizione e il valore del permesso. La colonna "Tipo" definisce se il permesso è concesso a livello di gruppo o di utente e il suo valore è determinato con l'ausilio della funzione `isUserPermission($perm)`.

Dipartimento di Ingegneria dell'Informazione - Università degli studi di Padova

Account utente

Account utente
Logout (client1)

Anagrafica utente

id	<input type="text" value="22"/>	Sola lettura
username	<input type="text" value="client1"/>	Sola lettura
psw_created	<input type="text" value="2012-02-05"/>	Sola lettura
psw_expires	<input type="text" value="1"/>	Sola lettura
id_group	<input type="text" value="0"/>	Sola lettura
surname	<input type="text" value="Doe"/>	Modificabile
name	<input type="text" value="John"/>	Modificabile
address	<input type="text" value="First Street"/>	Modificabile
email	<input type="text" value="john@doe.com"/>	Modificabile
phone	<input type="text" value="6817964"/>	Modificabile

Cambia password

Conferma password

Nuovo campo

Permessi

#	Nome	Descrizione	Attivo	Tipo
1	INSPRENO	Inserimento prenotazione	Falso	Gruppo
2	CANCPREN	Cancellazione prenotazione	Falso	Gruppo
3	INSESAMI	Inserimento esami	Falso	Gruppo
4	VISUPAZI	Visualizzazione cartella pazienti	Falso	Gruppo
5	INSPAZIE	Inserimento paziente	Falso	Gruppo
6	MODPAZIE	Modifica anagrafica paziente	Falso	Gruppo
7	MODEXAWK	Modifica esame in worklist	Falso	Gruppo
8	ADDEXAWK	Aggiunta esame in worklist	Falso	Gruppo
9	DELEXWKL	Eliminazione esame in worklist	Falso	Gruppo
10	ADDREPOR	Aggiungi referto	Falso	Gruppo
11	MODREPOR	Modifica referto	Falso	Gruppo
12	SHOWREPO	Visualizza referto	Falso	Gruppo
13	SIGNREPO	Firma referto	Falso	Gruppo
14	SHOWIMAG	Visualizza immagini	Falso	Gruppo
15	MERGEPAT	Merge anagrafico	Falso	Gruppo
16	ALTERSTU	Spostamento studio tra 2 pazienti	Falso	Gruppo
17	MNGUSRGR	Gestione utenze e gruppi	Falso	Gruppo
18	MNGWRKPL	Caricamento postazioni di lavoro	Falso	Gruppo
19	MNGROOM	Gestione stanze	Falso	Gruppo
20	MNGCALEN	Creazione agende	Falso	Gruppo
21	MNGEXAMS	Caricamento e modifica prestazioni	Falso	Gruppo
22	MNGPRINT	Gestione stampanti referti e etichette	Falso	Gruppo
23	MNGWARDS	Gestione reparti richiedenti	Falso	Gruppo
24	MNGTURNI	Gestione turni nelle sale	Falso	Gruppo
25	MNGSTATS	Gestione statistiche	Falso	Gruppo
26	ACCETPAT	Accetta paziente	Falso	Gruppo
27	EXECEXAM	Esegui esame	Falso	Gruppo
28	MODSTATO	Modifica stato esame	Falso	Gruppo
29	SETTECH	Modifica tecnico esame	Falso	Gruppo
30	SETMEDIC	Modifica responsabile esame	Falso	Gruppo
31	DIGITALS	Firma elettronica referto	Falso	Gruppo
32	MNGCERT	Gestione/creazione certificati	Falso	Gruppo
33	MNTRLOG	Monitoraggio log	Falso	Gruppo

Progetto di tesi sviluppato da Massimiliano Sciacco (max.sciacco@gmail.com), presso l'università degli studi di Padova, in collaborazione con il Policlinico universitario.

Figura 8.4. Pagina account di un utente normale.

I dati a sinistra sono modificabili con il metodo `__set($field, $value)` premendo il tasto “Salva modifiche”. I campi in sola lettura sono riconosciuti con il metodo `isEditable($field)` e non sono modificabili (i campi sono impostati a `readonly`). La pagina permette anche di aggiungere nuovi campi all'array utente. Come si può notare in Figura 8.4, infatti, sono presenti due riquadri, dove inserire il nome del nuovo campo e il suo valore associato. Premendo il tasto aggiungi, la nuova variabile viene aggiunta a quelle già esistenti (Figura 8.5). Infine, i campi “Cambia password” e “Conferma password” permettono la modifica controllata della password tramite il metodo `changePassword($newPsw)`.

email	<input type="text" value="john@doe.com"/>	Modificabile
phone	<input type="text" value="6817964"/>	Modificabile
new_field	<input type="text" value="new_value"/>	Modificabile

Cambia password

Conferma password

#	Nome	Descrizione	Attivo	Tipo
9	DELEXWKL	Eliminazione esame in worklist	Falso	Gruppo
10	ADDREPOR	Aggiungi referto	Falso	Gruppo
11	MODREPOR	Modifica referto	Falso	Gruppo
12	SHOWREPO	Visualizza referto	Falso	Gruppo
13	SIGNREPO	Firma referto	Falso	Gruppo
14	SHOWIMAG	Visualizza immagini	Falso	Gruppo

Figura 8.5. Aggiunta di un nuovo campo "new_field" alla lista delle variabili.

Trattandosi di un utente normale, privo di qualsiasi permesso, i link alle pagine di amministrazione non vengono visualizzati. Per fare un confronto con un superutente, si osservi la Figura 8.6. Come si può notare, nel menù in alto, sono presenti quattro link che in Figura 8.4 non sono presenti. Quei collegamenti fanno riferimento alle pagine di amministrazione racchiuse dal riquadro tratteg-

giato di Figura 8.1. Il controllo dei permessi dell'utente viene fatto chiamando la funzione `hasPermission($perm)`. In particolare la pagina “Gestione utenti” e quella “Gestione gruppi”, controlla se l'utente dispone del permesso “MNGUSRGR” (numero 17 nella lista di Figura 8.6); la pagina “Gestione certificati” controlla il permesso “MNGCERT” (numero 32) e la pagina “Monitor Log” controlla il permesso “MNTRLOG”. Poiché il gruppo “superutente” (che nel sistema è indicato con `id_group = 1`) dispone di tutti i permessi, l'utente “max” ha accesso a tutte le pagine del sistema.

Dipartimento di Ingegneria dell'Informazione - Università degli studi di Padova

Account utente

Account utente | Amministrazione: [Gestione utenti](#) | [Gestione gruppi](#) | [Gestione certificati](#) | [Monitor Log](#) | [Logout \(max\)](#)

Anagrafica utente

id	<input type="text" value="20"/>	Sola lettura
username	<input type="text" value="max"/>	Sola lettura
psw_created	<input type="text" value="2012-02-05"/>	Sola lettura
psw_expires	<input type="text" value="1"/>	Sola lettura
id_group	<input type="text" value="1"/>	Sola lettura
surname	<input type="text" value="Planck"/>	Modificabile
name	<input type="text" value="Max"/>	Modificabile
address	<input type="text" value="Via vai"/>	Modificabile
email	<input type="text" value="max@planck.com"/>	Modificabile
phone	<input type="text" value="4101947"/>	Modificabile

Cambia password

Conferma password

Nuovo campo Valore

Permessi

#	Nome	Descrizione	Attivo	Tipo
1	INSPRENO	Inserimento prenotazione	Vero	Gruppo
2	CANCPREN	Cancellazione prenotazione	Vero	Gruppo
3	INSESAMI	Inserimento esami	Vero	Gruppo
4	VISUPAZI	Visualizzazione cartella pazienti	Vero	Gruppo
5	INSPAZIE	Inserimento paziente	Vero	Gruppo
6	MODPAZIE	Modifica anagrafica paziente	Vero	Gruppo
7	MODEXAWK	Modifica esame in worklist	Vero	Gruppo
8	ADDEXAWK	Aggiunta esame in worklist	Vero	Gruppo
9	DELEXWKL	Eliminazione esame in worklist	Vero	Gruppo
10	ADDREPOR	Aggiungi referto	Vero	Gruppo
11	MODREPOR	Modifica referto	Vero	Gruppo
12	SHOWREPO	Visualizza referto	Vero	Gruppo
13	SIGNREPO	Firma referto	Vero	Gruppo
14	SHOWIMAG	Visualizza immagini	Vero	Gruppo
15	MERGEPAT	Merge anagrafico	Vero	Gruppo
16	ALTERSTU	Spostamento studio tra 2 pazienti	Vero	Gruppo
17	MNGUSRGR	Gestione utenze e gruppi	Vero	Gruppo
18	MNGWRKPL	Caricamento postazioni di lavoro	Vero	Gruppo
19	MNGROOM	Gestione stanze	Vero	Gruppo
20	MNGCALEN	Creazione agende	Vero	Gruppo
21	MNGEXAMS	Caricamento e modifica prestazioni	Vero	Gruppo
22	MNGPRINT	Gestione stampanti referti e etichette	Vero	Gruppo
23	MNGWARDS	Gestione reparti richiedenti	Vero	Gruppo
24	MNGTURNI	Gestione turni nelle sale	Vero	Gruppo
25	MNGSTATS	Gestione statistiche	Vero	Gruppo
26	ACCEPAT	Accetta paziente	Vero	Gruppo
27	EXECEXAM	Esegui esame	Vero	Gruppo
28	MODSTATO	Modifica stato esame	Vero	Gruppo
29	SETTECH	Modifica tecnico esame	Vero	Gruppo
30	SETMEDIC	Modifica responsabile esame	Vero	Gruppo
31	DIGITALS	Firma elettronica referto	Vero	Gruppo
32	MNGCERT	Gestione/creazione certificati	Vero	Gruppo
33	MNTRLOG	Monitoraggio log	Vero	Gruppo

Progetto di tesi sviluppato da Massimiliano Sciacco (max.sciacco@gmail.com), presso l'università degli studi di Padova, in collaborazione con il Policlinico universitario.

Figura 8.6. Pagina account di un superutente.

Ma nascondere i link non è sufficiente a impedire l'accesso a una pagina riservata. Cosa accade se l'utente normale di prima prova a scrivere il nome della pagina direttamente nella barra degli indirizzi? Viene accolto da una pagina di accesso negato. All'inizio di ogni pagina riservata, è presente, infatti, un controllo sul permesso necessario ad accedere a quella pagina. Se il permesso non è attivo, si viene automaticamente reindirizzati alla pagina di Figura 8.7.

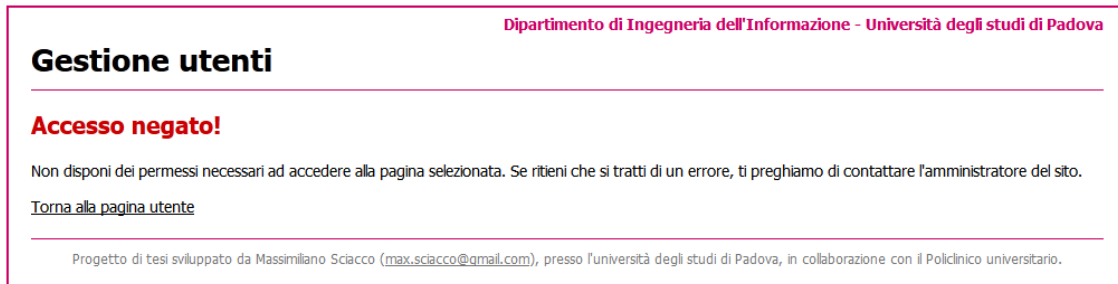


Figura 8.7. Reindirizzamento alla pagina di accesso negato.

Per completare la panoramica sull'autorizzazione, vediamo un altro esempio illustrato in Figura 8.8.

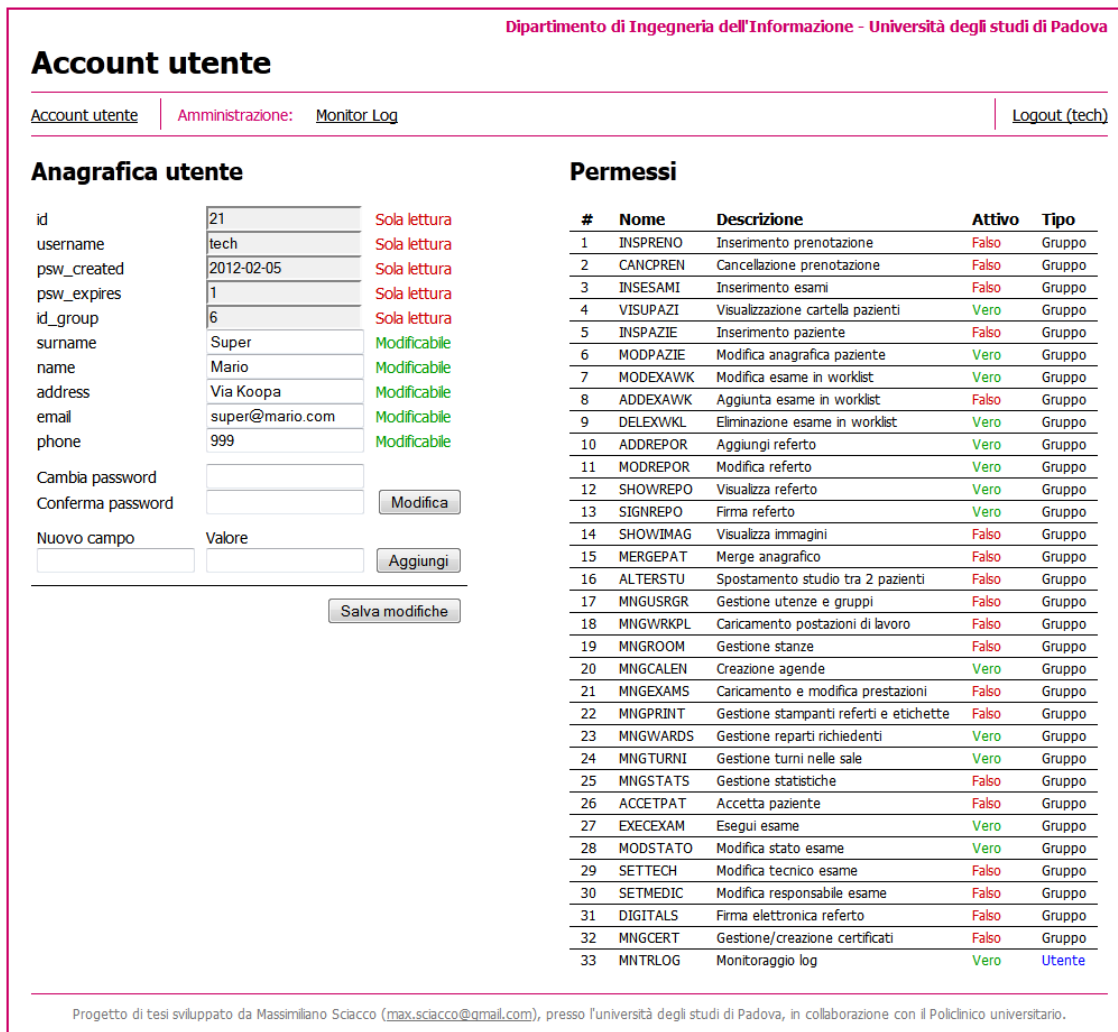


Figura 8.8. Pagina di account per un utente del gruppo “tecnici”.

L'utente “tech”, appartenente al gruppo dei tecnici (`id_group = 6`) presenta un privilegio utente per l'accesso al monitoraggio dei log (permesso numero 33 a destra). Il fatto che si tratti di un permesso a livello utente, è evidenziato dalla colonna “Tipo”, che presenta la scritta blu “Utente” in corrispondenza del permesso. A confermare l'attivazione del permesso, è la presenza del link alla pagina di monitoraggio dei log, “Monitor Log”, nel menù di navigazione in alto.

8.3 Pagine di amministrazione

La pagina di monitoraggio del log si presenta come Figura 8.9. L'intera navigazione del sistema, salva continuamente nuovi record di log nel database grazie agli oggetti della classe `Log`. L'esempio illustrato mostra un filtraggio dei log con priorità 5 (`LOG_WARNING`), di tipo 6 (`ALERT`) che vanno dal 5 febbraio 2012 in poi. I messaggi descrivono dei tentativi ripetuti da parte dell'utente con certificato `FAD9` di accedere al sito con credenziali sbagliate. La discrepanza temporale tra un tentativo e l'altro non ci fanno pensare a un tentativo di password guessing.

Il lettore scrupoloso avrà notato si tratta della stessa pagina presentata in Figura 2.10 per descrivere un attacco avanzato di SQL Injection con l'operatore `UNION`. Naturalmente quella vulnerabilità è stata lasciata apposta per mostrare l'attacco, ed è stata in seguito rimossa con adeguati meccanismi di convalida dell'input.

Dipartimento di Ingegneria dell'Informazione - Università degli studi di Padova

Log Monitor

Account utente | Amministrazione: [Monitor Log](#) | [Logout \(tech\)](#)

Ricerca log

Ident. Priorità Tipo Messaggio Seriale Cert. Da* A*

* Le date devono essere inserite nel formato: aaaa-mm-gg. Cerca

ID	Timelog	Ident.	Priorità	Tipo	Messaggio	Seriale Cert.
184	2012-02-05 11:09:43	audit	5	6	[FAD9] An unauthenticated user tried to login with an invalid psw.	FAD9
185	2012-02-05 11:09:50	audit	5	6	[FAD9] An unauthenticated user tried to login with an invalid psw.	FAD9
194	2012-02-09 13:59:24	audit	5	6	[FAD9] An unauthenticated user tried to login with an invalid psw.	FAD9
195	2012-02-09 13:59:28	audit	5	6	[FAD9] An unauthenticated user tried to login with an invalid psw.	FAD9
196	2012-02-09 13:59:32	audit	5	6	[FAD9] An unauthenticated user tried to login with an invalid psw.	FAD9
199	2012-02-09 13:59:55	audit	5	6	[FAD9] An unauthenticated user tried to login with an invalid psw.	FAD9
206	2012-02-12 12:55:02	audit	5	6	[FAD0] An unauthenticated user tried to login with an invalid psw.	FAD0
211	2012-02-12 14:57:04	audit	5	6	[FAD0] An unauthenticated user tried to login with an invalid psw.	FAD0
214	2012-02-12 15:26:02	audit	5	6	[FAD3] An unauthenticated user tried to login with an invalid psw.	FAD3
215	2012-02-12 15:26:05	audit	5	6	[FAD3] An unauthenticated user tried to login with an invalid psw.	FAD3
218	2012-02-12 15:26:53	audit	5	6	[FAD3] An unauthenticated user tried to login with an invalid psw.	FAD3

Progetto di tesi sviluppato da Massimiliano Sciacco (max.sciacco@gmail.com), presso l'università degli studi di Padova, in collaborazione con il Policlinico universitario.

Figura 8.9. Pagina di monitoraggio del log.

Finiamo il giro dell'applicazione descrivendo le pagine mancanti:

- `user_management.php`: esteticamente simile a `user_account.php`, permette a un utente con il permesso "MNGUSRGR" di modificare i dati anagrafici di un qualsiasi utente del sistema. Non è possibile modificare la password naturalmente. Tutte le modifiche fanno uso del magic method `__set($field, $value)` della classe `User`. La pagina `user_manager.php` si appoggia, infatti, alla classe `UserManager`, che a sua volta gestisce gli utenti istanziando oggetti di tipo `User`.
- `group_management.php`: raffigurata in Figura 8.10, permette a un utente con il permesso "MNGUSRGR" di modificare i permessi dei singoli gruppi e visionare gli utenti appartenenti a quel gruppo.
- `cert_management.php`: raffigurata in Figura 8.11, permette a un utente con il permesso "MNGCERT" di visionare i certificati creati e revocati, e di creare nuovi certificati client.

Seleziona il gruppo

Tecnico ▾

Permessi del gruppo Tecnico

#	Nome	Descrizione	Attivo	Cambia
0	INSPRENO	Inserimento prenotazione	Falso	CAMBIA
1	CANCPREN	Cancellazione prenotazione	Falso	CAMBIA
2	INSESAMI	Inserimento esami	Falso	CAMBIA
3	VISUPAZI	Visualizzazione cartella pazienti	Vero	CAMBIA
4	INSPAZIE	Inserimento paziente	Falso	CAMBIA
5	MODPAZIE	Modifica anagrafica paziente	Vero	CAMBIA
6	MODEXAWK	Modifica esame in worklist	Vero	CAMBIA
7	ADDEXAWK	Aggiunta esame in worklist	Falso	CAMBIA
8	DELEXWKL	Eliminazione esame in worklist	Vero	CAMBIA
9	ADDRPORA	Aggiungi referto	Vero	CAMBIA
10	MODREPORA	Modifica referto	Vero	CAMBIA
11	SHOWREPO	Visualizza referto	Vero	CAMBIA
12	SIGNREPO	Firma referto	Vero	CAMBIA
13	SHOWIMAG	Visualizza immagini	Falso	CAMBIA
14	MERGEPAT	Merge anagrafico	Falso	CAMBIA
15	ALTERSTU	Spostamento studio tra 2 pazienti	Falso	CAMBIA
16	MNGUSRGR	Gestione utenze e gruppi	Falso	CAMBIA
17	MNGWRKPL	Caricamento postazioni di lavoro	Falso	CAMBIA
18	MNGROOM	Gestione stanze	Falso	CAMBIA
19	MNGCALEN	Creazione agende	Vero	CAMBIA
20	MNGEXAMS	Caricamento e modifica prestazioni	Falso	CAMBIA
21	MNGPRINT	Gestione stampanti referti e etichette	Falso	CAMBIA
22	MNGWARDS	Gestione reparti richiedenti	Vero	CAMBIA
23	MNGTURNI	Gestione turni nelle sale	Vero	CAMBIA
24	MNGSTATS	Gestione statistiche	Falso	CAMBIA
25	ACCETPAT	Accetta paziente	Falso	CAMBIA
26	EXECEXAM	Esegui esame	Vero	CAMBIA
27	MODSTATO	Modifica stato esame	Vero	CAMBIA
28	SETTECH	Modifica tecnico esame	Falso	CAMBIA
29	SETMEDIC	Modifica responsabile esame	Falso	CAMBIA
30	DIGITALS	Firma elettronica referto	Falso	CAMBIA
31	MNGCERT	Gestione/creazione certificati	Falso	CAMBIA
32	MNTRLOG	Monitoraggio log applicativo	Falso	CAMBIA

Utenti nel gruppo

ID	Username	Gruppo	Nome	Cognome	Indirizzo	Email	Telefono
21	tech	6	Mario	Super	Via Koopa	super@mario.com	999

Figura 8.10. Pagina di gestione dei gruppi.

Tutte queste pagine implementano i meccanismi di controllo degli accessi descritti in precedenza. La loro esistenza ha il solo scopo di dimostrare la bontà di tali meccanismi e non di fornire un insieme completo di funzionalità per l'amministrazione dell'intero sistema.

Dipartimento di Ingegneria dell'Informazione - Università degli studi di Padova

Gestione certificati

Account utente | Amministrazione: [Gestione utenti](#) | [Gestione gruppi](#) | [Gestione certificati](#) | [Monitor Log](#) | [Logout \(max\)](#)

Lista dei certificati creati

Stato	Scadenza	Data di revoca	Seriale
V	12-12-29 12:07:02	--	FACE
R	12-12-29 13:13:54	12-01-26 12:44:28	FACE
V	13-01-01 15:08:20	--	FAD0
R	13-01-01 15:49:55	12-01-26 13:42:54	FAD1
R	13-01-01 16:06:13	12-01-26 13:43:00	FAD2
V	13-01-03 09:21:53	--	FAD3
V	13-01-03 09:58:11	--	FAD4
R	12-04-13 09:32:12	12-01-26 13:00:13	FAD8

Creazione di un nuovo certificato client

Dati CSR

Country (2 letter code):

State or province name (full name):

Locality name (eg. city):

Organization name (eg. company):

Organizational unit name (eg. section):

Common name:

Email address:

Dati Certificato

Scadenza (in giorni):

Sovrascrivere eventuali file con lo stesso common name?

Progetto di tesi sviluppato da Massimiliano Sciacco (max.sciacco@gmail.com), presso l'università degli studi di Padova, in collaborazione con il Policlinico universitario.

Figura 8.11. Pagina di gestione e creazione dei certificati.

Capitolo 9

Conclusioni

In quest'ultimo capitolo sono riassunte tutte le principali nozioni finora affrontate. Chiudono il trattato alcuni possibili miglioramenti realizzabili e alcune considerazioni personali sul sistema informativo complessivo.

9.1 Conclusioni

Con la loro continua evoluzione, le applicazioni web rappresentano uno strumento formidabile per qualsiasi azienda che debba mettere in rete la propria applicazione affinché sia accessibile allo staff. I vantaggi legati ai costi ridotti, la semplicità di distribuzione e aggiornamento del software, i numerosi strumenti open source e le potenzialità comunicative offerte da questa tecnologia, hanno visto moltissime attività trasferirsi sul web. Ma con ogni nuova tecnologia, nascono nuovi problemi legati alla sicurezza dell'identità, delle transazioni, ecc., in particolar modo in un ambiente fortemente "anonimo" come quello di internet. Gli sviluppatori devono prestare particolare attenzione alle meccaniche di autenticazione e autorizzazione, cuore pulsante di ogni sistema di sicurezza.

Abbiamo visto tre paradigmi di autenticazione:

- qualcosa che si conosce;
- qualcosa che si possiede;
- un tratto unico.

Questi paradigmi si traducono in tre grandi categorie di autenticazione: con username e password, con chiavi e certificati o con sistemi biometrici. Delle prime due categorie sono state descritte varie tecniche, dalla più semplice, alla più sofisticata. L'autenticazione con username e password basata su form è sicuramente la più diffusa perché facilmente adattabile allo stile e alle esigenze del servizio, oltre che fornire svariate funzionalità extra che l'autenticazione di base HTTP non ha. Il sistema basato su certificati si è verificato decisamente più sicuro, soprattutto se legato ad una parola chiave. Tuttavia presenta forti limitazioni dal punto di vista economico: ottenere centinaia di certificati garantiti da enti certificatori autorizzati, può venire a costare diverse migliaia di euro. Sui sistemi biometrici invece, non si è discusso molto vista la loro inadeguatezza ad un ambiente sanitario. Essi forniscono un livello di sicurezza molto alto, ma sono spesso costosi, imprecisi e lenti, caratteristiche assolutamente inaccettabili in un'azienda ospedaliera.

I principali attacchi a cui sono soggetti i sistemi di autenticazione sono di tre tipi:

1. rubare le credenziali;
2. assumere un'identità altrui;
3. bypassare il controllo di autenticazione.

Il primo fa uso di tecniche come il password guessing o l'eavesdropping per indovinare le password associate a specifici nomi utente. Il secondo, invece, cerca di carpire il significato dei valori dei token di sessione, così da modificarlo e ottenere quindi l'identità di un altro utente. Il terzo infine, cerca di superare i controlli di autenticazione sfruttando una debolezza del meccanismo di controllo, solitamente legato al tipo di input inserito dall'utente. È il caso degli attacchi di tipi input injection, di cui SQL injection è forse il più famoso. Un canale di trasmissione sicuro, token cifrati e casuali e una buona validazione dell'input sono la migliore contromisura a questi attacchi.

Abbiamo quindi visto come i meccanismi di autorizzazione seguano modelli diversi: Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role-Based Access Control (RBAC), Context-Aware Access Control (CAAC). I principali attacchi ai meccanismi di autorizzazione sfruttano la manomissione dei token di accesso per modificare i parametri utilizzati dalle Access Control List (ACL) per determinare i permessi. L'horizontal e vertical privilege escalation sono due esempi di manomissione dei token per ottenere l'identità di un utente con lo stesso livello di autorizzazione o maggiore. Un'attenta configurazione delle impostazioni di accesso di Apache, unita a token imprevedibili e un robusto sistema di log, rappresentano un buon punto di partenza per la realizzazione di un sistema di autorizzazione sufficientemente sicuro.

Infine abbiamo visto i diversi tipi di log e la loro importanza nell'individuazione delle anomalie o intrusioni. In particolare, sono state presentate due strategie per l'individuazione automatica delle anomalie: quella basata su regole statiche e quella su regole dinamiche. La prima può essere realizzata secondo i due modelli, positivo (black-list) e negativo (white-list). Il modello a black-list considera tutto normale come politica di default, e definisce cosa viene considerato anomalo (la lista nera appunto); mentre il modello white-list è l'esatto contrario. Le regole dinamiche invece si basano su politiche che dipendono dal contesto e richiedono una forte fase di "allenamento" in cui vengono distinti i casi normali da quelli da segnalare.

In tutta questa analisi non si sono mai perse di vista le direttive consigliate da IHE, per una corretta implementazione dei sistemi di autenticazione, autorizzazione e audit trail in ambito sanitario.

Implementazioni

Tra i meccanismi descritti sono stati quindi implementati i metodi di autenticazione con username e password basata su form, combinata a quella con chiavi e certificati per fornire un sistema di sicurezza a "due fattori". L'autorizzazione è realizzata sul modello RBAC inserendo all'interno degli stessi certificati il ruolo associato all'utente. L'intera soluzione è stata realizzata con una libreria ad oggetti PHP. Il sistema complessivo è risultato robusto, flessibile e sicuro ai principali metodi di attacco discussi. Siamo inoltre convinti dell'enorme beneficio che la programmazione ad oggetti può apportare, in termini di modularità, integrabilità, riutilizzo e organizzazione del codice, al sistema informativo in uso presso la Sezione di Radiologia del Dipartimento di Scienze Medico-Diagnostiche e Terapie Speciali dell'Università di Padova.

Per il log applicativo e l'audit trail si è utilizzato un approccio analogo che vede l'implementazione con factory pattern di una classe di Log generica, facilmente estendibile. L'analisi del carico di lavoro ha mostrato le criticità legate all'enorme mole di dati da gestire suggerendo backup periodici del database e sistemi dedicati. Per l'audit trail in particolare, i tempi di scrittura di un record su file XML sono risultati nettamente inferiori (di quasi 100 volte) a quelli richiesti per scrivere sul database. Soluzioni basate su XML ridurrebbero notevolmente il carico dell'audit trail sull'applicazione.

Il sistema complessivo è stato valutato in un ambiente di prova verosimile per dimostrarne l'effettiva applicabilità. I risultati sono stati incoraggianti e parte di esso è già stato installato sul MARiS.

9.2 Miglioramenti e considerazioni personali

L'autenticazione con certificati è stata realizzata con l'intento di valutarne l'effettiva applicabilità nell'ambiente operativo. Il passo successivo è quello di trasferire i certificati dal browser a dispositivi di sicurezza appositi, come smart card o security token. È importante tuttavia, che queste innovazioni non pesino eccessivamente sui bilanci dell'azienda.

Sull'audit trail

A un sistema molto sicuro ma lento, è preferibile uno più accessibile ma con prestazioni migliori. Rafforzare i meccanismi di audit trail diventa quindi fondamentale in un sistema del genere. La sicurezza, prima che dall'applicazione deve provenire dagli utenti. Gli utilizzatori diventano più coscienti se sanno che ogni loro azione viene tracciata e salvata in archivi altamente sicuri.

Proprio riguardo il tracciamento dell'audit, si può pensare di adottare un middleware per la gestione dei record (il già citato mirth per esempio). I record di audit vengono inviati al middleware in formato XML, il quale poi si occuperà di salvarli nel repository sicuro per gli audit. Questo porterebbe due benefici: intanto si svincola l'applicazione dall'operare con il database, poi si avrebbe il vantaggio di avere un buffer di memoria dove accodare i record in caso di sovraccarico del sistema.

Sul log applicativo

Il capitolo 4 suggerisce diversi metodi per l'individuazione delle intrusioni tramite l'utilizzo di regole statiche o dinamiche. Un lavoro futuro potrebbe essere quello di realizzare un motore che riconosca dinamicamente le anomalie. Tramite un algoritmo di apprendimento automatico (reti neurali, algoritmi genetici, support vector machine, ecc.) è possibile "insegnare" al sistema a riconoscere il traffico valido da quello anomalo. Il sistema deve attraversare una fase di apprendimento in cui gli viene fornito solo traffico lecito ed eterogeneo per includere quante più situazioni possibili. È importante che questo traffico sia assolutamente sicuro. Dopo un'adeguata fase di apprendimento, il motore sarà in grado di riconoscere automaticamente le anomalie segnalandole all'amministratore di sistema.

Sulla gestione utenti

Una piccola nota va fatta sul sistema di salvataggio dei dati adottato: tutte le informazioni utente, comprensive di permessi, sono salvate in variabili di sessione per essere facilmente trasferite tra una pagina e l'altra limitando al minimo gli accessi al database. Sebbene questa soluzione sia nettamente più sicura che salvare le informazioni sui cookie di sessione, per esempio, essa è pur sempre vulnerabile ad attacchi di eavesdropping. Fortunatamente il canale sicuro mitiga questo rischio. Inoltre va specificato che in caso di host condivisi, le variabili superglobali di PHP sono accessibili dai vari virtual host. Tipicamente non è il caso delle aziende sanitarie che dispongono di server dedicati. Una configurazione adeguata quindi, garantisce un livello di sicurezza più che sufficiente. La sessione, infatti, è penetrabile solamente da qualcuno in grado di eseguire un hacking a livello di server (per il quale sono richiesti metodi spesso molto sofisticati e determinati privilegi utente). Esistono

comunque diverse tecniche contro il furto delle sessioni: controllo dell'IP o utilizzo di nonce. In entrambi i casi però, la perdita di usabilità non giustifica la scelta. Controllando l'IP infatti, gli utenti che si trovano dietro un firewall per esempio, o che più genericamente non dispongono di un IP fisso, sono costretti a ri-autenticarsi ad ogni cambio. Utilizzando un nonce variabile a ogni pagina si ottiene lo spiacevole inconveniente di non poter ritornare alla pagina precedente senza “romperla”.

Sull'accesso al database

Per questioni di compatibilità col vecchio sistema informatico, si è preferito fare uso delle funzioni di base di accesso al database fornite da PHP. Il passaggio alla più completa libreria `mysqli`, permetterebbe di evitare tutti i controlli sugli input usati nelle query al database, essendo già gestiti dai prepared statement della classe. Oltre all'evidente incremento di sicurezza, ciò svincolerebbe lo sviluppatore dal preoccuparsi di SQL injection e di concentrarsi maggiormente su altri aspetti cruciali del suo lavoro. Non solo, la classe `mysqli` implementa anche dei meccanismi di query al database molto efficienti, che aumenterebbero notevolmente le prestazioni complessive del sistema.

Sulle politiche di accesso

Una sfida interessante sarebbe implementare il meccanismo di politiche descritto dai profili di controllo degli accessi IHE. Come spiegato nel paragrafo 3.5.4, l'accesso a determinate risorse cliniche, deve intersecare le politiche di soggetto richiedente, oggetto richiesto e proprietario dell'oggetto (solitamente il paziente). In questo momento il sistema prevede un legame solamente tra il soggetto richiedente e l'oggetto richiesto. Ampliando il controllo degli accessi anche al proprietario dell'oggetto, si potrebbe per esempio fare in modo che ogni medico abbia accesso solo ed esclusivamente alle cartelle cliniche dei pazienti che ha in cura, e solo se questi hanno dato il consenso all'accesso ai propri dati. Le complicazioni derivanti da tale implementazione sono legate alle situazioni di emergenza: se il medico curante non è disponibile e il paziente ha bisogno di un intervento d'urgenza, deve essere possibile bypassare le politiche di accesso come caso straordinario.

Il complesso dei permessi potrebbe essere esteso a un sistema di gestione delle politiche di accesso più completo e flessibile, in grado per esempio di determinare non solo il tipo di permesso da abilitare, ma anche il periodo di attivazione (per esempio l'accesso a un determinato strumento medico potrebbe essere abilitato solamente per la durata del turno in cui quell'utente ha accesso allo strumento), la postazione utilizzata (una ben precisa workstation per esempio) o il contesto in corso (uno stato di emergenza). Un meccanismo simile evolverebbe l'attuale modello RBAC (Role-Based Access Control) a CAAC (Context-Aware Access Control).

Sulla persistenza dei dati

Una caratteristica prevista dal profilo IHE è la persistenza dei dati sui client per le situazioni di rete instabile o inaccessibile. Il profilo specifica che ogni workstation deve essere in grado di lavorare indipendentemente dal server e deve pertanto disporre di tutte le informazioni e funzioni necessarie a operare correttamente, per la durata ritenuta necessaria, al termine del quale tutte le informazioni in più devono essere rimosse. Sebbene condivisibile dal punto di vista operativo, questo requisito va contro ogni logica alla base delle applicazioni web, dove i client sono trattati come thin-client. Im-

plementare un meccanismo simile renderebbe inutile ogni vantaggio legato alla struttura web, oltre che complicare la gestione dei dati.

Sul modello progettuale

Come ultima considerazione personale, ritengo che l'intero sistema informativo beneficerebbe di un'organizzazione più strutturata del codice e dei suoi componenti. La modularizzazione del software è utile sia a chi utilizza i moduli che a chi li progetta in quanto, essendo trattato come "black box" (scatola nera), l'utilizzatore non deve conoscere come vengono eseguite le funzioni, ma semplicemente cosa fanno, e lo sviluppatore può lavorare sul proprio modulo indipendentemente e parallelamente ad altri sviluppatori.

Bibliografia

- [1] M. S. Joel Scambray, *Hacking Exposed: Web Applications 2nd Edition*, Osborne: McGraw-Hill, 2002.
- [2] S. M. F. R. T. Holz, *New Threats and Attacks on the World Wide Web*, IEEE Computer Society: Security and Privacy, 2006.
- [3] R. M. B. E. M. M. Jamie Riden, «Know your Enemy: Web Application Threats,» 2007. [Online]. Available: <http://www.honeynet.org/papers/webapp/>.
- [4] V. L. C. S. Joel Scambray, *Hacking Exposed: Web Applications 3rd Edition*, Osborne: McGraw-Hill, 2011.
- [5] Acunetix, «Acunetix Web Application Security,» 2011. [Online]. Available: <http://www.acunetix.com/>.
- [6] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, Cambridge: Wiley, 2001.
- [7] IHE International, "IHE IT Infrastructure White Paper - HIE Security and Privacy through IHE Profiles," 22 Agosto 2008. [Online]. Available: http://www.ihe.net/Technical_Framework/index.cfm.
- [8] HIPAA LLC, "HIPAA.com," 2009. [Online]. Available: <http://www.hipaa.com/>.
- [9] OECD, "Organization for Economic Co-operation and Development," [Online]. Available: <http://www.oecd.org>.
- [10] IHE International, "IHE changing the way healthcare connects," [Online]. Available: <http://www.ihe.net/>.
- [11] National Electrical Manufacturers Association, "Digital Imaging and Communications in Medicine (DICOM). Part 1: Introduction and Overview," 2011. [Online]. Available: http://medical.nema.org/Dicom/2011/11_01pu.pdf.
- [12] HL7.org, "Health Level 7 International," 2007. [Online]. Available: <http://www.hl7.org/>.
- [13] IHE International, "IHE Profiles," [Online]. Available: <http://www.ihe.net/profiles/index.cfm>.
- [14] IHE International, "IHE Radiology Technical Framework, Volume 1 - Integration Profiles," 18 Febbraio 2011. [Online]. Available: http://www.ihe.net/Technical_Framework/index.cfm.
- [15] IHE International, "IHE IT Infrastructure (ITI) Technical Framework, Volume 1 - Integration Profiles," 19 Agosto 2011. [Online]. Available: http://www.ihe.net/Technical_Framework/index.cfm.
- [16] dcm4che.org, «Open Source Clinical Image and Object Management,» [Online]. Available: <http://www.dcm4che.org/>.
- [17] Mirth Corporation, «Mirth Corporation,» [Online]. Available: <http://www.mirthcorp.com/>.
- [18] *Decreto legislativo 30 giugno 2003, n. 196, Codice in materia di protezione dei dati personali.*

- [19] *art. 29 d.l. 25 giugno 2008, n. 112, conv., con mod., con l. 6 agosto 2008, n. 133; art. 34 del Codice; Prov. Garante 27 novembre 2008.*
- [20] Il garante per la protezione dei dati personali, «Misure e accorgimenti prescritti ai titolari dei trattamenti effettuati con strumenti elettronici relativamente alle attribuzioni delle funzioni di amministratore di sistema,» 27 Novembre 2008. [Online]. Available: <http://www.garanteprivacy.it/garante/doc.jsp?ID=1577499>.
- [21] Confindustria, «Consultazione pubblica su misure e accorgimenti prescritti ai titolari dei trattamenti effettuati con strumenti elettronici relativamente alle attribuzioni delle funzioni di amministratore di sistema.,» Roma, 2009.
- [22] *Decreto legislativo 7 marzo 2005, n. 82 - Codice dell'amministrazione digitale.*
- [23] V. Hauser, "THC-HYDRA," 1 Ottobre 2011. [Online]. Available: <http://www.thc.org/thc-hydra/>. [Accessed Gennaio 2012].
- [24] D. Flam, "Webcracker," 2010. [Online]. Available: <http://www.securityfocus.com/tools/706>.
- [25] HooBieNet, "Brutus," 30 Aprile 2001. [Online]. Available: <http://www.hoobie.net/brutus/index.html>.
- [26] Carnegie Mellon University, "The Official CAPTCHA Site," 2000-2010. [Online]. Available: <http://www.captcha.net/>.
- [27] Wikibooks, "Base64," [Online]. Available: <http://en.wikipedia.org/wiki/Base64>.
- [28] Tick Tock Computers, LLC, "Social-Engineer.Org : Security Through Education," [Online]. Available: <http://www.social-engineer.org/>.
- [29] W3Schools, «HTML URL Encoding Reference,» 1999-2012. [Online]. Available: http://www.w3schools.com/tags/ref_urlencode.asp.
- [30] Wikipedia, «MD5,» [Online]. Available: <http://en.wikipedia.org/wiki/MD5>.
- [31] T. Greening, "Ask and Ye Shall Receive: A Study in Social Engineering," *SIGSAC Review*, vol. 14, no. 2, pp. 9-14, Aprile 1996.
- [32] ITU-T Publications, "X.509," 2008. [Online]. Available: <http://www.itu.int/rec/T-REC-X.509/en>.
- [33] IHE International, «IHE IT Infrastructure White Paper - Access Control,» 28 Settembre 2009. [Online]. Available: http://www.ihe.net/Technical_Framework/index.cfm.
- [34] Wikipedia, «OSI model,» [Online]. Available: http://en.wikipedia.org/wiki/OSI_model.
- [35] R. Meyer, «Detecting Attacks on Web Applications from Log Files,» Sans Institute, 2008.
- [36] Jan Goyvaerts, «Regulare Expressions Reference,» [Online]. Available: <http://www.regular-expressions.info/reference.html>.
- [37] J. N. Menendez, «A Guide to Understanding Audit in Trusted Systems,» National Computer Security Center, 28 Luglio 1987. [Online]. Available: http://csirt.org/color_%20books/NCSC-TG-001-2.pdf.
- [38] M. S. Karen Kent, "Guide to Computer Security Log Management," National Institute of Standards and Technology, Gaithersburg, Settembre 2006.

-
- [39] Y. M. J. S. F. Miao, «RFC 5425,» Marzo 2009. [Online]. Available: <http://tools.ietf.org/html/rfc5425>.
- [40] IHE International, "IHE IT Infrastructure (ITI) Technical Framework, Volume 2a - Transactions Part A," 19 Agosto 2011. [Online]. Available: http://www.ihe.net/Technical_Framework/index.cfm.
- [41] DICOM Standard Committee, "Digital Imaging and Communications in Medicine (DICOM), Supplement 95: Audit Trail Messages," Rosslyn, 2010.
- [42] G. Marshall, "Security Audit and Access Accountability Message XML Data Definitions for Healthcare Applications," 2004.
- [43] D. B. M. M. N. M. Henry S. Thompson, "XML Schema Part 1: Structures," 24 Ottobre 2000. [Online]. Available: <http://www.w3.org/TR/2000/CR-xmlschema-1-20001024/>.
- [44] Grok-A-Lot LLC, "RFC3881," 19 Gennaio 2012. [Online]. Available: <http://www.rfc3881.net/>.
- [45] R. H. R. J. J. V. Erich Gamma, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1994.
- [46] The Apache Software Foundation, «Licenses,» [Online]. Available: <http://www.apache.org/licenses/>.
- [47] Wikipedia, «ISO 8601,» 6 Febbraio 2012. [Online]. Available: http://en.wikipedia.org/wiki/ISO_8601.
- [48] The Apache Software Foundation, «Stopping and Restarting - Apache HTTP Server,» 2011. [Online]. Available: <http://httpd.apache.org/docs/2.0/stopping.html>.
- [49] ASF BugZilla, «Bug 14104 - not documented: must restart server to load new CRL,» 21 Novembre 2011. [Online]. Available: https://issues.apache.org/bugzilla/show_bug.cgi?id=14104.
- [50] W3C, «XQuery 1.0: An XML Query Language (Second Edition),» 14 Dicembre 2010. [Online]. Available: <http://www.w3.org/TR/xquery/>.
- [51] The PHP Group, «PHP: Classes and Objects,» [Online]. Available: <http://php.net/manual/en/language.oop5.php>.
- [52] V. Hauser, "THC-HYDRA," 1 Ottobre 2011. [Online]. Available: <http://www.thc.org/thc-hydra/>.

Appendice

Acronimi

ACL	Access Control List
ACS	Access Control System
AdS	Amministratore di Sistema
ASCII	American Standard Code for Information Interchange
ATNA	Audit Trail and Node Authentication
B2B	Business-to-Business
CA	Certificate Authority
CAAC	Context-Aware Access Control
CAPTCHA	Completely Automated Public Turing Tests to Tell Computers and Humans Apart
CN	Common Name
CRC	Cyclic Redundancy Check
CRL	Certificate Revocation List
CSR	Certificate Signing Request
CSRF	Cross-Site Request Forgery
DAC	Discretionary Access Control
DBMS	DataBase Management System
DICOM	Digital Imaging and Communications in Medicine
DN	Distinguished Name
DoS	Denial of Service
EHR	Electronic Healthcare Record
EUA	Enterprise User Authentication
HCP	Health Care Provider
HIE	Health Information Exchange
HIPAA	Health Insurance Portability and Accountability Act
HMAC	Hashed Message Authentication Code
HTML	HyperText Markup Language

HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol over Secure Socket Layer
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IHE	Integrating the Healthcare Enterprise
ITU-T	International Telecommunication Union - Telecommunication Standardization Bureau
MAC	Mandatory Access Control (da non confondersi con MAC address, dove MAC sta per Medium Access Control)
MD5	Message Digest algorithm 5
NCSC	National Computer Security Center
OECD	Organization for Economic Cooperation and Development
PEP/PDP	Policy Enforcement and Policy Decision
PHI	Protected Healthcare Information
PKI	Public Key Infrastructure
PWP	Personnel White Pages
RAID	Redundant Array of Independent Disks
RBAC	Role-Based Access Control
RIS	Radiology Information System
SSI	Single-Sign-In
SSL	Secure Sockets Layer
SSPR	Self-Service Password Reset
ST	Security Token
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UML	Unified Modelling Language
URI	Uniform Resource Identifiers
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSRF	Cross-Site Request Forgery
XSS	Cross Site Scripting
XUA	Cross-Enterprise User Assertion

Elenco delle figure

Figura 1.1. I componenti di un'architettura web application.	4
Figura 1.2. L'architettura a 3-livelli delle applicazioni web.	7
Figura 1.3. Policy environment.	9
Figura 1.4. Profili di integrazione dell'infrastruttura IHE.	12
Figura 2.1. Un'esecuzione di Brutus per un sistema di autenticazione base HTTP.	28
Figura 2.2. La finestra di autenticazione base di un browser.	29
Figura 2.3. Flusso esecutivo di una richiesta di autenticazione HTTP base.	29
Figura 2.4. Funzionamento dell'autenticazione con digest.	30
Figura 2.5. Flusso esecutivo di un'autenticazione basata su form.	31
Figura 2.6. Modifica di un cookie con Burp.	33
Figura 2.7. Esempio di attacco man-in-the-middle.	34
Figura 2.8. Una form di login con controllo delle credenziali basato su database.	37
Figura 2.9. Una pagina di monitoraggio dei log vulnerabile a SQL injection.	40
Figura 2.10. Un attacco di SQL injection eseguito con l'operation UNION.	40
Figura 2.11. Funzionamento della mutua autenticazione con certificati.	43
Figura 3.1. Schema di relazione tra attore e ACS.	53
Figura 3.2. Derivazione dei permessi tramite il principio "Needs-to-Know".	53
Figura 3.3. Relazioni "Needs-to-Use" e accesso mediato.	54
Figura 3.4. Relazioni tra domini in un sistema con thin client.	54
Figura 4.1. Individuare gli attacchi nella rete.	60
Figura 4.2. Flusso di eventi in un messaggio di audit.	67
Figura 4.3. Diagramma di funzionamento del profilo ATNA.	69
Figura 5.1. Schema UML di un factory method.	76
Figura 5.2. Diagramma UML di un singleton.	77
Figura 5.3. Manipolazione dell'header tramite Live HTTP Headers.	78
Figura 6.1. Porzione di codice di login del Crono con parte del popolamento dati utente.	84
Figura 6.2. Prima versione dello schema delle tabelle utente.	85
Figura 6.3. Schema UML della classe Permission.	90
Figura 6.4. Tentativo di session fixation fallito.	95
Figura 6.5. Risposta del server a un tentativo di connessione senza certificato.	100
Figura 6.6. Certificato server auto-firmato non affidabile.	102
Figura 6.7. Seconda versione dello schema delle tabelle utente.	103
Figura 7.1. Schema delle tabelle di log.	107
Figura 7.2. Schema delle tabelle di audit trail.	110

Figura 8.1. Mappa dell'ambiente di prova.....	115
Figura 8.2. Diagramma di flusso dell'autenticazione.	116
Figura 8.3. Risposte dell'applicazione alla presenza o meno di un certificato valido.	117
Figura 8.4. Pagina account di un utente normale.	118
Figura 8.5. Aggiunta di un nuovo campo "new_field" alla lista delle variabili.	118
Figura 8.6. Pagina account di un superutente.	119
Figura 8.7. Reindirizzamento alla pagina di accesso negato.	120
Figura 8.8. Pagina di account per un utente del gruppo "tecnici"	120
Figura 8.9. Pagina di monitoraggio del log.....	121
Figura 8.10. Pagina di gestione dei gruppi.	122
Figura 8.11. Pagina di gestione e creazione dei certificati.	122

Normativa sulla privacy

Articoli 31-34, sulla sicurezza dei dati e dei sistemi

Art. 31. Obblighi di sicurezza

1. I dati personali oggetto di trattamento sono custoditi e controllati, anche in relazione alle conoscenze acquisite in base al progresso tecnico, alla natura dei dati e alle specifiche caratteristiche del trattamento, in modo da ridurre al minimo, mediante l'adozione di idonee e preventive misure di sicurezza, i rischi di distruzione o perdita, anche accidentale, dei dati stessi, di accesso non autorizzato o di trattamento non consentito o non conforme alle finalità della raccolta.

Art. 32. Particolari titolari

1. Il fornitore di un servizio di comunicazione elettronica accessibile al pubblico adotta ai sensi dell'articolo 31 idonee misure tecniche e organizzative adeguate al rischio esistente, per salvaguardare la sicurezza dei suoi servizi, l'integrità dei dati relativi al traffico, dei dati relativi all'ubicazione e delle comunicazioni elettroniche rispetto ad ogni forma di utilizzazione o cognizione non consentita.
2. Quando la sicurezza del servizio o dei dati personali richiede anche l'adozione di misure che riguardano la rete, il fornitore del servizio di comunicazione elettronica accessibile al pubblico adotta tali misure congiuntamente con il fornitore della rete pubblica di comunicazioni. In caso di mancato accordo, su richiesta di uno dei fornitori, la controversia è definita dall'Autorità per le garanzie nelle comunicazioni secondo le modalità previste dalla normativa vigente.
3. Il fornitore di un servizio di comunicazione elettronica accessibile al pubblico informa gli abbonati e, ove possibile, gli utenti, se sussiste un particolare rischio di violazione della sicurezza della rete, indicando, quando il rischio è al di fuori dell'ambito di applicazione delle misure che il fornitore stesso è tenuto ad adottare ai sensi dei commi 1 e 2, tutti i possibili rimedi e i relativi costi presumibili. Analoga informativa è resa al Garante e all'Autorità per le garanzie nelle comunicazioni.

Art. 33. Misure minime

1. Nel quadro dei più generali obblighi di sicurezza di cui all'articolo 31, o previsti da speciali disposizioni, i titolari del trattamento sono comunque tenuti ad adottare le misure minime individuate nel presente capo o ai sensi dell'articolo 58, comma 3, volte ad assicurare un livello minimo di protezione dei dati personali.

Art. 34. Trattamenti con strumenti elettronici

1. Il trattamento di dati personali effettuato con strumenti elettronici è consentito solo se sono adottate, nei modi previsti dal disciplinare tecnico contenuto nell'allegato B), le seguenti misure minime:
 - a) autenticazione informatica;
 - b) adozione di procedure di gestione delle credenziali di autenticazione;
 - c) utilizzazione di un sistema di autorizzazione;
 - d) aggiornamento periodico dell'individuazione dell'ambito del trattamento consentito ai singoli incaricati e addetti alla gestione o alla manutenzione degli strumenti elettronici;
 - e) protezione degli strumenti elettronici e dei dati rispetto a trattamenti illeciti di dati, ad accessi non consentiti e a determinati programmi informatici;
 - f) adozione di procedure per la custodia di copie di sicurezza, il ripristino della disponibilità dei dati e dei sistemi;
 - g) tenuta di un aggiornato documento programmatico sulla sicurezza;
 - h) adozione di tecniche di cifratura o di codici identificativi per determinati trattamenti di dati idonei a rivelare lo stato di salute o la vita sessuale effettuati da organismi sanitari.
- 1-bis. Per i soggetti che trattano soltanto dati personali non sensibili e che trattano come unici dati sensibili e giudiziari quelli relativi ai propri dipendenti e collaboratori, anche se extracomunitari, compresi quelli relativi al coniuge e ai parenti, la tenuta di un aggiornato documento programmatico sulla sicurezza è sostituita dall'obbligo di autocertificazione, resa dal titolare del trattamento ai sensi dell'articolo 47 del testo unico di cui al decreto del Presidente della Repubblica 28 dicembre 2000, n. 445, di trattare soltanto tali dati in osservanza delle misure minime di sicurezza previste

dal presente codice e dal disciplinare tecnico contenuto nell'allegato B). In relazione a tali trattamenti, nonché a trattamenti comunque effettuati per correnti finalità amministrativo-contabili, in particolare presso piccole e medie imprese, liberi professionisti e artigiani, il Garante, sentiti il Ministro per la semplificazione normativa e il Ministro per la pubblica amministrazione e l'innovazione, individua con proprio provvedimento, da aggiornare periodicamente, modalità semplificate di applicazione del disciplinare tecnico contenuto nel citato allegato B) in ordine all'adozione delle misure minime di cui al comma 1.

1-ter. Ai fini dell'applicazione delle disposizioni in materia di protezione dei dati personali, i trattamenti effettuati per finalità amministrativo-contabili sono quelli connessi allo svolgimento delle attività di natura organizzativa, amministrativa, finanziaria e contabile, a prescindere dalla natura dei dati trattati. In particolare, perseguono tali finalità le attività organizzative interne, quelle funzionali all'adempimento di obblighi contrattuali e precontrattuali, alla gestione del rapporto di lavoro in tutte le sue fasi, alla tenuta della contabilità e all'applicazione delle norme in materia fiscale, sindacale, previdenziale-assistenziale, di salute, igiene e sicurezza sul lavoro.

Articoli 75-76, sul trattamento dei dati personali in ambito sanitario:

Art. 75. Ambito applicativo

1. Il presente titolo disciplina il trattamento dei dati personali in ambito sanitario.

Art. 76. Esercenti professioni sanitarie e organismi sanitari pubblici

1. Gli esercenti le professioni sanitarie e gli organismi sanitari pubblici, anche nell'ambito di un'attività di rilevante interesse pubblico ai sensi dell'articolo 85, trattano i dati personali idonei a rivelare lo stato di salute:
 - a) con il consenso dell'interessato e anche senza l'autorizzazione del Garante, se il trattamento riguarda dati e operazioni indispensabili per perseguire una finalità di tutela della salute o dell'incolumità fisica dell'interessato;
 - b) anche senza il consenso dell'interessato e previa autorizzazione del Garante, se la finalità di cui alla lettera a) riguarda un terzo o la collettività.
2. Nei casi di cui al comma 1 il consenso può essere prestato con le modalità semplificate di cui al capo II.
3. Nei casi di cui al comma 1 l'autorizzazione del Garante è rilasciata, salvi i casi di particolare urgenza, sentito il Consiglio superiore di sanità.

Allegato B, disciplinare tecnico in materia di misure minime di sicurezza:

Sistema di autenticazione informatica

1. Il trattamento di dati personali con strumenti elettronici è consentito agli incaricati dotati di credenziali di autenticazione che consentano il superamento di una procedura di autenticazione relativa a uno specifico trattamento o a un insieme di trattamenti.
2. Le credenziali di autenticazione consistono in un codice per l'identificazione dell'incaricato associato a una parola chiave riservata conosciuta solamente dal medesimo oppure in un dispositivo di autenticazione in possesso e uso esclusivo dell'incaricato, eventualmente associato a un codice identificativo o a una parola chiave, oppure in una caratteristica biometrica dell'incaricato, eventualmente associata a un codice identificativo o a una parola chiave.
3. Ad ogni incaricato sono assegnate o associate individualmente una o più credenziali per l'autenticazione.
4. Con le istruzioni impartite agli incaricati è prescritto di adottare le necessarie cautele per assicurare la segretezza della componente riservata della credenziale e la diligente custodia dei dispositivi in possesso ed uso esclusivo dell'incaricato.
5. La parola chiave, quando è prevista dal sistema di autenticazione, è composta da almeno otto caratteri oppure, nel caso in cui lo strumento elettronico non lo permetta, da un numero di caratteri pari al massimo consentito; essa non contiene riferimenti agevolmente riconducibili all'incaricato ed è modificata da quest'ultimo al primo utilizzo e, successivamente, almeno ogni sei mesi. In caso di trattamento di dati sensibili e di dati giudiziari la parola chiave è modificata almeno ogni tre mesi.
6. Il codice per l'identificazione, laddove utilizzato, non può essere assegnato ad altri incaricati, neppure in tempi diversi.
7. Le credenziali di autenticazione non utilizzate da almeno sei mesi sono disattivate, salvo quelle preventivamente autorizzate per soli scopi di gestione tecnica.

8. Le credenziali sono disattivate anche in caso di perdita della qualità che consente all'incaricato l'accesso ai dati personali.
9. Sono impartite istruzioni agli incaricati per non lasciare incustodito e accessibile lo strumento elettronico durante una sessione di trattamento.
10. Quando l'accesso ai dati e agli strumenti elettronici è consentito esclusivamente mediante uso della componente riservata della credenziale per l'autenticazione, sono impartite idonee e preventive disposizioni scritte volte a individuare chiaramente le modalità con le quali il titolare può assicurare la disponibilità di dati o strumenti elettronici in caso di prolungata assenza o impedimento dell'incaricato che renda indispensabile e indifferibile intervenire per esclusive necessità di operatività e di sicurezza del sistema. In tal caso la custodia delle copie delle credenziali è organizzata garantendo la relativa segretezza e individuando preventivamente per iscritto i soggetti incaricati della loro custodia, i quali devono informare tempestivamente l'incaricato dell'intervento effettuato.
11. Le disposizioni sul sistema di autenticazione di cui ai precedenti punti e quelle sul sistema di autorizzazione non si applicano ai trattamenti dei dati personali destinati alla diffusione.

Sistema di autorizzazione

12. Quando per gli incaricati sono individuati profili di autorizzazione di ambito diverso è utilizzato un sistema di autorizzazione.
13. I profili di autorizzazione, per ciascun incaricato o per classi omogenee di incaricati, sono individuati e configurati anteriormente all'inizio del trattamento, in modo da limitare l'accesso ai soli dati necessari per effettuare le operazioni di trattamento.
14. Periodicamente, e comunque almeno annualmente, è verificata la sussistenza delle condizioni per la conservazione dei profili di autorizzazione.

Altre misure di sicurezza

15. Nell'ambito dell'aggiornamento periodico con cadenza almeno annuale dell'individuazione dell'ambito del trattamento consentito ai singoli incaricati e addetti alla gestione o alla manutenzione degli strumenti elettronici, la lista degli incaricati può essere redatta anche per classi omogenee di incarico e dei relativi profili di autorizzazione.
16. I dati personali sono protetti contro il rischio di intrusione e dell'azione di programmi di cui all'art. 615-*quinquies* del codice penale, mediante l'attivazione di idonei strumenti elettronici da aggiornare con cadenza almeno semestrale.
17. Gli aggiornamenti periodici dei programmi per elaboratore volti a prevenire la vulnerabilità di strumenti elettronici e a correggerne difetti sono effettuati almeno annualmente. In caso di trattamento di dati sensibili o giudiziari l'aggiornamento è almeno semestrale.
18. Sono impartite istruzioni organizzative e tecniche che prevedono il salvataggio dei dati con frequenza almeno settimanale.

Documento programmatico sulla sicurezza

19. Entro il 31 marzo di ogni anno, il titolare di un trattamento di dati sensibili o di dati giudiziari redige anche attraverso il responsabile, se designato, un documento programmatico sulla sicurezza contenente idonee informazioni riguardo:
 - 19.1. l'elenco dei trattamenti di dati personali;
 - 19.2. la distribuzione dei compiti e delle responsabilità nell'ambito delle strutture preposte al trattamento dei dati;
 - 19.3. l'analisi dei rischi che incombono sui dati;
 - 19.4. le misure da adottare per garantire l'integrità e la disponibilità dei dati, nonché la protezione delle aree e dei locali, rilevanti ai fini della loro custodia e accessibilità;
 - 19.5. la descrizione dei criteri e delle modalità per il ripristino della disponibilità dei dati in seguito a distruzione o danneggiamento di cui al successivo punto 23;
 - 19.6. la previsione di interventi formativi degli incaricati del trattamento, per renderli edotti dei rischi che incombono sui dati, delle misure disponibili per prevenire eventi dannosi, dei profili della disciplina sulla protezione dei dati personali più rilevanti in rapporto alle relative attività, delle responsabilità che ne derivano e delle modalità per aggiornarsi sulle misure minime adottate dal titolare. La formazione è programmata già al momento dell'ingresso in servizio, nonché in occasione di cambiamenti di mansioni, o di introduzione di nuovi significativi strumenti, rilevanti rispetto al trattamento di dati personali;
 - 19.7. la descrizione dei criteri da adottare per garantire l'adozione delle misure minime di sicurezza in caso di trattamenti di dati personali affidati, in conformità al codice, all'esterno della struttura del titolare;

- 19.8. per i dati personali idonei a rivelare lo stato di salute e la vita sessuale di cui al punto 24, l'individuazione dei criteri da adottare per la cifratura o per la separazione di tali dati dagli altri dati personali dell'interessato.

Ulteriori misure in caso di trattamento di dati sensibili o giudiziari

20. I dati sensibili o giudiziari sono protetti contro l'accesso abusivo, di cui all'art. 615-ter del codice penale, mediante l'utilizzo di idonei strumenti elettronici.
21. Sono impartite istruzioni organizzative e tecniche per la custodia e l'uso dei supporti rimovibili su cui sono memorizzati i dati al fine di evitare accessi non autorizzati e trattamenti non consentiti.
22. I supporti rimovibili contenenti dati sensibili o giudiziari se non utilizzati sono distrutti o resi inutilizzabili, ovvero possono essere riutilizzati da altri incaricati, non autorizzati al trattamento degli stessi dati, se le informazioni precedentemente in essi contenute non sono intelligibili e tecnicamente in alcun modo ricostruibili.
23. Sono adottate idonee misure per garantire il ripristino dell'accesso ai dati in caso di danneggiamento degli stessi o degli strumenti elettronici, in tempi certi compatibili con i diritti degli interessati e non superiori a sette giorni.
24. Gli organismi sanitari e gli esercenti le professioni sanitarie effettuano il trattamento dei dati idonei a rivelare lo stato di salute e la vita sessuale contenuti in elenchi, registri o banche di dati con le modalità di cui all'articolo 22, comma 6, del codice, anche al fine di consentire il trattamento disgiunto dei medesimi dati dagli altri dati personali che permettono di identificare direttamente gli interessati. I dati relativi all'identità genetica sono trattati esclusivamente all'interno di locali protetti accessibili ai soli incaricati dei trattamenti ed ai soggetti specificatamente autorizzati ad accedervi; il trasporto dei dati all'esterno dei locali riservati al loro trattamento deve avvenire in contenitori muniti di serratura o dispositivi equipollenti; il trasferimento dei dati in formato elettronico è cifrato.

Misure di tutela e garanzia

25. Il titolare che adotta misure minime di sicurezza avvalendosi di soggetti esterni alla propria struttura, per provvedere alla esecuzione riceve dall'installatore una descrizione scritta dell'intervento effettuato che ne attesta la conformità alle disposizioni del presente disciplinare tecnico.
26. Il titolare riferisce, nella relazione accompagnatoria del bilancio d'esercizio, se dovuta, dell'avvenuta redazione o aggiornamento del documento programmatico sulla sicurezza.

FAQ relative ai log sulle misure e accorgimenti prescritti ai titolari dei trattamenti effettuati con strumenti elettronici relativamente alle attribuzioni delle funzioni di amministratore di sistema, del 27 novembre 2008:

Relativamente all'obbligo di registrazione degli accessi logici degli AdS, sono compresi anche i sistemi client oltre che quelli server?

Anche i client, intesi come "postazioni di lavoro informatizzate", sono compresi tra i sistemi per cui devono essere registrati gli accessi degli AdS.

Nei casi più semplici tale requisito può essere soddisfatto tramite funzionalità già disponibili nei più diffusi sistemi operativi, senza richiedere necessariamente l'uso di strumenti software o hardware aggiuntivi. Per esempio, la registrazione locale dei dati di accesso su una postazione, in determinati contesti, può essere ritenuta idonea al corretto adempimento qualora goda di sufficienti garanzie di integrità.

Sarà comunque con valutazione del titolare che dovrà essere considerata l'idoneità degli strumenti disponibili oppure l'adozione di strumenti più sofisticati, quali la raccolta dei log centralizzata e l'utilizzo di dispositivi non riscrivibili o di tecniche crittografiche per la verifica dell'integrità delle registrazioni.

Cosa si intende per access log (log-in, log-out, tentativi falliti di accesso, altro?...)

Per access log si intende la registrazione degli eventi generati dal sistema di autenticazione informatica all'atto dell'accesso o tentativo di accesso da parte di un amministratore di sistema o all'atto della sua disconnessione nell'ambito di collegamenti interattivi a sistemi di elaborazione o a sistemi software.

Gli event records generati dai sistemi di autenticazione contengono usualmente i riferimenti allo "username" utilizzato, alla data e all'ora dell'evento (timestamp), una descrizione dell'evento (sistema di elaborazione o software utilizzato, se si tratti di un evento di log-in, di log-out, o di una condizione di errore, quale linea di comunicazione o dispositivo terminale sia stato utilizzato...).

Laddove il file di log contenga informazioni più ampie, va preso tutto il log o solo la riga relativa all'access log?

Qualora il sistema di log adottato generi una raccolta dati più ampia, comunque non in contrasto con le disposizioni del Codice e con i principi della protezione dei dati personali, il requisito del provvedimento è certamente soddisfatto. Comunque è sempre possibile effettuare un'estrazione o un filtraggio dei logfile al fine di selezionare i soli dati pertinenti agli AdS.

Come va interpretata la caratteristica di completezza del log? Si intende che ci devono essere tutte le righe? L'adeguatezza rispetto allo scopo della verifica deve prevedere un'analisi dei rischi?

La caratteristica di completezza è riferita all'insieme degli eventi censiti nel sistema di log, che deve comprendere tutti gli eventi di accesso interattivo che interessino gli amministratori di sistema su tutti i sistemi di elaborazione con cui vengono trattati, anche indirettamente, dati personali. L'analisi dei rischi aiuta a valutare l'adeguatezza delle misure di sicurezza in genere, e anche delle misure tecniche per garantire attendibilità ai log qui richiesti.

Come va interpretata la caratteristica di inalterabilità dei log?

Caratteristiche di mantenimento dell'integrità dei dati raccolti dai sistemi di log sono in genere disponibili nei più diffusi sistemi operativi, o possono esservi agevolmente integrate con apposito software. Il requisito può essere ragionevolmente soddisfatto con la strumentazione software in dotazione, nei casi più semplici, e con l'eventuale esportazione periodica dei dati di log su supporti di memorizzazione non riscrivibili. In casi più complessi i titolari potranno ritenere di adottare sistemi più sofisticati, quali i log server centralizzati e "certificati".

È ben noto che il problema dell'attendibilità dei dati di audit, in genere, riguarda in primo luogo la effettiva generazione degli auditabile events e, successivamente, la loro corretta registrazione e manutenzione. Tuttavia il provvedimento del Garante non affronta questi aspetti, prevedendo soltanto, come forma minima di documentazione dell'uso di un sistema informativo, la generazione del log degli "accessi" (login) e la loro archiviazione per almeno sei mesi in condizioni di ragionevole sicurezza e con strumenti adatti, in base al contesto in cui avviene il trattamento, senza alcuna pretesa di instaurare in modo generalizzato, e solo con le prescrizioni del provvedimento, un regime rigoroso di registrazione degli usages data dei sistemi informativi.

Cosa dobbiamo intendere per evento che deve essere registrato nel log? Solo l'accesso o anche le attività eseguite?

Il provvedimento non chiede in alcun modo che vengano registrati dati sull'attività interattiva (comandi impartiti, transazioni effettuate) degli amministratori di sistema.

Quali sono le finalità di audit che ci dobbiamo porre con la registrazione e raccolta di questi log?

La raccolta dei log serve per verificare anomalie nella frequenza degli accessi e nelle loro modalità (orari, durata, sistemi cui si è fatto accesso...). L'analisi dei log può essere compresa tra i criteri di valutazione dell'operato degli amministratori di sistema.

La registrazione degli accessi è relativa al sistema operativo o anche ai DBMS?

Tra gli accessi logici a sistemi e archivi elettronici sono comprese le autenticazioni nei confronti dei data base management systems (DBMS), che vanno registrate.

Articoli 67-70 del Capo VI sul Codice dell'amministrazione legale:

Art. 67. Modalità di sviluppo ed acquisizione

1. Le pubbliche amministrazioni centrali, per i progetti finalizzati ad appalti di lavori e servizi ad alto contenuto di innovazione tecnologica, possono selezionare uno o più proposte utilizzando il concorso di idee di cui all'articolo 57 del decreto del Presidente della Repubblica 21 dicembre 1999, n. 554.
2. Le amministrazioni appaltanti possono porre a base delle gare aventi ad oggetto la progettazione, o l'esecuzione, o entrambe, degli appalti di cui al comma 1, le proposte ideative acquisite ai sensi del comma 1, previo parere tecnico di congruità del CNIPA; alla relativa procedura è ammesso a partecipare, ai sensi dell'articolo 57, comma 6, del decreto del Presidente della Repubblica 21 dicembre 1999, n. 554, anche il soggetto selezionato ai sensi del comma 1, qualora sia in possesso dei relativi requisiti soggettivi.

Art. 68. Analisi comparativa delle soluzioni

1. Le pubbliche amministrazioni, nel rispetto della legge 7 agosto 1990, n. 241, e del decreto legislativo 12 febbraio 1993, n. 39, acquisiscono, secondo le procedure previste dall'ordinamento, programmi informatici a seguito di una valutazione comparativa di tipo tecnico ed economico tra le seguenti soluzioni disponibili sul mercato:

- a. sviluppo di programmi informatici per conto e a spese dell'amministrazione sulla scorta dei requisiti indicati dalla stessa amministrazione committente;
 - b. riuso di programmi informatici sviluppati per conto e a spese della medesima o di altre amministrazioni;
 - c. acquisizione di programmi informatici di tipo proprietario mediante ricorso a licenza d'uso;
 - d. acquisizione di programmi informatici a codice sorgente aperto;
 - e. acquisizione mediante combinazione delle modalità di cui alle lettere da a) a d).
1. Le pubbliche amministrazioni nella predisposizione o nell'acquisizione dei programmi informatici, adottano soluzioni informatiche che assicurino l'interoperabilità e la cooperazione applicativa, secondo quanto previsto dal decreto legislativo 28 febbraio 2005, n. 42, e che consentano la rappresentazione dei dati e documenti in più formati, di cui almeno uno di tipo aperto, salvo che ricorrano peculiari ed eccezionali esigenze.
 2. Per formato dei dati di tipo aperto si intende un formato dati reso pubblico e documentato esaurientemente.
 3. Il CNIPA istruisce ed aggiorna, con periodicità almeno annuale, un repertorio dei formati aperti utilizzabili nelle pubbliche amministrazioni e delle modalità di trasferimento dei formati.

Art. 69. Riuso dei programmi informatici

1. Le pubbliche amministrazioni che siano titolari di programmi applicativi realizzati su specifiche indicazioni del committente pubblico, hanno obbligo di darli in formato sorgente, completi della documentazione disponibile, in uso gratuito ad altre pubbliche amministrazioni che li richiedono e che intendano adattarli alle proprie esigenze, salvo motivate ragioni.
2. Al fine di favorire il riuso dei programmi informatici di proprietà delle pubbliche amministrazioni, ai sensi del comma 1, nei capitolati o nelle specifiche di progetto è previsto ove possibile, che i programmi appositamente sviluppati per conto e a spese dell'amministrazione siano facilmente portabili su altre piattaforme.
3. Le pubbliche amministrazioni inseriscono, nei contratti per l'acquisizione di programmi informatici, di cui al comma 1, clausole che garantiscano il diritto di disporre dei programmi ai fini del riuso da parte della medesima o di altre amministrazioni.
4. Nei contratti di acquisizione di programmi informatici sviluppati per conto e a spese delle amministrazioni, le stesse possono includere clausole, concordate con il fornitore, che tengano conto delle caratteristiche economiche ed organizzative di quest'ultimo, volte a vincolarlo, per un determinato lasso di tempo, a fornire, su richiesta di altre amministrazioni, servizi che consentano il riuso delle applicazioni. Le clausole suddette definiscono le condizioni da osservare per la prestazione dei servizi indicati.

Art. 70. Banca dati dei programmi informatici riutilizzabili

1. Il CNIPA, previo accordo con la Conferenza unificata di cui all'articolo 8 del decreto legislativo 28 agosto 1997, n. 281, valuta e rende note applicazioni tecnologiche realizzate dalle pubbliche amministrazioni, idonee al riuso da parte di altre pubbliche amministrazioni.
2. Le pubbliche amministrazioni centrali che intendono acquisire programmi applicativi valutano preventivamente la possibilità di riuso delle applicazioni analoghe rese note dal CNIPA ai sensi del comma 1, motivandone l'eventuale mancata adozione.