Politecnico di Torino

Tesina per l'esame di Sicurezza dei sistemi informatici

OpenAttestation access control

Autore:
Alessio Vallero (s192474)

Tutor:
Roberto Sassu

May 17, 2014

Indice

1 Introduzione

Negli ultimi vent'anni, Internet ha subito una rapida ed importante diffusione, grazie alla parallela crescita dell'uso dei PC in tutto il mondo. È quindi cresciuta la quantità (e la qualità) dei servizi offerti dal Web soprattutto con l'arrivo delle ben note connessioni a banda larga tramite sistemi DSL.

Purtroppo, un aspetto negativo di questo evento è stato la nascita delle minacce basate sul Web con l'obiettivo, da parte dei cybercriminali, di sfruttare eventuali vulnerabilità per ottenere una via d'accesso ai computer. Soltanto nel 2013 [1] gli attacchi sul Web si sono triplicati rispetto ai 24 mesi precedenti, crescendo quindi di oltre il 370% considerando soltanto attacchi di dimensioni significative, dove le conseguenze si riflettono a livello economico, legale e di immagine. Non sono però soltanto questi gli unici obiettivi per cui i cybercriminali agiscono.

Riferendosi sempre alle statistiche dell'ultimo anno, fonti di rilievo come la CNN (emittente televisiva statunitense all-news visibile via cavo nell'America centrosettentrionale e, in tutto il resto del mondo, grazie alla tecnologia satellitare), hanno rilevato [2] un aumento, rispetto al 2012, degli attacchi tramite malware e URL malevoli (anche su dispositivi mobili come smartphones e tablets, anch'esse piattaforme in rapida ascesa di utilizzo) con lo scopo di inviare dati privati come le password agli attaccanti; oppure inserendo il PC in una rete di computer infetti che il cybercriminale potrà poi usare per effettuare ulteriori attacchi in futuro.

Tra le contromisure che si stanno studiando per questi problemi, risulta in significante crescita l'utilizzo delle procedure di Remote Attestation (RA), ossia un processo che consente di accertare l'integrità dei componenti di un macchina attraverso la rete mediante l'uso di opportuni hardware e software in modo da poter garantire la realizzazione della cosiddetta informatica fidata (Trusted Computing) sul dispositivo che la utilizza. L'obiettivo dichiarato dell'informatica fidata è quello di produrre computer più sicuri mediante l'uso di opportuni hardware e software. Il termine deriva da trust (in italiano fiducia) ma assume un significato particolare: non significa che sia affidabile dal punto di vista dell'utente, ma piuttosto che debba essere considerato fidato secondo i canoni imposti dai produttori di tali dispositivi.

Questa tecnologia è ancora in corso di sviluppo, ed è nata dalle specifiche del Trusted Computing Group, consorzio statunitense nato nel 2003 e formato da un gruppo di membri fissi quali AMD, Hewlett-Packard, IBM, Infineon, Intel, Lenovo, Microsoft, Sun Microsystems e, dal 2008 in poi, anche da Fujitsu Limited, Seagate Technology e Wave Systems.

Intel in particolare, ha creato un SDK (Software Development Kit) chiamato OpenAttestation che si basa su questa tecnologia emergente. È scritto con il linguaggio Java e permette di gestire la verifica d'integrità, ossia la protezione dei dati e delle informazioni nei confronti delle modifiche del contenuto, accidentali oppure effettuate da una terza parte, degli host di un cloud (servizio offerto ai clienti da un provider, che permette di effettuare un'elaborazione e/o un'archiviazione dei dati tramite CPU e software distribuiti attraverso la rete), ponendosi come intermediario tra gli host che gestisce ed il software di clouding. Il primo fornisce la possibilità di richiedere una verifica sull'affidabilità di uno o più dei suoi host, che per contro devono quindi fornire l'elenco delle proprie misure.

Tra i componenti di OpenAttestation (d'ora in poi chiamato OAT, per brevità) ci sono, come vedremo con maggiori dettagli nella prossima sezione, tre API RESTful che per ragioni di sicurezza si basano su HTTPS. Tuttavia, l'implementazione interna della versione in esame, la 1.6, non impone alcun meccanismo di autenticazione nè di controllo degli accessi, pertanto qualunque utente ha la possibilità di utilizzare tutti i servizi offerti senza essere riconoscibile dal sistema. L'obiettivo è quindi quello di integrare in OAT queste funzionalità, ottenendo il riconoscimento dell'utente e, sulla base della sua identità, garantendo (con un termine inglese

detto “enforcement”) che egli possa compiere solo le azioni per cui è stato opportunamente autorizzato. L’autenticazione infatti, consente di determinare se un utente è in effetti chi afferma di essere, mentre l’autorizzazione permette di determinare se un utente può accedere ad una particolare risorsa (partendo quindi dal presupposto di conoscerne l’identità tramite un precedente processo di autenticazione) e che cosa è permesso fare agli utenti non autenticati. Per ottenere queste funzionalità, verrà modificata opportunamente la struttura del database di OAT, verranno aggiunti i servizi per la creazione degli utenti e delle relative autorizzazioni che possiede sul sistema, verrà aggiornato il servizio delle richieste d’attestazione rendendolo soggetto esplicitamente ad autenticazione ed autorizzazione e verranno automaticamente filtrate alcune informazioni statistiche presentate da OAT. Verrà quindi resa disponibile la possibilità di creare gli utenti subito dopo aver installato OAT, impostando i privilegi, le azioni eseguibili ed il formato delle stesse. Pertanto sarà possibile:

- Rendere i comandi attuali autenticati, comprese le richieste di attestazione;
- Filtrare le informazioni già esistenti in base ai privilegi previsti.

Saranno inoltre aggiunte delle informazioni statistiche riguardo le richieste effettuate da un utente.

2 Background

Per comprendere meglio il lavoro necessario per raggiungere i nostri scopi, occorre presentare i concetti su cui si dovrà operare e conoscere le funzionalità di OAT coinvolte. Vedremo quindi ora di presentare in questa sezione tutte le informazioni necessarie.

2.1 Il Trusted Computing

Il concetto di informatica fidata/sicura (il cosiddetto Trusted Computing, d’ora in poi citato come TC per brevità), nasce con l’obiettivo [3] di fornire una piattaforma informatica che permetta di rilevare eventuali problemi sul sistema in uso, rendendo i computer più sicuri mediante l’uso di opportuni hardware e software.

Le specifiche del TC vennero rilasciate dal Trusted Computing Group (TCG), un consorzio formato da importanti aziende quali IBM, AMD, Intel, Microsoft, Sony e Sun Microsystems e sono tutt’ora oggetto di acceso dibattito da parte degli esperti, poichè la loro definizione di sicurezza segue una direzione diversa dalle altre tecnologie finora esistenti. Il TC infatti, mira a garantire le funzionalità sopracitate utilizzando alcuni meccanismi sul sistema e rendendolo di fatto più sicuro per l’utente ma in un certo senso anche “contro”lo stesso, poichè in questo modo i dispositivi che implementeranno tale tecnologia potranno, oltre che proteggere il software da manomissioni, imporre restrizioni su applicazioni ritenute non desiderabili dai produttori (ad esempio perchè inaffidabili, pericolose per l’utente o per gli interessi del produttore stesso); concetto che può senz’altro mettere a rischio la privacy degli utenti stessi e rendere difficoltoso l’utilizzo di software senza licenza [4], fattore che può risultare inaccettabile per qualcuno.

Il più importante componente di tipo hardware adottato nel TC, è un microchip conosciuto con il nome Trusted Platform Module (TPM). Esso viene fabbricato seguendo le buone pratiche di ingegnerizzazione e di revisione industriale necessarie e racchiude una serie di funzioni, dette “Root of Trusts”[5] che vengono sempre considerate fidate dal sistema operativo e che servono a controllare il processore crittografico. Ciascuna di queste funzioni fornisce le principali funzionalità del TC, tra cui:

- Crittografia dei dati;
- Rilevamento e segnalazione di modifiche non autorizzate al sistema operativo od ai programmi;
- Rilevamento di rootkits.
- Protezione hardware della memoria per prevenire letture o scritture inappropriate su altre parte della memoria.
- Supporto hardware per la gestione dei diritti digitali.

Le funzioni che devono essere implementate da una piattaforma, per poterla considerare fidata, sono sostanzialmente tre: realizzazione, memorizzazione e report dei livelli d'integrità, atti a quantificare l'attuale riduzione dei rischi. Le Root of Trust rappresentano dunque il primo elemento di una catena fiduciaria atta ad attestare l'affidabilità dell'intero sistema, per questo motivo il TPM si occupa di proteggerle da manomissioni esterne. Fanno eccezione solo alcune parti che prendono il nome di Trusted Building Block, che solitamente includono soltanto le funzioni di inizializzazioni del chip.

2.2 Remote Attestation

La Remote Attestation (RA) è il processo che consente di accertare l'integrità dei componenti di un macchina attraverso la rete e può essere usato per cercare di tutelare da numerosi problemi di fiducia in termini di sicurezza, quali l'esecuzione dei software, il rilascio di contenuti ai clienti ed in generale tutti quei contesti in cui è assolutamente necessario essere certi dell'identità di un interlocutore.

Il TCG ha definito tre elementi importanti per il processo d'attestazione:

- il Requestor: elemento che fornisce informazioni o richieste al Verifier;
- il Verifier: elemento che riceve le informazioni dal Requestor e si occupa di elaborarle;
- la Relying Party: elemento che utilizza l'esito elaborato dal Verifier.

Le misure scambiate fra Requestor e Verifier avranno forma d'attestazione remota di tipo binary attestation, atta a valutare l'integrità dei componenti attraverso il digest del codice eseguito e dai file di configurazione. Per effettuare le misurazioni in ambiente Linux, viene sfruttato l'Integrity Measurement Architecture (IMA), componente sviluppato da IBM ed incluso nel kernel. Esso usa il file che deve essere acceduto dall'applicativo come input per una funzione di hash SHA-1, ottenendo una stringa univoca da 160 bit.

In ogni caso, affinchè il Verifier possa contare su misure affidabili occorre evitare qualunque alterazione, sia interne al programma e sia provenienti da terze parti. Vedremo tra poco il meccanismo con cui è possibile ottenere questa importante garanzia.

Osservando questi elementi (schematizzati in Fig. 1) si può evincere come l'obiettivo sia quindi quello di provare ad una terza parte (remota) che il proprio sistema operativo, con i relativi applicativi software, è intatto ed affidabile.

Ovviamente non è possibile conoscere a priori tutte le applicazioni esistenti in un dato momento, per cui il Verifier deve riconoscere quali siano i software in esecuzione sulla macchina che si desidera attestare e quest'ultima, tramite IMA, deve poi inviare le misure. Pertanto

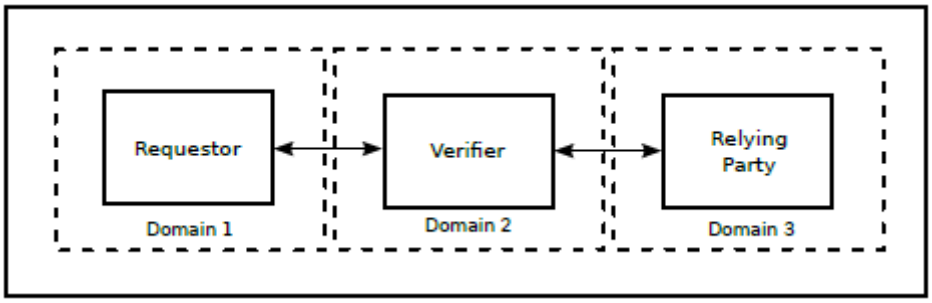

Figura 1: Modello TCG di attestazione remota - Citata da [6]

è necessario che i sistemi di attestazione siano sufficientemente flessibili, e che garantiscano privacy e completezza nelle misure d'attestazione.

Sono stati quindi definiti [7] cinque principi fondamentali che vanno seguiti dalle architetture di attestazione:

- Le informazioni devono essere aggiornate: ossia rispecchiare l'attuale sistema in esecuzione.
- Le informazioni devono essere complete: i meccanismi d'attestazione devono fare in modo che le informazioni siano accessibili a tutti gli strumenti di valutazione interni.
- Devono esserci vincoli sulla divulgazione: deve essere possibile applicare meccanismi di controllo su quali misure vanno inviate all'entità che dovrà iniziare il processo di attestazione. Tali politiche, devono poter distinguere il tipo di informazione da rilasciare e permettere di richiedere lo stato dell'attestatore prima di rilasciarle allo stesso. Questo principio può essere in qualche modo visto come una parziale perdita della privacy, poichè può esporre la piattaforma a degli attacchi nati in seguito alla rilevazione delle vulnerabilità attuali del sistema.
- La semantica utilizzata deve essere chiara ed esplicita: principio fondamentale per provare l'identità di componente. In questo modo infatti, l'attestatore può prendere decisioni più precise durante il processo di attestazione.
- Il meccanismo deve essere affidabile: il servizio di attestazione deve poter identificare in modo inequivocabile ciascun componente.

È possibile far sì che la decisione di giudicare affidabile una parte del codice di un software venga presa basandosi su una WhiteList (un elenco di software conosciuti per essere affidabili). Questo perchè una BlackList potrebbe venir manipolata periodicamente anche solo di pochi bytes ed esser così resa ingestibile a causa della sua dimensione. Tuttavia, gestire una WhiteList di software affidabili può risultare davvero complicato e questo può essere considerato come un limite sugli attuali protocolli di attestazione. Come d'altronde può essere considerato un limite il fatto che ogni modifica o aggiornamento sui software vada comunque registrata sul sistema di attestazione ricalcolando gli hash per i verificatori; questo problema può essere considerato gestibile solo se tali modifiche non impattino in modo arbitrario su tutto il codice, poichè in questo caso il numero di file binari da verificare nel sistema crescerebbe in modo esponenziale, creando grossi problemi di gestione.

Vedremo un caso reale di sistema di Remote Attestation, descrivendo la procedura d'attestazione eseguita da OAT.

2.3 Il Cloud Computing

Il Cloud Computing è una tecnologia complessa e di ampio respiro, le cui caratteristiche distintive sono di difficile individuazione e per cui ad oggi non esiste una definizione globalmente accettata. In termini semplicistici, il Cloud Computing è una tecnologia che permette il salvataggio e l'accesso di dati e programmi attraverso la rete Internet anzichè dall'Hard Disk del proprio computer. Da qui il termine "Cloud", usato come metafora per indicare, mediante la rappresentazione con diagrammi di flusso, la gigantesca struttura lato server di Internet come una nuvola che accetta connessioni e collegamenti e che distribuisce informazioni.

Va comunque sottolineato che il Cloud Computing non è un disco rigido del nostro computer. Quando vengono salvati dei dati oppure eseguiti dei programmi dal nostro disco rigido, si sta utilizzando una risorsa locale, poichè tutto ciò che ci serve è fisicamente vicino a noi, permettendo accessi rapidi e semplici ai dati. Il Cloud non è neanche un server personale dedicato. Salvare dei dati in una rete casalinga o lavorativa non significa che si sta utilizzando un Cloud.

Per essere considerato Cloud infatti, l'accesso a dati e programmi deve essere effettuato sulla rete Internet o, quanto meno, tali dati devono essere sincronizzati con altre informazioni sulla rete.

Esiste tuttavia un concetto di Cloud completamente differente quando si sta ragionando in termini aziendali. Alcune aziende infatti, scelgono di implementare il cosiddetto Software-as-a-Service (SaaS), dove le stesse aziende iscrivono un'applicazione all'accesso ad Internet. Oppure il Platform-as-a-Service (PaaS), dove un'azienda crea le proprie applicazioni personalizzate per farle utilizzare a tutti nella propria compagnia. Oppure ancora la Infrastructure-as-a-Service (IaaS), dove l'azienda fornisce la propria struttura come spina dorsale "affitabile"da altre aziende.

Il Cloud Computing quindi è ormai diventato un vero e proprio business.

2.4 OpenAttestation

Ora che sono stati trattati i concetti generali di Trusted Computing, Remote Attestation e Cloud Computing è possibile entrare nel dettaglio di OAT; soffermandosi in particolare sul funzionamento della Remote Attestation in questo prodotto, ma anche sulle funzionalità su cui si andrà ad operare per raggiungere gli scopi che ci siamo prefissati.

OAT è un Software Development Kit (SDK) sviluppato da Intel per gestire la verifica d'integrità degli host di un cloud. Si basa sull'Host Integrity at Startup (HIS) messo a punto dal National Information Assurance Research Laboratory (NIARL), che consente di misurare e riportare lo stato di piattaforme equipaggiate con il TPM [8]. Il servizio che viene offerto ha quindi come destinatario generiche applicazioni di clouding sviluppate da terze parti, a cui vengono fornite API per ottenere e verificare i PCR di host specifici, valutandone così l'integrità. Il servizio d'attestazione offerto, si pone come intermediario tra il software di clouding e gli host che questo gestisce. Il primo ha la possibilità di richiedere una verifica sull'affidabilità di uno o più dei suoi host, che per contro devono fornire l'elenco delle proprie misure.

2.4.1 I componenti principali

Sullo schema in Fig. 2 si può notare come OAT disponga di 6 componenti fondamentali:

- API HTTPS per il servizio di attestazione: utili per attestare lo stato degli Hosts abilitando il software ISV a richiedere lo stato di integrità degli stessi. È un servizio fondamentale di OAT che vedremo nel dettaglio tra poco e che sarà oggetto di modifiche per aggiungere l'autenticazione alle richieste di attestazione;
- API HTTPS per il servizio della WhiteList: fornisce le API per gestire una lista di misure note e affidabili. Va utilizzato in fase di configurazione post-installazione di OAT e verrà opportunamente aggiornato con nuovi servizi che ci permetteranno le registrazione (e la eventuale successiva modifica o eliminazione) degli utenti;
- Historical Integrity Reporting Portal: portale scritto in PHP che permette un'ulteriore interfacciamento con il software di clouding, consentendo l'accesso ad una cronologia dei report d'integrità forniti dagli host. Anch'esso oggetto di modifica, come vedremo, in modo tale da visualizzare solo le informazioni a cui l'utente autenticato può avere accesso;
- API HTTPS per gli HostAgent: utilizzata per realizzare la meccanica interna del SDK (invio dei report);
- PrivacyCA: elemento che verifica l'EK degli host e che produce il relativo certificato (EK Certificate) che gli host a loro volta utilizzano per richiedere un AIK Certificate (AIC), ossia un certificato legato alla cosiddetta Attestation Identity Key, chiave utilizzata per firmare le informazioni prodotte dal TPM durante l'esecuzione del protocollo per la RA;
- Appraiser: è il motore di OAT e si occupa di verificare le misure ricevute dagli host.

Tutti questi componenti sono corredati da una documentazione esplicativa piuttosto scarsa, che può senz'altro rallentare la fase di sviluppo e che testimoniano come il codice sia ancora in fase di evoluzione. Tuttavia, per i nostri scopi, sarà necessario modificare in modo sensibile solamente alcuni di questi elementi, che analizzeremo quindi tra poco con maggior dettaglio.

2.4.2 Il CommandTool

Introducendo i componenti principali di OAT abbiamo avuto modo di incontrare tre API RESTful. Per utilizzarle correttamente, vengono sfruttati degli script BASH presenti nella cartella CommandTool di OAT. Essi effettuano un primo controllo sulla correttezza dei parametri passati prima di effettuare la chiamata ai Web Service RESTful ed segnalano eventuali problemi all'utente suggerendo il formato corretto con cui utilizzare il comando. In secondo luogo effettuano la chiamata al Web Service vero e proprio, gestendo in modo trasparente all'utilizzatore la parte di autenticazione del server mediante certificato (generato proprio tramite un comando del CommandTool in fase di configurazione post-installazione) tramite il tool "curl".

Dovremo quindi inserire un nuovo comando che permetta di aggiungere, modificare o eliminare gli utenti ed i loro permessi, e modificare il comando già esistente atto ad effettuare le richieste di attestazione, il cosiddetto "pollhosts".

2.4.3 Le API RESTful

Come già accennato, OAT dispone di 3 API RESTful (Whitelist, Attestation Service e HostAgent). Ci soffermeremo ora nell'introduzione delle prime due, in quanto oggetto delle nostre modifiche per l'integrazione della nuova API di registrazione degli utenti e la relativa richiesta di attestazione autenticata.

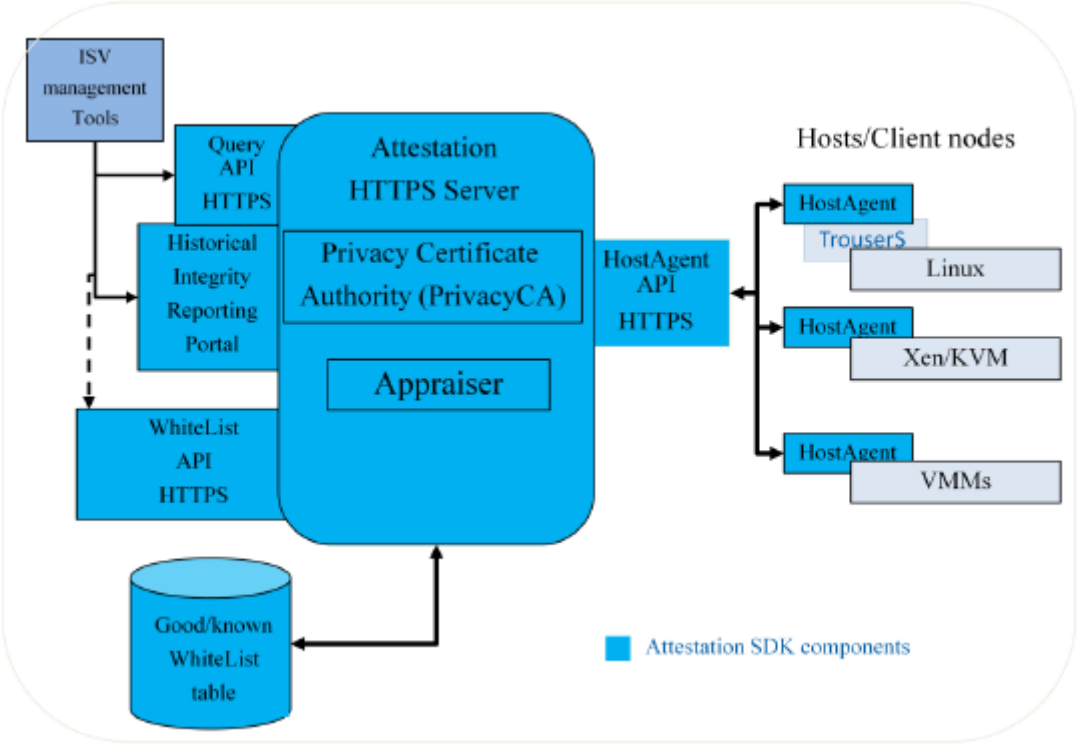

Figura 2: Componenti del servizio di attestazione di OpenAttestation - Citata da [8]

Whitelist

Questa API non si occupa solo della lista da cui prende il nome, ma agisce anche su altri quattro elementi di OAT: Operating System (OS), Original Equipment Manufacturer (OEM), Measured Launch Environment (MLE) e, appunto, la WhiteList.

I primi due comandi consentono all'utente di manipolare le informazioni relative al sistema operativo (nome, versione e descrizione) ed al produttore (nome e descrizione), informazioni che potranno poi essere associate agli host del cloud, il terzo comando invece è leggermente più articolato e di fatto permette l'accesso alle informazioni sul MLE, ovvero il primo modulo del sistema operativo che andrà ad essere misurato. Esso infatti può assumere il valore VMM oppure BIOS (da notare che, a seconda del valore impostato, variano le altre informazioni richieste per la registrazione).

In questa API andremo ad aggiungere, come vedremo nel dettaglio nella prossima sezione, i servizi della nuova API di registrazione degli utenti, opportunamente chiamati medianti nuovi comandi inseriti nel CommandTool.

Attestation Service

Quest'altra API si occupa di fare da tramite tra il software di clouding ed il servizio d'attestazione. Essa infatti consente di inoltrare una richiesta di attestazione, registrare (POST), modificare (PUT), cancellare (DELETE) ed ottenere (GET) le informazioni sugli host del cloud. Si possono associare molteplici attributi all'HostName, da indirizzo IP e porta tramite cui contattare l'host, ai riferimenti sul suo sistema operativo.

Il servizio su cui andremo ad operare però, che è poi anche il fulcro di tutta l'API, è quello che si occupa della richiesta di attestazione, inoltrata tramite POST con il comando PollHosts. Essa riceve in input una lista di host per cui richiedere una verifica d'integrità e fornisce come output, per ogni host della richiesta, tre informazioni: l'host name, la data in cui è stata fatta la richiesta di attestazione (vtime) ed il trust lvl, che rappresenta il livello di affidabilità dell'host

e che può assumere i valori trusted, untrusted, unknown, pending o timeout. Un esempio di richiesta e risposta di questo comando può quindi essere il seguente:

```
1  {
2  "hosts": [
3      "localhost",
4      "ubuntu1104"
5      ]
6  }
7  {
8      "hosts": [{
9      "host_name": "localhost",
10     "trust_lvl": "trusted",
11     "vtime": "2012-10-15T22:36:58.836+08:00"
12 },
13 {
14     "host_name": "ubuntu1104",
15     "trust_lvl": "trusted",
16     "vtime": "2012-10-15T22:36:58.836+08:00"
17 }]
18 }
```

Listato 1: Esempio di richiesta d'attestazione e relativa risposta (citato da [8])

Come già accennato, andremo ad aggiungere a questo servizio un meccanismo di autenticazione, in modo tale che queste richieste possano essere effettuate soltanto dagli utenti per cui è stato previsto questo tipo di permesso.

OAT adotta un meccanismo di attestazione iniziato dall'host, che di fatto interroga l'Appraiser (il motore di questo SDK, ha il compito di verificare le misure ricevute dagli host) periodicamente per accertarsi della presenza di nuove richieste d'attestazione. Il processo, ben schematizzato nella documentazione di OAT (vedi Fig. 3), è sintetizzabile in 4 fasi:

- L'host invio la richiesta all'Appraiser;
- L'Appraiser risponde all'host con un nonce (numero casuale utilizzato solo una volta) e con i PCR richiesti, ottenendoli da una maschera di bit presente nei file di configurazione, chiamata PCR SELECT;
- L'host ottiene dal TPM una Quote (elemento contenente HostName, i PCR richiesti ed il nonce, tutti derivanti dal punto precedente) cifrata con la chiave privata dell'AIK;
- L'Appraiser avvia una serie di verifiche sulle informazioni ricevute. Con l'HostName estrae l'AIC dal suo database e lo verifica in base al certificato ottenuto dalla PrivacyCA. L'AIC è quindi utile per la verifica della firma sulla Quote, il cui contenuto va anch'esso validato controllando la corrispondenza del nonce ricevuto con quello inviato inizialmente. Infine, viene validato il contenuto dei PCR con il report precedente e con le entry della WhiteList.

2.4.4 Il portale

Il cosiddetto Historical Integrity Reporting Portal di OAT, sviluppato con il linguaggio PHP, permette di godere di una panoramica più dettagliata sullo stato di salute degli host e sulle varie informazioni relative al sistema d'attestazione. Tramite questo portale, è infatti possibile visualizzare la lista dei report ricevuti (Fig. 4) con un ampio intervallo di informazioni utili,

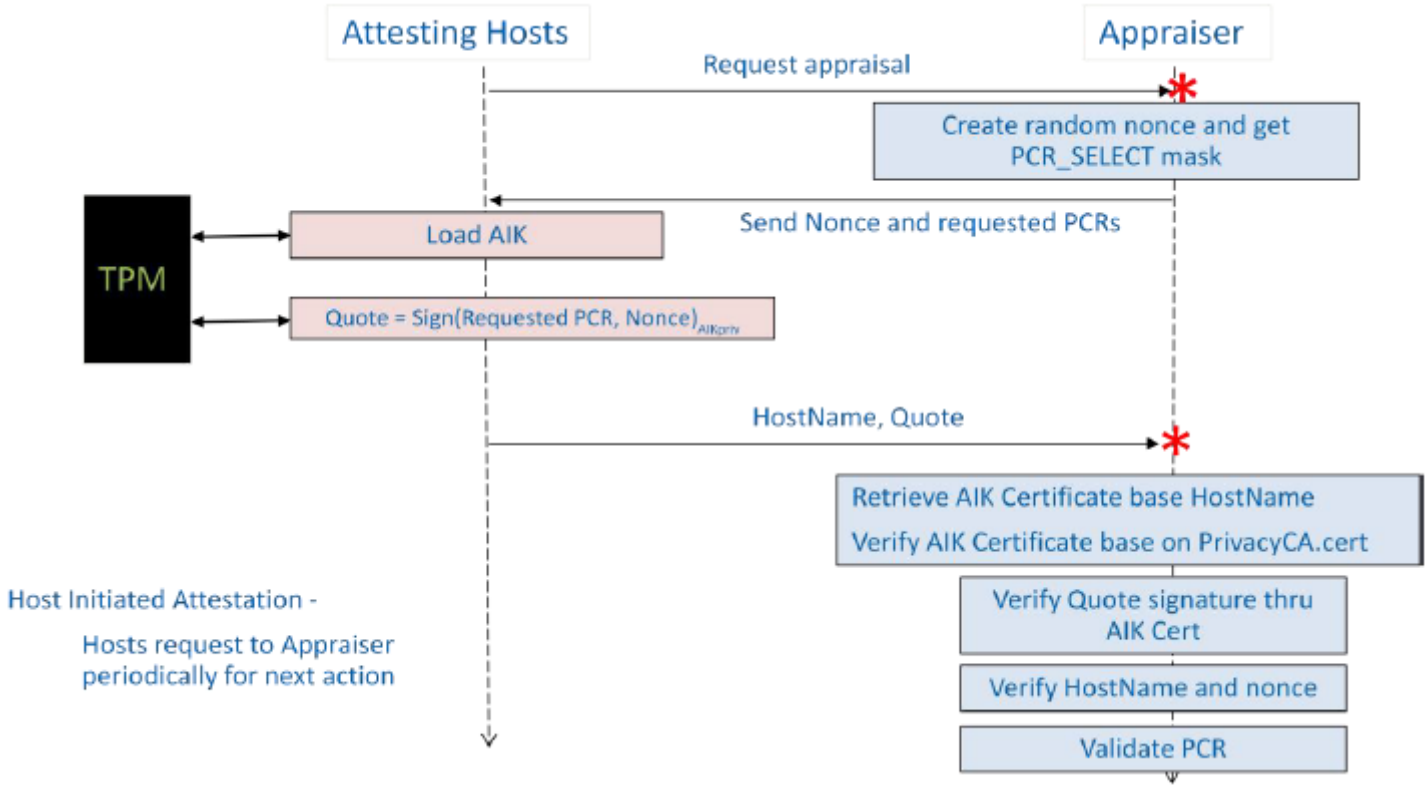

Figura 3: Attestazione remota in OpenAttestation - Citata da [8]

Report		PCR	Sig	Timestamp	Machine	User
	8			2014-02-08 14:06:55	node-110	root
	7			2014-02-08 13:06:47	node-110	root
	6			2014-02-08 13:06:47	node-110	root
	5			2014-02-05 23:15:32	node-110	root
	4			2014-02-05 23:12:10	node-110	root
	3			2014-02-05 20:45:35	node-110	root
	2			2014-02-04 11:47:24	node-110	root
	1			2014-02-04 11:47:15	node-110	root

Figura 4: Lista dei report sull'Historical Integrity Reporting Portal

quali la macchina e l'utente che l'ha generato, la data della sua creazione, l'esito delle verifiche su PCR e firma digitale oltre alla visualizzazione completa in formato XML dello stesso.

Il portale organizza anche in modo molto semplice, ma allo stesso tempo ordinato ed efficace, tutte le altre statistiche che possono essere utili in fase di analisi. Sono infatti presenti:

- gli Alert per le verifiche non andate a buon fine, con i relativi report collocati in forma tabellare;
- una lista completa delle macchine registrate presso il servizio di attestazione con le relative informazioni d'integrità;
- i valori più frequenti riscontrati per ciascun PCR;
- le statistiche riassuntive delle attestazioni portate a termine.

Le modifiche apportate, riguardano la verifica dei permessi di chi cerca di utilizzare le pagine del portale (vedremo nella prossima sezione come) ed il filtering automatico di quali report di integrità è possibile visualizzare, sempre sulla base dei permessi dell'utente utilizzatore. È stata inoltre aggiunta una nuova pagina per visualizzare le richieste d'attestazione effettuate da ciascun utente.

2.4.5 Vantaggi e svantaggi di OAT

Il punto di forza principale di questa versione di OAT, è senz'altro l'Historical Integrity Reporting Portal, che permette in maniera piuttosto semplice ed intuitiva, di avere accesso alle informazioni di lavoro più importanti per un utilizzatore, come descritto poco sopra.

Un altro fattore importante è poi il trust level restituito dal comando PollHost per la richiesta delle attestazioni, che permette di sfruttare, per chi lo desidera, meccanismi di reazione automatica in base allo stato dell'host; ma è comunque possibile utilizzarlo più semplicemente per finalità diagnostiche.

Una pecca è invece, come già segnalato in precedenza, la scarsa documentazione reperibile per questo SDK, fattore che penalizza sicuramente la fase di sviluppo e che inoltre può influire negativamente e quindi "spaventare"eventuali nuovi utilizzatori che dovessero deciderne l'integrazione sui loro prodotti già esistenti. Questo fattore non può essere sottovalutato in ottica di una futura crescita della comunità di utilizzatori del prodotto, che possono aiutare a migliorare e velocizzare l'evoluzione del SDK.

Un aspetto importante è proprio la mancanza del concetto di utente utilizzatore e quindi di accesso ed utilizzo autenticato delle risorse dall'esterno (voluto da parte degli sviluppatori Intel di OAT per concedere agli utilizzatori la completa libertà di implementazione di un proprio meccanismo personalizzato). Tuttavia questo può essere considerato un lato positivo poichè, in questo modo, chi utilizza il prodotto può creare tale servizio in base alle proprie precise esigenze (proprio come è stato fatto in questa sede).

3 Nuova API per il controllo degli accessi

Ora che sono stati presentati gli elementi su cui si è andato ad operare, è possibile iniziare ad esporre lo sviluppo che è stato fatto sul codice.

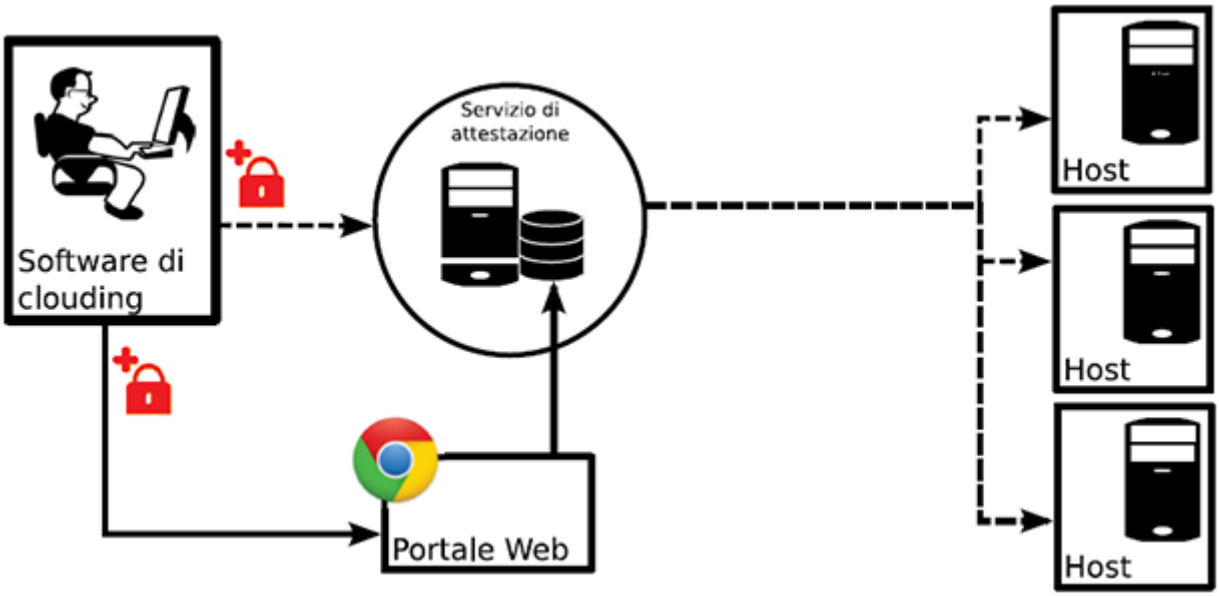

Figura 5: Modello del nuovo controllo degli accessi su OAT

L'obiettivo è introdurre l'autenticazione sugli attuali servizi resi disponibili da OAT ed impostare un meccanismo di controllo dei permessi attivi sugli stessi. Inizialmente verrà applicato solo su una parte dei servizi esistenti; ossia sulle richieste di attestazione, sul portale (nei report d'integrità e nella nuova pagina con le richieste di attestazione) e sulle nuove API introdotte per definire gli utenti.

Il modello risultante in OAT (schematizzato in Fig. 5), farà quindi sì che ciascun utente possa:

- Eseguire le azioni solo se previste e se i relativi parametri di ingresso sono in un formato valido;
- Visualizzare solo le sue informazioni sul portale, mediante un filtro automatico basato sulle credenziali di accesso.

In questo modo, anche le richieste d'attestazione potranno essere fatte solo da chi ne ha il permesso e solo sugli host definiti, rendendo quindi possibile tracciarne lo storico e legandolo all'utente che le ha effettuate ed aggiungere ulteriori analisi statistiche finora impraticabili senza l'esistenza del concetto di utente all'interno del SDK.

Per introdurre questa funzionalità, è quindi necessario aggiungere:

- Alcune API per la registrazione degli utenti, da utilizzare dopo aver installato l'SDK, con annessa impostazione dei permessi previsti;
- I parametri per l'accesso autenticato sui servizi esistenti;
- Un meccanismo di autenticazione sul portale, con cui filtrare in automatico le informazioni visibili in base all'utente che ha fatto l'accesso, compresa la nuova pagina contenente le richieste d'attestazione.

3.1 Autenticazione dei Client in OAT

Prima di vedere nel dettaglio come sono state sviluppate le funzionalità che ci servono, introduciamo le due possibili tecniche di autenticazione per i client che sono state oggetto della nostra valutazione durante la fase di analisi iniziale e le problematiche incontrate.

Per integrare un meccanismo di autenticazione, occorre configurare opportunamente due software utilizzati da OAT: l'HTTP Server ed il Tomcat di Apache. L'Historical Integrity Reporting Portal si poggia su Apache HTTP Server e gli amministratori possono lavorare [9] inserendo le direttive desiderate in comuni file di testo, tra i quali ssl.conf oppure il file .htaccess, il cui utilizzo è necessario per la gestione decentralizzata della configurazione e che va inserito direttamente nell'albero del portale Web dove deve agire. In questo file è possibile abilitare i più comuni tipi di autenticazione quali il Secure Sockets Layer (SSL) o la Basic Authentication, che sono state entrambe oggetto della nostra valutazione e che vedremo nel dettaglio tra poco.

Se si decide di utilizzare il SSL è inoltre necessario configurare anche Apache Tomcat, utilizzato da OAT per gestire i Web Services di tipo RESTful introdotti in precedenza.

Per creare i certificati x.509 dei client, è preferibile prevedere un formato universale che permetta di utilizzare senza problemi tali certificati anche al di fuori di OAT, caratteristica che garantirebbe una certa flessibilità nell'utilizzo, dato che in questo modo sarebbe di conseguenza consentito anche l'utilizzo di certificati già esistenti.

Occorre segnalare che OAT, in fase di installazione, utilizza una sua Certification Authority con la quale crea un certificato self-signed per il server. Sarebbe quindi necessario modificare la procedura di installazione, in modo tale da poter definire una CA vera e propria di OAT, da usare non soltanto nella generazione del certificato per il server, ma anche quelli per i client. Il carico di lavoro richiesto in questo senso è molto elevato rispetto alle tempistiche a disposizione e pertanto si è scelto di optare, in prima battuta, sulla Basic Authentication (che in fase di rilascio futuro verrà sostituita con l'autenticazione tramite certificati). In questo modo è comunque possibile testare la robustezza dell'algoritmo del nuovo controllo degli accessi e dei permessi, obiettivo principale che ci si è posti in questa sede.

3.1.1 Certificato SSL x.509 per i Client

Per integrare tale soluzione, di fatto è necessario che il nuovo comando, denominato oat_user, crei un certificato per l'utente da utilizzare per autenticarsi sia durante le richieste d'attestazione della PollHost sia nel Historical Integrity Reporting Portal.

Questo tipo di client authentication richiede un certificato nel formato x.509 per ciascun client contenente le informazioni dell'utente.

La CA rilascia quindi un certificato che accoppia una chiave pubblica di cui sopra ad un Nome Distintivo (Distinguished Name) seguendo la tradizione del X.500. In generale, il Nome Distintivo identifica univocamente un'entità, e contiene le seguenti informazioni (tutte riferite all'entità presentata nel certificato):

- Common Name (CN): un nome che può distinguere un singolo utente oppure una qualsiasi altra entità, come un DNS o un Web Server;
- Title (T): il titolo associato;
- Organization name (O): il nome dell'organizzazione di cui fa parte;
- Organizational Unit name (OU): unità dell'organizzazione, che può contenere più valori, ma con un'istanza soltanto per ogni entry. L'ordine in questo caso è molto importante, poichè viene utilizzato come gerarchia delle unità dell'organizzazione, con la prima atta a rappresentare il livello più alto;
- Locality name (L): località;

- State or Province name (S): stato o provincia;
- Country (C): paese.

L'RFC3280 [10] non specifica quali campi siano obbligatori e quali no, ma il Nome Distintivo dell'emittente non deve essere vuoto.

L'idea, nel nostro caso, è quella di sfruttare il Nome Distintivo per identificare univocamente un utente.. È ben noto come la client authentication avvenga quando il server richiede al client il relativo certificato, durante la fase di SSL handshake sulla rete. Da questo si può evincere come sia il server a richiedere al client di autenticarsi e non viceversa.

Un'ipotesi di sviluppo quindi potrebbe essere questa:

- Modificare la procedura d'installazione inserendosi in tale meccanismo, creando un'opportuna CA da utilizzare per creare i certificati lato client;
- Generare tali certificati in fase di creazione degli utenti;
- Abilitare il Tomcat di OAT ad autenticare mediante certificati X.509 per i client nelle chiamate ai servizi RESTful (ed usarli nei comandi che prevedono autenticazione);
- Abilitare in Apache HTTP Server l'autenticazione attraverso certificati X.509.

Appare evidente come, a parte per la creazione del certificato, occorre agire lato server per abilitare questo meccanismo, inserendo nel file ssl.conf di Apache HTTP Server le seguenti direttive (ove, ovviamente, tutti i nomi e percorsi, devono corrispondere a quelli realmente utilizzati):

```
1 SSLCertificateFile /u01/app/myCA/certs/my_certificate.crt
2 SSLCertificateKeyFile /u01/app/myCA/private/my_key.key
3 SSLCertificateChainFile /u01/app/myCA/certs/rootCA.crt
4 SSLCACertificateFile /u01/app/myCA/certs/rootCA.crt
5 SSLVerifyClient require
6 SSLVerifyDepth 10
```

Listato 2: Direttive per abilitare la SSL Client Authentication in Apache

Per generare la chiave privata ed il relativo certificato nel formato PKCS, è possibile usare il tool openssl. I passi da seguire in questo senso sono i seguenti:

- Generare la coppia chiave privata/certificato per un server:

```
1 openssl genrsa -out server.key 1024
2 openssl req -key server.key -new -out server.req
3 openssl x509 -req -in server.req -CA CA.pem -CAkey privkey.pem -
      CAserial file.srl -out server.pem
```

- Generare la coppia chiave privata/certificato per un client:

```
1 openssl genrsa -des3 -out client.key 1024
2 openssl req -key client.key -new -out client.req
3 openssl x509 -req -in client.req -CA CA.pem -CAkey privkey.pem -
      CAserial file.srl -out client.pem
```

Risulta poi necessario agire anche in Apache Tomcat, utilizzato da OAT (nella versione 6.0.29) per gestire i Web Services di tipo RESTful introdotti. Per abilitare un meccanismo di sicurezza quale l'autenticazione dei client su SSL, occorre infatti configurarne alcuni comportamenti [11].

I passi previsti in questo senso sono:

- Preparazione di un certificato per il Keystore: i formati supportati al momento sono JKS, PKCS11 e PKCS12. È possibile firmare i certificati con la propria CA e creare uno di questi formati tramite dei tool quali, per esempio, openssl.
- Modifica del file di configurazione di Tomcat: è sufficiente abilitare la proprietà SSLEngine e configurare un SSL Connector nel file $CATALINA_BASE/conf/server.xml.

A questo punto sul portale di OAT verrà effettuato un accesso automatico mediante la classica procedura di SSL handshake ed il Distinguished Name del certificato sarà facilmente ottenibile dalla variabile PHP $_SERVER ed utilizzabile per impostare il filtro automatico nelle query eseguite sulle pagine (in modo del tutto simile a quanto mostrato nel dettaglio tra poco con la Basic Authentication).

Nei Web Service RESTful di OAT invece, si potrà ottenere il certificato utilizzato per l'autenticazione SSL mediante il parametro di ingresso javax.servlet.http.HttpServletRequest. In questo modo è possibile disporre anche qui dei dati dell'utente necessari, impostati nel Distinguished Name del certificato.

3.1.2 Basic Authentication

Questo tipo d'autenticazione segue uno schema [12] in cui i client vanno autenticati, per ogni dominio, con uno user-ID ed una password. Il server quindi servirà la richiesta solo se riesce a validare questa coppia di dati, senza altri parametri opzionali. Tali dati, vengono passati attraverso la rete in chiaro, pertanto questo schema è da considerarsi non sicuro, se non viene associato a meccanismi esterni (quali, ad esempio, il SSL). Tuttavia, come già accennato precedentemente, questa soluzione è stata adottata soltanto temporaneamente per via delle tempistiche limitate a disposizione, per riuscire ad ottenere in prima battuta un prototipo logico funzionante e robusto soprattutto per l'algoritmo di controllo degli accessi, e verrà sostituita con l'autenticazione mediante certificati x.509 descritta poco fa.

Per poterla integrare in OAT, sono necessarie due modifiche al codice sorgente del SDK, una sull'Apache HTTP Server ed una sugli header delle chiamate alle API RESTful che si desidera rendere autenticate. Per quanto concerne il nostro HTTP Server, risulta pertanto necessario:

- Installare il modulo mod-auth-mysql per Apache;
- Imporre la ricerca di eventuali file .htaccess nella configurazione base dello stesso Apache, ossia quei file che sovrascrivono alcuni comportamenti standard da tenere sui portali Web;
- Creare un file .htaccess per OAT, impostando un AuthType di tipo Basic ed i dati con cui interfacciarsi al DB MySQL (se si decide di effettuare l'autenticazione sfruttando un DB).

È stato quindi aggiunto in OAT un comando per l'editor di flusso sed da eseguire al termine dell'installazione, che abiliti la ricerca del file .htaccess (comando visibile nella riga 443 del file postinst in Installer/DPKG-OAT-Appraiser-Base/DEBIAN allegato) ed il file .htaccess stesso

(presente nella cartella Source/Portal dei sorgenti allegati). I dati del controllo accessi (utenti compresi) vengono pertanto salvati sul database MySQL di OAT e, come accennato, con questo file è possibile interfacciarsi ad esso. In tal senso, sono particolarmente degni di nota i seguenti campi:

- Auth_MySQL_Username_Field: colonna della tabella che contiene lo Username dell'utente;
- Auth_MySQL_Password_Field: colonna della tabella che contiene la Password dell'utente;
- Auth_MySQL_Password_Clause: eventuale clausola aggiuntiva in linguaggio SQL da associare all'autenticazione. In questa sede risulta utile perchè, come vedremo, gli utenti non potranno essere cancellati fisicamente dal DB a causa dei vincoli relazionali tra le tabelle e quindi in questo modo sarà possibile creare una clausola aggiuntiva sulla colonna DELETED che indicherà se l'utente è attivo, oppure se è stato cancellato;
- Auth_MySQL_Encryption_Types: assume particolare importanza poichè indica il tipo di cifratura usato sui valori contenuti nella colonna Password degli utenti. Può assumere i seguenti valori:
  - Plaintext: password in chiaro;
  - Crypt_DES: password data in pasto alla funzione UNIX standard crypt(), usando un hashing DES;
  - Crypt_MD5: password data in pasto alla funzione UNIX standard crypt(), usando un hashing MD5;
  - Crypt: password data in pasto alla funzione UNIX standard crypt(), senza preferenze sull'hashing usato;
  - PHP_MD5: hashing MD5 ottenuta con il metodo adottato dal PHP;
  - MySQL: Lo schema di hashing usato dalla funzione PASSWORD() di MySQL;
  - SHA1Sum: hashing di tipo SHA-1.

Tra questi valori possibili, in questa sede si è scelto di salvare le password degli utenti con il formato SHA1Sum.

In ogni caso, tutti i metodi che usano le funzioni di hashing sopracitate, sono di per sè insicuri oggigiorno. Questa problema, e le relative scelte fatte in merito, saranno trattate con maggior dettaglio tra poco.

A differenza dell'autenticazione tramite certificato, con la Basic Authenticaton non è necessario modificare la configurazione predefinita di Apache Tomcat ma è sufficiente aggiungere un header nella chiamata sui Web Service che viene fatta negli script bash del CommandTool che prevedono autenticazione.

Infatti, OAT prevede già l'utilizzo di un Auth-blob nell'header di ogni richiesta d'attestazione e gli sviluppatori incoraggiano appunto chi utilizza questo SDK ad implementare la funzione ISV_Authentication_Verification(), definita proprio per questo scopo. Tale header è stato pertanto sfruttato non soltanto per le richieste d'attestazione, ma anche per tutti i servizi su cui è stato deciso che doveva essere applicata un'autenticazione. Infatti, tramite l'oggetto javax.servlet.http.HttpServletRequest in input ad ogni WS, è possibile estrarre username e password passati.

3.1.3 Problematiche del salvataggio su DB delle password

Come accennato nella sezione precedente, i tipi predefiniti di cifratura ammessi dal modulo mod-auth-mysql di Apache sono ormai da considerarsi obsoleti. Nel caso del SHA-1 infatti, è stato ormai dimostrato dai crittoanalisti come sia possibile ottenere collisioni mediante un teorico attacco di complessità 2^{61} e, a causa del numero ormai limitato di combinazioni possibili (la lunghezza della stringa prodotta è infatti di soli 20 byte), le attuali potenze di calcolo disponibili sul mercato rendono praticabili attacchi di tipo forza bruta. Sarebbe opportuno (quantomeno), che tale l'hashing delle password venga fatto su più colonne (USERNAME e PASSWORD per esempio); oppure utilizzare un sale, per proteggersi da attacchi di tipo dizionario. Come si può notare nella documentazione ufficiale di Apache, non vi è modo di definire tali combinazioni su questo modulo nel file .htaccess [13], quindi sarebbe necessario optare per l'uso del comando htpasswd per la generazione degli utenti, soluzione che tuttavia non permette di interfacciarsi ad un database. In questa situazione quindi, le opzioni possibili sono due:

- Modificare e ricompilare il codice del modulo mod-auth-mysql, aggiungendo all'algoritmo di autenticazione predefinito un meccanismo che tiene conto di colonne multiple o del sale;
- Mantenere duplicate le informazioni d'autenticazione, con una copia sul database ed un'altra sul file htpasswd.

Tuttavia, vista la volontà di sostituire poi la Basic Authentication con un'autenticazione mediante certificati x.509, questo problema è stato ritenuto ininfluente poichè non affliggerà mai gli utenti finali ed anche perchè, come detto, l'obiettivo principale che ci si è posti in questa sede era quello di ottenere, in prima battuta, un meccanismo funzionante e robusto soprattutto per il controllo degli accessi. Per questo motivo si è quindi optato per evitare di introdurre informazioni duplicate o di effettuare modifiche ai sorgenti ufficiali dei moduli di Apache.

3.2 Struttura del controllo accessi

Ora che sono state esposte caratteristiche e scelte fatte riguardo al metodo di autenticazione da adottare, è possibile descrivere nel dettaglio la struttura del controllo accessi applicata ad OAT.

Le informazioni vengono salvate nel database di OAT secondo la struttura relazionale visibile in Fig. 6 ed integrata nel file init.sql che lo stesso OAT utilizza durante l'installazione per creare il DB (modifiche visibili dalla linea 123 in poi di tale file, presente in Installer/FilesForLinux dei sorgenti allegati). La tabella USERS conterrà gli utenti, con le seguenti caratteristiche:

- USERNAME: Nome utente che lo identifica univocamente;
- PASSWORD: Credenziale di accesso, ottenuta con la funzione di hashing SHA-1;
- DELETED: Se false, l'utente è attivo. Se true invece, è cancellato (non avviene cancellazione fisica sul DB a causa del nuovo vincolo relazionale tra le tabelle USERS e attest_request, quest'ultima già presente in precedenza in OAT ed a sua volta relazionata alla tabella audit_log).

La tabella PERMISSIONS_TYPES invece, contiene i tipi di permesso predefiniti, con le seguenti caratteristiche:

- CLASS: classe dell'oggetto. Ad esempio Host, User o PCR; ossia gli oggetti a cui si riferisce questo tipo di permesso;
- OPERATION: operazione che si esegue. Ad esempio Add, Edit o Delete, ossia l'azione che viene eseguita sull'oggetto CLASS;
- PAR_NAME: parametro dell'operazione sulla classe. Ad esempio Username o Password di uno User, oppure HostName di una richiesta d'attestazione;
- IS_ENFORCED: indica se questo tipo di permesso è soggetto ad un controllo sul formato specifico per ogni utente. Quindi se impostato a false, significa che non ci sono vincoli in tal senso e qualunque valore utilizzato viene considerato valido. Se impostato a true invece, è necessario che gli utenti abbiano definita la loro regola sul formato di validità del parametro nella tabella USERS_PERMISSIONS descritta tra poco (nel caso sia assente, tutti i valori vengono considerati non validi).

Va sottolineato come le tuple di questa tabella vengano scelte dallo sviluppatore e pre-inserite automaticamente in fase d'installazione di OAT, pertanto l'unica modifica ammessa dopo che il SDK è stato configurato, sarà sul flag IS_ENFORCED.

La tabella USERS_PERMISSIONS contiene i permessi di ogni utente, con le seguenti caratteristiche:

- ID_USERS: ID dell'utente a cui si riferisce il tipo di permesso, in chiave esterna alla tabella USERS;
- ID_PERMISSIONS_TYPES: ID del tipo di permesso, in chiave esterna sulla tabella PERMISSIONS_TYPES;
- VALUE: è di fatto il cuore di questo meccanismo per il controllo accessi, insieme alla colonna IS_ENFORCED della tabella PERMISSIONS_TYPES, poichè indica i valori ammessi per questo permesso, espresso sotto forma di una classica espressione regolare.

Per interfacciarsi a tale struttura sul DB, è quindi presente un Web Service RESTful (con i metodi POST, PUT e DELETE) per manipolare l'utente ed un altro per i suoi permessi, con un'ulteriore servizio (con il solo metodo PUT) per modificare il valore della colonna IS_ENFORCED, necessario per attivare/disattivare il meccanismo di controllo accessi. Come per i Web Service già esistenti, anche per quelli appena introdotti si potrà interagire mediante il CommandTool, con nuovi appositi comandi che verranno esposti tra poco.

Dopo aver installato OAT quindi, seguirà una fase di configurazione che l'amministratore del sistema dovrà effettuare per definire tutti gli utenti e le relative azioni a cui ciascuno di loro può avere accesso. Inizialmente tutti i tipi di permesso avranno l'enforcing sul valore disabilitato, permettendo così la creazione di tale struttura senza doversi autenticare. L'idea quindi, è di definire innanzitutto gli utenti, con i loro username e le loro password, dopodichè definire i loro permessi sui servizi, con i valori ammessi per i parametri di input degli stessi, ed infine abilitare l'enforcing sui tipi di permesso, portando a true la loro colonna IS_ENFORCED.

Finita questa fase di configurazione occorre disabilitare la possibilità di modificare questa struttura, agendo semplicemente su un parametro chiamato edit_permission_type_enabled del file OpenAttestation.properties (file già esistente in precedenza su OAT e presente in Source/HisWebServices/src dei sorgenti allegati) e settandolo a false. Da questo momento in poi, tutti i comandi per cui è stata prevista l'autenticazione (nella nostra versione soltanto oat_pollhosts per effettuare le richieste d'attestazione) e gli stessi comandi per manipolare utenti e permessi, richiederanno username e password ed effettueranno l'autenticazione e l'autorizzazione dell'utente sul comando stesso.

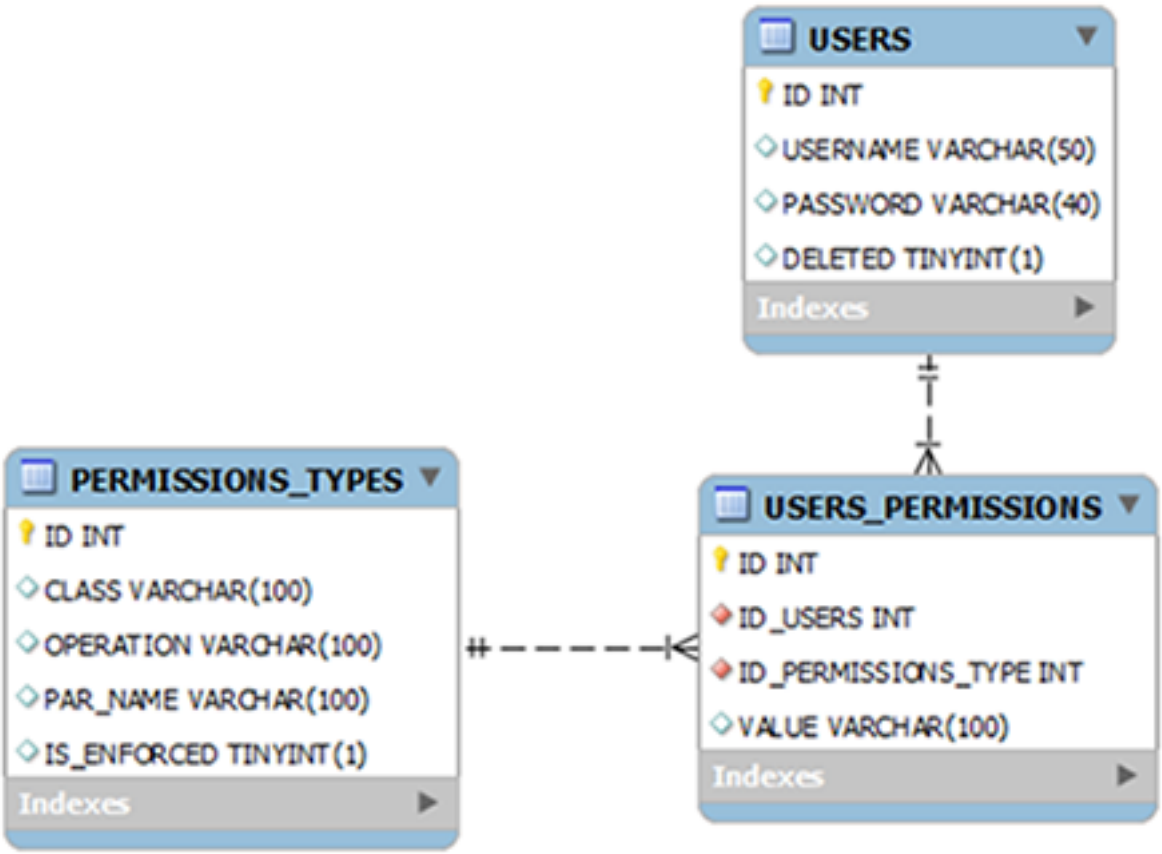

Figura 6: Struttura sul DB del nuovo controllo degli accessi

3.2.1 L'API di registrazione degli utenti

La nuova API è stata inserita nel servizio WhiteList di OAT (WLMService) e, come accennato, si fonda su un nuovo Web Service RESTful che gestisce aggiunta, modifica e cancellazione di un utente ed il cui codice è in Source/WLMService/src/com/intel/openAttestation/manifest/resource/UserResource.java dei sorgenti allegati. Questi metodi del servizio vengono richiamati con un nuovo comando apposito nel CommandTool chiamato oat_user che prende in input username e password desiderati, sui quali è necessaria l'autenticazione se la fase di configurazione iniziale è terminata. Pertanto solo gli utenti per cui è stato previsto il permesso potranno aggiungere, modificare o eliminare altri utenti (o sè stessi).

Naturalmente un tipo di permesso che può essere sempre sensato impostare su un utente è quello con CLASS, OPERATION e PAR_NAME rispettivamente di valore User, Edit, Username; che può essere settato con VALUE della USERS_PERMISSIONS esattamente uguale allo Username dello stesso utente e che quindi, come si pu'o facilmente intuire, fa sì che l'utente possa effettuare modifiche solo su sè stesso (ad esempio modificando la propria password, campo che ovviamente può essere anch'esso oggetto di restrizioni sui valori ammessi dal controllo accessi, magari impostando come validi soltanto caratteri alfanumerici).

3.2.2 Le API per i permessi degli utenti

Le API per gestire i permessi del controllo accessi sono due ed anch'esse si trovano ora nel servizio WhiteList di OAT. La prima si interfaccia alla tabella USERS_PERMISSIONS ed è atta ad aggiungere, modificare e rimuovere un permesso su un utente (il codice sorgente di

questa API si trova nel file Source/WLMService/src/com/intel/openAttestation/manifest/resource/UserPermissionResource.java allegato). Si può richiamare con un nuovo comando apposito nel CommandTool chiamato oat_user_permission che prende in input i seguenti parametri:

- Username dell'utente sul quale impostare il permesso;
- Class, Operation e ParName del tipo di permesso;
- Value da settare (come detto in forma di espressione regolare) sul permesso per questo utente.

Anche su quest'ultimo comando è necessaria l'autenticazione se la fase di configurazione iniziale è terminata. Pertanto anche qui, solo gli utenti per cui è stato previsto il permesso potranno aggiungere, modificare o eliminare i permessi stessi (suoi o di altri utenti). In questo senso può essere sensato, per esempio, aver configurato un utente che funga da amministratore del sistema, che sia l'unico in grado di manipolare tali permessi al termine della fase di configurazione iniziale di OAT.

La seconda API invece, riguarda i tipi di permesso della tabella PERMISSIONS_TYPES (il cui codice sorgente è nel file Source/WLMService/src/com/intel/openAttestation/manifest/resource/PermissionTypeResource.java allegato) e contiene un unico WS di tipo PUT, che viene richiamato con un nuovo comando apposito nel CommandTool chiamato oat_permission_type, che permette la sola modifica del flag IS_ENFORCED di tale tabella. Come già accennato in precedenza, quest'ultima API è utilizzabile solo se il parametro edit_permission_type_enabled del file OpenAttestation.properties è impostato a true ed è utile nella fase finale della configurazione del controllo accessi, quando sui permessi opportuni occorre forzare al valore true tutti gli IS_ENFORCED previsti.

3.3 Autenticazione sui servizi esistenti

Terminata la fase di configurazione con l'abilitazione degli enforcing desiderati sugli utenti previsti sul sistema, i comandi pre-esistenti (e quelli nuovi che prevederanno autenticazione) del CommandTool dovranno quindi contenere due ulteriori parametri chiamati -uname (Username) e -upwd (Password dell'utente). Come già anticipato all'inizio di questa sezione, nella parte riguardante la Basic Authentication, tali valori arriveranno ai WS tramite l'header Authblob, su cui OAT prevede già cifratura (insieme al payload) attraverso il protocollo HTTPS. Su ciascun comando quindi, verrà effettuata l'autenticazione, mentre nel relativo WS verrà valutata la validità dei parametri di input, in base al flag IS_ENFORCED e, se esso risulta abilitato, al formato ritenuto valido per quell'utente. Oltre ai comandi introdotti (oat_user, oat_user_permission ed oat_permission_type), dei servizi già esistenti in OAT soltanto quello atto a servire le richieste di attestazione è stato integrato con il meccanismo di controllo accessi (come è possibile vedere nel file sorgente AttestationService/resource/HOSTResource.java allegato, nel servizio pollHosts che inizia nella riga 445 di tale file), mentre per le altre API l'integrazione verrà valutata in seguito. È stata quindi aggiunta la colonna id_users alla tabella attest_request (riga 147 del file d'installazione Installer/FilesForLinux/init.sql allegato) per legare la richiesta d'attestazione all'utente che l'ha effettuata. Di conseguenza, tali richieste vanno ora sempre autenticate, indipendentemente dalle autorizzazioni impostate.

Attest Request	Host Name	Request Time	Report	Request Host	Count	Validate Time	User
3	node-110	2014-02-22 16:56:58	7	127.0.0.1	1	2014-02-22 16:57:03	alessio_user
2	node-110	2014-02-22 16:56:21	5	127.0.0.1	1	2014-02-22 16:56:26	alessio_user

Figura 7: La griglia nella nuova pagina delle richieste d'attestazione

3.4 Autenticazione sul portale e filtering automatico

Come detto, in questa sede il portale di OAT è stato integrato con la Basic Authentication mediante l'opportuna configurazione di Apache HTTP Server; quindi ora, nella fase di apertura dello stesso, vengono richieste le credenziali d'accesso.

Tali credenziali sono reperibili sul codice all'interno della variabile globale $_SERVER del PHP [14], che contiene un array di informazioni quali path, header, posizione degli script ed anche le credenziali con cui viene effettuato l'accesso, indicizzate per la precisione con "PHP_AUTH_USER"per quanto riguarda lo username, e con "PHP_AUTH_PW"per ciò che concerne la password, e sono state sfruttate per filtrare opportunamente sia la pagina dei report d'integrità che quella nuova contenente le richieste d'attestazione.

3.4.1 Il filtro sui report d'integrità

I report d'integrità saranno soggetti al PERMISSIONS_TYPES di tipo Host, Read_Report, HostName (come visibile nella riga 31 file Source/Portal/reports.php allegato). Saranno pertanto visibili solo i report con HostName previsto nei permessi, a meno che venga lasciato l'IS_ENFORCED di tale tipologia di permesso a false (in tal caso i report saranno tutti visibili).

Tale meccanismo può ad esempio essere utilizzato per far sì che gli utenti vedano soltanto i report degli host per cui possono fare le richieste d'attestazione, associando tale permesso con il PERMISSIONS_TYPES di tipo Host, Attest, HostName; ma si può ovviamente scegliere qualunque altra combinazione eventualmente necessaria, quale potrebbe essere quella su un ipotetico utente amministratore per cui si vuole far sì che siano visibili i report di tutti gli host del sistema (impostando nella colonna VALUE del permesso l'espressione regolare ".* ", che significa appunto qualunque HostName esistente). Per il resto, la pagina è rimasta invariata.

3.4.2 La nuova pagina delle richieste d'attestazione

Le richieste d'attestazione, ora legate all'utente e finora assenti nel portale, saranno presentate con una nuova pagina avente la griglia visibile in Fig. 7: Questa pagina, il cui codice sorgente è presente nel file Source/Portal/attestation_requests.php allegato, è soggetta al PERMISSIONS_TYPES Host, Read_Attest, Username. Saranno pertanto visibili solo le richieste degli utenti previsti. Può essere quindi anche qui applicabile una soluzione che preveda, per gli utenti considerati "standard", che siano visibili solo le richieste d'attestazione effettuate dall'utente stesso; mentre per un eventuale utente amministratore sarà per esempio impostabile la possibilità di vedere le richieste d'attestazione di tutti gli utenti (anche qui impostando nella colonna VALUE del permesso l'espressione regolare ".* ", che significa appunto qualunque Username esistente).

4 Conclusioni

Con questa tesina ci si è posti l'obiettivo di arricchire le funzioni di sicurezza messe a disposizione da OAT con un meccanismo di controllo degli accessi che implementa, per gli utilizzatori, servizi di autenticazione e autorizzazione sulle azioni e sul portale. Come metodo di autenticazione che fornisce le credenziali di accesso tra client e server è stata scelta la Basic Authentication, utile in prima battuta per valutare la funzionalità dello stesso controllo accessi integrato. Ogni modifica a OAT è stata realizzata nel pieno rispetto della sua progettazione e del suo stile originale, cercando di apportare la minor quantità possibile di modifiche al codice sorgente e considerando ogni nuova funzione da integrare come una costruzione su basi già consolidate, piuttosto che una rivisitazione delle stesse. Nel rispetto di questa filosofia sono stati dunque creati i comandi nel CommandTool, le API RESTful (sia i Web Services in sè, che le query sul database effettuate al loro interno) e le modifiche al portale ed alle routine di installazione.

La struttura adottata per il controllo accessi, porta con sè i seguenti vantaggi:

- I comandi che la utilizzano possono ora esser eseguiti solo previa autorizzazione;
- I parametri di ingresso delle azioni sono soggetti a maggiori controlli sul loro formato;
- La funzionalità offerta è generica ed estremamente potente, poichè qualunque gerarchia interna sugli utenti risulta realizzabile.

Appare invece uno svantaggio la complessità medio-alta per definire la struttura iniziale di utenti e permessi, nel caso essa venga effettuata comando per comando. Tuttavia, questa pecca può esser facilmente scavalcabile mediante l'utilizzo di script BASH preconfezionati che contengono gruppi di comandi in cascata. Può essere infatti creato un unico script con cui generare un ipotetico utente amministratore, su cui richiamare il comando oat_user per creare l'utente, i comandi oat_user_permission con i suoi permessi, che possono appunto essere, trattandosi di un amministratore, permessi di manipolazione sugli utenti e sui permessi stessi. Oppure può essere confezionato uno script per gli utenti generici su cui richiamare anche qui il comando oat_user per creare l'utente, i comandi oat_user_permission con cui impostare il permesso di modifica alla sua password, il permesso su quali host poter effettuare le richieste d'attestazione e di conseguenza visualizzare nel portale i report d'integrità e le sue stesse richieste. Ma possono anche essere pensati utenti con caratteristiche diverse, a seconda delle esigenze previste sul sistema. Può infine risultare molto comodo creare un unico script per forzare il valore della colonna IS_ENFORCED di tutti i PERMISSIONS_TYPES, senza dover eseguire il comando oat_permission_type per ogni tipo di permesso.

4.1 Punti aperti

È sicuramente opportuno sostituire l'attuale Basic Authentication con la più sicura client authentication su SSL mediante certificati, o quantomeno associare un ulteriore meccanismo di sicurezza esterno alla stessa Basic Authentication. In quest'ultimo caso però, dovrà anche essere oggetto di attenzione il formato di salvataggio nel database delle password degli utenti che, come detto, ora è il risultato della ormai poco sicura funzione di hash SHA-1 (una delle funzioni di hashing predefinite del file .htaccess di Apache HTTP Server).

Risolti questi punti ancora in sospeso, non resterà che integrare il controllo accessi su tutte le API di OAT, con la stessa filosofia con cui è già stato integrato su alcune API per questa tesina.

4.2 Prospettive future

Nel caso si scegliesse di applicare la client authentication su SSL mediante certificati, appare molto promettente la possibilità di prevedere un formato opportuno per gli stessi; che permetta il loro utilizzo anche su prodotti esterni e l'utilizzo di certificati già esistenti per altre applicazioni in OAT, velocizzando così la fase iniziale di configurazione e rendendo ancora più appetibile questa funzionalità.

A
Manuale utente

Questo manuale descrive le modalità di installazione del server e del client di OAT, nonchè i comandi per l'utilizzo della nuova API per il controllo degli accessi. I seguenti comandi vanno sempre eseguiti da linea di comando con i privilegi di amministratore e sono validi su sistema operativo Ubuntu 12.04 LTS. È necessario aver compilato il codice sorgente di OAT, come riportato in questa sede nel manuale del programmatore, sezione "Compilare OAT". In alcuni comandi è necessario sostituire il contenuto di eventuali parentesi angolari con il valore corretto suggerito nelle stesse.

A.1
Server

- Disabilitare il firewall del server:

```
1 ufw disable
```

- Installare i moduli esterni necessari per il server:

```
1 apt-get install apache2
2 apt-get install mysql-client mysql-server mysql-common
3 apt-get install php5 php5-mysql
4 apt-get install openssl
5 apt-get install openjdk-6-jdk
```

- Disinstallare eventuali versioni precedenti dell'Attestation Server:

```
1 dpkg -P oat-appraiser-base-oatapp
```

- Installare l'Attestation Server:

```
1 dpkg -i /tmp/debbuild/DEBS/x86_64/OAT-Appraiser-Base-OATapp-1.0.0-2.
     x86_64.deb
```

A.2
Client

- Installare i moduli esterni necessari per il client:

```
1 apt-get install trousers
2 apt-get install libtspi1
3 apt-get install openjdk-6-jre
```

- Modificare il daemon script del modulo trousers e riavviarne il servizio:

```
1 sed -i 's/--chuid \${USER}//g' /etc/init.d/trousers
2 service trousers restart
```

- Installare il client:

```
service OATClient stop
wget http://localhost/ClientInstallForLinux.zip
unzip ClientInstallForLinux.zip
cd ClientInstallForLinux
sh general-install.sh
service OATClient start
```

A.3 Configurazione lato verificatore

```
cd <PATH-DI-OAT>/CommandTool
bash oat_cert -h node-110
bash oat_oem -a -h node-110 '{"Name":"OEM1","Description":"Test id"}'
bash oat_os -a -h node-110 '{"Name":"precise","Version":"v1234","
    Description":"Test1"}'
bash oat_mle -a -h node-110 '{"Name":"node-110-precise","Version":"123","
    OsName":"precise","OsVersion":"v1234","Attestation_Type":"PCR","
    MLE_Type":"VMM","Description":"Test ad"}'
bash oat_host -a -h node-110 '{"HostName":"node-110","IPAddress":"<
    CLIENT_IP_ADDRESS>","Port":"9999","VMM_Name":"node-110-precise","
    VMM_Version":"123","VMM_OSName":"precise","VMM_OSVersion":"v1234","
    Email":"","AddOn_Connection_String":"","Description":""}'
bash oat_pcrwhitelist -a -h node-110 '{"pcrName":"0","pcrDigest":"7
    D94A15BE0295A3743FC259B07202FF42550B369","mleName":"node-110-precise
    ","mleVersion":"123","osName":"precise", "osVersion":"v1234"}'
```

A.4 API di controllo degli accessi

I tipi di permesso preimpostati in questa sede sono i seguenti:

Class	Operation	ParName	Descrizione
Host	Read_Report	HostName	Gli host per cui è possibile visualizzare il report nel portale.
Host	Attest	HostName	Gli host per cui è possibile effettuare richieste d'attestazione.
Host	Read_Attest	Username	Gli utenti per cui è possibile visualizzare nel portale le richieste d'attestazione effettuate.
User	Add	Username	Il formato ammesso per gli username degli utenti per poterli aggiungere.
User	Add	Password	Il formato ammesso per le password degli utenti per poterli aggiungere.
User	Edit	Username	Gli username degli utenti che è possibile modificare.
User	Edit	Password	Il formato ammesso per le password degli utenti per poterle modificare.
User	Delete	Username	Gli username degli utenti che è possibile cancellare.
UserPermission	Add	Username	Gli username degli utenti per cui è possibile aggiungere permessi.
UserPermission	Add	Class	Le classi per cui è possibile aggiungere permessi.
UserPermission	Add	Operation	Le operazioni per cui è possibile aggiungere permessi.
UserPermission	Add	ParName	I parametri per cui è possibile aggiungere permessi.
UserPermission	Add	Value	Il formato ammesso dei valori per cui è possibile aggiungere permessi.
UserPermission	Edit	Username	Gli username degli utenti per cui è possibile modificare permessi.
UserPermission	Edit	Class	Le classi per cui è possibile modificare permessi.
UserPermission	Edit	Operation	Le operazioni per cui è possibile modificare permessi.
UserPermission	Edit	ParName	I parametri per cui è possibile modificare permessi.
UserPermission	Edit	Value	Il formato ammesso dei valori per cui è possibile modificare permessi.
UserPermission	Delete	Username	Gli username degli utenti per cui è possibile cancellare permessi.
UserPermission	Delete	Class	Le classi per cui è possibile cancellare permessi.
UserPermission	Delete	Operation	Le operazioni per cui è possibile cancellare permessi.
UserPermission	Delete	ParName	I parametri per cui è possibile cancellare permessi.

I nuovi comandi introdotti in questa sede sono:

- Aggiungere un utente:

```
1 bash oat_user -a -h <HOSTNAME_OF_OAT-APPRAISER> -uname <USERNAME> -
    upwd <PASSWORD> '{"Username":"<DESIRED_USERNAME>","Password":"<
    DESIRED_PASSWORD>"}'
```

- Modificare un utente:

```
1 bash oat_user -e -h <HOSTNAME_OF_OAT-APPRAISER> -uname <USERNAME> -
    upwd <PASSWORD> '{"Username":"<EXISTING_USERNAME>","Password":"<
    NEW_DESIRED_PASSWORD>"}'
```

- Eliminare un utente:

```
1 bash oat_user -d -h <HOSTNAME_OF_OAT-APPRAISER> -uname <USERNAME> -
    upwd <PASSWORD> '{"Username":"<DESIRED_USERNAME>"}'
```

- Aggiungere un permesso ad un utente:

```
1 bash oat_user_permission -a -h <HOSTNAME_OF_OAT-APPRAISER> -uname <
    USERNAME> -upwd <PASSWORD> '{"Username":"<
    USERNAME_OF_USER_WITH_THIS_PERMISSION>","Class":"<CLASS>","
    Operation":"<OPERATION>","ParName":"<PARNAME>","Value":"<VALUE
    >"}'
```

- Modificare un permesso di un utente:

```
1 bash oat_user_permission -e -h <HOSTNAME_OF_OAT-APPRAISER> -uname <
    USERNAME> -upwd <PASSWORD> '{"Username":"<
    USERNAME_OF_USER_WITH_THIS_PERMISSION>","Class":"<CLASS>","
    Operation":"<OPERATION>","ParName":"<PARNAME>","Value":"<VALUE
    >"}'
```

- Eliminare un permesso di un utente:

```
1 bash oat_user_permission -d -h <HOSTNAME_OF_OAT-APPRAISER> -uname <
    USERNAME> -upwd <PASSWORD> '{"Username":"<
    USERNAME_OF_USER_WITH_THIS_PERMISSION>","Class":"<CLASS>","
    Operation":"<OPERATION>","ParName":"<PARNAME>"}'
```

- Modificare il parametro IS_ENFORCED di un tipo di permesso:

```
1 bash oat_permission_type -e -h <HOSTNAME_OF_OAT-APPRAISER> -uname <
    USERNAME> -upwd <PASSWORD> '{"Class":"<CLASS>","Operation":"<
    OPERATION>","ParName":"<PARNAME>","IsEnforced":"<
    IS_ENFORCED_VALUE>"}'
```

- Script esterni alla patch ma presenti nel materiale allegato:
  - Creare l'utente admin: questo utente è in grado di alterare tutti gli utenti ed i permessi e può visualizzare tutte le richieste d'attestazione ed i report d'integrità.

```
1 bash oat_create_admin -upwd <DESIRED_PASSWORD>
```

– Creare un utente atto alla gestione di tutti gli utenti del sistema: questo utente è in grado di alterare tutti gli utenti esistenti, aggiungendone nuovi, modificando quelli già presenti o cancellandoli.

```
1 bash oat_create_usermanager_user -uname <DESIRED_USERNAME> -upwd
      <DESIRED_PASSWORD>
```

– Modificare i valori IS_ENFORCED di tutti i tipi di permesso contemporaneamente.

```
1 bash oat_change_enforcing -enforced <ENFORCING_VALUE>
```

B
Manuale del programmatore

Verranno descritte nel dettaglio le procedure e le funzioni introdotte, oltre alle modifiche apportate a quelle preesistenti, iniziando l'analisi proprio da queste ultime. Infine saranno descritte le modalità con cui applicare le suddette modifiche sotto forma di patch per il repository GIT della versione 1.6 di OAT e la relativa compilazione del codice per ottenere un file .deb finale con cui poter effettuare l'installazione su sistema operativo Linux Ubuntu 12.04 LTS.

B.1
Modifiche a procedure e funzioni

B.1.1
CommandTool

- "**oat_pollhosts**": il comando è stato modificato per permettere di inserire le credenziali di accesso autenticato. È infatti ora necessario l'inserimento dei parametri -uname e -upwd immediatamente dopo il parametro HOSTNAME_OF_OAT-APPRAISER. Verrà quindi innanzitutto controllata la presenza di tali parametri per valutare la posizione effettiva delle informazioni relative alla richiesta d'attestazione (dalla linea 30 alla 42) e successivamente verrà costruita una stringa avente il formato USERNAME#PASSWORD, passata come header della richiesta sul WS PollHosts (linea 47).

B.1.2
Installer

- "**DPKG-OAT-Appraiser-Base/DEBIAN/postinst**": il file è stato aggiornato con l'inserimento di un comando per l'editor di flusso sed nella riga 443 atto alla sostituzione della regola AllowOverride None con AllowOverride All nel file 000-default di Apache HTTP Server per il portale di OAT. In questo modo, verrà cercato il file .htaccess con le direttive della Basic Authentication;

- "**FilesForLinux/init.sql**": il file è stato modificato con la creazione (dalla riga 123 in poi) delle tabelle USERS, PERMISSIONS_TYPES e USERS_PERMISSIONS ed il relativo inserimento delle tuple per i tipi di permesso previsti sul sistema. È presente un vincolo UNIQUE per la tabella PERMISSIONS_TYPES atto a garantire l'univocità delle colonne CLASS, OPERATION e PAR_NAME, un altro vincolo UNIQUE sulla

tabella USERS_PERMISSIONS che ha lo stesso scopo ma sulle colonne ID_USERS e ID_PERMISSIONS_TYPES e due chiavi esterne per queste ultime due colonne riferite, rispettivamente, alle tabelle USERS e PERMISSIONS_TYPES. È stata infine spostata la creazione della tabella attest_request dopo quella della tabella USERS, poichè è ora presente una colonna aggiuntiva sulla stessa (riga 147), denominata id_users, che è chiave esterna sulla colonna ID della tabella USERS e che quindi obbliga che prima venga fatta la creazione di quest'ultima tabella.

B.1.3 AttestationService

- "**src/hibernateOat.cfg.xml**": sono stati aggiunti, dalla riga 35 alla 37, i percorsi ai file xml atti al mapping hibernate con le nuove tabelle USERS, USERS_PERMISSIONS e PERMISSIONS_TYPES;
- "**src/com/intel/openAttestation/AttestationService**":
  - "**bean/OpenAttestationResponseFault.java**": è stato aggiunto, alla riga 57, un nuovo tipo di FaultCode da usare in caso si presenti un errore di tipo Forbidden Access nei WS che fanno il check delle autorizzazioni;
  - "**resource/AttestService.java**": È stato aggiunto un commento in testa al metodo ISV_Autherntication_module, per indicare che tale funzione è in realtà inutilizzata, poichè vi è una copia della stessa nel modulo HisAppraiser, che è visibile a tutti i WS che ci servono grazie alle dipendenze preesistenti su OAT;
  - "**resource/HOSTResource.java**": sono stati aggiunti autenticazione ed autorizzazione sul WS pollHosts, a partire dalla riga 485 fino alla 508. L'autenticazione viene fatta passando l'HttpServletRequest al metodo di autenticazione in HisAppraiser, se fallisce viene restituito il FaultCode 401. L'autorizzazione viene controllata dapprima analizzando l'IS_ENFORCED sul tipo di permesso Host, Attest, HostName con il metodo ISV_Permission_Type_Enforcement in HisAppraiser e, se di valore true, vengono controllati gli HostName leggendo il VALUE relativo nella tabella USERS_PERMISSIONS, il tutto attraverso il metodo doAuthorization, anch'esso in HisAppraiser, che riceve in input ciascun oggetto ParNameContainer (di cui quindi ora è presente un import in testa al file) necessario. Inoltre, è stato aggiunto il parametro userId al metodo addRequests (riga 649) che associa l'utente alle richieste d'attestazione appena fatte, settando la relativa property sugli oggetti AttestRequest in mapping hibernate con la tabella attest_request.

B.1.4 HisAppraiser

- "**src/gov/niarl/hisAppraiser/hibernate**":
  - "**domain/AttestRequest.java**": è stata aggiunta al fondo del file la proprietà UsersId con getter e setter;
  - "**mapping/attestRequest.hbm.xml**": è stata aggiunta al fondo del file il mapping tra la proprietà UsersId e la colonna id_users della tabella attest_request;
  - "**util/AttestService.java**": sono stati aggiunti, dalla riga 146 in poi, i metodi per autenticazione, enforcement, autorizzazione e hashing. Nel primo, vengono ricavate

le credenziali d'accesso dall'header Auth-blob, splittandone il contenuto sul carattere #, impostato nel comando in CommandTool per separare username e password. Di quest'ultima, viene ricavato l'hash SHA-1 con il metodo getHash ed infine viene utilizzato un oggetto UserDAO per interrogare il DB verificando i dati d'accesso; viene ritornato l'ID dell'utente se si è autenticati, -1 altrimenti. Nel secondo, in input vi sono i dati utili per identificare il tipo di permesso e tramite un oggetto PermissionTypeDAO viene letto dal DB il valore della sua colonna IS_ENFORCED. Nel terzo, viene sfruttato un oggetto UserPermissionDAO per interrogare il DB sulla base del tipo di permesso e la lista di ParName da verificare; viene restituito true se si è autorizzati, false altrimenti. Nel metodo getHash infine, viene sfruttata la classe Java MessageDigest per ottenere il Digest del messaggio passato, con l'algoritmo desiderato. Tali bytes vengono convertiti in stringa esadecimale prima di essere restituiti;

  - "**util/AttestUtil.java**": tra i parametri letti dal file OpenAttestation.properties e restituiti tramite proprietà da questa classe, vi è ora edit_permission_type_enabled, utile per sapere se è possibile fare modifiche sulla tabella PERMISSIONS_TYPES.

B.1.5 HisWebServices

- "**src/OpenAttestation.properties**": al fondo di questo file, è ora presente il valore edit_permission_type_enabled, inizializzato a true per favorire la configurazione iniziale di OAT dopo l'installazione.

B.1.6 Portal

- "**includes/navigation.php**": è stato aggiunto un listitem per la nuova pagina attestation_requests, in modo che sia selezionabile nella barra di navigazione del portale;

- "**reports.php**": in questo file viene ricavata (dalla riga 25 alla 32) l'espressione regolare relativa al tipo di permesso Host, Read_Report, HostName. Se l'autenticazione fosse disabilitata, oppure se l'IS_ENFORCED relativo fosse false, l'espressione regolare impostata sarà ".*"(quindi tutti gli HostName vengono considerati validi). È pertanto stata adeguata la query sull'audit_log ed i filtri sulla tabella dei report. Inoltre, è stato aggiunto il filtro single, che permette di visualizzare un unico report con l'id passato, utile alla nuova pagina delle richieste d'attestazione.

B.1.7 WLMService

- "**src/hibernateOat.cfg.xml**": sono stati aggiunti, dalla riga 31 alla 33, i percorsi ai file xml atti al mapping hibernate con le nuove tabelle USERS, USERS_PERMISSIONS e PERMISSIONS_TYPES;

- "**src/com/intel/openAttestation/manifest**":

  - "**bean/OpenAttestationResponseFault.java**": è stato aggiunto, alla riga 48, un nuovo tipo di FaultCode da usare in caso si presenti un errore di tipo Forbidden Access nei WS che fanno il check delle autorizzazioni.

B.2 Procedure e funzioni introdotte

B.2.1 CommandTool

- "**oat_user**": il comando aggiunto è atto all'interazione con l'API degli utenti, per la loro aggiunta, modifica o cancellazione. Segue lo stile dei comandi già esistenti, con il controllo sui parametri passati e le chiamate HTTPS ai relativi WS. Contiene inoltre i parametri -uname e -upwd per le credenziali di accesso autenticato, passati sull'header della richiesta sui WS;
- "**oat_user_permission**": il comando aggiunto è atto all'interazione con l'API dei permessi degli utenti, per la loro aggiunta, modifica o cancellazione. Lo stile dei contenuti è lo stesso di oat_user;
- "**oat_permission_type**": il comando aggiunto è atto all'interazione con l'API dei tipi di permesso, per la modifica del valore IS_ENFORCED. Lo stile dei contenuti è lo stesso di oat_user ed oat_user_permission.

B.2.2 HisAppraiser

- "**src/gov/niarl/hisAppraiser/hibernate**":
  - "**dao/PermissionTypeDAO.java**": file che si occupa delle query sul DB per i tipi di permesso. Contiene il solo metodo getPermissionTypeEnforcement che restituisce il valore della colonna IS_ENFORCED per il tipo di permesso richiesto, identificato dai parametri classValue, operationValue e parnameValue passati;
  - "**dao/UserDAO.java**": file che si occupa delle query sul DB per gli utenti. Contiene il solo metodo getAuthenticatedUserId che restituisce l'ID dell'utente, identificato dai parametri username e password passati, o -1 se non viene trovato (e quindi i dati di autenticazione non sono validi);
  - "**dao/UserPermissionDAO.java**": file che si occupa delle query sul DB per i permessi degli utenti. Contiene il solo metodo doAuthorization che riceve i dati per identificare i tipi di permesso e la lista di valori da verificare. Se almeno un permesso per quell'utente non viene trovato, ovviamente non si è considerati autorizzati e viene restituito false. Se invece vengono trovati tutti, verrà estratta l'espressione regolare associata a ciascun ParName per questo utente e controllato il match con il relativo valore passato (presente nella proprietà Value del ParNameContainer) e, se almeno un tipo di permesso non ha un formato autorizzato, si viene considerati non autorizzati; altrimenti, nel caso tutti i match diano esito positivo, viene concessa l'autorizzazione, uscendo con true da questo metodo;
  - "**domain/PermissionType.java**": file che contiene la classe atta al mapping hibernate con la tabella PERMISSIONS_TYPES, contenente tutte le proprietà necessarie e corrisponenti alle colonne della stessa tabella;
  - "**domain/User.java**": come per il file PermissionType.java, ma riferito alla tabella USERS;
  - "**domain/UserPermission.java**": come per i file PermissionType.java e User.java, ma riferito alla tabella USERS_PERMISSIONS;

- "**mapping/PermissionType.hbm.xml**": file che contiene il mapping hibernate con la tabella PERMISSIONS_TYPES, contenente tutte le sue colonne mappate appunto con le proprietà della classe PermissionType;
- "**mapping/User.hbm.xml**": come per il file PermissionType.hbm.xml, ma riferito alla tabella USERS mappata sulla classe User;
- "**mapping/UserPermission.hbm.xml**": come per i file PermissionType.hbm.xml e User.hbm.xml, ma riferito alla tabella USERS_PERMISSIONS mappata sulla classe UserPermissions;
- "**util/ParNameContainer.java**": all'interno di questo file vi è una classe di appoggio per il metodo doAuthorization della classe UserPermissionDAO, che contiene le informazioni riguardo ad una coppia ParName-Value di un tipo di permesso.

B.2.3 Portal

- "**includes/users_permissions_functions.php**": questo file contiene un metodo PHP per ottenere dal DB l'espressione regolare nella colonna VALUE di un permesso legato ad un utente, dati username dell'utente proprietario del permesso e Class, Operation e ParName del tipo di permesso;
- "**.htaccess**": file di configurazione per Apache HTTP Server, specifico per OAT. Contiene i dati per l'accesso al DB ed i dati della tabella USERS che servono ad Apache per effettuare l'accesso mediante Basic Authentication; ossia la colonna USERNAME, la colonna PASSWORD (a cui associare il controllo sul flag DELETED per controllare se l'utente è stato cancellato) ed il formato con cui viene salvata, che nel nostro caso è il risultato ottenuto dalla funzione di hash SHA-1 sul testo della password in chiaro. Il campo AuthUserFile /dev/null serve ad indicare che non vengono usati file generati con htpasswd. Se questo file venisse rinominato con qualunque altro nome, l'autenticazione sul portale verrebbe disabilitata ed OAT si comporterebbe allo stesso modo in cui si comportava prima dell'integrazione del controllo accessi;
- "**attestation_requests.php**": questa è la nuova pagina del portale che contiene le richieste d'attestazione effettuate, presentate su una tabella. Tale tabella viene filtrata sulla base del tipo di permesso Host, Read_Attest, Username associato all'utente che ha fatto l'accesso. Se l'autenticazione fosse disabilitata, oppure se l'IS_ENFORCED relativo fosse false, l'espressione regolare impostata sarà ".*"(ossia si vedrebbero le richieste d'attestazione di tutti gli utenti). Questa pagina segue lo stile utilizzato su tutto il portale e, sulle righe della tabella, l'host contiene anche un'icona di collegamento che una volta cliccato reindirizza alla pagina machine.php; l'utente ha anch'esso un'icona aggiuntiva che reindirizza allo stesso modo alla pagina user.php ed il report contiene un'icona che reindirizza alla pagina reports.php, che sarà filtrata in modalità single per contenere soltanto il report d'interesse (se l'utente ha il permesso Read_Report valido per l'host a cui fa riferimento).

B.2.4 WLMService

- "**src/com/intel/openAttestation/manifest**":

- "**bean/UserPermissionsBean.java**": questo nuovo file contiene la classe atta a mappare in un oggetto il JSON in arrivo dal comando oat_user_permission sui WS della classe UserPermissionResource;
- "**hibernate/dao/PermissionTypeDAO.java**": questa nuovo file contiene la classe atta ad effettuare le query sul DB riferite alla tabella PERMISSIONS_TYPES;
- "**hibernate/dao/UserDAO.java**": come per il PermissionTypeDAO, ma riferito alle query sulla tabella USERS;
- "**hibernate/dao/UserPermissionDAO.java**": come per il PermissionTypeDAO e la UserDAO, ma riferito alle query sulla tabella USERS_PERMISSIONS;
- "**hibernate/domain/PermissionType.java**": file che contiene la classe atta al mapping hibernate con la tabella PERMISSIONS_TYPES, contenente tutte le proprietà necessarie e corrisponenti alle colonne della stessa tabella;
- "**hibernate/domain/User.java**": come per il file PermissionType.java, ma riferito alla tabella USERS;
- "**hibernate/domain/UserPermission.java**": come per i file PermissionType.java e User.java, ma riferito alla tabella USERS_PERMISSIONS;
- "**hibernate/mapping/PermissionType.hbm.xml**": file che contiene il mapping hibernate con la tabella PERMISSIONS_TYPES, contenente tutte le sue colonne mappate appunto con le proprietà della classe PermissionType;
- "**hibernate/mapping/User.hbm.xml**": come per il file PermissionType.hbm.xml, ma riferito alla tabella USERS mappata sulla classe User;
- "**hibernate/mapping/UserPermission.hbm.xml**": come per i file PermissionType.hbm.xml e User.hbm.xml, ma riferito alla tabella USERS_PERMISSIONS mappata sulla classe UserPermissions;
- "**resource/PermissionTypeResource.java**": questo file contiene un unico WS chiamato editUserPermission. Tale metodo effettua l'autenticazione, controlla se sul file .properties di OAT le modifiche alla tabella sono abilitate ed infine agisce sulla colonna IS_ENFORCED del tipo di permesso desiderato;
- "**resource/UserResource.java**": in questo file vi sono tre WS atti rispettivamente ad aggiungere, modificare o cancellare gli utenti. In tali metodi viene effettuata l'autenticazione, controllato il valore della colonna IS_ENFORCED e, se necessario, valutata la validità dei parametri passati stando ai permessi dell'utente che ha effettuato l'accesso;
- "**resource/UserPermissionResource.java**": come per il file UserResource.java ma riferito ai permessi degli utenti.

B.3 Compilare OAT

I seguenti comandi vanno sempre eseguiti da linea di comando con i privilegi di amministratore e sono validi su sistema operativo Ubuntu 12.04 LTS. In alcuni comandi è necessario sostituire il contenuto di eventuali parentesi angolari con il contenuto corretto suggerito nelle stesse. Mi è stata infatti fornita una macchina con il suddetto sistema operativo, contenente il pacchetto di OAT nella sua versione 1.6 (scaricabile dal relativo repository GIT, all'indirizzo: https://github.com/OpenAttestation/OpenAttestation).

- Installazione dei pacchetti base necessari:

```
1 apt-get install openjdk-6-jdk
2 apt-get install libtspi-dev
3 apt-get install zip
4 apt-get install ant
5 apt-get install g++
6 apt-get install make
```

- Applicare le modifiche tramite la patch fornita:

```
1 cd <PATH-DI-OAT>
2 sudo cat <PATH-DELLA-PATCH>/alessio_vallero_oat_access_control.patch
      | patch -p1
3 sudo chmod -R +x CommandTool
```

- I sorgenti di OAT non includono le librerie necessarie a runtime, occorre pertanto scaricarle:

```
1 cd <PATH-DI-OAT>/Source
2 sudo bash download_jar_packages.sh -s /home/alessio/OpenAttestation
3 sudo bash distribute_jar_packages.sh
```

- Compilare il server di OAT:

```
1 cd <PATH-DI-OAT>/Installer
2 sudo bash deb.sh -s <PATH-DI-OAT>/Source
```

Una volta che la compilazione è andata a buon fine, si potrà trovare il file deb generato in /tmp/debbuild/DEBS/x86_64/OAT-Appraiser-Base-OATapp-1.0.0-2.ubuntu.x86_64.deb.

Riferimenti

[1] La Stampa, "Cybercrimine, la crescita è inarrestabile", Ottobre 2013, http://www.lastampa.it/2013/10/02/tecnologia/cybercrimine-la-crescita-inarrestabile-in-un-anno-attacchi-sul-web-triplicati-YSwnt6Iekp6uow1YzOI92O/pagina.html

[2] CNN, "Cyber-criminals are targeting phones and bank info", Febbraio 2013, http://edition.cnn.com/2013/02/21/tech/mobile/mcafee-threats-report/index.html

[3] Trusted Computing Group, "Trusted computing", Febbraio 2014, http://www.trustedcomputinggroup.org/trusted_computing

[4] InterLex, "Il Trusted Computer viola la mia sicurezza", Ottobre 2003, http://www.interlex.it/675/trustedcomp.htm

[5] WhatIs.com, "Roots of Trust (RoT)", Gennaio 2014, http://whatis.techtarget.com/definition/Roots-of-Trust-RoT

[6] TCG Infrastructure Working Group, "Reference Architecture for Interoperability (Part I), specification version 1.0", Giugno 2005, http://www.trustedcomputinggroup.org/files/resource_files/8770A217-1D09-3519-AD17543BF6163205/IWG_Architecture_v1_0_r1.pdf

[7] WPI, "Principles of Remote Attestation", Gennaio 2011, http://web.cs.wpi.edu/~guttman/pubs/remote_attest.pdf

[8] OAT's github repository, "OpenAttestation SDK Overview", Dicembre 2012, https://github.com/OpenAttestation/OpenAttestation/blob/master/docs/Overview.pdf

[9] Archive Apache, "Apache HTTP Server Documentation Version 2.2", Giugno 2006, http://archive.apache.org/dist/httpd/docs/httpd-docs-2.2.2.en.pdf

[10] Network Working Group, "Internet X.509 Public Key Infrastructure", Aprile 2002, http://www.ietf.org/rfc/rfc3280.txt

[11] Apache Tomcat 6.0, "Apache Tomcat", Gennaio 2014, http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html#Quick_Start

[12] Network Working Group, "HTTP Authentication: Basic and Digest Access Authentication", Giugno 1999, http://www.ietf.org/rfc/rfc2617.txt

[13] Apache HTTP Server documentation, "Authentication and Authorization", Marzo 2014, http://httpd.apache.org/docs/current/howto/auth.html

[14] php.net, "$_SERVER", Febbraio 2014, http://www.php.net/manual/en/reserved.variables.server.php