Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Corso di Laurea Triennale in Informatica

Studio e implementazione di protocolli di autenticazione per la connessione di sistemi degli utenti in laboratori virtuali

Tesi di Laurea in Progetto di Sistemi Virtuali

Relatore:
Chiar.mo Prof.
RENZO DAVOLI

Presentata da:
MICHELE CUCCHI

Sessione III
Anno Accademico 2010-2011

Siamo come nani sulle spalle di giganti,
così che possiamo vedere più cose di loro e più lontane,
non certo per l'altezza del nostro corpo,
ma perchè siamo sollevati e portati in alto
dalla statura dei giganti.
Bernardo di Chartres

Introduzione

Negli ultimi anni la continua evoluzione e lo sviluppo delle capacità di calcolo e memorizzazione, unite al costante incremento delle possibilità di accesso alla rete Internet, tramite dispositivi sempre più diffusi e portatili, ha permesso la nascita di tecnologie come il *Cloud Computing*, che consentono il trasferimento delle capacità di storage ed elaborazione, da singole macchine, distribuite e indipendenti, a posizioni unificate e centralizzate, in grado di fornire queste caratteristiche come servizi, usufruibili tramite i collegamenti di rete.

L'idea della possibile applicazione di queste tecniche, anche in contesti didattici ed accademici, ha portato alla sperimentazione di soluzioni per la conversione dell'infrastruttura di laboratori didattici di informatica, al paradigma del *Cloud*, sostituendo gli strumenti presenti, come gli elaboratori fisici usati dagli utenti, con i servizi forniti da un sistema centralizzato, raggiungibile grazie alla possibilità di accesso veloce ai collegamenti di rete. Tale facoltà consente il collegamento dei dispositivi portatili, di coloro che usufruiscono del laboratorio, con un insieme di calcolatori, fisici o virtuali, che fornisce i servizi richiesti.

L'applicazione di tale tecnologia, nell'ambito di questo contesto, ha evidenziato la presenza di problematiche e rischi, legati soprattutto alla sicurezza delle operazioni di accesso ai dati e ai servizi, legata molto frequentemente alla sua implementazione nei meccanismi, di connessione alla rete e trasporto delle informazioni.

La realizzazione della sicurezza nei sistemi di comunicazione e nell'accesso

ai servizi è legata alla possibilità di identificare con certezza gli utenti utilizzatori e l'identità degli interlocutori, con cui si sta iniziando uno scambio di informazioni, per evitare possibili abusi e violazioni alla riservatezza dei dati.

In questo studio si intende fornire una possibile soluzione, al problema dell'identificazione univoca degli utilizzatori del servizio di rete, per l'accesso all'infrastruttura virtuale di un laboratorio didattico, realizzato tramite l'applicazione del paradigma di cloud computing.

Si propone un sistema che renda l'autenticazione, degli utenti, propedeutica e vincolante per l'accesso all'infrastruttura di rete, studiato per l'affiancamento e l'integrazione con i protocolli e le strutture già presenti nel sistema informatico del *dipartimento di Scienze dell'Informazione* dell'*Università di Bologna*.

Si mostra quindi lo studio delle possibilità di integrazione e di utilizzo del protocollo di autenticazione IEEE 802.1X, in una struttura per l'accesso alla rete ad alta velocità, realizzata con tecnologia IEEE 802.3 Gigabit Ethernet, focalizzandosi principalmente sull'efficienza e l'integrazione, più trasparente possibile, del protocollo scelto con le infrastrutture presenti.

Viene mostrata la soluzione ad alcune problematiche di sicurezza e compatibilità, emerse durante lo studio del protocollo, grazie allo sviluppo di componenti software elaborate tramite l'impiego di tecnologie libere.

La struttura del documento è suddivisa nei seguenti capitoli:

Scenario : Si descrive il contesto dello studio, focalizzandosi in particolare sui protocolli e le tecnologie utilizzate, descrivendone le possibili problematiche.

Implementazione : In questo capitolo è trattata in dettaglio l'implementazione dell'idea di soluzione al problema principale oggetto della tesi, introducendo l'applicazione software oggetto di specifica nel capitolo successivo.

Linux Neighbor Logging System : Il capitolo descrive in dettaglio lo sviluppo e l'implementazione dell'applicazione software, che consente la soluzione al problema oggetto della ricerca.

Sviluppi futuri : Vengono delineati i possibili sviluppi futuri della soluzione elaborata, non integrati nel presente studio per mancanza di tempo.

Conclusioni : La conclusione del lavoro riassume le fasi essenziali della ricerca, che hanno portato allo sviluppo della soluzione.

Indice

Elenco delle figure

Capitolo 1

Scenario

Negli ultimi anni nell'ambiente informatico si stanno diffondendo sempre di più l'uso delle tecniche integrate della virtualizzazione e del *Cloud Computing*, ossia la delocalizzazione e l'astrazione di risorse di elaborazione, di memorizzazione e di contenuti, distribuiti e raggiungibili attraverso la rete, con l'obiettivo di ottenere maggior flessibilità, ottimizzazione delle risorse e contenimento dei costi.

Sta diventando sempre più comune l'utilizzo di questa tecnologia per fornire infrastrutture, piattaforme e software come servizi, è ancora poco applicata in quelle realtà che gestiscono in proprio i loro servizi, per necessità particolari, come possono essere, servizi sperimentali o laboratori didattici di informatica.

1.1 Laboratorio didattico virtuale

Il *laboratorio didattico* è quel luogo dove lo studioso di una qualche disciplina scientifica trova gli strumenti per la verifica, la pratica e la sperimentazione di teorie ed idee. In particolare le discipline scientifiche hanno il laboratorio come punto centrale, in quanto è proprio il metodo scientifico a richiedere la verifica e la ripetizione tramite esperimenti delle teorie elaborate.

In modo particolare l'informatica, scienza che ha come obiettivo lo studio e la sperimentazione di metodi algoritmici, per l'elaborazione e la trasmissione, nel tempo e nello spazio di informazioni in modo automatico ; si serve di laboratori formati da gruppi di calcolatori elettronici, interconnessi tra loro, con cui l'informatico può sperimentare ed applicare nuovi concetti ed idee.

1.1.1 L'evoluzione

Il passo successivo del laboratorio di informatica è la sua *dematerializzazione* ovvero, l'applicazione delle tecnologie del cloud computing e della virtualizzazione al laboratorio stesso, trasformando un aula di elaboratori fisici, usufruibili da studenti e sperimentatori ad orari fissi, in un luogo fisso *fisico*, in una sua rappresentazione virtuale, ospitata in elaboratori centralizzati e raggiungibile attraverso le reti. Rendendo indipendente l' elaboratore locale dell'utilizzatore, dalla piattaforma di calcolo su cui i servizi del laboratorio vengono ospitati.

Consentendo inoltre l'utilizzo dei servizi in un luogo diverso, magari molto lontano, da quello in cui i servizi vengono forniti.

Un ulteriore vantaggio è l'abbattimento dei costi relativi alla manutenzione fisica degli elaboratori, concentrando e consolidando in poche macchine tutti i servizi.

1.1.2 Problemi e Requisiti

L'applicazione dei paradigmi del cloud computing ad un sistema come un laboratorio informatico, consente la focalizzazione su alcune problematiche comuni a questo tipo di tecnologie:

Allocazione delle risorse agli utenti

L'allocazione delle risorse agli utenti deve avvenire in modo equo, evitando la monopolizzazione dell'utilizzo del cloud da parte di pochi soggetti a scapito

di altri.

Sicurezza delle informazioni

Le informazioni affidate al cloud devono essere memorizzate garantendone la confidenzialità, l'integrità e la disponibilità ai soggetti autorizzati.

Accesso remoto

Deve essere possibile la connessione dei sistemi degli utenti con le infrastrutture del cloud, alla massima velocità possibile, utilizzando un meccanismo di autenticazione sicura per l'identificazione univoca degli utenti.

Tutti questi problemi hanno già possibili soluzioni in tecnologie esistenti, altri necessitano di ultriori sviluppi.

In particolare la connessione dei sistemi utenti con l'infrastruttura di cloud richiede l'adattamento di una possibile soluzione, ai requisiti della virtualizzazione di un laboratorio didattico di informatica.

L'obiettivo di questo studio è proporre una possibile soluzione per questo problema.

1.2 Protocolli e dispositivi di rete

Nella descrizione ed organizzazione di un architettura di rete si presta molto bene l'utilizzo di un modello a livelli di astrazione, in cui ogni livello, agisce come una *scatola nera* fornendo un interfaccia, visibile al livello superiore, per l'uso delle sue funzionalità, utilizzando quelle del livello inferiore.

Secondo quest'architettura ogni livello comunica logicamente con il suo omologo nel sistema ricevente, ma le informazioni transitano attraverso tutti i livelli inferiori, fino al più basso, per poi risalire i layer nel sistema destinatario.

Attraverso questa astrazione si consente all'implementatore di concentrarsi sullo sviluppo delle funzioni del livello selezionato tralasciando la visione globale.

1.2.1 Modello ISO/OSI

L'architettura *OSI*, mostrata nella figura 1.1, è stata una delle prime ad essere definite per i sistemi di comunicazione di rete, sviluppata dalla *ISO*[1], stabilisce 7 livelli di astrazione, partendo dallo strato più basso troveremo:

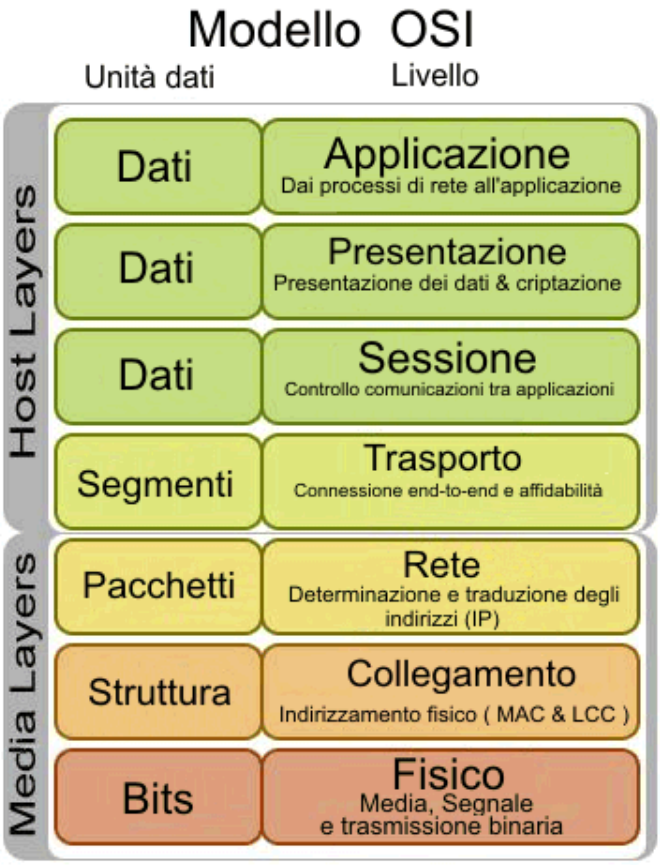

Figura 1.1: Modello OSI

[1]International Organization for Standardization: organismo internazionale per la definizione di norme tecniche.

Livello Fisico

Gestisce la trasmissione di flussi di bits, attraverso il medium trasmissivo scelto.

A questo livello lavorano i *modulatori*[2] e *demodulatori* delle interfacce di rete che si occupano di convertire i bits in segnali adatti al mezzo trasmissivo utilizzato.

Livello di Collegamento Dati

Gestisce la trasmissione di aggregati delimitati di bits chiamati *frame*, tra host direttamente connessi, connessi tramite un *commutatore*, o connessi con un mezzo trasmissivo *broadcast*[3], eseguendo opzionalmente anche operazioni di controllo del flusso e degli errori.

A questo livello generalmente lavorano le componenti logiche delle schede di interfaccia ed i drivers dei sistemi operativi.

Spesso è suddiviso in ulteriori due sottolivelli che svolgono operazioni logiche specifiche:

- Logical Link Control: fornisce i servizi di *multiplexing* per fare condividere a più protocolli di livello superiore l'accesso al mezzo trasmissivo. Fornisce anche controllo del flusso e alcuni meccanismi di gestione degli errori.

- Medium Access Control: gestisce l'accesso al mezzo trasmissivo e fornisce un sistema di indirizzamento tra gli host interessati nella comunicazione.

[2]Modulatore/Demodulatore: dispositivo che consente di modificare un segnale elettromagnetico, sulla base di un secondo segnale contenente informazione, con il fine di trasmettere l'informazione del secondo segnale.

[3]Broadcast: modalità di trasmissione dell'informazione, inviata da un nodo di una rete e ricevuta da tutti i nodi collegati.

Livello di Rete

Gestisce la trasmissione di aggregati di dati chiamati *pacchetti*, tra host non direttamente collegati, fornendone un meccanismo di indirizzamento.

Consente l'interconnessione di reti di livello 2.

Livello di Trasporto

Fornisce ai livelli superiori l'astrazione di un canale di comunicazione tra host *punto-punto*, spesso questo livello implementa meccanismi per rendere il canale affidabile, ovvero tecniche per la creazione di un servizio orientato alla connessione, di ritrasmissione eventuale dei pacchetti, di controllo di flusso e congestione e si occupa, inoltre, di mantenere il corretto ordine di consegna dei i pacchetti.

Livello di Sessione

Questo livello fornisce tecniche di controllo e gestione di una sessione di comunicazione, ovvero sistemi per l'aggregazione di diversi canali di trasporto del livello inferiore che possono essere parte della stessa applicazione.

Livello di Presentazione

In questo livello sono presenti i servizi di conversione del formato dei dati, dell'*endianness*[4], conversione del set dei caratteri, compressione dati e spesso di crittografia.

Livello di Applicazione

Questo livello fornisce i protocolli ed i servizi per la comunicazione tra i processi su diversi host, si appogga sui servizi dei livelli inferiori per realizzare la trasmissione.

[4]Endianness: metodo di indicizzazione dei bytes della memoria, big endian conta i byte partendo dal più significativo, little endian partendo dal meno significativo.

Solitamente troviamo implementato tutto il modello solo sugli host terminali, mentre nei sistemi all'interno della rete troviamo solo fino al livello di trasporto o di rete.

Per un approfondimento rimandiamo a [1]

1.2.2 Modello TCP/IP

Il modello *TCP/IP* continua ad utilizzare l'astrazione a livelli del modello *ISO/OSI*, ma riduce il numero dei livelli a soli 4, figura 1.2, importata da [2]:

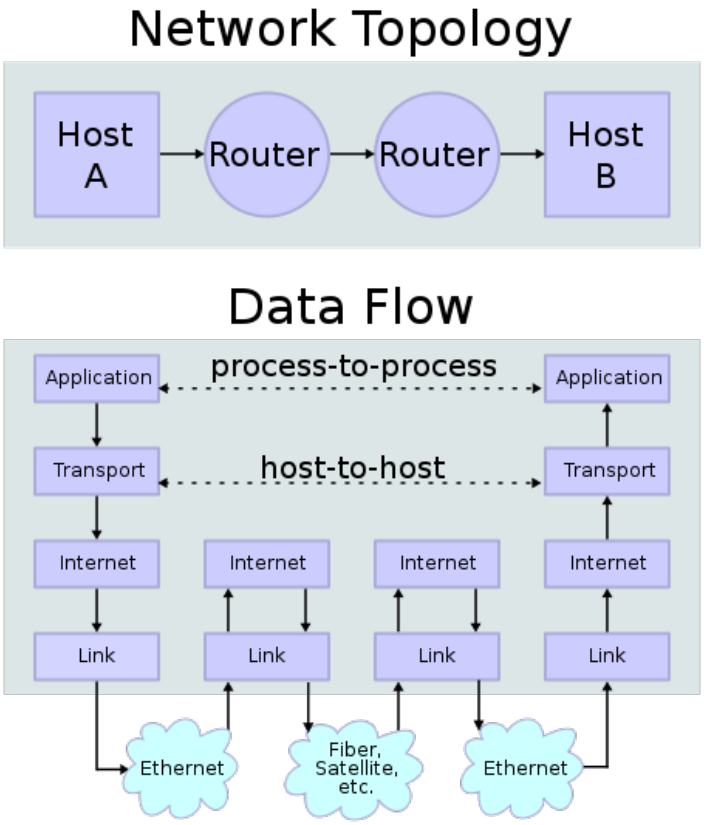

Figura 1.2: Modello TCP/IP

Collegamento Dati

Il *TCP/IP* utilizza i servizi di questo livello per fornire intercomunicazione tra reti non direttamente connesse.

Può utilizzare vari tipi di reti di livello 2.

Livello di Rete

In questo livello è presente un unico protocollo, il protocollo *IP Internet Protocol*, che consente l'interconnessione tra più reti non direttamente collegate, favorendo la creazione di un unica rete logicamente interconnessa.

Livello di Trasporto

Il livello di trasporto fornisce due soli protocolli di trasporto, *Trasmission Control Protocol TCP* e *User Datagram Protocol UDP*, il primo consente la creazione di un canale sicuro e affidabile di comunicazione orientato alla connessione, mentre il secondo fornisce un canale di comunicazione non affidabile orientato alla consegna di datagrammi.

Applicazione

Fornisce un insieme di protocolli applicativi destinati alla comunicazione diretta tra le applicazioni, come ad esempio: *HTTP*[5], *FTP*[6], *Telnet*[7], *SMTP*[8].

L'architettura di *TCP/IP* non è però rigida e predeterminata, infatti un protocollo applicativo può utilizzare direttamente i livelli inferiori scavalcando quello immediatamente sottostante, viene lasciata alla libera scelta del programmatore di rete.

[5]HTTP: Hyper Text Transfer Protocol: protocollo di trasferimento di ipertesti, è il principale protocollo di trasferimento di informazioni nel web.

[6]FTP: File Transfer Protocol protocollo usato per il trasferimento di file attraverso la rete.

[7]Telnet: protocollo per l'accesso alle interfacce di comando di elaboratori remoti.

[8]Simple Mail Transfer Protocol: protocollo per la trasmissione di email sulla rete.

1.2.3 Dispositivi Hardware

Canali di comunicazione

Possiamo trattare come componenti hardware anche i mezzi fisici di comunicazione come:

1. Cavi di rame: su cui le informazioni vengono inviate sotto forma di segnali elettrici modulati;

2. Fibre ottiche: dove l'informazione viene scambiata come segnale luminoso;

3. Spazio Libero: dove le informazioni vengono inviate sotto forma di onde elettromagnetiche che a seconda della lunghezza d'onda prendono il nome di:

   - Radio Frequenza: con lunghezze d'onda maggiori o uguali a 1mm;
   - Infrarossi: con lunghezze d'onda comprese tra 1mm e 700nm;
   - Laser: con lughezze d'onda comprese tra 700nm e 400nm.

Interfacce di rete

Le interfacce di rete sono i dispositivi che consentono l'accesso al sistema di trasmissione fisico, agiscono tra lo strato fisico e quello del collegamento dati.

Hub, concentratori, ripetitori

Sono dispositivi, che agiscono al livello fisico, utilizzati principalmente nelle reti su cavo con tipologia a stella, in cui tutti gli host vengono collegati ad un punto centrale di concentrazione delle comunicazioni.

Molto spesso ripetono ed amplificano i segnali ricevuti, prima della ritrasmissione.

Switch, bridge

Gli *switch* o *bridge* sono dispositivi con la stessa funzione del concentratore, che però lavorano al livello collegamento dati, analizzano i *frame* che ricevono, per individuare l'host esatto a cui vanno inviati, ottimizzando l'utilizzo della banda della rete.

Access Point

Un *Access Point* radio è un dispositivo che lavora a livello datalink, fungendo da ponte tra la rete cablata e quella radio.

Consente ai client con interfacce di rete radio, di collegarsi ai servizi della rete cablata.

Router, firewall

I *Router* sono i dispositivi che interconnettono più reti di livello 2, formando la rete del livello 3.

Si occupano dell'instradamento dei pacchetti tra reti diverse realizzando la commutazione a livello 3.

L'instradamento può essere realizzato tramite le connessioni dirette con altri host e reti, oppure tramite la consegna dei pacchetti ad altri router di cui si è a conoscenza.

1.2.4 Protocolli di Livello Collegamento dati

I protocolli di rete del livello collegamento dati, comunemente usati nelle reti locali sono:

- IEEE 802.3: definisce una tecnologia di comunicazione per la realizzazione di reti locali con supporto cablato;
- IEEE 802.11: definisce uno standard di comunicazione per l'implementazione di reti LAN senza fili;

Nella sottosezione seguente si può trovare una breve analisi del protocollo IEEE 802.3 Ethernet.

IEEE 802.3 Ethernet

Ethernet è tecnologia di rete, per la realizzazione di reti cablate, non orientata alla connessione, di tipo CSMA/CD ovvero:

- Carrier Sense: ogni nodo della rete riesce sempre a distinguere se il mezzo trasmissivo è occupato o meno;
- Multiple Access: il mezzo trasmissivo è ad accesso multiplo, ovvero è condiviso tra tutti i nodi collegati;
- Collision Detection: ogni nodo resta sempre in ricezione anche quando trasmette, riuscendo a rilevare quando la trasmissione interferisce con un'altra.

Il protocollo Ethernet può utilizzare come supporti fisici di trasmissione:

- cavo di rame bilanciato in doppini intrecciati
- cavo coassiale sbilanciato
- fibre ottiche.

Le trasmissioni avvengono in *banda base*[9] con *bandwidth* di 10, 100, 1.000 o 10.000 Mbps a seconda del supporto e dell'implementazione del protocollo utilizzata.

Ogni trasmissione nel canale condiviso raggiunge tutti i nodi collegati, creando un unico dominio di collisione e rendendo possibili solo trasmissioni *half-duplex*[10].

[9]Trasmissione in banda base: trasmissione in cui non vi è modulazione del segnale, ma viene utilizzata tutta la banda del canale.

[10]Half-Duplex: tipo di trasmissione bidirezionale, in cui può trasmettere solo un nodo per volta, mentre tutti gli altri ricevono.

In origine prevedeva una topologia a *bus*, che connetteva tra loro tutti i nodi della rete. Si è passati successivamente ad una topologia a *stella* in cui tutti i nodi sono connessi ad un unico punto centrale di concentrazione o più punti interconnessi tra loro.

Per aumentare le velocità di collegamento ed ottenere trasmissioni *full-duplex*[11] si è reso necessario introdurre una segmentazione del dominio di collisione in più domini, uno per ogni nodo della rete, aggiungendo dei dispositivi di commutazione, che consentono di indirizzare i frame ricevuti, direttamente al nodo a cui sono destinati, creando un collegamento commutato punto-punto tra i due nodi.

La più piccola entità inviabile nella rete Ethernet è il *frame*, che ha la struttura seguente, tabella 1.1:

Campo	PRE	SFD	DA	SA	Len/Type	Payload	FCS
Bytes	7	1	6	6	2	46-1500	4

Tabella 1.1: Frame Ethernet

- Preambolo: i primi 7 byte sono necessari per la sincronizzazione degli adattatori di rete.
- Delimitatore di inizio Frame: 1 byte contenente una sequenza di bit stabilita come limite di inizio del frame.
- Indirizzo destinazione: 6 byte per l'indirizzo dell'host destinatario.
- Indirizzo sorgente: 6 byte per l'indirizzo dell'host sorgente.
- Lunghezza o tipo: 2 byte per la lunghezza o il tipo del pacchetto.
- Payload: da 46 a 1500 bytes per i dati trasportati.

[11] Full-Duplex: tipo di trasmissione bidirezionale, in cui i nodi possono trasmettere contemporaneamente.

- Frame check sequence: 4 byte di sequenza di controllo in cui si trova il valore di CRC del pacchetto, per la verifica degli errori di trasmissione.

Gli indirizzi ethernet consistono di una sequenza univoca di 6 byte, rappresentati come una stringa di 6 coppie di cifre esadecimali, es: 01:00:0c:cc:cc:cc, che vengono memorizzati dal costruttore nel *firmware* dell'interfaccia di rete.

Sono consentiti indirizzi:

- Unicast: i frame con un indirizzo destinazione di questo tipo sono normalmente ricevuti solo dal nodo a cui sono destinati
- Multicast: i frame con un indirizzo multicast di destinazione sono ricevuti dai nodi che dichiarano di volere ricevere dati dal quel gruppo multicast
- Broadcast: i frame inviati con questo indirizzo di destinazione sono normalmente ricevuti da tutti i nodi della rete.

Gli indirizzi multicast hanno il primo bit settato ad 1, mentre l'indirizzo broadcast è un indirizzo speciale con tutti i bit settati ad 1.

Normalmente le interfacce di rete ethernet ricevono tutti i frame trasmessi sulla rete, ma inoltrano al host solo quelli esplicitamente destinati ad esso, tranne il caso in cui l'interfaccia venga settata in modo *promiscuo*, consentendo l'inoltro all'host di tutti i frame ricevuti.

1.2.5 Protocolli di Livello Rete

Il protocollo di livello 3, più usato in ogni dispositivo di rete e su cui si basa l'indirizzamento dell'intera rete mondiale è Internet Protocol:

Internet Protocol

Nato come protocollo di interconnessione tra reti di livello datalink. Fornisce metodi per l'indirizzamento univoco degli host interconnessi.

Esistono due versioni successive del protocollo di cui la seconda è l'evoluzione di quella originaria:

IPv4 La prima versione introdotta, che utilizza uno schema di indirizzi numerici di 32 bit, consente l'indirizzamento di massimo 2^{32} host. Gli indirizzi sono rappresentati normalmente stampando i valori dei 4 bytes, che li compongono, separati da punti es. 192.168.0.1. Questa versione del protocollo è attualmente utilizzata, anche se lo spazio di indirizzamento si sta esaurendo.

IPv6 Seconda versione del protocollo, implementata principalmente per superare le limitazioni alla capacità di indirizzamento del protocollo v4. Consente l'indirizzamento di massimo 2^{128} host. Gli indirizzi sono normalmente rappresentati stampando i valori dei 16 bytes che li compongono, sotto forma di una stringa esadecimale, a coppie di 2 byte alla volta, utilizzando come carattere separatore i due punti.
Esempio di indirizzo IPv6: 2001:760:2e00:ff00:0000:0000:0000:0001

1.3 Sicurezza ed AAA

La sicurezza nell'ambiente informatico si può mettere in atto attraverso il rispetto di politiche di:

- Confidenzialità: assicurare che solo i soggetti autorizzati possano avere accesso alle informazioni;
- Integrità: garantire la protezione delle informazioni da modifiche ed alterazioni non volute;
- Disponibilità: garantire alle entità autorizzate l'accesso alle informazioni;
- Non ripudio: garantire che un soggetto non possa negare un'azione da lui eseguita.

Queste politiche possono essere realizzate attraverso meccanismi detti di *AAA*, acronimo di:

Authentication

L'autenticazione assicura l'identificazione univoca dei soggetti, che vogliono accedere ai sistemi ed alle informazioni, attraverso un identità digitale verificata tramite il possesso di credenziali quali password, certificati digitali o metodi biometrici.

Authorization

L'autorizzazione stabilisce le azioni che possono essere eseguite, sui sistemi e sulle informazioni, dai soggetti autenticati, stabilendo restrizioni diverse a seconda del soggetto in esame.

Accounting

L'accounting consente la misurazione del consumo delle risorse utilizzate, dai vari soggetti, consentendo la verifica a posteriori, il tracciamento e la prevenzione di eventuali abusi, nell'accesso alle informazioni.

Per approfondimenti si rimanda a [5].

1.3.1 RADIUS

Acronimo per *Remote Authentication Dial-In User Service*, è un protocollo di *AAA*, considerato lo *standard de-facto* per l'autenticazione e l'accounting su reti IP e per l'accesso remoto. [4]

Viene utilizzato principalmente in ambienti caratterizzati dalla presenza di vari dispositivi di accesso, con necessità di autenticazione di un gran numero di utenti, come reti di campus e organizzazioni.

Consente di mantenere i database delle credenziali di autenticazione in una posizione centralizzata senza duplicare i dati in ogni dispositivo di connessione, spesso integrato e caratterizzato da insufficienti risorse hardware. Fornisce inoltre alcuni schemi basilari di protezione contro attacchi di *sniff-*

ing[12] della fase di autenticazione, tramite un meccanismo di cifratura a chiave condivisa.

Protocollo

Radius utilizza il protocollo di trasporto *UDP* per la comunicazione tra il server ed i dispositivi di connessione, la struttura del pacchetto radius è la seguente, figura 1.3:

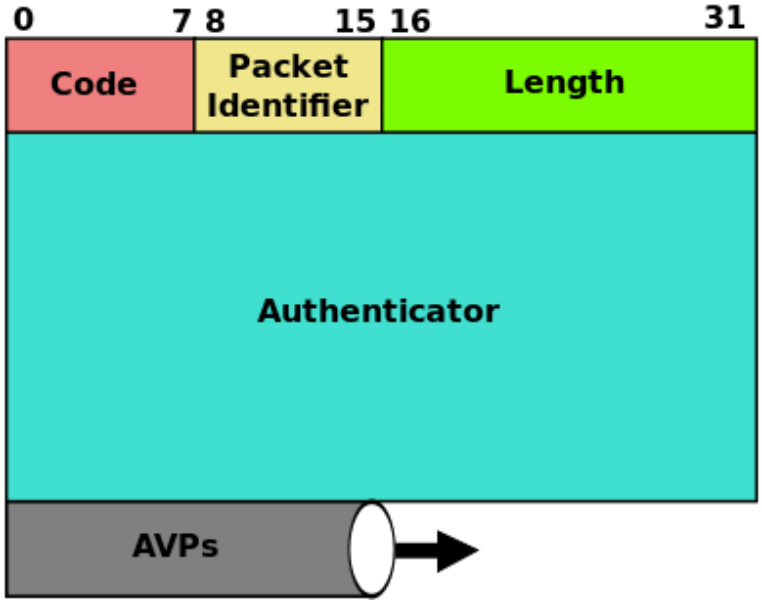

Figura 1.3: Pacchetto Radius

Il campo più importante è il codice che stabilisce il tipo di pacchetto, i codici più usati sono i seguenti:

- Access-Request: codice 1, pacchetto che transita dal dispositivo di connessione al server per richiedere l'accesso;
- Access-Accept: codice 2, pacchetto che transita dal server al dispositivo per autorizzare l'accesso;

[12]Sniffing: tecnica di attacco che prevede l'intercettazione delle comunicazioni tra due soggetti.

- Access-Reject: codice 3, pacchetto che transita dal server al dispositivo per rifiutare l'accesso;

- Accounting-Request: codice 4, pacchetto proveniente dal dispositivo e diretto al server, è una richiesta relativa al meccanismo di accounting;

- Accounting-Response: codice 5, pacchetto proveniente dal server destinato al dispositivo, è una risposta ad una richiesta di accounting;

- Access-Challenge: codice 11, pacchetto proveniente dal server verso il dispositivo, si richiedono ulteriori informazioni al client.

Per la verifica delle credenziali possono essere utilizzati tre protocolli diversi di autenticazione, ai cui viene applicato, nel tratto tra il server e l'autenticator, il meccanismo di cifratura a chiave condivisa del protocollo radius:

Password Authentication Protocol

Protocollo di autenticazione, generalmente utilizzato per la verifica dell'accesso tramite collegamenti punto-punto, richiede l'invio delle credenziali, al server di verifica, senza alcun tipo di cifratura, rendendolo un meccanismo non sicuro.

Challenge Authentication Protocol

Meccanismo di autenticazione che utilizza un challenge inviato dal server al client in risposta ad una richiesta di accesso. I due soggetti condividono una password segreta, necessaria al client per creare un hash *md5*[13] del challenge, che viene verificato dal server tramite la stessa password, accettando o meno la richiesta di accesso. Per aumentare la sicurezza questa procedura viene ripetuta più volte a connessione già stabilita.

[13]Message Digest 5: algoritmo i hashing di un messaggio.

Extensible Authentication Protocol

Framework di autenticazione standard molto flessibile, generalmente utilizzato per l'autenticazione delle reti wireless. Consente la definizione di una interfaccia comune per il trasporto di vari metodi di autenticazione, a seconda dei requisiti dell'ambiente di utilizzo, con vantaggi e svantaggi. Generalmente *EAP* stesso non fornisce funzioni di protezione e cifratura dell'autenticazione, affidandosi ai protocolli inferiori, o lasciando ai metodi trasportati la definizione di meccanismi di sicurezza [3].

I metodi più comuni o maggiormente utilizzati sono:

EAP-MD5 : è l'applicazione del *CHAP* utilizzato all'interno di *EAP*, non è un metodo consigliato nel caso sia necessario un alto livello di sicurezza. Consente l'utilizzo solamente di credenziali user-password ed è vulnerabile agli attacchi a dizionario. Non fornisce possibilità di mutua autenticazione, nel caso in cui la rete autentichi il client ed il client la rete.

EAP-TLS : *TLS*[14] applicato ad *EAP* consente un elevato livello di sicurezza. Le normali credenziali vengono sostituite con certificati, verificati da una *PKI*[15], ed obbligatori sia lato client che lato server, realizzando la mutua autenticazione. Lo svantaggio è il costo computazionale e di manutenzione, elevato rispetto ai metodi basati su credenziali a causa della complessità dei sistemi software necessari.

EAP-TTLS : *Tunneled Transport Layer Security* è un metodo di autenticazione basato sulla creazione di un tunnel, nato come evoluzione del *EAP-TLS*, per rimuovere la necessità del possesso di certificati per i client. L'autenticazione viene effettuata in due fasi: nella prima viene creato un tunnel cifrato e verificato tramite i servizi del protocollo *TLS*,

[14]Transport Layer Security: protocollo che consente di rendere sicuro il canale di trasporto al livello 4, utilizzando crittografia e firme digitali.

[15]PKI: Public Key Infrastructure è un autorità di certificazione per la verifica dell'identità di un utente e della sua chiave pubblica.

nella seconda fase viene utilizzato un altro metodo di autenticazione, trasportato all'interno del tunnel, per verificare l'idendità del client, al termine della verifica il tunnel cifrato viene distrutto. Lo schema di autenticazione interno può essere un altro metodo *EAP* oppure un vecchio protocollo come *PAP*, *CHAP*, *MS-CHAP*[16], *MS-CHAPv2*[17].

EAP-PEAP : *Protected EAP* è insieme ad *EAP-TLS*, lo standard *EAP* più utilizzato e supportato al mondo. Si basa anch'esso sulla creazione di un tunnel cifrato in cui veicolare l'effettivo protocollo di autenticazione. Il protocollo trasportato è dipendente dalla versione di *PEAP*. Nel caso della versione 0, la più diffusa, l'unico protocollo utilizzabile è l'*EAP-MS-CHAPv2*, che vincola all'utilizzo di password in chiaro o all'applicazione del risultato dell'algoritmo di hash *NT* applicato alla password. La versione 1 prevede l'applicazione del metodo *EAP-GTC*[18], consentendo l'autenticazione basata su smart-card e one-time-password.

Autenticazione ed Autorizzazione

La procedura di autenticazione-autorizzazione prevede che il client si colleghi al dispositivo di accesso, denominato *Remote Access Service*, utilizzando il protocollo di autenticazione scelto, eseguendo una richiesta di accesso, attraverso il media di livello 2 utilizzato. Il *RAS* invia al server *RADIUS* un pacchetto di tipo *Access-Request* contenente le credenziali di accesso, incapsulate nel protocollo di autenticazione, ed altre informazioni di indentificazione dell'utente. Il server *RADIUS* risponde con un pacchetto di responso che può essere: *Access-Accept* se la richiesta è stata accolta, *Access-Challenge* se la richiesta necessita di ulteriori informazioni, *Access-Reject* se la richiesta

[16]MS-CHAP: Implementazione Microsoft del protocollo CHAP, utilizza la cifratura DES ed MD4 invece di quella MD5.

[17]MS-CHAPv2: versione 2 del protocollo MS-CHAP, prevede un handshake della connessione tra client e server utilizzando l'identificativo di connessione e stringhe pseudo casuali. Prevede l'utilizzo dell'algoritmo di hashing NT.

[18]EAP-GTC: Generic Token Card, metodo di autenticazione basato sull'uso di smart-card a cui sono associate one-time password.

è stata rifiutata. Nel caso di responso positivo il server può restituire ulteriori informazioni necessarie al client, come indirizzi IP, parametri di *QoS*, il massimo tempo di accesso al servizio, parametri della *VLAN* etc. Una volta ricevuto il responso il dispositivo di accesso applica la risposta del server *RADIUS*, richiedendo ulteriori informazioni o consentendo-rifiutando la richiesta di accesso.

Accounting

Una volta completata ed autorizzata l'autenticazione, il protocollo consente l'avvio di una sessione di *Accounting*. Il dispositivo di accesso invia al server un pacchetto di tipo *Accounting-Request*, con l'attributo *start*, contenente varie informazioni relative al client ed ai servizi in uso, necessarie per eseguire operazioni di fatturazione del servizio, controllo delle informazioni e logging. Durante tutta la durata della sessione periodicamente vengono inviati pacchetti con l'attributo *interim update*, necessari all'aggiornamento regolare delle informazioni registrate. Al momento di chiusura della sessione, si ha l'invio di un pacchetto con l'attributo *stop*, per terminare la registrazione delle informazioni e stabilire la terminazione della sessione.

1.4 Tecnologie di AAA e sicurezza su rete Ethernet

Le politiche di *AAA* possono essere implementate a diversi livelli ed in diversi dispositivi delle reti, con i corrispondenti vantaggi e svantaggi.

1.4.1 Autenticazione al livello 2

L'autenticazione al livello 2 prevede che ogni client che desidera effettuare una trasmissione di dati attraverso la rete, debba effettuare un autenticazione, altrimenti lo switch di accesso rifiuterà completamente ogni informazione proveniente dal client.

Normalmente si prevede che la porta dello switch, a cui il client è collegato consenta solo il transito di traffico di autenticazione ; il normale traffico inizia ad essere accettato in caso di completamento positivo delle operazioni di verifica delle credenziali.

Vantaggi e Svantaggi

L'autenticazione obbligatoria per l'accesso al mezzo trasmissivo consente un effettivo controllo sui soggetti autorizzati all'effettuare operazioni sulla rete. Il principale svantaggio è l'assenza della crittografia sul traffico generico, perchè generalmente si utilizza un canale cifrato solo per rendere sicura la fase di autenticazione.

IEEE 802.1X

Lo standard *802.1X* è un modello architetturale per il controllo degli accessi alle porte di una rete *LAN*, fornisce uno schema di autenticazione, basato sul framework *EAP*, per i dispositivi che desiderano collegarsi ad una rete, cablata o wireless, della famiglia *IEEE 802*. Prevede l'incapsulamento dei pacchetti *EAP* in pacchetti di rete Ethernet *802.3* o *802.11*, realizzando il protocollo *EAP over LAN*.

Il protocollo *802.1X* prevede tre entità che partecipano al processo di connessione, figura 1.4, importata da [2]:

- Supplicant: il client che desidera connettersi alla rete, anche se generalmente si intende come supplicant il software, in esecuzione nel client, che esegue le richieste di connessione.
- Authenticator: il dispositivo di accesso alla rete, che provvede ad inoltrare il protocollo *EAP* dal client all'authenticazion server ed a mantenere il client disconnesso dalla rete fino ad autenticazione ultimata positivamente.
- Authentication Server: il server che implementa le politiche di *AAA*, e prende la decisione di accettare o rifiutare la richiesta di connessione,

generalmente è un server *RADIUS*, collegato al database centralizzato delle credenziali.

La procedura di autenticazione, su rete Ethernet, prevede il blocco della porta di accesso, da parte dell'*authenticator*, consentendo il transito al solo traffico *EAPoL*, fino alla conclusione positiva della procedura di connessione, quando l'authenticator apre la porta al traffico generico.

Il dispositivo di accesso oltre a garantire la sicurezza mantenendo il controllo sulle porte agisce da come una sorta di *proxy*, ricevendo i pacchetti *EAP* provenienti dal supplicant, incapsulati nei frame ethernet, e ritrasmettendoli tramite il protocollo *RADIUS* al server di autenticazione.

La sicurezza dell'autenticazione è garantita dal metodo di *EAP* scelto per l'operazione.

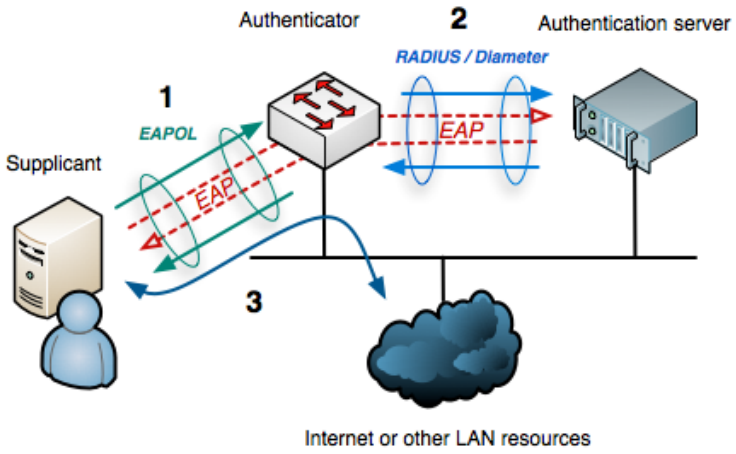

Figura 1.4: Protocollo 802.1X

L'uso di *802.1X* può provocare in alcuni contesti problemi di sicurezza, perchè il protocollo non prevede dopo che l'autenticazione è andata a buon fine, la necessità di controllare l'identità del client, rendendo possibili attacchi di *MAC spoofing*[19] e *Denial of Service*[20] dovuti alla trasmissione di falsi

[19]MAC Spoofing: falsificazione dell'indirizzo hardware dell'interfaccia di rete.

[20]Denial of Service: tipo di attacco, che ha come obiettivo la negazione del servizio.

pacchetti *EAP-Logoff*[21].

1.4.2 Autenticazione al livello 3

L'autenticazione al livello 3 è una possibile soluzione, per l'autenticazione delle richieste di accesso ad Internet da una rete interna, che ha un solo punto di collegamento verso la rete pubblica. In questo caso essendo presente un solo punto di transito, si può pensare di implementare le politiche di *AAA*, nel router stesso, il quale provvederà a rifiutare il forwarding di tutti i pacchetti dall'interno verso l'esterno se l'indirizzo ip sorgente non è presente tra quelli dei clients che si sono autenticati.

In questo caso quindi il client ottiene direttamente dalla connessione al segmento di rete interna un indirizzo pubblico, che però non viene instradato se il client non è autenticato.

Vantaggi e Svantaggi

In questo caso l'autenticazione avviene solo nel punto di interfaccia tra la rete interna e quella pubblica, quindi sfuggono al controllo i soggetti che effettuano operazioni solamente all'interno della rete, rendendo la sicurezza della rete interna pari a quella di una rete non autenticata.

Vite

Un esempio di questa soluzione è l'utilizzo della suite *Vite*, sviluppata da Mauro Amico, Renzo Davoli, Marco Giordani e Leonardo Macchia; che consente al router di accesso di inserire regole dinamiche di filtraggio dei pacchetti, basate sull'autenticazione *SSH* degli utenti che richiedono l'accesso ad Internet. Fino a che la sessione ssh dell'utente sul router rimane aperta, viene mantenuta anche la regola che consente il forwarding dei pacchetti, quando la sessione viene chiusa anche il forwarding viene interrotto.

[21]EAP-Logoff: pacchetti di disconnessione del protocollo EAP.

Queste operazioni sono correlate al logging, per esigenze di controllo, degli indirizzi IP e Ethernet del client connesso.

1.4.3 VPN: Rete Privata Virtuale

Una rete privata virtuale è una tecnologia che permette di emulare la presenza di un collegamento fisico dedicato tra reti e/o host presenti in sedi diverse, utilizzando la rete pubblica come canale di trasporto, mantenendo il più possibile la stessa confidenzialità fornita da un collegamento fisico dedicato.

Il rispetto di questo requisito può essere ottenuto realizzando tramite software e/o hardware appositi, un tunnel cifrato di collegamento tra le due entità in causa, quindi un canale di comunicazione sicuro all'interno di un medium pubblico, che non fornisce un livello di sicurezza affidabile.

La cifratura del canale è sempre abbinata a meccanismi di autenticazione sicura, che consentono di stabilire esattamente l'identità delle entità in comunicazione ed evitare possibilità di intercettazione dei dati o falsificazione delle identità, per esempio tramite attacchi *man in the middle*[22].

Possiamo suddividere le VPN in 2 macrotipi:

VPN Host to LAN

Il canale virtuale fornisce un collegamento sicuro tra un singolo client ed una rete di elaboratori considerata sicura. Solitamente viene scelta questa tipologia per garantire collegamenti sicuri di una macchina isolata, come può essere quella di un lavoratore, in collegamento con la rete aziendale, o di uno studente che deve interagire da remoto con i sistemi universitari.

[22]Man in the middle: attacco in cui un soggetto malevolo si inserisce in una comunicazione in corso, in modo trasparente agli interlocutori, con l'intento di intercettare le informazioni.

VPN LAN to LAN

L'altra tipologia consente l'interconnessione sicura di intere reti, generalmente utilizzata quando vi è la necessità di collegare due sedi remote della stessa organizzazione senza l'utilizzo di costosi circuiti dedicati.

La tecnologia di tunneling delle VPN può essere a sua volta suddivisa in due tipi principali, a seconda del tipo di pacchetto che vi transita all'interno:

VPN Bridged

Se i pacchetti veicolati sono di livello 2, generalmente Ethernet, si è in presenza di una VPN *bridged* in quanto le due reti o l'host e la rete remota sono collegate al livello collegamento dati, quindi il collegamento virtuale, può trasportare qualunque tipo di protocollo di terzo livello, supportato dal *datalink* scelto, insieme ai *multicast/broadcast* di livello 2. Il risultato logico è che le due reti sono collegate come se fossero un'unica rete di livello 2.

VPN Routed

Nel caso in cui i pacchetti veicolati siano di livello 3, il collegamento avviene solo tramite il tipo di protocollo di rete scelto, *IPv4* o *IPv6*. Si è quindi in presenza di una VPN di tipo *routed*, in quanto l'instradamento avviene tramite regole di routing statiche o dinamiche. L'interconnessione ottenuta è quindi sfruttabile solo dal livello 3 in su.

Protocolli e implementazioni

Esistono vari protocolli utilizzabili per l'implementazione di VPN, alcuni generici come *IPSec*, *SSL/TLS*, altri dedicati per questo obiettivo come *L2TP* e *PPTP*. Alcune implementazioni prevedono l'utilizzo contemporaneo di alcuni di essi, per soddisfare i requisiti che potrebbero essere assolti singolarmente.

- IPSec: è uno standard che si prefigge di rendere sicure le reti IP, aggiungendo la capacità di cifratura ed autenticazione forte al livello rete, così

da rendere trasparente questa funzione ai livelli superiori. Può essere utilizzato sia in comunicazioni host to host, nel caso di comunicazioni tra due singoli sistemi, che network to network quando vengono rese sicure le comunicazioni tra i router perimetrali di due reti.

- SSL/TLS: standard che prevede l'implementazione di capacità di sicurezza ed integrità nelle comunicazioni, al livello di trasporto, utilizzando tecniche di crittografia sia simmetrica che asimmetrica, per garantire la riservatezza dei dati e Message Authentication Code insieme a firme digitali per garantire l'autenticazione ed il non ripudio dei messaggi.

- Layer 2 Tunnelling Protocol: protocollo che consente di creare un tunnel di livello 2 punto-punto tra host differenti, utilizzando i meccanismi offerti da altri protocolli di livello inferiore per garantire riservatezza ed autenticazione, tipicamente *IPSec*.

- Point to Point Tunneling Protocol: protocollo che tramite la tecnologia *Generic Routing Encapsulation*[23] consente la creazione di un tunnel per il trasporto di frame *PPP*[24] punto-punto, che possono veicolare a loro volta pacchetti IP, consentendo la comunicazione tra i sistemi interessati. L'implementazione della sicurezza nelle comunicazioni viene affidata totalmente alle funzionalità offerte dal protocollo *PPP*.

- OpenVPN[25]: è un implementazione di *VPN*, viene utilizzato per creare tunnel crittografati punto-punto, tra host che hanno eseguito una mutua autenticazione, attraverso chiavi condivise, certificati digitali e credenziali utente. Per la gestione dell'autenticazione e cifratura viene utilizzato il protocollo *SSL/TLS*.

[23]GRE: protocollo progettato per il tunneling, è pensato per veicolare altri protocolli di rete.

[24]Point to Point Protocol: protocollo di livello collegamento dati, utilizzato per il collegamento diretto punto-punto tra due nodi.

[25]OpenVPN: software per l'implementazione di VPN, utilizza SSL/TLS. http://openvpn.net

- VDE: Virtual Distributed Ethernet[26] non è propriamente un'implementazione di *VPN*, ma può essere utilizzato con successo per la creazione di reti private virtuali, perchè consente di veicolare frame *Ethernet*, attraverso qualunque stream di dati, consentendo la creazione di *VPN* di tipo *bridged*. La gestione della cifratura e dell'autenticazione viene lasciata ai meccanismi dello stream dati a cui si appoggia.

Vantaggi e Svantaggi

L'utilizzo delle *VPN*, nell'autenticazione delle connessioni cablate, consente di autenticare e cifrare tutto il traffico di rete tra i due host in comunicazione, consentendo di mantenere un canale confidenziale di comunicazione, proteggendo anche dalla possibilità di avere soggetti fraudolenti che si spacciano per i legittimi destinatari delle comunicazioni.

Gli svantaggi evidenti nell'utilizzo di questa tecnologia sono: l'*overhead* introdotto dalla cifratura applicata a tutto il traffico e l'utilizzo della comunicazione in tunnel che potrebbe provocare riduzioni del MTU[27] dei pacchetti a causa degli header aggiunti. Un altro svantaggio è la richiesta di configurazioni supplementari spesso non standard delle connessioni di rete.

[26]VDE: Virtual Distributed Ethernet, rete Ethernet distribuita virtuale, suite software sviluppata dal Prof. Renzo Davoli nell'ambito del progetto Virtualsquare. http://vde.sourceforge.net/

[27]Maximum transmission unit: dimensione massima del pacchetto dati inviabile.

Capitolo 2

Implementazione

Il problema che si vuole risolvere in questo lavoro è la connessione di elaboratori client e la loro integrazione, con la rete di un laboratorio didattico virtuale.

I sistemi generalmente utilizzati, sono calcolatori portatili dotati di capacità di connessione sia a reti cablate che wireless e generalmente gli utenti preferiscono, per comodità d'uso, l'utilizzo della connessione senza fili.

In un laboratorio virtuale, l'accesso alla rete deve, però, essere il più possibile trasparente, senza provocare, rallentamenti nell'utilizzo dei servizi, quindi si è privilegiato lo studio e la sperimentazione, per l'implementazione di un meccanismo di connessione via cavo ad alta velocità, in grado di garantire, alta larghezza di banda e basso ritardo, per avvicinare maggiormente l'esperienza d'uso, a quella di un terminale fisico locale.

2.1 Obiettivi e Requisiti

L'obiettivo principale è fornire una connessione alla rete affidabile e con la massima velocità possibile, utilizzando quindi la tecnologia di rete maggiormente disponibile anche sugli elaboratori portatili. La scelta è ricaduta quindi sul protocollo *IEEE 802.3 Gigabit Ethernet*, rispettando alcuni requisiti definiti vincolanti:

Sicurezza

L'utilizzo della rete e dei servizi deve essere possibile solo ai soggetti autorizzati e in possesso delle credenziali d'accesso.

Devono essere implementati meccanismi per garantire:

- autenticazione ed autorizzazione: per prevenire accessi illegali ed usi impropri del servizio;
- logging e accounting: per registrare le modalità ed i tempi di utilizzo del servizio, permettendo la ricostruzione di eventuali violazioni;

Queste caratteristiche sono normalmente richieste dalle reti senza fili, che per la morfologia stessa del mezzo trasmissivo, non sono possibili da limitare all'interno di uno spazio chiuso. Nelle nelle reti cablate, di solito, non sono necessarie, perchè è sufficiente la limitazione, del dovere ottenere l'accesso fisico al canale di trasmissione.

Efficienza e scalabilità

I meccanismi di autenticazione, autorizzazione e gestione delle risorse devono mantenere la massima efficienza nell'utilizzo delle risorse computazionali e di banda della rete per mantenere la scalabilità all'aumento del numero dei client e dispositivi di accesso.

Interoperabilità

Le tecnologie ed i protocolli utilizzati devono essere interoperabili al massimo e supportati dai più diffusi sistemi operativi utilizzati dai clients. Il più possibile senza richiedere l'installazione di software supplementari a quelli già forniti nei sistemi operativi.

Libertà degli utenti

Deve essere consentita la massima libertà di utilizzo lecito e sperimentazione per gli utenti, soprattutto nel segmento di rete interna del laborato-

rio. Va evitata per quello che sia possibile la definizione di filtri sul traffico o regole di traffic shaping, realizzando un controllo non invasivo sugli accessi alla rete, in modo da registrare solo l'utilizzo di risorse pubbliche, come indirizzi visibili dalla rete esterna, per esigenze di prevenzione e ricostruzione di eventuali abusi.

Software Libero

Un requisito importante, per l'implementazione, è l'utilizzo di software libero, ed il rilascio dei protocolli e delle soluzioni sviluppate con licenza libera, per consentire a chiunque, l'uso, lo studio, la sperimentazione ed il miglioramento.

2.2 Contesto di applicazione

L'ambiente in cui si svolge la sperimentazione è l'infrastruttura di rete del dipartimento di Scienze dell'Informazione dell'Università di Bologna, in cui convivono e vengono erogati servizi critici ed essenziali per il buon funzionamento della struttura stessa. Si è quindi cercato di eseguire la sperimentazione e lo studio nel modo meno invasivo possibile, evitando per quanto possibile di creare disservizi e fastidi agli utilizzatori dei servizi.

2.2.1 Infrastruttura di rete

La rete del dipartimento si compone di un infrastruttura di interconnessione fisica, basata sull'utilizzo del protocollo *IEEE 802.3 Gigabit Ethernet*, in grado di sostenere trasferimenti dati fino a velocità di 1 Gbps.

L'infrastruttura fisica è suddivisa in sottoreti logiche VLAN, caratterizzate da funzioni differenti, realizzate per mezzo del protocollo *IEEE 802.1Q*[1].

[1]IEEE 802.1Q: protocollo che consente la definizione di reti virtuali, con diverso dominio di collisione all'interno della stessa LAN fisica. Il protocollo modifica i frame Ethernet aggiungendo un campo che contiene un indice numerico che abilita all'uso di 2^{12} reti diverse.

Tutte le VLAN raggiungono i router perimetrali, che consentono l'instradamento dei pacchetti IPv4 e IPv6 verso il *CeSIA*[2] dell'Università di Bologna, punto di passaggio per il transito del traffico verso la rete pubblica.

2.2.2 Infrastruttura di autenticazione

Tutti i servizi funzionanti nella rete dipartimentale richiedono l'autenticazione univoca degli utenti tramite credenziali personali. Il meccanismo di autenticazione è basato sull'obsoleto sistema di *Network Information Service*[3] che consente la condivisione delle mappe, contenenti le credenziali di autenticazione, tra un server centralizzato e clients periferici.

Esiste inoltre un sistema di autenticazione centralizzato gestito dal *CeSIA*, per tutto l'Ateneo di Bologna, che utilizza la tecnologia di gestione delle directory *Active Directory*[4] proprietaria di Microsoft.

2.2.3 Laboratori didattici

La maggior parte dei servizi dipartimentali vengono forniti da elaboratori equipaggiati con sistema operativo *GNU/Linux*, solo una piccola parte utilizza sistemi proprietari.

Nel dipartimento sono presenti due laboratori didattici, in cui viene consentito l'utilizzo, anche da remoto, di elaboratori client, equipaggiati con sistema operativo *Ubuntu GNU/Linux*[5], interconnessi al meccanismo di autenticazione NIS.

[2]CeSIA: Centro Servizi Informatici di Ateneo, organismo che sovrintende la gestione e la manutenzione della rete di Ateneo.

[3]NIS: protocollo utilizzato su Unix e GNU/Linux per la condivisione distribuita dei dati di autenticazione, tra i server ed i clients.

[4]Active Directory: sistema implementato da Microsoft per la gestione di Single Sign On e servizi di directory, utilizza vari altri protocolli tra cui DHCP, LDAP e Kerberos.

[5]Ubuntu GNU/Linux: distribuzione del sistema operativo GNU/Linux, www.ubuntu.com.

2.2.4 Connessioni alla rete

Negli edifici del dipartimento, ma in particolare nei laboratori didattici, viene fornita la possibilità di accesso alla rete, con i propri dispositivi portatili, tramite i servizi:

- ALMAWIFI: rete senza fili, gestita dal *CeSIA*, utilizza il protocollo di autenticazione *IEEE 802.1X* ed il *SSO*[6] tramite le credenziali gestite da Active Directory.
- CSNET: rete cablata e senza fili, gestita dal dipartimento, utilizza il software di autenticazione Vite, con gestione delle credenziali tramite il sistema NIS.

2.3 Sperimentazione

Lo studio e la sperimentazione mirano a trovare una possibile soluzione, per l'autenticazione dei clients intenzionati all'utilizzo dell'accesso alla rete cablata Gigabit Ethernet, mantenendo l'integrazione con il presente sistema di credenziali NIS, ma predisposto per la conversione verso una struttura più avanzata, sicura e flessibile.

La scelta del protocollo di autenticazione è basata sull'esperienza ottenuta dall'utilizzo pregresso di alcune di queste tecnologie in altri progetti e necessità già esistenti all'interno della rete dipartimentale e da alcuni esperimenti svolti su di un ulteriore modello di autenticazione mai applicato in questo contesto.

2.3.1 VPN e autenticazione a livello rete

I metodi di autenticazione e protezione degli accessi basati sul controllo al livello rete e sulla creazione di reti private virtuali, sono già stati applicati con successo in progetti e servizi forniti dalla rete dipartimentale.

[6]SSO: Single Sign On.

In particolare l'autenticazione al 3° livello viene attualmente utilizzata per la protezione dell'accesso ad Internet fornito tramite la connessione wireless e cablata *CSNET*, destinata all'utilizzo da parte del solo personale dipartimentale.

La tecnologia *VPN* è utilizzata per l'autenticazione dell'accesso ad Internet sperimentale fornito dal progetto *Virtualsquare*[7], tramite l'utilizzo del software *VDE*.

Dall'esperienza di utilizzo e configurazione di queste tecnologie si è potuto verificare la loro aderenza ai requisiti posti per l'utilizzo in questo progetto:

Sicurezza

Questo requisito viene rispettato da entrambe le tecnologie, perchè tutte due prevedono almeno la cifratura forte della fase di autenticazione, ed in più il metodo VPN consente la cifratura totale del traffico.

Utilizzo del software libero

Entrambe le tecnologie sono già oggi implementate utilizzando solamente software libero.

Libertà dell'utente

Tutti e due i metodi non limitano la libertà di sperimentazione e di uso da parte degli utenti, anzi le VPN ne rendono massima la sicurezza.

Interoperabilità e semplicità

Entrambi i metodi non consentono la massima interoperabilità e semplicità, perchè richiedono a volte l'installazione di software supplementare ed i protocolli di VPN non sono disponibili *as is* per tutti i sistemi operativi, riducendo la semplicità d'uso per gli utenti.

[7]Virtualsquare: progetto di ricerca, sul mondo della virtualizzazione, portato avanti dal prof. Renzo Davoli. www.virtualsquare.org

Efficienza e scalabilità

Nel caso delle VPN la scalabilità e l'efficienza sono molto compromesse, perchè la cifratura totale del traffico comporta un aumento delle richieste di calcolo e possibili riduzioni della larghezza di banda, nel caso di connessione contemporanea di molti utenti.

2.3.2 Autenticazione a livello datalink

Fino ad ora non esiste nella rete dipartimentale, un sistema che utilizzi un protocollo di autenticazione che preveda la protezione dell'accesso alla rete al 2° livello, percui si è reso necessaria l'implementazione di una piattaforma di test, utilizzando sistemi sperimentali in dotazione ai progetti *Virtualsquare* e *Admstaff*[8].

L'infrastruttura di test prevede l'utilizzo del protocollo *IEEE 802.1X*, abbinato al metodo di autenticazione *Protected-EAP* del framework *EAP*, sfruttando come directory di autenticazione, il server *LDAP*[9] che gestisce le credenziali utente, del cluster di servizi sperimentali del gruppo Admstaff.

Come unico metodo di autenticazione è stato scelto *EAP-PEAP*, perchè risulta essere il meccanismo che offre il miglior compromesso tra sicurezza ed interoperabilità tra tutti i sistemi operativi e i dispositivi hardware.

Infrastruttura di test

L'infrastruttura fisica comprende:

- un gruppo di client multipiattaforma, ogni client ha installato uno dei sistemi operativi più diffusi;

[8]Admstaff: http://admstaff.students.cs.unibo.it/

[9]LDAP: Lightweight Directory Access Protocol protocollo che offre servizi di modifica ed interrogazione di directory di dati, specialmente utilizzato come base per meccanismi di autenticazione.

- una macchina server con sistema operativo *GNU/Linux Debian*[10];
- uno switch Gigabit Ethernet, che consenta il supporto al protocollo *IEEE 802.1X* con metodo *EAP-PEAP*;

L'infrastruttura logica prevede:

- il gruppo di client portatili multipli, per simulare le macchine degli utenti che richiedono l'accesso cablato alla rete;
- un gruppo di client portatili o fissi, per simulare le macchine stabilmente collegate all'infrastruttura, utilizzabili come client per la gestione delle credenziali d'accesso;
- un server di autenticazione RADIUS;
- un server directory LDAP [10], per la memorizzazione e gestione delle credenziali;
- uno o più dispositivi di accesso, che consentono l'autenticazione dei client portatili;

Per semplificare, nell'infrastruttura di test, il server eseguirà le funzioni del RADIUS, directory LDAP e client per la gestione delle credenziali.

Lo switch è collegato al server RADIUS tramite una *VLAN 8021q* dedicata, diversa da quella utilizzata dai client.

Tutte le operazioni tra RADIUS, server LDAP e tools di modifica password, avvengono tramite l'interfaccia di *loopback* locale del server.

I software utilizzati sono i seguenti, tutti installati sul server di test:

- FreeRadius: implementazione libera del protocollo RADIUS;
- OpenLdap: implementazione libera del protocollo LDAP [9];

[10]Debian: distribuzione molto diffusa del sistema operativo GNU/Linux. www.debian.org

- PAM: Pluggable Authentication Module[11], framework di autenticazione per la verifica e la manutenzione centralizzata delle credenziali di accesso ai sistemi GNU/Linux.

Obiettivo della sperimentazione

Al termine della sperimentazione si vuole ottenere l'accesso autenticato alla rete cablata per i client multipiattaforma. Gli utenti devono ottenere l'accesso alla risorsa di rete semplicemente connettendo l'elaboratore portatile alle prese di collegamento ed inserendo, quando richieste, le credenziali di accesso. Non devono essere richieste altre operazioni, a parte piccole configurazioni del sistema operativo.

Requisiti Protected EAP

Il metodo di EAP scelto, per scelte progettuali, consente l'utilizzo solamente del protocollo di autenticazione *MS-CHAPv2*, che effettua la verifica delle credenziali utente utilizzando come password, il risultato dell'algoritmo di hash *NT*. Costringendo a memorizzare nella directory o la password in chiaro o la password sottoforma di hash *NT*. Questo requisito rende incompatibile l'utilizzo del metodo, con le credenziali standard di un sistema GNU/Linux, perchè per la verifica della password viene memorizzato l'output della funzione di hashing *CRYPT*, totalmente diverso ed incompatibile con *NT*.

Hashing NT

Si vuole evitare l'introduzione di ulteriore complessità, come quella che si otterrebbe, aggiungendo alla directory LDAP, gli schemi descrittivi dei campi

[11]PAM: framework che consente la centralizzazione in un unica API di alto livello, di vari metodi per l'autenticazione, permettendo ai programmi che necessitano l'uso di autenticazioni di essere implementati indipendentemente da uno specifico schema di autenticazione. Consente il miglioramento della sicurezza accentrando in una sola posizione gli eventuali problemi che si potrebbero verificare.

del software Samba, per la compatibilità verso i protocolli Windows, necessari solamente alla memorizzazione del digest NT. Quindi si è provveduto a studiare lo sviluppo di una soluzione che prevedesse l'aggiunta alla directory, dei soli campi contenenti i password digest di entrambi gli algoritmi CRYPT ed NT.

L'operazione è stata resa possibile dalla caratteristica del server LDAP, pensata per mantenere l'interoperabilità tra sistemi con metodi di hashing diversi, che consente di memorizzare più di un campo, contenente l'hash della password. La memorizzazione del digest avviene, codificando l'hash in base64 ed anteponendovi un tag identificativo, standard per il tipo di algorimo usato. Di conseguenza i client che accedono alla directory devono essere in grado di decodificare l'insieme tag-digest.

Il server LDAP non ha, però la possibilità nativa di utilizzare l'algoritmo di digest NT, quindi è stato necessario creare un sistema per il calcolo e la verifica dell'hash realizzato come un modulo dinamico dell'eseguibile, grazie alla possibilità di utilizzo di librerie dinamiche.

Modulo dinamico OpenLdap per le operazioni di hash NT

Il modulo aggiunge al server OpenLdap la possibilità di gestire hash di tipo NT, in tutte le operazioni di autenticazione, incluse nel protocollo. Ciò consente ai client che supportano le funzionalità di *extended operation*, tramite la versione 3 del protocollo LDAP, di accedere ai campi contenenti i digest delle password, lasciando l'esecuzione delle operazioni di verifica e creazione degli hash al server stesso.

L'algoritmo per il calcolo dell'hash NT prevede l'applicazione di due operazioni principali, ai dati del testo in chiaro:

1. conversione del set di caratteri da quello originario all'UTF-16LE;

2. esecuzione dell'algoritmo di hash MD4 al risultato dell'operazione precedente;

Il modulo di calcolo, implementato per il server OpenLdap, prevede l'invocazione delle funzioni di calcolo e verifica che applicano l'algoritmo precedente, da parte delle due funzioni handler principali, specificate di seguito.

La funzione seguente per la verifica dell'hash, viene invocata dal server per il controllo delle credenziali.

```
/* Check if digest in passwd->bv_val is equal to the hash
generated from plaintext password in cred->bv_val */
static int checkNtHash(
        const struct berval *scheme,
        const struct berval *passwd,
        const struct berval *cred,
        const char **text )
{
        unsigned char ntHash[MD4_DIGEST_LEN];
        char hashEncoded[LUTIL_BASE64_ENCODE_LEN
                (MD4_DIGEST_LEN)+1];

        if(calcNtHash(cred->bv_val, cred->bv_len, ntHash)
                != __LOCAL_ERROR)
        {
                lutil_b64_ntop(ntHash, MD4_DIGEST_LEN,
        hashEncoded, LUTIL_BASE64_ENCODE_LEN(MD4_DIGEST_LEN)+1);
                return (strcmp(hashEncoded, passwd->bv_val));
        }
        else
                return  LUTIL_PASSWD_ERR;
}
```

La funzione seguente per il calcolo dell'hash, viene invocata dal server per l'aggiornamento delle credenziali.

```
/* Generate the digest from plaintext password in
passwd->bv_val and use pw_string64 function to base64
encode the digest and concatenate with its scheme.
```

```
Finally the scheme+digest is returned in hash */
static int createNtHash(
        const struct berval *scheme,
        const struct berval *passwd,
        struct berval *hash,
        const char **text )
{
        unsigned char ntHash[MD4_DIGEST_LEN];

        if(calcNtHash(passwd->bv_val, passwd->bv_len,
                ntHash) != __LOCAL_ERROR)
        {
                struct berval digest;
                digest.bv_val = (char *) ntHash;
                digest.bv_len = sizeof(ntHash);

                return pw_string64(scheme, &digest, hash, NULL);
        }
        else
                return  LUTIL_PASSWD_ERR;
}
```

La funzione di inizializzaione viene invocata dal server al caricamento del modulo, per la definizione delle funzioni handler, delle richieste di verifica e calcolo dell'hash.

Di seguito la specifica della funzione applicata al modulo implementato.

```
/* Module entry point function */

int init_module(int argc, char *argv[]) {
        return lutil_passwd_add((struct berval *) &ntscheme,
                checkNtHash, createNtHash);
}
```

Le API di OpenLdap forniscono anche funzioni ausiliarie che consentono

di creare e manipolare strutture dati interne al server, necessarie nella creazione e verifica dei digest [9].

Configurazione OpenLDAP

La configurazione opportuna del server LDAP, mediante la direttiva *moduleload* consente il caricamento e l'abilitazione del modulo dinamico NT, precedentemente installato nella directory del server destinata alle librerie ; per la distribuzione Debian essa è emph/usr/lib/ldap.

Tramite l'uso delle direttive *password-hash* seguite dal tag identificativo dell'algoritmo di hashing, $\{NT\}$ o $\{CRYPT\}$, si indica al server la volontà di utilizzare solo gli algoritmi di digest specificati.

L'ultima direttiva di configurazione: *password-crypt-salt-format*, che consente la definizione del formato di salting per l'algoritmo CRYPT, deve essere settata al formato usato dalla distribuzione Linux scelta, generalmente è utilizzato un salt ad 8 caratteri con algoritmo di digest *SHA512*, quindi il parametro della direttiva deve essere: *6%.8s*.

Configurazione PAM

L'attivazione della direttiva *pam_password exop*, nel modulo di accesso ad LDAP della libreria PAM, installata nell'elaboratore Linux di gestione delle credenziali, permette l'utilizzo delle *Ldap Extended Operation*, per le operazioni di modifica password, ottenendo l'aggiornamento trasparente e contemporaneo di entrambi i digest CRYPT ed NT. [14]

Configurazione FreeRADIUS

Il software FreeRadius, a cui è collegato lo switch di accesso, viene configurato per l'utilizzo della directory LDAP. Successivamente si utilizza la direttiva *password_attribute userPassword* per dichiarare l'esatto campo della directory, in cui sono memorizzati i digest delle password, e attivando la funzionalità di autodecodifica del tipo di hash, sulla base del tag identificati-

vo, si consente al server di accedere direttamente all'hash NT per completare le procedure di autenticazione.

Il processo di autenticazione

La procedura di autenticazione secondo il protocollo *IEEE 802.1X* avviene secondo i seguenti passi:

1. Il client si collega fisicamente alla porta di rete e richiede l'accesso. La porta consente solo il passaggio di traffico EAP fino alla fine della procedura autenticativa.

2. Lo switch riceve i pacchetti EAP e li reincapsula in pacchetti RADIUS inviandoli al server.

3. Il server esegue le operazioni di verifica e genera una risposta EAP, ritrasmessa allo switch incapsulata in RADIUS.

4. Lo switch riceve la risposta del RADIUS ed esegue l'operazione corrispondente al tipo del messaggio ricevuto: apertura della porta o rifiuto della connessione.

5. Attivazione delle funzionalità di accounting delle risorse, tramite lo switch.

Punti deboli

La sperimentazione ha permesso di evidenziare sul campo, alcune implementazioni del protocollo *IEEE 802.1X*, che possono essere sfruttati come punti deboli:

- Viene autenticato e registrato solamente il MAC address Ethernet del client, il protocollo non prevede l'accounting dell'indirizzo IP assegnato all'host.

- Il protocollo non prevede alcun controllo sul client già autenticato.

- La deautenticazione si basa sullo stato del link Ethernet, se lo stato del collegamento passa da *up* a *down*, l'equivalente dello scollegamento del cavo, il client si considera deautenticato.

Il primo problema, non è un punto debole e si può risolvere facilmente mediante l'utilizzo di software per il monitor e la registrazione delle corrispondenze tra i MAC address e gli indirizzo IP, come ad esempio *Arpwatch* per IPv4 o *NDMon* per IPv6.

I rimanenti due problemi sono vere e proprie possibili vulnerabilità del protocollo e possono portare al bypass del meccanismo di autenticazione nei casi in cui:

- venga collegato, un dispositivo simile ad un hub, tra un client legittimo e lo switch, consentendo dopo la corretta autenticazione, l'apertura della porta al traffico generico per tutti gli host che si collegano all'hub;
- un soggetto malevolo, interponendo sempre un hub tra client e switch, utilizzi un MAC address clonato al client legittimo, per l'utilizzo improprio della connessione, al termine della sessione del client legittimo;

La soluzione possibile per la riduzione se non l'annullamento di questi due problemi è la configurazione corretta dei parametri dello switch, ausiliari al protocollo di autenticazione:

1. L'impostazione forzata del massimo limite di MAC address transitabili per porta, ad un indirizzo, comporta la soluzione del primo problema.
2. Per ridurre al minimo gli effetti del secondo problema, può essere settato nello switch un breve timeout di riautenticazione, forzando la deautenticazione di un eventuale soggetto fraudolento, non in possesso delle legittime credenziali.

Rispetto dei requisiti

Al termine della sperimentazione si può evidenziare l'aderenza della soluzione esaminata ai requisiti iniziali:

Sicurezza : Il protocollo utilizzato offre un buon livello di sicurezza perchè le credenziali, vengono verificate e scambiate all'interno di un tunnel cifrato, anche se il traffico generico non viene cifrato.

Software Libero e libertà utente : La soluzione è sviluppata utilizzando solamente software libero e l'utente mantiene la libertà di utilizzo e sperimentazione perchè è obbligatoria l'autenticazione ma non vi è alcun blocco o filtro sull'uso legittimo della rete.

Efficienza e scalabilità : La soluzione offre una buona efficienza e scalabilità, poichè non sono presenti operazioni invasive sul traffico della rete. L'unica operazione, con alte richieste computazionali, è la creazione del tunnel cifrato di autenticazione, che però è una frazione del tempo totale della sessione di collegamento.

Il server OpenLdap consente una buona scalabilità nell'accesso alla directory per via di varie tecniche di ottimizzazione e caching nell'accesso ai dati.

Interoperabilità e semplicità : L'utilizzo di questa soluzione risulta essere la migliore per compatibilità e semplicità d'uso, perchè ogni sistema operativo testato ha già al suo interno il software client richiesto dal protocollo, senza necessità di installare programmi supplementari.

La semplicità d'uso risulta molto alta perchè si richiede all'utente la sola operazione di inserimento delle credenziali ed eventualmente piccole semplici configurazioni[12].

2.4 Installazione di produzione

Per l'installazione definitiva di produzione si è scelto di adattare al contesto presente, il modello sperimentato dell'autenticazione al livello datalink,

[12] Ad esempio la selezione manuale del protocollo di autenticazione EAP-PEAP

perchè consente di rispettare maggiormente i requisiti più importanti richiesti dal progetto.

2.4.1 Configurazione fisica

L'adattamento dell'infrastruttura fisica di test al contesto operativo di produzione, non prevede modifiche sostanziali allo schema di collegamento stabilito durante l'esperimento. L'unica differenza è l'aggiunta di un server in più, necessario alla separazione dei servizi RADIUS e LDAP, prima ospitati da un unico elaboratore, su macchine diverse, per esigenze di sicurezza.

2.4.2 Configurazione logica

La configurazione logica prevede modifiche più sostanziali, per l'adattamento dell'infrastruttura di test al contesto operativo. La struttura di base rimane invariata, comprendendo quindi le stesse entità funzionali, ma per necessità di integrazione con i meccanismi già presenti, alcuni servizi vengono sostituiti tramite i loro omologhi già attivi nella struttura attuale.

Il servizio di accesso alla rete richiede, inoltre, l'aggiunta nel sistema di produzione, delle funzionalità di routing ed assegnamento automatico di indirizzi IP pubblici ai clients, non presente nel sistema di test. Il servizio può essere svolto dallo stesso elaboratore che esegue il server RADIUS.

Elaboratori client fissi

La funzione svolta, nel sistema di test, dall'elaboratore utilizzato per la gestione delle credenziali di accesso, viene eseguita, nella struttura di produzione, dal cluster di client condivisi presente nel laboratorio didattico rimanente.

Server RADIUS e LDAP

I servizi dei server RADIUS e LDAP, eseguiti in un solo elaboratore nell'infrastruttura di test, richiedono nel passaggio al sistema di produzione, per

ragioni di sicurezza, la separazione su macchine diverse.

VLAN e indirizzamento IP

Per ragioni di sicurezza il traffico di rete generico dei client viene separato dal traffico RADIUS ed LDAP su due VLAN differenti, configurando lo switch per etichettare con un identificativo diverso i frame provenienti dai client, rispetto a quelli scambiati tra i server.

L'indirizzamento IP dei vari soggetti prevede quattro raggruppamenti di indirizzi:

- Indirizzi pubblici IPv4 e IPv6: si assegnano due sottoreti pubbliche, una per ogni protocollo, destinate all'utilizzo, in regime di autoconfigurazione, da parte dei clients. Un indirizzo per ogni subnet deve essere assegnato all'interfaccia del server RADIUS, facente funzione di router.

- Indirizzi pubblici di transito: si instradano i pacchetti provenienti e destinati agli indirizzi pubblici dei clients, attraverso sottoreti di transito già utilizzate in precendenza nell'infrastruttura.

- Indirizzi pubblici server: è necessario assegnare al server LDAP un indirizzo pubblico, per consentire la sua visibilità, dal cluster di macchine client del laboratorio.

- Indirizzi privati: si assegnano indirizzi IPv4 privati, non instradabili, per la comunicazione all'interno delle VLAN, destinate all'interconnessione dei server RADIUS e LDAP.

2.4.3 Sistema di gestione credenziali

Il meccanismo attivo per la gestione delle credenziali dei clients presenti nei laboratori didattici è basato sull'obsoleto Network Information Service, che consente solamente funzionalità limitate alla manipolazione degli identificativi utente e password di un sistema Linux, senza offrire le funzionalità

supplementari necessarie all'integrazione con i protocolli RADIUS e IEEE 802.1X.

La sostituzione completa del sistema non è praticabile perchè comporterebbe sicuri *denial of service* e disservizi inaccettabili dagli utenti del sistema informatico. Si procede quindi all'affiancamento del sistema NIS con una configurazione del server di directory LDAP, adattata per il funzionamento in parallelo con i servizi già presenti, ed elaborata per consentire l'eventuale sostituzione di NIS in un prossimo futuro.

Configurazione LDAP

Si installa il server LDAP e lo si attiva seguendo lo schema di configurazione elaborato nella sperimentazione. La directory viene, quindi popolata, clonando le informazioni già presenti nel NIS, grazie all'utilizzo delle utilities *MigrationTools*[13], che consentono una conversione dei record di autenticazione dal formato proprio di NIS, ad uno compatibile con la directory LDAP.

Ogni volta che si procederà all'aggiunta di nuove credenziali utente al sistema NIS, dovrà essere ripetuta la procedura di conversione, per le sole nuove credenziali aggiunte, avendo cura di cambiare una volta la password, per permettere la creazione anche dell'hash di tipo NT.

Configurazione clients

Per mantenere la consistenza di entrambe le directory delle credenziali e la trasparenza nei confronti degli utenti, in modo che possano continuare ad aggiornare le password tramite l'uso delle procedure canoniche, si devono configurare opportunamente, i clients del rimanente laboratorio didattico, per informarli del servizio parallelo al NIS.

[13]MigrationTools: utility scritte in perl, per la conversione e l'importazione dei dati contenuti in directory NIS, nel corrispondente servizio LDAP. Sono sviluppate dalla PADL Software Pty Ltd. http://www.padl.com/OSS/MigrationTools.html

L'operazione può essere facilmente effettuata utilizzando una configurazione dedicata della libreria PAM, che importa in parte quella elaborata nella sperimentazione [14]

La configurazione deve consentire l'autenticazione sulle macchine e la modifica delle password di due tipi di utenti:

- utente locale: identificativo presente solo nel file /etc/passwd locale
- utente distribuito: utente presente sia nel database NIS, che nella directory LDAP

Autenticazione Per l'autenticazione non è necessario modificare alcuna configurazione, perchè la libreria PAM viene opportunamente settata, anche per l'utilizzo di LDAP, al momento dell'installazione del modulo dedicato alla comunicazione con i server LDAP.

Modifica Password Il funzionamento corretto della possibilità di modifica delle password, necessita invece di una configurazione più complessa, dovendo discriminare il tipo di utenza per cui si sta aggiornando il record. Si mostra di seguito la configurazione elaborata per consentire il funzionamento previsto:

```
password [success=done default=ignore] pam_unix.so obscure sha512
password [success=ok default=ignore] pam_unix.so obscure sha512 nis
password [success=done user_unknown=ignore default=die] pam_ldap.so
use_authtok try_first_pass
```

La prima linea consente la modifica della password di un utente locale della macchina, se l'operazione termina correttamente, la valutazione del file non procede, invece se l'utente non esiste nei files locali la valutazione prosegue.

La seconda linea consente la modifica della password di un utente presente nel NIS e se la modifica termina correttamente, la valutazione

prosegue sulla terza linea che consente l'aggiornamento delle credenziali LDAP, mediante l'utilizzo degli identificativi e password forniti per la modifica precedente, parametro *try_first_pass*.

Nel caso di esito negativo viene eseguita la clausola ignore, sia per NIS che per LDAP.

Mediante queste configurazioni gli utenti possono aggiornare la propria password mediante una sola chiamata del comando */usr/bin/passwd*.

Per gli utenti importati dal NIS, l'aggiornamento della password è propedeutico al primo utilizzo del servizio, per consentire la generazione iniziale del password digest NT, in quanto le credenziali NIS, contengono solamente l'hash della password di tipo CRYPT e non esiste un operazione in grado di convertire il risultato di una funzione hash, nell'output di un altra, senza l'utilizzo dei dati in chiaro.

2.4.4 Servizi ausiliari di rete

Il corretto funzionamento del servizio di accesso alla rete dipende anche dalla presenza di alcuni servizi essenziali come l'assegnamento automatico degli indirizzi IP ed il server DNS.

La presenza delle funzionalità di accounting e logging consente di prevenire usi illeciti del servizio ed in caso di necessità, ricostruire eventuali violazioni commesse.

Autoconfigurazione IP

L'indirizzamento dei clients che accedono al servizio è realizzato tramite l'uso di entrambe le versioni del protocollo IP, riservando per il segmento di rete del laboratorio le sottoreti pubbliche:

- IPv4: *130.136.46.0/24* che comprende 254 indirizzi utilizzabili, di cui 2 vengono riservati per il router ed utilizzi futuri.

- IPv6: *2001:760:2e00:f02e::/64*, sottorete con prefisso 64 bit, consente l'autoconfigurazione dell'indirizzo tramite il MAC address della scheda di rete.

Il router risponde agli indirizzi:

- IPv4: *130.136.46.254*
- IPv6 *2001:760:2e00:f02e::1*

L'assegnamento degli indirizzi è realizzato mediante la configurazione, a bordo della macchina che svolge il compito di router, dei software:

- ISC-DHCPD: con cui viene implementato il protocollo DHCPv4 per l'assegnamento automatico degli IPv4;
- RADVD: tramite il quale si consente il funzionamento del meccanismo di autoconfigurazione intrinseco al protocollo IPv6;

Accounting e logging

La funzionalità di accounting fornita dal protocollo RADIUS consente il logging preciso delle operazioni di autenticazione degli utenti. In particolare permette la registrazione e memorizzazione, corredate di timestamp, delle associazioni tra l'identificativo utente e l'indirizzo MAC Ethernet del dispositivo utilizzato per la connessione, sia nel momento di avvio che in quello di chiusura della sessione.

La registrazione di queste informazioni non basta, però, a garantire la possibilità di ricostruire eventuali violazioni o operazioni illecite commesse eventualmente dagli utenti, perchè l'unico indirizzo raggiungibile all'esterno della rete è l'indirizzo IP, che però non viene registrato.

L'utilizzo di indirizzi ip pubblici facilita l'implementazione di soluzioni per la registrazione delle associazioni tra IP e MAC address, escludendo l'utilizzo di operazioni a livello trasporto come il *connection-tracking*, onerose dal punto di vista della banda di rete.

Queste operazioni sarebbero necessarie nel caso dell'utilizzo di indirizzi privati mascherati con un sistema di *NAT*[14], che non utilizzando una corrispondenza univoca tra gli indirizzi pubblici ed i privati, avrebbe spezzato la corrispondenza IP-utente.

Le due soluzioni esistenti analizzate per realizzare il logging delle associazioni tra MAC ed IP, prevedono l'utilizzo dei software:

- ArpWatch: Sistema software che utilizzando la libreria *pcap*[15], analizza il traffico del protocollo *ARP*[16], memorizzando le associazioni rilevate tra IP e MAC e notificando tramite messaggi email e/o registrazione di log, eventuali modifiche alle coppie precedentemente rilevate.

  Il programma può registrare solamente le associazioni che prevedono l'uso di indirizzi IPv4.

- NDMon: Sistema software omologo di ArpWatch, svolge la medesima funzione, ma tramite l'utilizzo del protocollo *ND*[17], quindi consente la registrazione delle sole associazioni di indirizzi IPv6.

Si è preferito non utilizzare le soluzioni già presenti, per privilegiare lo sviluppo di un'unica soluzione che consentisse sia il logging di indirizzi IPv4 che IPv6 e che non utilizzasse la libreria pcap per la cattura del traffico di rete, ma una soluzione più efficiente.

La sperimentazione ha portato allo sviluppo del software *Linux Neighbor Logging System*, che viene descritto nel capitolo dedicato.

[14]NAT: Network Address Translation, sistema che consente il risparmio sull'utilizzo degli indirizzi IP pubblici, sostituendo l'indirizzo sorgente dei pacchetti provenienti da indirizzi privati, con l'indirizzo pubblico esterno del router, che esegue l'algoritmo di NAT.

[15]Libpcap: libreria che fornisce delle API per la cattura del traffico di rete.

[16]ARP: Address Resolution Protocol, protocollo per la mappatura degli indirizzi IPv4 assegnati alle macchine sui corrispondenti indirizzi Ethernet.

[17]ND: Neighbor Discovery protocol è il protocollo che svolge la funzione dell'ARP per gli indirizzi IPv6, implementa anche ulteriori funzionalità come l'autoconfigurazione degli indirizzi ed il router advertisment.

Capitolo 3

Linux Neighbor Logging System

La necessità di analizzare e registrare le corrispondenze tra indirizzi IP e MAC Ethernet, degli host client di una rete, in modo più efficiente possibile ha portato alla sperimentazione e realizzazione di questa soluzione software.

3.1 Soluzione: obiettivi e requisiti

L'obiettivo da raggiungere è la possibilità di memorizzare le corrispondenze tra gli indirizzi di livello datalink e network di tutti gli host di una rete, il cui traffico è obbligato al transito attraverso dei punti ben definiti, come possono essere dei router o firewall perimetrali.

Si tratta quindi di aggiungere, sul router perimetrale, l'esecuzione di un operazione di log ad ogni attivazione del protocollo di *Neighboring*, necessario alla mappatura degli indirizzi di rete su quelli datalink.

3.1.1 Requisiti

Lo sviluppo deve tenere conto dei requisiti di:

Efficienza

L'uso delle risorse di elaborazione e memoria deve essere il più contenuto possibile per non inficiare la velocità di instradamento dei pacchetti, che potrebbe causare perdita di prestazioni della rete.

Configurabilità

Va considerata la possibilità di eseguire l'operazione di logging solo delle coppie con determinati indirizzi o gruppi di tali, da cui la necessità di configurare le caratteristiche di logging a runtime.

GNU/Linux

La soluzione deve essere sviluppata per il funzionamento in ambiente GNU/Linux.

IPv4-6 compatibile

La soluzione software deve dare la possibilità di memorizzare sia le associazioni tra indirizzi IPv4 e MAC address che quelle con gli indirizzi IPv6.

3.1.2 Soluzione possibile

La soluzione immediata può essere la registrazione e memorizzazione, corredata da timestamp, delle entry della tabella associativa in cui il kernel Linux memorizza, le corrispondenze tra gli indirizzi di rete e collegamento dati, riportate grazie all'utilizzo dei protocolli di mappatura ARP e ND.

3.2 Kernel Linux e Neighbor

Nel kernel Linux la gestione dei Neighbor [8] è implementata da un sottosistema dedicato, con struttura ed interfaccia indipendente, sia dal protocollo datalink che da quello di rete.

3.2.1 Funzionamento generale

Il sottosistema di gestione dei Neighbor è progettato per assolvere i compiti di:

- mappatura degli indirizzi level 3 su quelli level 2;
- caching tramite un struttura hash table, delle corrispondenze, tra indirizzi gia risolte;
- caching di frame level 2 degli indirizzi datalink già risolti, per velocizzare le future richieste di trasmissione dati agli host in cache;

Mappatura indirizzi

La procedura di funzionamento generica prevede che un protocollo di livello superiore, richieda al sottosistema di Neighboring, un indirizzo di livello datalink.

Nel caso di indirizzo presente in cache, la richiesta viene subito soddisfatta. Nel caso, invece, di cache-miss viene eseguita la procedura di risoluzione, che basandosi sul tipo dell'indirizzo al livello rete, attiva il protocollo apposito per la corretta mappatura.

L'aggiunta di record alla cache può avvenire, inoltre, in caso di ricezione di pacchetti notificatori gratuiti dei protocolli di risoluzione.

Invalidazione record

Ad ogni record della tabella cache è associato un timeout, scaduto il quale, una routine di *garbage collection*[1] del kernel si occupa di rimuovere le entry non più aggiornate.

[1]Garbage collection: procedura di recupero della memoria utilizzata da dati non più utilizzati, o inconsistenti.

Notifiche

Ogni volta che la tabella cache dei neighbor viene aggiornata, sia per operazioni di aggiunta che per quelle di rimozione, il sottosistema del kernel, provvede ad inviare un messaggio di notifica, per eventuali applicazioni in spazio utente, al gruppo multicast *RTMGRP_NEIGH* del protocollo *Netlink*.

3.2.2 Protocollo Netlink

Il protocollo *Netlink* [6],[7] è un meccanismo full-duplex, di *Inter Process Comunication*[2] tra processi in spazio utente e tra spazio utente e kernel.

L'interfaccia lato userspace è utilizzabile attraverso il meccanismo dei *socket*[3], mentre in spazio kernel è utilizzabile tramite un insieme di *API*.

Generalmente viene utilizzato dai software di controllo dei sottosistemi di rete del kernel, come *Iptables*[4] ed *Iproute2: suite userspace per la configurazione dei parametri di rete del kernel Linux.*.

Full-duplex e Multicast

La comunicazione tramite Netlink è bidirezionale full-duplex, ovvero può essere iniziata, indifferentemente, sia dal lato Kernel che dal lato client, inoltre il protocollo implementa il supporto alle comunicazioni multicast, consentendo l'invio di informazioni punto-multipunto tramite l'uso di indirizzi di gruppo.

L'indirizzamento del destinatario dei messaggi avviene utilizzando il *PID* del processo ricevente ; il kernel utilizza *PID* uguale a 0.

[2]Inter Process Comunication: tecniche fornite dal sistema operativo per consentire la comunicazione ed il trasferimento di dati tra i processi.

[3]Socket: oggetto software su cui si può eseguire operazioni di lettura e scrittura, che rappresenta l'astrazione del canale per la comunicazione tra entità di una rete o tra processi.

[4]IPTables: utility userspace che utilizza il protocollo Netlink , per la configurazione del sottosistema di firewalling Netfilter del kernel Linux.

Interfaccia userspace

Lato userspace può essere utilizzato tramite le primitive di gestione dei socket, specificando con la canonica funzione *socket*, la famiglia *AF_NETLINK*, con tipo di socket *SOCK_RAW* e protocollo scelto tra quelli implementati da Netlink.

Si mostra un esempio dell'utilizzo della funzione di libreria *int socket()*, per l'apertura di socket *Netlink*, della famiglia *NETLINK_ROUTE*:

```
int fd = socket(AF_NETLINK, SOCK_RAW, NETLINK_ROUTE);
```

Protocolli

I protocolli più utilizzati sono:

1. NETLINK_ROUTE: consente l'utilizzo di operazioni che configurano i vari sottosistemi di rete, tra cui:
   - gestione link;
   - gestione neighbor;
   - gestione indirizzi;
   - sottosistema routing;
   - filtraggio rete;
   - quality of service;
2. NETLINK_FIREWALL: fornisce l'accesso ad un'interfaccia per la ricezione dei pacchetti dal firewall;
3. NETLINK_NFLOG: fornisce un'interfaccia di comunicazione tra il sottosistema Netfilter ed il tool userspace IPTables

3.3 Funzionamento

L'applicazione *Linux Neighbor Logging System* permette di eseguire il logging delle associazioni IP - MAC address della macchina su cui è in esecuzione, tramite l'elaborazione e l'analisi delle notifiche sull'aggiornamento della tabella dei Neighbor, ricevute attraverso il gruppo multicast NETLINK, *RTMGRP_NEIGH*.

3.3.1 Opzioni da riga di comando

L'eseguibile, consente la modifica di alcune opzioni di configurazione dell'applicazione tramite parametri della riga di comando.

Output dei dati

Si permette la selezione della locazione dove stampare i dati in output tramite i parametri:

- *-O* oppure *–stdout* consente la stampa su standard output
- *-F* oppure *–filelog* filename consente la stampa su file di testo
- *-s* oppure *–syslog* consente la stampa tramite il sottosistema Syslog

Filtri

Si consente la definizione di filtri per vincolare il logging alle opzioni di selezione definite, sulla base dei parametri:

- *-I* oppure *–interfaces int1,int2...* consente la selezione delle coppie ricevute tramite le interfacce selezionate
- *-A* oppure *–addrfamily inet—inet6* consente la selezione delle coppie con indirizzo level 3 della famiglia selezionata

- *-S* oppure *–subnets sub/mask,sub/mask1,..* consente la selezione delle sottoreti a cui devono appartenere gli indirizzi delle coppie che si vuole selezionare

I filtri possono essere definiti insieme o singolarmente.

Modalità background e debug

Le modalità: esecuzione in background e debug possono essere attivate con le seguenti opzioni:

- *-D* oppure *–debug* attiva la modalità di debug
- *-d* oppure *–daemonize* attiva la modalità esecuzione in background

3.3.2 Cache timeout

Un'ulteriore funzionalità implementata è una routine di controllo basata su timeout, in grado di verificare se una coppia ricevuta è già stata analizzata e memorizzata.

Alla ricezione di una nuova coppia viene verificata la sua presenza in un'area di memoria del programma, utilizzata come cache, se viene trovata, la nuova coppia non viene registrata nel log, altrimenti verrà registrata nella cache, insieme al timestamp di ricezione, quindi scritta nel log.

Periodicamente una funzionalità di garbage collection analizza la cache per rimuovere le coppie, il cui tempo di ricezione è più vecchio di un certo timeout. Il timeout è modificabile solo nel sorgente del programma.

3.4 Implementazione

L'applicazione è stata scritta completamente utilizzando il linguaggio C ed un modello di progettazione modulare, per concentrare nel medesimo file

sorgente, gli elementi comuni dell'implementazione, aumentando la possibilità di riuso delle procedure software e per mantenere al minimo la complessità delle singole componenti. Di seguito viene mostrata l'implementazione delle strutture di memorizzazione per i dati significativi provenienti dalle notifiche, insieme alla suddivisione modulare del programma.

3.4.1 Strutture dati

Le principali strutture dati implementate sono quelle che organizzano la memorizzazione dei dati registrati dalle notifiche ricevute:

neighBourBlock

La struttura ha una corrispondenza univoca con le notifiche ricevute e consente di memorizzare, per ogni notifica, le informazioni significative ottenute tramite Netlink, per consentire il funzionamento del programma.

La struttura neighBourBlock è definita come segue:

```
struct neighBourBlock {
        unsigned char addressFamily;

        // network level address can be inet or inet6 address
        union {
                unsigned char inetAddr[INETLEN];
                unsigned char inet6Addr[INET6LEN];
        };
        // local link address, only ethernet is supported
        unsigned char etherAddr[ETH_ALEN];

        // interface index and name
        unsigned int if_index;
        char if_name[IF_NAMESIZE];

        // timestamp
```

```
        time_t last_seen;

        // ptr for hash table collision list
        struct neighBourBlock *t_next;
        struct neighBourBlock *t_prev;

} __attribute__((packed));
```

La specifica dei campi è la seguente:

`unsigned char addressFamily`

contiene il codice della famiglia dell'indirizzo di livello rete, AF_INET per IPv4 o AF_INET6 per IPv6;

`unsigned char inetAddr[INETLEN]`

contiene l'eventuale indirizzo IPv4, memorizzato sotto la forma di un vettore di 4 bytes;

`unsigned char inet6Addr[INET6LEN]`

contiene l'eventuale indirizzo IPv6, nella forma di un vettore di 16 bytes;

`unsigned char etherAddr[ETH_ALEN]`

contiene l'indirizzo hardware ethernet, sotto la forma di un vettore di 6 bytes;

`unsigned int if_index`

contiene il codice numerico assegnato dal sistema operativo all'interfaccia di rete a cui si riferisce la notifica Netlink;

`char if_name[IF_NAMESIZE]`

contiene l'identificativo ASCII dell'interfaccia di rete corrispondente all'indentificativo numerico memorizzato;

`time_t last_seen`

instante temporale di ricezione della notifica, consente la verifica del time-out di garbage collection e viene aggiornato all'ultimo istante temporale di ricezione di una notifica già memorizzata

`struct neighBourBlock *t_next`

puntatore a struttura neighBourBlock, viene utilizzato insieme al corrispondente *t_prev* per l'implementazione di una struttura dati *lista concatenata bilinkata*, per la gestione delle eventuali collisioni nella struttura hashtable.

3.4.2 Moduli

La suddivisione modulare del sorgente segue la separazione in macroaree destinate all'implementazione dei singoli gruppi di funzionalità del programma:

Ciclo principale

Il modulo principale è implementato nel file *nlSystem.c* che contiene le funzioni di livello più alto, tra cui:

- `int main(int argc, char *argv[])`

  funzione punto di ingresso, l'esecuzione del programma parte da questa funzione;

- `void mainLoop(void)`

  procedura che ripete le operazioni del ciclo principale, viene interrotta solamente con l'esecuzione del gestore di un segnale di terminazione;

- `void initStruct(void)`

  procedura per l'esecuzione delle operazioni di inizializzazione delle strutture dati della memoria e l'apertura dei file e socket descriptor necessari;

- `void pktSave(struct neighBourBlock *neighBour)`

  funzione che richiama le procedure necessarie all'avvio delle operazioni di verifica e logging di una notifica ricevuta;

- `void setSignalHandlers(void)`

  funzione di configurazione per i gestori dei segnali ricevibili dal sistema operativo;

Elaborazione notifiche

La funzionalità di elaborazione delle notifiche è svolta dalle procedure presenti nel file *nl2Neigh.c*, che estraggono dal messaggio Netlink, i dati significativi:

- indirizzo IP, v4 o v6;
- indirizzo datalink;
- famiglia indirizzo IP;
- identificativo interfaccia;

per poi memorizzarli, includendo il timestamp di ricezione della notifica, in una struttura di tipo *neighBourBlock*.

Nel caso della ricezione di una notifica per dati già presenti, il timeout di garbage collection, associato a quella struttura, viene resettato.

Le funzioni presenti nel modulo sono le seguenti:

- `struct neighBourBlock *packetConverter(struct nlmsghdr *nlMsgHdr)`

  esegue le operazioni di importazione e memorizzazione nella struttura neighBourBlock corrispondente, per i dati ricevuti dalla notifica Netlink;

- `struct neighBourBlock *parseNlPacket(struct nlmsghdr *nlMsgHdr)`

  esegue il controllo sul tipo di notifica ricevuta prima di richiamare la funzione di memorizzazione dei dati;

- `inline int packetTest(struct nlmsghdr *nlMsgHdr, int len)`

  utilizza le funzioni fornite da Netlink per eseguire un controllo di consistenza sulla notifica ricevuta dal kernel;

Gestione del logging

In questo modulo, implementato nel file *logging.c*, sono presenti le funzioni di conversione dei dati dalla forma memorizzata, al formato stringa adatto alla stampa, insieme alle procedure per la stampa su: *file*, *standard output* e sistema *Syslog*.

Sono riportate le principali procedure:

- `void logWrite(struct neighBourBlock *neigh)`

  funzione principale di logging, a seconda della modalità di configurazione, richiama le procedure per eseguire il logging sui dispositivi di output scelti;

- `static void str2FdPrint(struct neighBourBlock *neigh, FILE *fd)`

  procedura per l'esecuzione del logging di una notifica, tramite file descriptor, quindi nel caso che sia stata selezionata la stampa su file di testo o standard output;

- `static void sysLogPrint(struct neighBourBlock *neigh)`

  funzione di esecuzione del logging tramite il sottosistema Syslog;

- `static void neigh2Ascii(struct neighBourBlock *neigh, char *printOutBuf, int outLen)`

  procedura di conversione dei dati dal formato di memorizzazione a quello ASCII adatto per la stampa;

Controllo dei timers

Le operazioni di garbage collection sono realizzate tramite l'utilizzo del segnale *SIGALRM*, inviato dal servizio di *interval timer* del sistema operativo, alla scadenza del tempo impostato.

In questo caso il timer è coincidente con il timeout che stabilisce il termine dell'intervallo di garbage collection per le strutture in memoria. Ad ogni scadenza dell'interval timer viene eseguita la funzione di gestione del segnale, che consente l'inizio dell'operazione di garbage collection.

Le principali procedure sono:

- `void intervalTimerStart(void)`

  funzione di configurazione dell'interval timer del sistema operativo, stabilisce anche la procedura di handler per il segnale *SIGALRM*;

- `inline void mask(void)`

  funzione di disabilitazione dei segnali, per evitare race condition;

- `inline void unMask(void)`

  funzione di riabilitazione dei segnali precedentemente disabilitati;

Implementazione dei filtri

Il modulo contiene tutte le operazioni per l'impostazione e la verifica dei filtri sui pacchetti. Le operazioni di impostazione e di settaggio dei filtri sono eseguite all'avvio dell'applicazione, mentre quelle di verifica vengono richiamate alla ricezione di ogni notifica.

Le funzioni principali di configurazione del filtraggio sono:

- `void filterSetAF(int af)`

  procedura di configurazione del filtro di selezione della famiglia degli indirizzi IP contenuti nelle notifiche;

- `void filterAddInterface(unsigned int int_index)`

  funzione di configurazione del filtro per la selezione dell'interfaccia di ricezione del pacchetto che viene notificato;

- `int filterAddSubnet(char *subNet)`

  procedura di definizione del filtro basato sulla selezione delle sottoreti IP a cui devono appartenere gli indirizzi contenuti nelle notifiche che si vuole loggare;

mentre le principali procedure di verifica:

- `int filter(struct neighBourBlock *neighBour)`

  principale procedura di filtro, richiama le singole funzioni che eseguono la selezione;

- `static inline int verifyAF(unsigned char af)`

  applica la selezione delle notifiche che rispettano il filtraggio basato sulla famiglia dell'indirizzo IP;

- `static inline int verifyInt(unsigned int if_index)`

  applica la selezione tramite l'interfaccia di ricezione del pacchetto;

- `static inline int verifySubnet(struct neighBourBlock *neighBour)`

  applica la ricezione sulla base delle sottoreti a cui appartiene l'indirizzo del pacchetto;

Tabella di hash

Per velocizzare la ricerca delle strutture *neighBourBlock*, contententi dati già ricevuti, si è scelto di utilizzare una struttura hashtable, in modo da mantenere il tempo per l'operazione di ricerca il più possibile costante, all'aumentare del numero delle strutture memorizzate.

Il modulo contiene le funzioni che implementano la ricerca delle strutture già presenti, insieme alle procedure di garbage collection, per i dati scaduti.

La verifica della presenza all'interno della tabella, di una struttura uguale a quella appena ricevuta, si effettua tramite il controllo della corrispondenza di tutti i dati contenuti nelle due strutture confrontate.

Le principali funzioni contenute nel modulo sono:

- `void ip_hash_gc(void)`

  handler del segnale *SIGALRM*, esegue la periodica rimozione dall'hashtable delle strutture con timeout scaduto;

- `static inline void delete_hash_entry(struct neighBourBlock *old)`

  funzione di eliminazione della struttura selezionata;

- `int neighHashFindAdd(struct neighBourBlock *neighBlock)`

  aggiunge una struttura all'hash table;

- `static inline int ip_hash(int len,unsigned char *addr)`

  funzione di hashing, restituisce un valore hash calcolato utilizzando i valori di un vettore di bytes, viene applicata all'indirizzo IP memorizzato, nella struttura da aggiungere all'hashtable;

Parametri di configurazione da riga di comando

La decodifica dei parametri della riga di comando viene effettuata dal modulo apposit,o che utilizza le funzioni fornite dagli altri sottosistemi, per settare le configurazioni dei vari parametri di funzionamento.

Le procedure principali sono le seguenti:

- `void parseCmdLine(int argc, char *argv[])`

  funzione principale di parsing dei parametri, richiama le singole funzioni per la configurazione dell'applicazione, sulla base dei parametri specificati;

- `static void filterSubnets(char *subNetsArg)`

  procedura per la gestione del parametro di filtraggio tramite sottoreti;

- `static void filterInterfaces(char *interfaces)`

  procedura per la gestione del parametro di filtraggio delle interfacce;

- `static void filterAddrFamily(char *addrFamily)`

  procedura per la gestione del parametro di filtraggio della famiglia di indirizzi IP;

- `static void sysLog(void)`

  funzione per l'abilitazione del logging tramite syslog;

- `static void stdOutput(void)`

  funzione per l'abilitazione del logging tramite standard output;

- `static void fileLog(char *logFileName)`

  procedura per l'abilitazione del logging tramite file di testo;

- `static void daemonize(void)`

  funzione che abilita la modalità di esecuzione in background;

3.4.3 Procedura di esecuzione

La procedura alla base del programma inizializza le strutture dati, apre i socket ed i file, quindi esegue un ciclo infinito che ripete le operazioni principali:

1. ricezione di una notifica Netlink;
2. sanity check e verifica del tipo della notifica ricevuta, per accettare solo le operazioni di aggiunta entry alla tabella Neighbor;
3. allocazione di una struttura per la memorizzazione dei dati ricevuti dalla notifica;
4. trattamento dei dati memorizzati a seconda della configurazione selezionata dall'utente:

- stampa immediata su standard output, se è selezionato il modo debug;

oppure

- applicazione dei filtri di selezione, e conseguente scarto dei pacchetti, o prosecuzione al passo successivo;
- verifica se un elemento uguale è già presente nella tabella hash, quindi in caso positivo dealloca la memoria oppure logga il pacchetto ricevuto;

L'uscita dal ciclo principale di esecuzione può avvenire solo tramite l'esecuzione degli handlers dei segnali:

SIGALRM : consente il funzionamento delle operazioni di garbage collection;

SIGHUP, SIGINT, SIGTERM : viene eseguita una funzione che chiude tutti i file descriptor e termina correttamente il programma;

SIGKILL, SIGSTOP : non sono intercettabili, il sistema operativo esegue l'azione di default determinata per quel tipo di segnale, generalmente la terminazione del processo;

RIMANENTI SEGNALI : tutti i rimanenti segnali vengono ignorati, tramite la configurazione dell'azione default *SIG_IGN*;

3.5 Analisi finale

L'applicazione sviluppata rispetta pienamente i requisiti originariamente analizzati:

Efficienza La soluzione è efficiente perchè non richiede grandi risorse computazionali, utilizzando una funzionalità già fornita dal kernel del sistema operativo, inoltre non introduce filtri o rallentamenti nei pacchetti.

Configurabilità L'applicazione offre la possibilità di impostare filtri e selezioni sui dati da registrare, per escludere o includere particolari informazioni.

GNU/Linux La soluzione è progettata per funzionare solo sul sistema operativo GNU/Linux, perchè utilizza il protocollo *Netlink* implementato nel kernel di questo sistema operativo.

Compatibilità contemporanea con IPv4 e IPv6 L'applicazione consente il logging delle coppie di indirizzi sia del protocollo IPv4 che di IPv6, consentendo anche la selezione dei due protocolli singolarmente.

3.5.1 Funzionalità aggiuntive

Si può pensare in futuro all'implementazione di funzionalità aggiuntive, che non è stato possibile sviluppare per mancanza di tempo, come la possibilità di configurazione tramite la riga di comando del timeout di garbage collection, attualmente modificabile solo dal sorgente.

3.5.2 Codice sorgente

I sorgenti dell'applicazione sono rilasciati con licenza GNU GPL versione 2 e sono disponibili per il download all'indirizzo:
http://sourceforge.net/projects/lnls/

Capitolo 4

Sviluppi Futuri

Questo studio di tesi è solamente una parte dell'ampio progetto relativo all'implementazione di un prototipo di laboratorio didattico virtuale, perciò esistono ulteriori possibilità di studio e sviluppo, inerenti alla parte di progetto realizzata, non analizzate per mancanza di tempo.

4.1 Integrazioni con DSA

Il sistema *DSA* dell'*Università di Bologna* consente l'autenticazione su tutti i servizi informatici dell'Ateneo mediante le credenziali centralizzate gestite dal *CeSIA*.

Un'ulteriore possibilità di sperimentazione può essere l'integrazione, completa e trasparente, dell'autenticazione per il servizio di accesso alla rete del prototipo di laboratorio virtuale, con il servizio DSA.

Tramite lo studio di una soluzione che consenta l'interoperabilità con il servizio DSA, implementato tramite applicazioni e protocolli proprietari, effettuando, le minori modifiche possibili all'architettura attuale.

4.2 SSO tra connessione alla rete ed accesso alle VM

Il progetto del laboratorio didattico virtuale prevederà l'utilizzo di macchine virtuali in sostituzione degli elaboratori fisici. Un ulteriore campo di sperimentazione e studio può essere la realizzazione della possibilità di integrazione dell'autenticazione dei sistemi virtuali, con quella necessaria per l'accesso alla rete, implementando un meccanismo di *Single Sign On*, che permetta agli utenti di effettuare una sola verifica delle credenziali di accesso, al momento della connessione alla rete.

Capitolo 5

Conclusioni

Questo lavoro di tesi ha evidenziato la possibilità di implementare un sistema centralizzato di autenticazione sicura, per la connessione di elaboratori portatili ad un'infrastruttura di rete cablata ad alta velocità, necessaria per consentire agli utenti, l'accesso più rapido possibile ai sistemi di un futuro laboratorio didattico virtuale. Condizione necessaria per rendere l'esperienza d'uso del laboratorio più simile possibile a quella di un'entità fisica.

Il primo passo è stato l'analisi e lo studio dei requisiti che la soluzione da elaborare avrebbe dovuto rispettare, ciò ha consentito di escludere a priori soluzioni già utilizzate ma non adatte al contesto di applicazione. Lo step successivo ha previsto l'applicazione della soluzione scelta in un contesto di sperimentazione, che ha permesso di evidenziare e risolvere alcune possibili mancanze, soprattutto di sicurezza, che non avrebbero consentito il rispetto dei requisiti analizzati inizialmente. Infine la soluzione sperimentata ed analizzata è stata applicata, nel contesto finale di produzione, con i risultati previsti e nel rispetto dei requisiti iniziali.

Questo lungo lavoro mi ha permesso di acquisire metodi e competenze per l'analisi e l'implementazione di progetti che richiedono l'integrazione di protocolli e tecnologie vari, rispettando criteri di progettazione funzionali alla compatibilità e riusabilità.

Per motivi di tempo non è stato possibile completare la soluzione elab-

orata, con ulteriori funzionalità utili alla realizzazione del progetto, a cui appartiene, ma si lasciano come possibili sviluppi futuri.

Appendice A

Modulo OpenLdap NT hash

```
/* OpenLDAP module for check and generate NT hash
 * developed as part of VirtualSquare project
 *
 * Copyright 2010 Michele Cucchi <cucchi@cs.unibo.it>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License, version 2, as
 * published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program; if not, write to the Free Software Foundation, Inc.,
 * 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 */
/* This module include headers and use some utility libraries of OpenLdap */
```

```
/* Nt hash OpenLDAP module */

#include <lber.h>
#include <lber_pvt.h>
#include <ac/string.h>
#include "lutil.h"

#include <iconv.h>
#include <string.h>
#include <assert.h>

// my implementation of md4 hashing algo
#include "md4.h"

#define __LOCAL_ERROR -1
#define BUFUTF16_LEN 256

static LUTIL_PASSWD_CHK_FUNC checkNtHash;
static LUTIL_PASSWD_HASH_FUNC createNtHash;
static const struct berval ntscheme = BER_BVC("{NT}");
static const char srcCharSet[] = "UTF-8";
static const char dstCharSet[] = "UTF-16LE";

/* function taken from openLdap libraries at libraries/liblutil/passwd.c */
static int pw_string64(
        const struct berval *sc,
        const struct berval *hash,
        struct berval *b64,
        const struct berval *salt )
{
        int rc;
        struct berval string;
        size_t b64len;
```

```
if( salt ) {
        /* need to base64 combined string */
        string.bv_len = hash->bv_len + salt->bv_len;
        string.bv_val = ber_memalloc( string.bv_len + 1 );

        if( string.bv_val == NULL ) {
                return LUTIL_PASSWD_ERR;
        }

        AC_MEMCPY( string.bv_val, hash->bv_val,
                hash->bv_len );
        AC_MEMCPY( &string.bv_val[hash->bv_len], salt->bv_val,
                salt->bv_len );
        string.bv_val[string.bv_len] = '\0';

} else {
        string = *hash;
}

b64len = LUTIL_BASE64_ENCODE_LEN( string.bv_len ) + 1;
b64->bv_len = b64len + sc->bv_len;
b64->bv_val = ber_memalloc( b64->bv_len + 1 );

if( b64->bv_val == NULL ) {
        if( salt ) ber_memfree( string.bv_val );
        return LUTIL_PASSWD_ERR;
}

AC_MEMCPY(b64->bv_val, sc->bv_val, sc->bv_len);

rc = lutil_b64_ntop(
        (unsigned char *) string.bv_val, string.bv_len,
```

```
                &b64->bv_val[sc->bv_len], b64len );

        if( salt ) ber_memfree( string.bv_val );

        if( rc < 0 ) {
                return LUTIL_PASSWD_ERR;
        }

        /* recompute length */
        b64->bv_len = sc->bv_len + rc;
        assert( strlen(b64->bv_val) == b64->bv_len );
        return LUTIL_PASSWD_OK;
}

/* Count bytes of an utf-16 string, terminate loop when read \0.
Return bytes number less string termination character.
Return maxNByte if total number of bytes is equal to maxNchar */
static unsigned int utf16ByteCount(
        unsigned short *str,
        unsigned int maxNByte )
{
        unsigned int counter = 0;

        while(counter < maxNByte)
        {
                if(*str == 0)
                        return counter;
                // if is part of an utf-16 surrogate pair
                else if((*str >= 0xD800) && (*str <= 0xDFFF))
                {
                        counter += 4;
                        str += 2;
                }
```

```
                else
                {
                        counter += 2;
                        str++;
                }
        }

        return counter;
}

/* Convert a string from srcCharSet to dstCharSet */
static int charSetConversion(
        char *srcStr,
        char *dstStr,
        size_t srcLen,
        size_t dstLen )
{
        iconv_t dc;

        int ret = 0;

        // open iconv descriptor
        dc = iconv_open(dstCharSet, srcCharSet);
        if(dc < 0)
                return __LOCAL_ERROR;

        if(iconv(dc, &srcStr, &srcLen, &dstStr, &dstLen) == (size_t) -1)
                return __LOCAL_ERROR;

        if(iconv_close(dc) < 0)
                return __LOCAL_ERROR;

        return ret;
```

```
}

/* Generate NT hash using MD4 hashing functions */
static int calcNtHash(
        char *plainText,
        size_t plainLen,
        unsigned char *digestText )
{
        int retVal = 0;
        char buf[BUFUTF16_LEN];

        memset(buf, 0, BUFUTF16_LEN);

        retVal = charSetConversion(plainText, buf, plainLen, BUFUTF16_LEN);

        if(retVal != __LOCAL_ERROR)
                MD4((unsigned char *) buf, utf16ByteCount((unsigned short *)
                        buf, BUFUTF16_LEN), digestText);
        else
                return retVal;

        return retVal;
}

/* Check if digest in passwd->bv_val is equal to the hash
generated from plaintext password in cred->bv_val */
static int checkNtHash(
        const struct berval *scheme,
        const struct berval *passwd,
        const struct berval *cred,
        const char **text )
{
        unsigned char ntHash[MD4_DIGEST_LEN];
```

```
        char hashEncoded[LUTIL_BASE64_ENCODE_LEN
                        (MD4_DIGEST_LEN)+1];

        if(calcNtHash(cred->bv_val, cred->bv_len,
                ntHash) != __LOCAL_ERROR)
        {
                lutil_b64_ntop(ntHash, MD4_DIGEST_LEN, hashEncoded,
                LUTIL_BASE64_ENCODE_LEN(MD4_DIGEST_LEN)+1);
                return (strcmp(hashEncoded, passwd->bv_val));
        }
        else
                return  LUTIL_PASSWD_ERR;
}

/* Generate the digest from plaintext password in passwd->bv_val and
use pw_string64 function to base64 encode the digest and concatenate
with its scheme. Finally the scheme+digest is returned in hash */
static int createNtHash(
        const struct berval *scheme,
        const struct berval *passwd,
        struct berval *hash,
        const char **text )
{
        unsigned char ntHash[MD4_DIGEST_LEN];

        if(calcNtHash(passwd->bv_val, passwd->bv_len,
                ntHash) != __LOCAL_ERROR)
        {
                struct berval digest;
                digest.bv_val = (char *) ntHash;
                digest.bv_len = sizeof(ntHash);

                return pw_string64(scheme, &digest, hash, NULL);
```

```
        }
        else
                return  LUTIL_PASSWD_ERR;
}

/* Module entry point function */
int init_module(int argc, char *argv[]) {
        return lutil_passwd_add((struct berval *) &ntscheme,
                checkNtHash, createNtHash);
}
```

```
/* MD4 Hashing Algorithm
 *
 * Copyright 2010 Michele Cucchi <cucchi@cs.unibo.it>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License, version 2, as
 * published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License along
 * with this program; if not, write to the Free Software Foundation, Inc.,
 * 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 */
/*
 * Implementation of the MD4 hash algorithm, as described in RFC 1320.
*/

/* If the message length is more than 2^64 bit is not
```

```
guarantee the correctness of the result */

#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

#include "md4.h"

// buffers start values
#define A 0x67452301
#define B 0xEFCDAB89
#define C 0x98BADCFE
#define D 0x10325476

// round 2 and 3 additive constants with value of square root 2 and 3
#define SQRT2 0x5A827999
#define SQRT3 0x6ED9EBA1

// 32 bit left Circular shift
static uint32_t leftRot(uint32_t value, uint32_t shift)
{
        if(shift >= (sizeof(uint32_t)*8))
                return value;
        else
                return ((value << shift) |
                (value >> ((sizeof(uint32_t)*8) - shift)));
}

// 32 bit right Circular shift
/*static uint32_t rightRot(uint32_t value, uint32_t shift)
{
        if(shift >= (sizeof(uint32_t)*8))
```

```
                return value;
        else
                return ((value >> shift) |
                (value << ((sizeof(uint32_t)*8) - shift)));
}
*/

// Algorithm logic functions
static uint32_t F(uint32_t X, uint32_t Y, uint32_t Z)
{
        return ((X & Y) | ((~X) & Z));
}

static uint32_t G(uint32_t X, uint32_t Y, uint32_t Z)
{
        return ((X & Y) | (X & Z) | (Y & Z));
}

static uint32_t H(uint32_t X, uint32_t Y, uint32_t Z)
{
        return (X ^ Y ^ Z);
}

// Bit padding
static uint32_t *padding(const uint8_t *msg, uint32_t bytesLen, uint32_t *N)
{
        // compute message length modulo 64
        int32_t mod64Len = (bytesLen % 64);
        // compute bit message length and save value in a quadword (8 word)
        uint64_t bitLen = (bytesLen * 8);

        // compute padding bytes number
        int32_t count = (56 - mod64Len);
```

```
        uint32_t newBytesCount = 0;

        if(count <= 0)
                newBytesCount = (64 + count);
        else
                newBytesCount = count;

        // compute buffer length
        uint32_t bufLen = (bytesLen + newBytesCount + 8);

        // return message length in words number
        *N = (bufLen / 4);

        // buffer allocation
        uint8_t *buf = malloc(bufLen);

        // if error on memory allocation abort program
        assert(!(buf == NULL));

        // copy message in the buffer
        memcpy(buf, msg, bytesLen);

        // set padding bytes
        memset((buf+bytesLen), 0x80, 1);
        memset((buf+bytesLen+1), 0, (newBytesCount - 1));

        // copy message length in bit in the last 8 bytes
        memcpy((buf+bytesLen+newBytesCount), &bitLen, 8);

        // return buffer address
        return ((uint32_t *) buf);
}
```

```
/* Message is the address of cleartext message, length is the
cleartext message length in bytes, digest is the address of a
buffer to store the digest of message, the buffer must be length
MD4_DIGEST_LEN bytes */
void MD4(const unsigned char *message, unsigned int length,
        unsigned char *digest)
{
        /* buffer: 4 32 bit word buffer initialized with
        a, b, c, d constants values*/
        uint32_t buffer[] = {A, B, C, D};

        /* N : message length in word, X working
        buffer of 16 word, i : main loop index*/
        uint32_t N = 0, X[16], i;

        // address of buffer contains padded message
        uint32_t *M = padding(message, length, &N);

        // Process message 16 word at a time
        for (i=0; i<(N/16); i++)
        {
                uint8_t j = 0;

                // copy in the working buffer the selected block of message
                for (j=0; j<16; j++)
                        X[j] = M[(i*16)+j];

                // save buffers previous values
                uint32_t prev0 = buffer[0], prev1 = buffer[1],
                prev2 = buffer[2], prev3 = buffer[3];

                /* Round 1 */
                for(j=0; j<4; j++)
```

```
{
        buffer[0] = leftRot((buffer[0] + F(buffer[1],
        buffer[2], buffer[3]) + X[(j*4)]), 3);
        buffer[3] = leftRot((buffer[3] + F(buffer[0],
        buffer[1], buffer[2]) + X[(j*4)+1]), 7);
        buffer[2] = leftRot((buffer[2] + F(buffer[3],
        buffer[0], buffer[1]) + X[(j*4)+2]), 11);
        buffer[1] = leftRot((buffer[1] + F(buffer[2],
        buffer[3], buffer[0]) + X[(j*4)+3]), 19);
}

/* Round 2. */
for(j=0; j<4; j++)
{
        buffer[0] = leftRot((buffer[0] + G(buffer[1],
        buffer[2], buffer[3]) + X[j] + SQRT2), 3);
        buffer[3] = leftRot((buffer[3] + G(buffer[0],
        buffer[1], buffer[2]) + X[j + 4] + SQRT2), 5);
        buffer[2] = leftRot((buffer[2] + G(buffer[3],
        buffer[0], buffer[1]) + X[j + (4*2)] + SQRT2), 9);
        buffer[1] = leftRot((buffer[1] + G(buffer[2],
        buffer[3], buffer[0]) + X[j + (4*3)] + SQRT2), 13);
}

/* Round 3. */
for(j=0; j<4; j++)
{
        uint8_t k = 0;

        if(j==1)
                k=2;
        else if(j==2)
                k=1;
```

```
                    else
                        k=j;

                    buffer[0] = leftRot((buffer[0] + H(buffer[1],
                    buffer[2], buffer[3]) + X[k] + SQRT3), 3);
                    buffer[3] = leftRot((buffer[3] + H(buffer[0],
                    buffer[1], buffer[2]) + X[k + (4*2)] + SQRT3), 9);
                    buffer[2] = leftRot((buffer[2] + H(buffer[3],
                    buffer[0], buffer[1]) + X[k + 4] + SQRT3), 11);
                    buffer[1] = leftRot((buffer[1] + H(buffer[2],
                    buffer[3], buffer[0]) + X[k + (4*3)] + SQRT3), 15);
                }

                /* Add to 4 buffer previous saved values */
                buffer[0] += prev0;
                buffer[1] += prev1;
                buffer[2] += prev2;
                buffer[3] += prev3;
        }

        /* end */

        // copy digest in the return buffer
        memcpy(digest, buffer, MD4_DIGEST_LEN);
}
```

Bibliografia

[1] Larry L. Peterson, Bruce S. Davie Reti di Calcolatori, Apogeo, 2004.

[2] Wikipedia: http://en.wikipedia.org/wiki/Main_Page.

[3] Ietf Network Working Group, RFC3748 - Extensible Authentication Protocol, 2004, http://tools.ietf.org/html/rfc3748.

[4] Ietf Network Working Group, RFC2865 - Remote Authentication Dial In User Service, 2000, http://datatracker.ietf.org/doc/rfc2865.

[5] Matt Bishop, Introduction to Computer Security, Addison-Wesley, 2004.

[6] Ietf Network Working Group, RFC3549 - Linux Netlink as an IP Services Protocol, 2003, http://datatracker.ietf.org/doc/rfc3549.

[7] The Linux Foundation, Generic Netlink Howto, 2009, http://www.linuxfoundation.org/collaborate/workgroups/networking/generic_netlink_howto.

[8] Christian Benvenuti, Understanding Linux Network Internals, O'Reilly, 2005.

[9] OpenLdap Community, OpenLdap Development Documentation, 2010, http://www.openldap.org/devel/contributing.html.

[10] J. Sermersheim, Novell, Inc, RFC4511 - Lightweight Directory Access Protocol (LDAP): The Protocol, 2006, http://datatracker.ietf.org/doc/rfc4511.

[11] Charles Hornig, RFC894 - A Standard for the Transmission of IP Datagrams over Ethernet Networks, 1984, http://tools.ietf.org/html/rfc894.

[12] Ietf Network Working Group, RFC4861 - Neighbor Discovery for IP version 6 (IPv6), 2007, http://datatracker.ietf.org/doc/rfc4861.

[13] HP, ProCurve Switches - Access Security Guide, 2009, http://www8.hp.com/it/it/support-drivers.html.

[14] A. Morgan, T. Kukuk, The Linux PAM System Administrator Guide, 2009, http://debian.securedservers.com/kernel/pub/linux/libs/pam/Linux-PAM-html/Linux-PAM_SAG.html