Università degli studi di Pisa

FACOLTÀ DI INGEGNERIA

Corso di Laurea Specialistica in Ingegneria Informatica

TESI DI LAUREA SPECIALISTICA

Un sistema integrato di analisi e filtraggio della posta elettronica anti-spam ed anti-bot

Candidato:
Fabio Cappellini

Relatori:
Prof. Gianluca Dini
Prof.sa Cinzia Bernardeschi

Anno Accademico 2007/2008

Sommario

Le *email (Elettronic MAIL)*, rappresentano uno degli strumenti di comunicazione più adottati al mondo. Proprio per la sua larga diffusione, l'email è soggetta ad attacchi quali: *SPAM*, *phishing* ed infezioni *bot*. Lo *SPAM* è sostanzialmente *posta spazzatura*, ovvero corrispondenza non desiderata, mentre il *phishing* è una tecnica più spregiudicata che sfrutta messaggi con *link* a copie di siti originali per carpire informazioni a utenti ignari. Ultimamente si stanno diffondendo le *botnet*, reti di macchine soggiogate attraverso l'uso di virus o altri software malevoli, al potere di un unico *botmaster* che le utilizza per inviare email con scopi illeciti. Analizzando tali attacchi e accertata la loro accresciuta virulenza, arrecante danni spesso di carattere economico, si è progettato e realizzato un sistema in grado di arginare il problema, in maniera poco invasiva rispetto all'*end user* che lo utilizza.

Sia la comunità scientifica sia le industrie di software hanno sviluppato metodi per contrastare gli attacchi sopra menzionati. In particolare, la comunità scientifica si è principalmente concentrata sulla protezione dalle infezioni botnet a livello di *core network*, mentre le industrie di software si sono soprattutto concentrate sul problema SPAM. I metodi ad oggi maggiormente utilizzati per il contrasto alla posta elettronica non desiderata, si basano principalmente sul filtraggio degli indirizzi ovvero il *listing*. In alternativa il filtraggio può essere compiuto sul contenuto sfruttando alcune leggi probabilistiche. Altri sistemi si basano sulla modifica dei protocolli SMTP e POP3 ma non sembrano, a tutt'oggi, una strada facilmente percorribile.

Dalla ricerca compiuta è nata l'idea di realizzare un servizio di protezione che andasse a riempire una mancanza tra le soluzioni già esistenti. In particolare si

è visto che il problema curato è quasi sempre quello dello SPAM, integrato in rari casi da sistemi anti-virus, trascurando la questione legata agli attacchi *bot* che possono insorgere. Tale mancanza può portare il sistema eventualmente colpito sia a diventare uno strumento con cui il *botmaster* può compiere altri *cyber crime*, sia a provocare contagi a catena in grado di interessare i sistemi ad esso noti. Stime attuali indicano che più di un quarto di tutti i personal computer connessi ad internet sono potenzialmente candidati a divenire parte di *botnet*.

Partendo da questi riscontri empirici si è pensato di realizzare un sistema integrato anti-spam ed anti-bot, garantendo così una difesa più ampia. Un altro obiettivo si è delineato scegliendo di proteggere un'intera rete locale, ponendo il sistema sulla macchina che fornisce ad essa la connettività internet. L'applicativo sviluppato è stato chiamato *Bloumail*, acronimo di *BLOck Unwanted MAIL*. Tale programma si pone come filtro per la posta elettronica, entrante ed uscente, e può integrarsi con il programma *Blobot*. Quest'ultimo con metodi in parte analoghi a quelli proposti nel seguito, si occupa di controllare il traffico web, garantendo la sicurezza della navigazione internet in caso di infezioni bot.

Bloumail si basa su due tecniche classiche di filtraggio, il *blacklisting* sugli indirizzi delle email ed il filtraggio probabilistico di *Bayes* sul loro contenuto. Facendo operare opportunamente tali meccanismi anche per la posta in uscita, ed aggiungendo un particolare strumento di autorizzazione fondato sull'uso di *CAPTCHA*, si è ottenuto il pregio di poter bloccare SPAM eventualmente prodotto da infezioni bot della macchina. In particolare l'uso di CAPTCHA permette di impedire l'accesso al sistema ai bot, garantendo che l'operatore sia un umano. In conclusione si sono eseguiti i test e le analisi atte a verificare sperimentalmente il corretto ed efficace funzionamento di quanto sviluppato.

La tesi si articola su sei capitoli. Nel primo si espongono i principi alla base del funzionamento e dell'architettura del servizio di email, presentando con attenzione le problematiche riscontrabili durante il suo utilizzo. Nel secondo capitolo si riporta una sintesi delle ricerche compiute sullo stato dell'arte nel

campo della difesa dagli abusi perpetrati attraverso la posta elettronica. Una sintetica descrizione dei protocolli SMTP e POP3 si trova nel terzo capitolo. Vincoli di progetto ed applicativi utilizzati con le rispettive motivazioni sono argomento del quarto capitolo. La completa trattazione del metodo ideato e la sua implementazione, comprendente le singole funzioni sviluppate e l'architettura che le governa sono riprese dal quinto capitolo. Il sesto ed ultimo capitolo descrive la sperimentazione, arricchita dalle conclusioni risultanti da questa esperienza.

Ringraziamenti

Un sincero e sentito ringraziamento porgo al primo relatore, professor Gianluca Dini, che mi ha dato la possibilità di scegliere questa tesi, dedicandomi parte del proprio tempo e fornendomi preziose indicazioni per la sua realizzazione. Vorrei inoltre ricordare, il secondo relatore professoressa Cinzia Bernardeschi, sempre disponibile anch'essa. In particolare però desidero ringraziare l'Ingegner Silvio La Porta che mi ha seguito ed aiutato ogni qual volta ho avuto dubbi o necessità di un consiglio, dimostrandosi un prezioso collaboratore durante tutto lo sviluppo della tesi. Un omaggio anche a chi ha affrontato prima di me problematiche simili alla mia, realizzando lo strumento *Blobot* che integra il sistema *Bloumail* che ho creato, e cioè Rossella Candela. Un grazie infine va a tutti coloro che direttamente od indirettamente mi hanno sostenuto durante questi anni di studi e soprattutto nel compimento di questa tesi.

Fabio Cappellini

Indice

Capitolo 1

La posta elettronica

La posta elettronica, anche detta *email* (*Electronic Mail*), è un servizio internet grazie al quale ogni utente può inviare o ricevere messaggi. Ne definisce il funzionamento Jon Postel nel 1972 e subito dopo Ray Tomlinson la rende utilizzabile su ARPANET, il progenitore di internet. All'inizio la sua utilità era quella di permettere lo scambio di messaggi prevalentemente tra le università, visto soprattutto il ristretto bacino di utenza a cui era permesso accedere a questa rete. Nel giro di pochi decenni il networking ha compiuto passi da gigante rendendo internet il più importante media cui oggi si dispone per comunicare con l'intero mondo. In egual misura, anche il servizio email è cresciuto, divenendo uno dei vettori di informazioni più utilizzati. Questa nuova realtà pone il sistema della posta elettronica di fronte a problematiche che al suo esordio neppure i progettisti riuscirono appieno ad immaginare.

Oggi ciascun *navigatore* possiede una o più *caselle email*, su cui vengono conservati i messaggi con la possibilità di essere consultati, scaricati sulla propria macchina o eliminati. La stessa casella può essere utilizzata sia per rispondere che per inviare messaggi a uno o più utenti. Il vero e proprio boom di questo servizio è dovuto a diversi fattori facilmente intuibili quali: l'enorme espansione di internet, la semplicità del servizio che per alcuni versi ricorda il suo fratello cartaceo, la sua quasi istantaneità e la gratuità dello stesso che consente corrispondenze con tutto il globo, un tempo assai più onerose e dispendiose in termini di tempo.

La modalità di accesso al servizio è asincrona, ovvero per la trasmissione di un messaggio non è necessario che il mittente ed il destinatario siano contemporaneamente attivi o collegati come invece ad esempio accade per le comunicazioni telefoniche. Le lettere inviate e ricevute possono stazionare in apposite *mailbox* amministrate dagli utenti stessi. La posta elettronica si basa su diversi protocolli, i più usati attualmente sono: l' SMTP per l'invio dei messaggi ed il POP3 per la ricezione degli stessi. La consegna al destinatario dei messaggi inviati normalmente *non* è garantita. Nel caso un server SMTP non riesca a consegnare un messaggio, tenta di inviare una qualche notifica al mittente per avvisarlo della mancata consegna, ma anche quest'ultima è a sua volta un messaggio (generato automaticamente dal server), e quindi neppure la sua consegna sarà assicurata.

In conclusione si può affermare che il protocollo SMTP, un po' come la maggior parte dei protocolli internet, offre un servizio di tipo *best-effort*, cioè non fornisce garanzie sul corretto invio dei messaggi e neppure sulla loro ricezione. Questo assicura una certa semplicità architetturale e quindi una maggior flessibilità, d'altro canto però limita la possibilità di impedire abusi, mancando soprattutto strumenti per garantire le autorizzazioni degli utenti ad usare il servizio e la stessa autenticazione dei mittenti.

1.1. La posta elettronica certificata

La posta elettronica certificata (PEC) è un servizio che permette di ottenere la garanzia di ricezione del messaggio da parte del destinatario e dell'integrità dello stesso sopperendo alle carenze del protocollo SMTP. In Italia oggi l'invio di un'email certificata (nelle forme stabilite dalla normativa vigente) è equiparato a tutti gli effetti di legge alla spedizione di una raccomandata cartacea con avviso di ricevimento.

Il meccanismo consiste nel fatto che il gestore di posta elettronica certificata, nel momento in cui prende in carico l'email del mittente, inoltra ad esso una ricevuta di accettazione, che attesta l'avvenuto invio. Nel momento invece in cui il gestore deposita il messaggio nella casella del destinatario, lo stesso manda al mittente una ricevuta di consegna che ne dimostra l'avvenuta ricezione. Sia la ricevuta di accettazione che la ricevuta di consegna sono in formato elettronico, e ad esse è apposta la firma digitale del gestore. Riassumendo nel circuito PEC vengono rilasciate tre ricevute ai fini della certificazione del messaggio di posta elettronica:

- *Di accettazione*, che attesta l'avvenuto invio dell'email dal gestore di posta elettronica certificata del mittente.

- *Di presa in carico*, che attesta il passaggio di responsabilità tra due distinti gestori di posta certificata, mittente e destinatario. Questa ricevuta viene scambiata tra i due gestori e non viene percepita dagli utilizzatori del servizio.

- *Di avvenuta consegna*, attestante che il messaggio è giunto a buon fine e che il destinatario ne ha piena disponibilità nella sua casella (anche se non ha ancora visto il messaggio).

In caso di un qualche errore e quindi di una situazione negativa esistono altri tre tipi di avvisi rilasciati dal sistema PEC:

- *Di non accettazione*, per virus o utilizzo di un mittente falso o utilizzo di destinatari in copia nascosta, vietati dalla PEC, o altri problemi.

- *Di mancata consegna*, che sarà inviata al mittente entro 24 ore.

- *Di rilevazione di virus informatici*.

I messaggi in ingresso al sistema PEC possono essere "imbustati" dal gestore in due differenti tipologie di *buste*:

- *Di trasporto*, se il messaggio proviene da una casella di PEC e supera tutti i controlli di esistenza, provenienza e validità della firma.

- *Di anomalia*, se il messaggio proviene da una casella email non-PEC oppure è malformato.

1.2. Struttura di un'email

Ogni messaggio email è costituito da una ben definita struttura indicata dagli standard riportati in particolare nell'RFC 822 e successivamente nel RFC 2822, che ha reso obsoleto il precedente. Si riporta di seguito un riassunto delle componenti essenziali di cui è composta un'email:

- Una *busta* (envelope): Contiene principalmente gli indirizzi email del mittente e dei destinatari, che vengono scambiati tra server attraverso il protocollo SMTP. Queste informazioni normalmente corrispondono a quelle che è possibile ritrovare nelle intestazioni, ma possono esserci delle differenze.

- Una sezione di *intestazioni* (header): Sono informazioni di servizio che servono a controllare l'invio del messaggio, o a tener traccia delle manipolazioni che subisce. Ciascuna intestazione è costituita da una riga di testo, con un nome seguito dal carattere ':' e dal corrispondente valore, ad esempio *Subject:* (*Oggetto:*), *From:* (*Da:*), *To:* (*A:*) ecc.

- Un *corpo del messaggio* (body): Originariamente composto di solo testo semplice, in seguito grazie all'introduzione dello standard MIME,

è stata introdotta la possibilità di inserire dei file *allegati*, ad esempio per inviare immagini o documenti.

L'RFC 2822 stabilisce altri importanti dettagli su come realizzare i campi esposti in precedenza, in particolare viene consigliato di non superare i 78 caratteri per ogni riga di intestazione, rinviando a capo nel caso fossero necessarie linee più lunghe. In questo frangente la nuova riga deve iniziare con un carattere di spaziatura come ad esempio una tabulazione. Inoltre viene stabilito che per separare l'*header* dal *body* è sufficiente un ritorno a capo (CRLF). Altre direttive dello stesso documento riguardano i singoli campi dell'intestazione con specifiche dettagliate sui vari significati e sulle loro formattazioni.

Per identificare mittente e destinatario si usano indirizzi email, questi hanno la forma *nomeutente@dominio*, dove *nomeutente* è un nome scelto dall'utilizzatore o dall'amministratore del server, che identifica in maniera univoca un utente (o un gruppo di essi). Il *dominio* invece è un nome DNS che viene usato per identificare univocamente il fornitore di servizi. L'indirizzo email può contenere qualsiasi carattere alfabetico e numerico (escluse le accentate) e alcuni simboli come il trattino basso (_) ed il punto (.), e può essere al massimo complessivamente composto da 320 caratteri.

1.3. Lo standard MIME

Per capire a fondo la struttura delle email occorre porre particolare attenzione alle specifiche MIME descritte in diversi RFC, i principali sono: 2045, 2046, 2047, 2077, 4288, 4289. Lo standard MIME (*Multipurpose Internet Mail Extensions*) è stato realizzato per superare alcune carenze che il servizio email presentava, oggi la stragrande maggioranza dei messaggi di posta elettronica utilizza questo standard.

In particolare, MIME permette il supporto a codifiche differenti alla ASCII 7 bit usata dal protocollo SMTP, cioè permette di usare il servizio con lingue diverse dall'inglese, ad esempio permettendo l'alfabeto latino composto da lettere non supportate dalla codifica precedente a 7 bit. Un altro vantaggio dell'approccio MIME è quello di poter inserire allegati di varia natura nelle email come immagini, audio, documenti ecc. Di seguito si è cercato brevemente di far comprendere come sfruttare questo standard:

1. Occorre definire nell'intestazione dell'email alcune righe per far conoscere all'interprete che sta leggendo un messaggio realizzato seguendo il metodo MIME.

    - *MIME-version*: Tipicamente 1.0, indica appunto che versione dello standard si utilizzerà.

    - *Content-type*: Questo campo servirà per specificare una coppia tipo/sottotipo di contenuto che potrà essere inserito nel messaggio ad esempio *text/plain* nel caso il campo sia unico e di tipo testo, *multipart/mixe* o *multipart/alternative* nel caso invece in cui il corpo della lettera possa contenere più campi.

    - *Boundary*: Indica un'etichetta che servirà per delimitare i vari campi MIME dell'email.

2. Nel corpo dell'email ogni campo MIME inizierà con una particolare stringa del tipo: "*--boundary*". Allo stesso modo terminerà con un etichetta come: "*--boundary--*". All'interno ogni campo conterrà una propria intestazione composta da specifiche del tipo:

    - *Content-type*: Ha lo stesso significato di quello definito in precedenza per l'intestazione.

- *Content-transfer-encoding*: Identifica il tipo di codifica che si è utilizzato in quel campo, ad esempio: *7bit, 8bit, binary, quoted-printable, base64.*

Si può vedere ora un esempio di un'email scritta utilizzando lo standard MIME, in particolare si è costruito un messaggio con campi multipli contenente testo e un file "*.txt*" come allegato. L'email qui proposta è stata generata automaticamente dal client di posta *Microsoft Outlook 2003*.

```
From: <mittente@server.com>
To: <ricevente@server.com>
Subject: titolo email
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="----=_NextPart_000_0004_01C8AF93.D3732090"

Messaggio multipart in formato MIME.

------=_NextPart_000_0004_01C8AF93.D3732090
Content-Type: multipart/alternative;
        boundary="----=_NextPart_001_0005_01C8AF93.D3732090"


------=_NextPart_001_0005_01C8AF93.D3732090
Content-Type: text/plain;
        charset="us-ascii"
Content-Transfer-Encoding: 7bit

corpo email

------=_NextPart_001_0005_01C8AF93.D3732090--

------=_NextPart_000_0004_01C8AF93.D3732090
Content-Type: text/plain;
        name="allegato.txt"
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment;
        filename="allegato.txt"

sono il file di testo allegato
------=_NextPart_000_0004_01C8AF93.D3732090—
```

Concludendo, come si è appena visto lo standard MIME rende molto più potente il servizio email permettendo di passare da semplici messaggi testuali a

veri e propri contenitori per files multimediali quali: immagini, suoni, video, documenti HTML o anche veri e propri programmi eseguibili. Questa maggior flessibilità ha portato il servizio di posta elettronica ad espandersi ulteriormente tra la popolazione mondiale, rendendo d'altro canto possibile veicolare nuovi e più potenti strumenti di attacco ai sistemi informatici, quali *virus* e *phishing* che si prenderanno in esame dettagliatamente più avanti.

1.4. Architettura del servizio email

Il servizio email per funzionare si basa su tre componenti essenziali, vengono di seguito riportati i termini identificativi con le loro rispettive caratteristiche:

- MUA (*Mail User Agent*): Sono i cosiddetti *client* di posta, vengono utilizzati per accedere alle casella di posta elettronica e per inviare i messaggi.

- MTA (*Mail Transfer Agent*): Sono dei *server* con la funzione di ricevere i messaggi email in arrivo ed in partenza, smistandoli ove necessario.

- MS (*Message Store*): Simili ai *server* MTA, inoltre questi immagazzinano i messaggi per gli utenti consentendone il *download* in un secondo momento.

La Figura 1.1 illustra il funzionamento del sistema di posta elettronica così come viene utilizzato oggi.

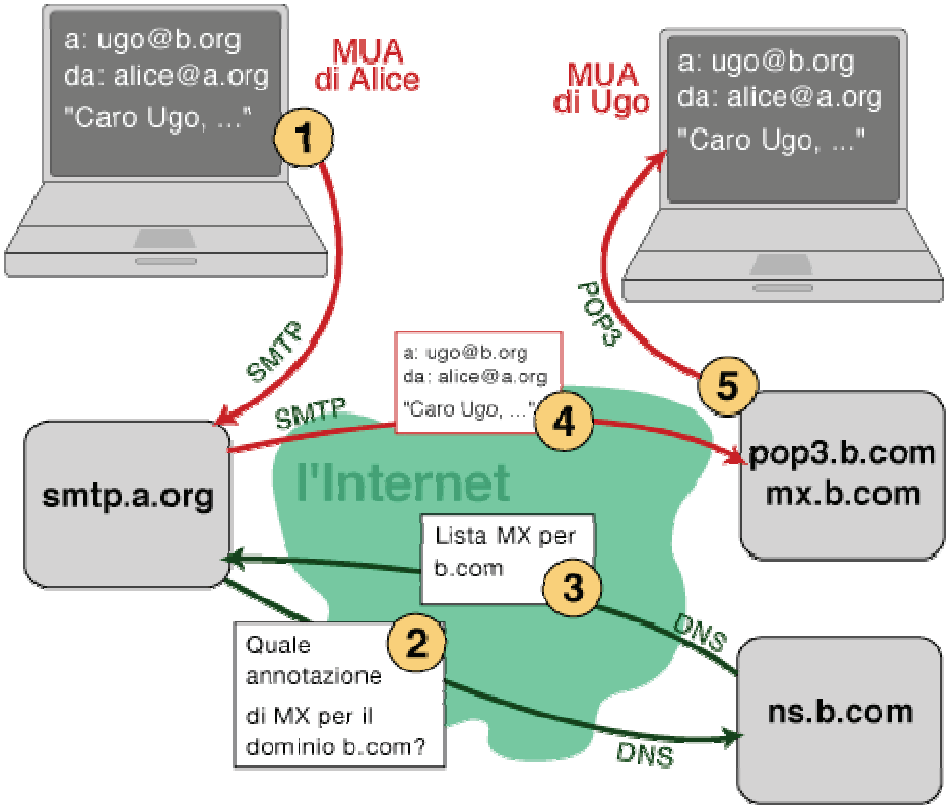


Figura 1.1 - Schema di funzionamento del servizio email (fonte wikipedia)

1. Alice compone un messaggio nel suo *editor* di posta, di solito contenuto in ogni client MUA, quest'ultimo provvede automaticamente a formattare il messaggio secondo lo standard descritto dai paragrafi precedenti. Quando l'email è stata interamente composta ed è pronta ad essere inviata, viene trasferita utilizzando il protocollo SMTP al MTA locale, in questo caso *smtp.a.org* che di solito si trova presso l' ISP (*Internet Service Provider*) di Alice.

2. L'MTA legge l'indirizzo destinatario specificato durante la connessione SMTP (non quello dell'intestazione del messaggio), nell'esempio *ugo@b.org*. L'MTA estrae il dominio dall'indirizzo e lo invia ad un server DNS (*Domain Name System*), in questo caso *ns.b.com*, per trovare il server MX (*Mail Exchange*) a cui inoltrare il messaggio.

3. Il server DNS risponderà all'MTA con la lista dei server a cui si può

inoltrare l'email per poter essere recapitata al dominio richiesto, in questo caso *mx.b.com*.

4. *smtp.a.org* può inviare il messaggio a *mx.b.com* sempre usando SMTP consegnandolo quindi alla *mailbox* di Ugo.

5. Bob dal suo client MUA può richiedere di ottenere l'email sulla propria macchina scaricandola dalla sua mailbox mediante una connessione basata sul protocollo POP3.

La procedura appena descritta rappresenta solo uno dei possibili scenari con cui oggi è possibile scambiarsi posta elettronica, di seguito si elencano alcune alternative:

- Gli interlocutori potrebbero far parte di una rete privata in cui il sistema email è gestito usando strumenti quali ad esempio *Microsoft Exchange* o *IBM Lotus Notes*. Queste reti, spesso aziendali, usano protocolli proprietari per la formattazione dei messaggi e per la loro consegna, solo nel caso in cui l'email voglia essere fatta uscire da questo dominio verrà convertita al formato descritto in precedenza e subirà ancora tutti i passaggi appena elencati.

- Se il mittente non ha un MUA sulla propria macchina, potrebbe utilizzare un *web mail service* che simula il client di posta in una pagina web. In questo caso il protocollo di scambio con l'MTA non sarà più SMTP ma quello utilizzato dalla connessione web.

- Sulla macchina del mittente potrebbe già essere presente un MTA e quindi la prima fase del trasferimento risulterebbe superflua.

- Il ricevente potrebbe usare altri protocolli al posto di POP3 per ricevere

la posta ad esempio l'IMAP oppure avvalersi anch'esso di un *web mail service*.

1.5. Lo SPAM

Con il termine *spam* si indica genericamente tutto ciò che in inglese è detto *junk-mail*, ovvero, posta spazzatura. Il termine rappresenta l'insieme di tutti quei messaggi non richiesti e privi di interesse per l'utente, contenenti spesso pubblicità se non addirittura programmi malevoli. Chi inoltra grosse quantità di spam ad un vasto numero di utenti è detto *spammer*. Questi utenti spesso malintenzionati si differenziano dagli *hackers* soprattutto per essere persone ben organizzate e con l'idea di ottenere ingenti ritorni economici dalle loro azioni.

Il termine trae origine da uno sketch comico del *Monty Python's Flying Circus* ambientato in un locale nel quale ogni piatto era a base di spam (un tipo di carne in scatola particolarmente reclamizzato). Si ritiene che il primo messaggio di spam via email della storia sia stato inviato il 1° maggio 1978 dalla DEC per pubblicizzare un suo nuovo prodotto. La lettera digitale fu spedita a tutti i destinatari ARPANET della costa ovest degli Stati Uniti. Il principale scopo dello spamming come appena presentato è costituito dalla pubblicità, il cui oggetto può andare dalle più comuni offerte commerciali a proposte di vendita di materiale pornografico o illegale, ma anche riguardare software pirata, farmaci senza prescrizione medica o discutibili progetti finanziari che possono rivelarsi veri e propri tentativi di truffa.

Per definizione lo spam viene inviato senza il permesso del destinatario ed è un comportamento ampiamente considerato inaccettabile dagli Internet Service Provider (ISP) e dalla maggior parte degli utenti di internet. Mentre questi ultimi trovano lo spam fastidioso e con contenuti spesso offensivi, gli ISP vi si oppongono soprattutto in considerazione dei costi del traffico generato dall'invio indiscriminato delle lettere che questa pratica produce.

Un gran numero di spammer utilizza informazioni personali false (nomi, indirizzi, numeri di telefono, ecc.) per creare account presso vari ISP. Spesso le informazioni anagrafiche sono il risultato di veri e propri furti di identità compiuti attraverso la rete. La possibilità di accedere indiscriminatamente ad un elevato numero di account intestati ad ignari utenti, permette a questi "pirati" di cambiare velocemente identità qualora la precedente sia scoperta e disattivata dall'ISP.

Gli spammer più evoluti spesso utilizzano software appositamente creato da loro stessi e dedicato all'esclusivo monitoraggio delle connessioni internet più vulnerabili e che presentano una scarsa protezione. Rilevati tali collegamenti non è difficile per il malintenzionato inserirsi e dirottare il traffico in modo da immettere i propri messaggi di spam direttamente nella connessione appena trovata. Questa pratica rende più difficile identificare la posizione di chi compie l'attacco e spesso l'unico a subire le conseguenze più pesanti dell'azione è l'ISP della vittima che può essere soggetto ad aspre reazioni e rappresaglie da parte degli attivisti che si mobilitano per tentare di fermare lo spammer. Tutte le forme di spamming sopracitate sono illegali, tuttavia raramente possono essere perseguite a causa proprio dell'impiego delle tattiche elusive appena esposte.I mittenti delle email pubblicitarie affermano spesso che il loro lavoro non è da considerarsi "spamming".

Quale tipo di attività costituisca spamming è motivo di dibattiti, e le argomentazioni divergono in base allo scopo per il quale è definito, oltre che dalle diverse legislazioni. Lo spamming è considerato un reato in vari paesi ed in Italia l'invio di messaggi non sollecitati è soggetto a sanzioni. Una tecnica più subdola di questa pratica, in quanto sfrutta l'ingenuità di molta gente, è quella detta *per interposta persona*. Per l'esattezza, si intende di solito l'invio di email commerciali ad alcuni destinatari conosciuti e spesso regolarmente iscritti ad una newsletter dello spammer, con l'invito a divulgare una certa promozione tra le persone da loro conosciute, invogliandoli magari con qualche piccolo compenso. Grazie a questo sistema sarà l'ingenuo destinatario a "spammare" le caselle di posta dei suoi conoscenti, in questo modo farà da scudo al vero colpevole che

invece guadagnerà da questo comportamento. Inoltre si abbasserà di molto la mole di lavoro del vero spammer, raggiungendo allo stesso tempo sempre più persone.

I termini *unsolicited commercial email*, UCE (email commerciale non richiesta) e *unsolicited bulk email*, UBE (email non richiesta in grandi quantità) sono usati per definire più precisamente ed in modo meno gergale i messaggi di spam. Molti utenti considerano tutti i messaggi UBE come spam, senza distinguere il loro contenuto, ma i maggiori sforzi legali contro lo spam sono concentrati sui messaggi UCE. Una piccola ma evidente porzione di messaggi non richiesti è anche di carattere non commerciale; alcuni esempi comprendono i messaggi di propaganda politica e le catene di Sant'Antonio, lettere che contengono informazioni allarmanti, promesse di facili guadagni o vere e proprie truffe, ed invitano ad inoltrare il messaggio ai propri conoscenti, finendo talvolta per circolare in rete mesi o anni.

1.6. Mezzi utilizzati dagli spammer

Per compiere azioni di spamming è necessario come prima cosa conoscere degli indirizzi a cui poter inoltrare la propria posta spazzatura. Gli indirizzi email vengono nella maggior parte dei casi raccolti in maniera automatica dalla rete mediante *spambots* cioè appositi programmi che sondano la rete alla ricerca di indirizzi esistenti. Gli spammer sono in grado anche di effettuare attacchi a sistemi privati per trafugare le *mailing list*, cioè elenchi di recapiti, in essi contenuti. Altro modo consiste nell'indovinare gli indirizzi usando nomi comuni o facendo veri e propri attacchi a dizionario utilizzando ad esempio le parole più ricorrenti di una certa lingua.

Uno studio della *Federal Trade Commission* americana [1] riporta le probabilità che si ha di ricevere spam a seconda di dove si fornisce il proprio

indirizzo. Risulta che si ha il 99% di possibilità nel caso delle *chat rooms*, l'86% per i *newsgroups* e pagine web, il 50% per le pagine web personali, il 27% per le bacheche virtuali e solo il 9% vengono ricavati dalle directories dei servizi email. Per arginare il problema molti consigliano di non rilasciare il proprio indirizzo su internet o nel caso fosse proprio necessario utilizzare tecniche di *address munging*, cioè scrivere l'indirizzo in una maniera difficilmente individuabile da un computer (es. "*myemail at domain dot com*").

Una volta venuto in possesso degli indirizzi a cui inoltrare le proprie lettere lo spammer ha bisogno di mezzi fisici per completare il suo attacco. In particolare, necessita di account di posta, intestati ad altri per non poter essere perseguibile ed in grandi quantità per poter inondare meglio la rete. Su internet le macchine di maggior interesse per uno spammer sono senza dubbio i *proxy* e gli *email relay*. Questi sistemi si interpongono tra un client ed un server, inoltrando le richieste e le risposte dall'uno all'altro. Il protocollo SOCKS realizza una forma di *proxy* a livello di trasporto che inoltra semplicemente le connessioni TCP e UDP tra client e server, senza analizzare i protocolli applicativi.

Gli *email relay* prima ricevono l'intero messaggio e poi provvedono a trasferirlo al successivo mail server. I *proxy* invece non prevedono una bufferizzazione di questo tipo ed hanno bisogno di sincronizzazione tra le due connessioni. Gli email relay modificano il campo *received from* dell'intestazione che contiene l'indirizzo del mittente ed il *timestamp* di quando è stato ricevuto. In questo modo si può sempre risalire al vero mittente del messaggio. I proxy invece si limitano a fare una semplice modifica dell'indirizzo IP del mittente, risultando in questo modo loro stessi i confezionatori del messaggio. Questa tecnica permette a chi compie gli attacchi di mantenere l'anonimato.

Gli spammer spesso usano hardware e risorse di rete "rubate" cioè utilizzate all'insaputa del vero possessore. I mezzi adoperati spesso sono: le *botnet*, cioè reti di sistemi infettati e sfruttati come email client, e gli *open relay server*. Queste ultime macchine permettono a chiunque di usarle, senza alcun permesso

precedentemente rilasciato, sia come sender che come receiver. Gli spammer spesso riescono inoltre ad abusare di questi sistemi accedendo alle liste degli utenti in esse contenuti. Gli *open relay server* in particolare sono cresciuti fino alla fine del 2001 raggiungendo la quantità di circa 225.000 [2]. Oggi a causa della vulnerabilità di tali sistemi agli attacchi si sta cercando di rendere i servizi disponibili solo dopo un'autenticazione dell'utente.

1.7. Evoluzione dello spam

Lo spam negli ultimi anni ha assistito ad un'evoluzione che un tempo non sarebbe mai potuta essere neppure ipotizzata, crescendo a ritmi incessanti e candidandosi ad essere uno dei problemi più grandi per la rete internet attuale. Allo scopo di tenere sotto controllo questo fenomeno sono nate varie ricerche ed enti come lo *Spamhaus Project*. Questa è una società di volontari fondata da Steve Linford nel 1998 che si occupa di studiare il mondo degli spammer e delle loro attività, è responsabile di tre anti-spam DNS Blocklist (detti DNSBLs, usati per pubblicare liste di indirizzi web che hanno a che fare con lo spam) largamente usati nel mondo.

Dal 1995 quando è stato aperto internet al commercio elettronico si è assistito ad un drammatico e stabile aumento del traffico legato allo spam [3]. Già nel 1997 *America Online* considerava come spam il 5-30% della sua totalità di email trattate, si parla di circa 0.5-3 milioni di e-mail al giorno di posta spazzatura [4]. La stima del 2006 fornita dallo Spamhaus parla di un livello di spam del 75% sull'intero traffico email [5]. Previsioni sul futuro dicono che per il 2015 si sfonderà il tetto del 95% [6] e se non vi si porrà freno in qualche modo questo indurrà gli utenti ad allontanarsi dall'utilizzo del servizio email per le loro comunicazioni, spostandosi su altri mezzi.

La differenza tra lo spamming e la posta spazzatura cartacea sta nel diverso

modello legato ai costi. La stampa e le spese postali della corrispondenza sono pagati dal mittente, mentre nel caso dello spam, il server del destinatario paga i costi maggiori, in termini di banda, tempo di elaborazione e spazio per immagazzinamento. Bisogna considerare che le email sono gratuite sia per il mittente che per il ricevente e che esistono spesso possibilità di accedere a internet mediante abbonamenti gratis. Tutti questi fattori portano ad avere costi veramente minimi per uno spammer. A causa di questa ricaduta di costi sui server destinatari, molti considerano lo spamming un furto o un equivalente di crimine che provoca danni economici.

Gli spammer spesso cercano e sfruttano sistemi vulnerabili come gli *open mail relay* e *server proxy* aperti. Essi abusano anche di risorse messe a disposizione per la libera espressione su internet, come i *remailer* anonimi. Come risultato, molte di queste risorse sono state oggi disattivate, negando la loro utilità agli utenti legittimi, mentre le risorse essenziali alla vita degli ISP devono essere rinforzate per sostenere il traffico con elevati investimenti che possono ricadere anche sui canoni di utilizzo della connessione. Altro costo è quello legato al tempo perso da chi usa il servizio email sul proprio posto di lavoro, si è calcolato che la perdita di produttività ammonti a circa l'1.4-3.1% [7]. Per avere un'idea, si parla di cinque settimane-uomo per ogni milione di messaggi spam, considerando di perdere un secondo per ognuno cancellato. Queste lettere digitali spesso trasportano anche virus o attacchi di tipo phishing che portano a compromettere la sicurezza dei sistemi coinvolti e quindi anche dei dati privati degli utenti in essi contenuti, con corrispondenti perdite finanziarie.

Il costo per ogni singolo utente legato allo spam in sé non è particolarmente elevato anzi spesso risulta quasi impercettibile a chi utilizza saltuariamente il servizio di posta elettronica. Il problema vero è che se si analizza il costo aggregato, si trovano cifre dell'ordine di miliardi di dollari, considerando unicamente gli Stati Uniti [8]. Solo recentemente si sta iniziando a comprendere la portata di questo problema, destinato a diventare sempre più una vera e propria piaga anche sotto il profilo economico.

Si potrebbe pensare che la questione spam cresca con l'aumentare degli utenti che si collegano ad internet, in realtà il fenomeno è più complesso, in quanto proprio la mancanza di spese sostenute dallo spammer per inoltrare i propri messaggi rende questa espansione incontrollata. Gli unici limiti sono posti dal numero di lettere che uno spammer riesce ad inoltrare in un giorno, dalla velocità con cui può reperire sempre nuovi indirizzi email e (ove sia il caso) dalla capacità di calcolo che gli consente di effettuare attacchi a dizionario sugli indirizzi. Ad ogni risposta che riesce ad ottenere lo spammer ottiene il suo profitto, per questo più email manda più possibilità ha di risposta e più profitto ne ricava.

1.8. Composizione dello spam

Per comprendere meglio la composizione ed i contenuti dei messaggi di spam si sono compiute varie ricerche. Analizzando nel dettaglio una recente indagine pubblicata per l'*International Conference on Convergence Information Technology* del 2007 [9] si possono apprendere i risultati di circa un anno (dal Gennaio 2006 al Febbraio 2007) di posta indesiderata raccolta. In totale si sono collezionate circa 400.000 email ricevute da un server aziendale attrezzato con una *spam-trap* (trappola per lo spam). Il server forniva servizi per 200 utenti, con 20 identificativi di gruppo e 200 account individuali.

Le conclusioni ottenute sembrano piuttosto interessanti e permettono di farsi un'idea sulla composizione dei messaggi di spam, sui loro contenuti e su come sono strutturati. Da questi studi si arriva alla conclusione che chi compie questo genere di azioni di solito tende a non cambiare il proprio modo di agire giorno per giorno ma a mantenerlo il più a lungo possibile, nel caso si sia rivelato efficace.

Uno dei pochi motivi in grado di far decidere ad uno spammer di modificare i propri metodi di attacco è legato all'evoluzione dei sistemi antispam che

riescono a rendere inefficaci i mezzi finora utilizzati. Questo periodo va sperimentalmente da otto mesi a un anno. Analizzando nel dettaglio un periodo di due settimane si è osservato che non c'è un'apparente relazione tra posta legittima e non, e che allo stesso tempo lo spam costituisce quasi sempre la percentuale maggiore del traffico totale.

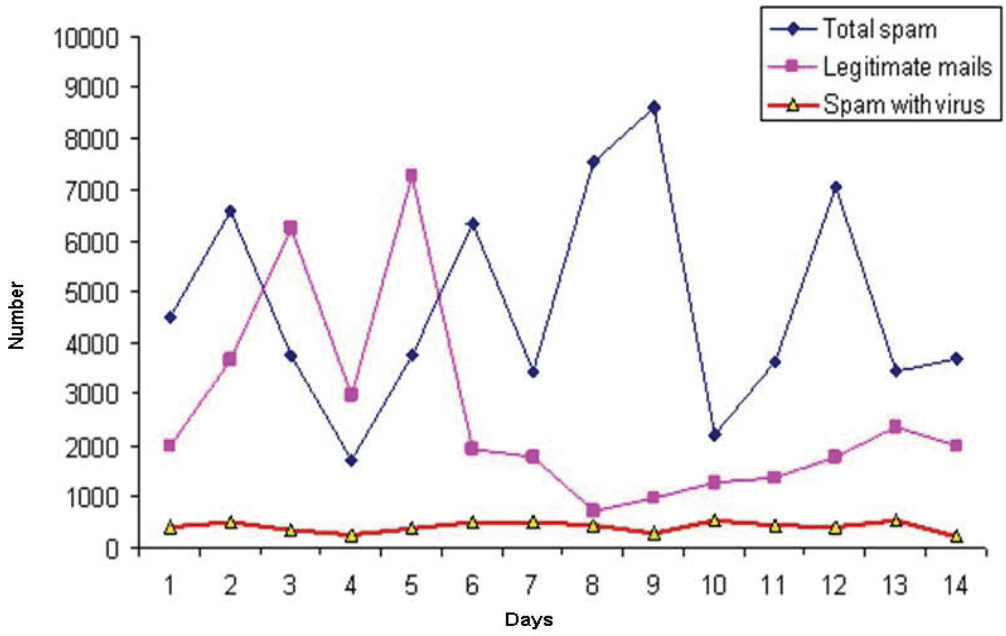


Figura 1.2 - Grafico relativo al traffico email in arrivo (fonte *Charaterizing Spam traffic and Spammers*)

Lo spam ha come oggetto nella maggior parte dei casi studiati: finanza, farmacia, promozioni commerciali, affari e pornografia. Le percentuali di spam con allegato rispetto a quello senza sono praticamente del 50% a testa. Il grafico riportato in Figura 1.3 rappresenta i due tipi di posta non legittima, evidenziando la mancanza di una forte relazione tra gli andamenti. Questo sta a significare l'esistenza di due tipologie di spammer diverse, che utilizzano tecnologie differenti per mandare i propri messaggi.

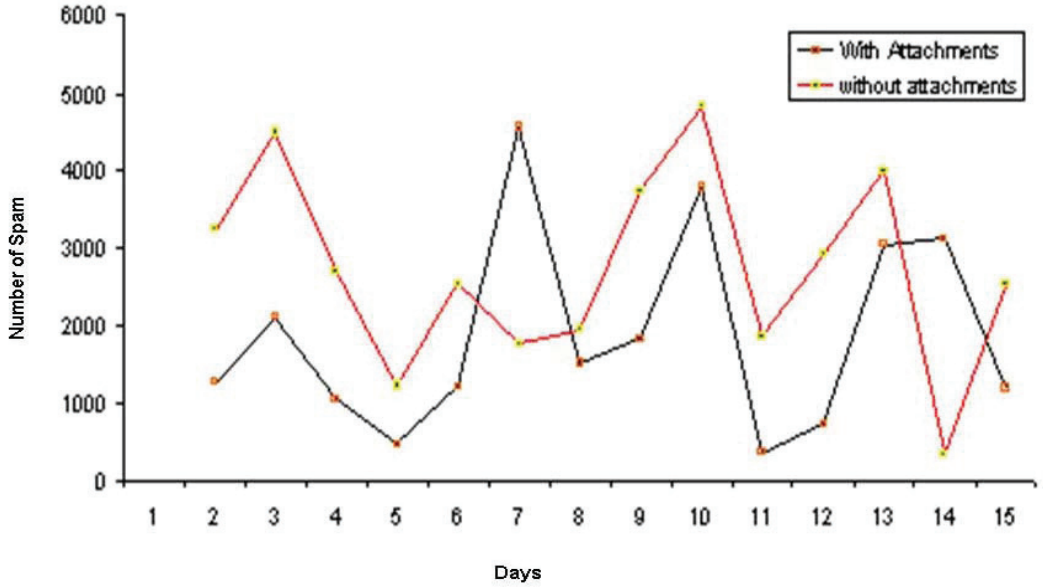


Figura 1.3 - Traffico spam con e senza allegati (fonte *Charaterizing Spam traffic and Spammers*)

Lo spam senza allegato di solito è di dimensione piuttosto piccola e può essere suddivisibile in lettere contenti solo testo o messaggi che riportano anche link ad un sito internet. Nel primo caso ci si trova di fronte molto spesso a vere e proprie truffe, inviate da mittenti individuali ad un numero ristretto di riceventi. La maggior parte degli indirizzi mittenti in questo caso non esiste, mentre quando è stato possibile individuare colui che ha spedito, risultava spesso africano ed utilizzante domini giapponesi. I messaggi che non riportano dei link nel loro corpo sono la minoranza rispetto al totale, tutti gli altri invece spesso conducono a pagine riguardanti prodotti farmaceutici.

Analizzando il traffico con allegati si può osservare che a sua volta è suddivisibile in email che trasportano immagini e vere e proprie *mail bomb,* contenenti worms e virus. Le lettere della prima categoria in prevalenza contengono allegati di tipo "*.gif*" ed in piccola parte "*.jpg*" mentre le altre sono quasi sempre portatrici di veri e propri programmi malevoli di estensione "*.exe*". Nel grafico di Figura 1.4 è stato raggruppato per contenuto lo spam e si può vedere che in massima parte contiene solo testo ed un'immagine, mentre non appaiono correlazioni evidenti tra le varie categorie.

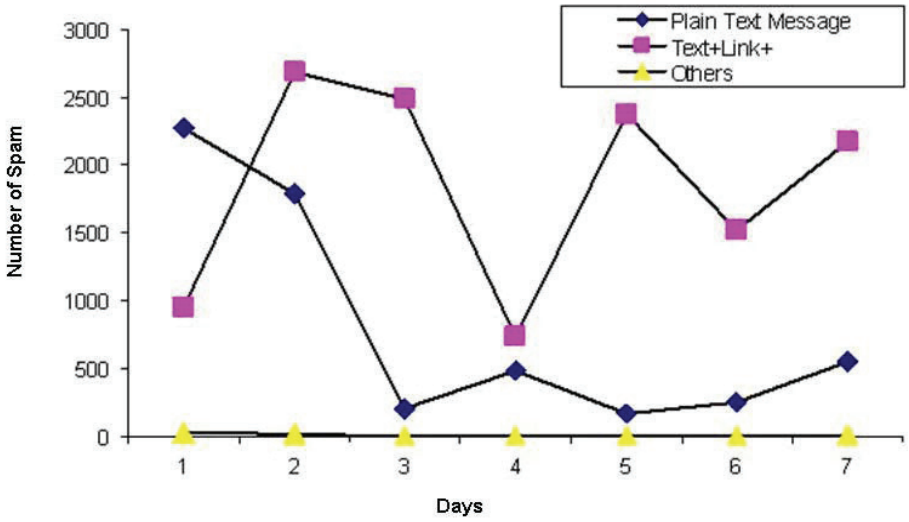


Figura 1.4 - Grafico relativo allo spam senza allegati (fonte *Charaterizing Spam traffic and Spammers*)

Le immagini utilizzate spesso sono costituite da testo e da un'URL che di solito non è direttamente *clickabile*, questo per non poter essere facilmente rilevati dai filtri di posta. Per lo stesso motivo anche il testo di sovente viene modificato tra le varie email sia per lunghezza che per contenuti pur mantenendo l'argomento, passando da una riga ad alcuni paragrafi.

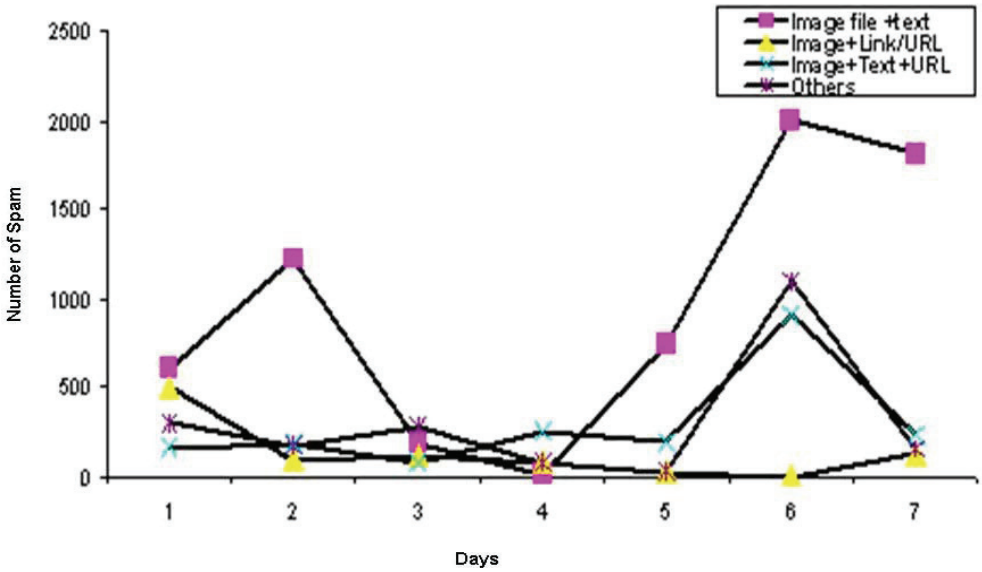


Figura 1.5 - Grafico relativo allo spam con allegati (fonte *Charaterizing Spam traffic and Spammers*)

Appare evidente che gli spammer utilizzino del software (spesso realizzato da loro stessi), permettendogli l'invio dei messaggi di posta spazzatura con allegati. Le principali caratteristiche di questi programmi sono:

1. Nascondere l'identità del mittente.

2. Scegliere a caso il testo del messaggio tra un elenco di possibilità.

3. Identificare gli *open relays* disponibili al momento in rete.

4. Permettere una capacità di invio su larga scala.

5. Definire l'ora e la durata dell'azione di inoltro.

Coloro che compiono gli attacchi avvalendosi di questi strumenti rappresentano la categoria degli spammer più sofisticati. Le analisi dimostrano che il 99% dei messaggi prodotti con questo metodo risulta essere stato inviato da account non esistenti. Un'altra classe di spammer meno evoluti invia le email senza allegati e ad un numero piuttosto ristretto di utenti, non usa software automatizzati, ma si avvale spesso di metodi per l'invio di posta gratuiti o rubando account altrui.

Si riportano i risultati degli ultimi studi compiuti riguardo allo spam, il primo fornito dal *Messaging and Anti-Abuse Working Group* (MAAWG) [10] stima per il 2007 tra il 75% e l'80% di email abusive sul totale del traffico di posta elettronica mondiale. Il secondo tratto dal *Global Internet Security Threat Report* di *Symantec* dell'aprile 2008 [11] fornisce un diagramma (Figura 1.6) circa la composizione dello spam evidenziando come gli argomenti più utilizzati per confezionare tali messaggi riguardino: prodotti commerciali, internet, finanza e salute.

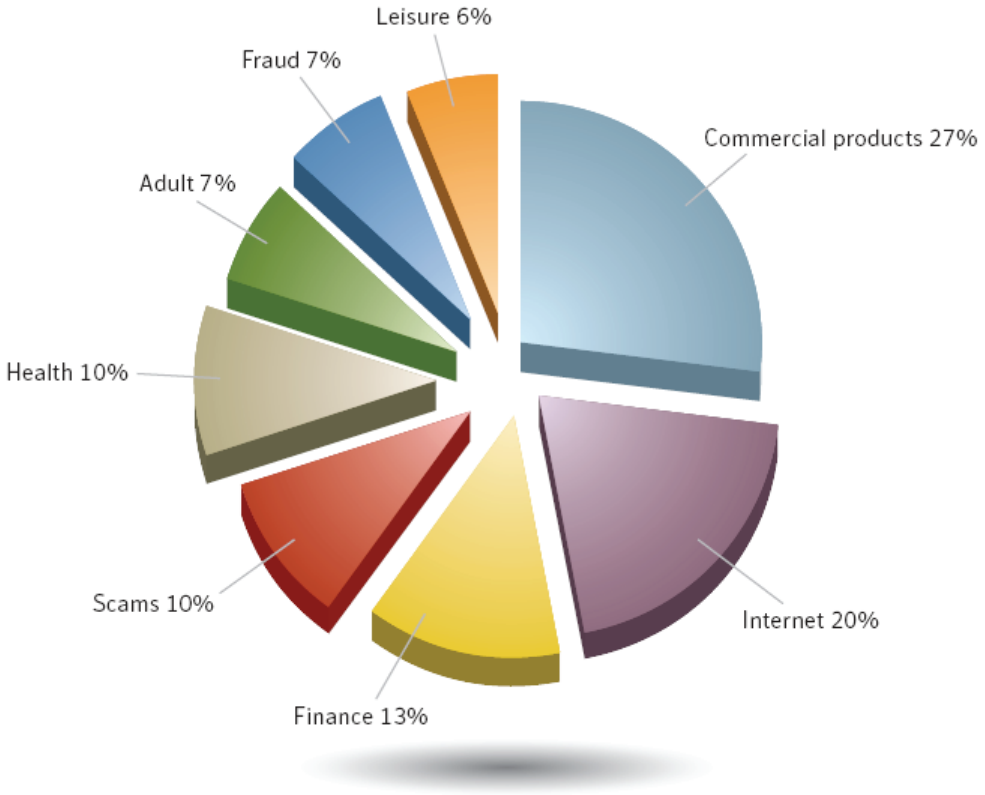


Figura 1.6 – Diagramma relativo alle categorie di argomenti più usati per lo spam (fonte Symantec)

1.9. Il phishing

Il *phishing* è una attività di truffa che sfrutta una tecnica di ingegneria sociale, cioè lo studio del comportamento individuale di una persona. Lo scopo è quello di ottenere l'accesso a informazioni personali o riservate con la finalità del furto di identità mediante l'utilizzo delle comunicazioni elettroniche, soprattutto attraverso email fasulle. Grazie a questi messaggi, l'utente ingannato è portato a rivelare dati personali, come numero di conto corrente, numero di carta di credito, codici di identificazione, ecc.

Il termine *phishing* è stato coniato nel 1996 e deriva da *fishing* (letteralmente "pescare"), probabilmente influenzato da *phreaking* che allude

all'uso di tecniche sempre più sofisticate per “pescare” dati finanziari e password di un utente. Un tipico attacco di phishing può essere strutturato come l’esecuzione di vari passi:

1. Il malintenzionato (*phisher*) spedisce al malcapitato ed ignaro utente un messaggio email che simula, nella grafica e nel contenuto, quello di una istituzione nota al destinatario (per esempio la sua banca, il suo provider web o un sito di aste online a cui è iscritto).

2. L’email contiene quasi sempre avvisi di particolari situazioni o problemi verificatesi con il proprio conto corrente privato o con uno qualsiasi dei suoi account (per esempio un addebito enorme o la scadenza dell’account ecc.).

3. L’email invita il destinatario a collegarsi ad un link, presente nel messaggio, per evitare l’addebito e/o per regolarizzare la sua posizione con l’ente o la società clonati.

4. Il link fornito *non* porta in realtà al sito web ufficiale ma ad una *copia fittizia* apparentemente simile all’originale, situata su di un server controllato dal phisher. Lo scopo è quello di richiedere ed ottenere dal destinatario dati personali particolari, normalmente con la scusa di una conferma o la necessità di effettuare una autenticazione al sistema. Spesso il sito è di buona fattura e può prevedere anche un offuscamento dell’indirizzo riportato sulla barra di navigazione. Per rendere ancora più efficace l’attacco una volta terminata l’operazione di inserimento dati, l’utente viene ridiretto alla pagina originale.

5. Il phisher utilizza questi dati per acquistare beni, trasferire somme di denaro o anche solo come “ponte” per ulteriori attacchi.

Secondo il *Global Phishing Report* dell’inizio 2008 pubblicato da *Symantec* [12], l’Italiano è la seconda lingua più utilizzata al mondo per confezionare le

email di phishing. Questo dato può essere legato ad una scarsa cultura informatica degli italiani, che porta spesso ad usare tecnologie nuove in maniera troppo disinvolta e senza prestare la giusta attenzione alle possibili insidie che possono nascondersi in rete. Non a caso i siti più clonati da queste truffe sono *Poste Italiane*, *eBay* e *PayPal*. Per frodi del genere, l'uso della lingua giusta è fondamentale: un utente italiano non darà alcun credito ad un messaggio della sua banca che sia scritto in inglese.

Mentre gli utenti cominciano a prendere coscienza dei rischi che possono provenire della rete, e quindi a controllare la veridicità degli indirizzi sui quali cliccano, i phishers rispondono ad esempio sovrapponendo immagini posticce alla barra di navigazione, nascondendo in questo modo l'indirizzo del sito truffaldino. Nell'ultimo periodo si sono scoperte anche email meno subdole contenenti addirittura minacce di morte, un esempio reale è il seguente: "Se ci dai il numero di carta di credito, valuteremo l'opportunità di non ucciderti".

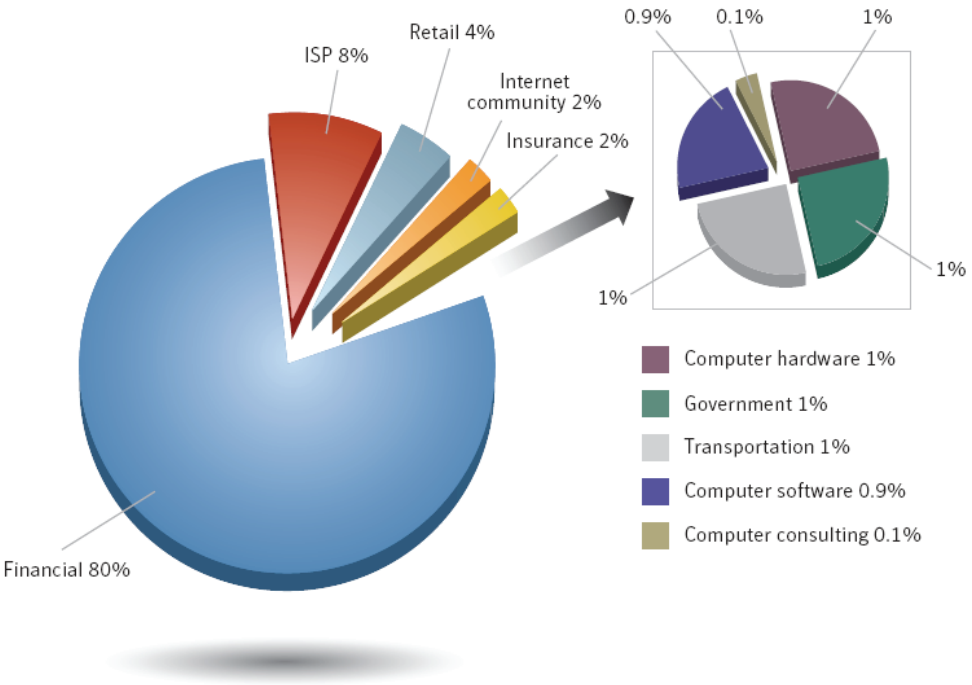


Figura 1.7 - Argomenti più utilizzati nei messaggi di phishing (fonte Symantec)

Un'altra dimostrazione della capacità evolutiva dei phishers è quella di saper sfruttare i problemi del nostro tempo, come attraverso l'invio di favolose offerte sui mutui, sapendo bene quanto oggi questa sia una difficoltà particolarmente avvertita, soprattutto negli Stati Uniti, alla luce della crisi dei *subprime*. La Figura 1.7 tratta da [11] circa la composizione del phishing evidenzia come l'argomento più utilizzato per confezionare tali messaggi sia la finanza.

1.10. I malware

Il termine *Malware* deriva dalla contrazione delle parole inglesi *mailcious* e *software*, dal significato letterale "programma malvagio" o più spesso *codice maligno*. Precisamente in questa categoria rientrano tutti gli eseguibili che hanno lo scopo di arrecare danni alla macchina sulla quale vengono avviati. Esistono vari tipi di questi software, gli esperti li raggruppano in famiglie e ad ognuna di esse possono appartenere diverse varianti, cioè versioni spesso identificate mediante l'uso delle lettere alfabetiche.

I malware si diffondo nella maggior parte dei casi attraverso la posta elettronica sotto forma di allegati oppure riportando nelle email un link ad un sito che li contiene. Altri metodi che ne facilitano la proliferazione sono i sistemi *peer-to-peer* per lo scambio di file tra utenti internet, le chat IRC o anche mezzi fisici quali i dispositivi di memoria USB. Una delle ultime ricerche della *SOPHOS* [13], importante ditta di sistemi anti-virus, rivela che circa l'80% del *web-based* malware è contenuto in siti web "innocenti" ma compromessi all'insaputa dei rispettivi amministratori.

Alcuni tipi di questi programmi per agire hanno bisogno di una breve interazione con l'utente che gli permetta di attivarsi, può bastare anche un semplice click del mouse. Essendo spesso i malware composti di più parti

interdipendenti, una volta entrati in funzione di solito vengono eseguiti altri componenti secondari ma non meno devastanti. Di seguito si riportano alcune delle caratteristiche salienti che identificano i programmi malevoli [14]:

1. Multifunzionalità e modularità: Vari tipi di software maligni possono essere utilizzati insieme per raggiungere lo scopo desiderato. Sono facilmente aggiungibili nuove funzionalità, permettendo di bypassare i sistemi antivirus ed i firewall, consentendo di abilitare in seconda battuta il download di altri applicativi simili e dando libero accesso alla macchina dall'esterno.

2. Utilizzabilità in rete ed interfacciamento *user-friendly*: Tali qualità lo rendono una potente arma nelle mani degli attacker con capacità di lanciare attacchi sofisticati anche oltre il loro livello di conoscenza.

3. Persistenza ed efficienza: Risultano sempre più difficilmente rilevabili e rimuovibili, presentando una costante evoluzione.

4. Potenzialità di contagio multipiattaforma: Non vengono interessati solo i *personal computer* ma anche i *personal digital assistants* (PDA) ed allo stesso modo anche i server ed i router.

5. Predisposizione ad altri attacchi: Il malware può essere il punto di partenza a cui far seguire altri tipi di assalti con delle vere e proprie azioni di *cybercrime*.

6. Redditività economica: Non è più solo un gioco di alcuni smaliziati programmatori o campo di studi e di ricerca ma rappresenta un vero e proprio modo per fare denaro.

Di seguito si riporta una breve classificazione del malware, visto la rapida evoluzione di questo campo, le definizioni sono spesso sfumate l'una sull'altra non lasciando netti margini tra di esse.

- Virus: Sono parti di codice che usano copiarsi all'interno di altri programmi, o in particolari sezioni del disco rigido.

- Worm: Non hanno bisogno di infettare altri file, modificano il sistema operativo della macchina ospite per essere eseguiti automaticamente, si possono replicare in rete.

- Backdoor: Consentono l'accesso non autorizzato al sistema, si diffondono di solito insieme a trojan o worm.

- Trojan: Contengono funzionalità lecite per indurre all'uso l'utente, inoltre dispongono di istruzioni dannose eseguite all'insaputa dell'utilizzatore, non riescono ad autoreplicarsi.

- Spyware: Rubano le informazioni private dell'utente trasmettendole a chi li diffonde.

- Dialer: Modificano il numero di telefono utilizzato per connettersi ad internet dalle macchine che accedono alla rete mediante sistemi dial-up.

- Hijacker: Provocano l'apertura di pagine web indesiderate mediante manipolazioni sui browser.

- Rootkit: Composti da driver e a volte copie modificate di programmi "normali", nascondono la presenza sul sistema di particolari file come spyware e trojan ai sistemi anti-virus.

- Rabbit: Esauriscono le risorse del computer creando copie di se stessi in memoria o su disco, a grande velocità.

Da uno studio di una associazione di banche del Regno Unito [15] si stima che le perdite dirette causate dal malware ai suoi associati ammontino a 12.2 Milioni di sterline per il 2004, 23.2 per il 2005 e 33.5 per il 2006, presentando un incremento del 90% dal 2004 e del 44% dal 2005. Nella Figura 1.8 sono riportati i numeri di incidenti legati ai soli trojan rilevati dalle banche del Regno Unito tra il 2005 ed il 2006, presentando anch'esse un andamento crescente.

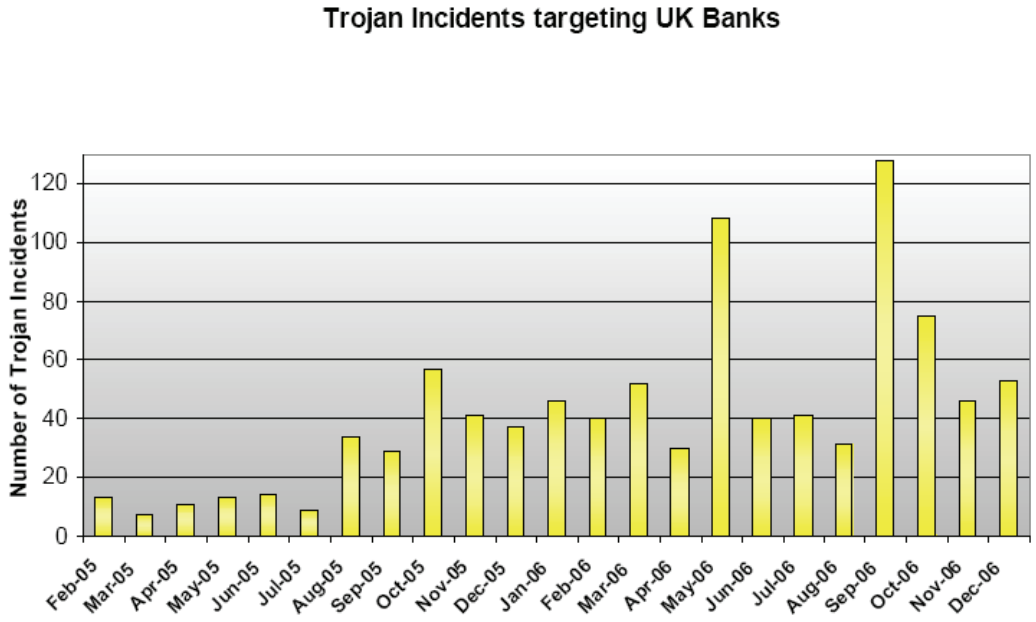


Figura 1.8 - Diagramma relativo al numero di attacchi trojan alle banche UK 2005/2006 (fonte OECD)

La crescita del fenomeno malware si può spiegare tenendo conto di alcuni fattori quali: l'aumento del numero di connessioni a banda larga, lo sviluppo di nuovi servizi disponibili in rete, le vulnerabilità riscontrabili nei software e nei sistemi operativi e la facilità con cui si possono colpire gli utenti internet medi.

1.11. Le botnet

Le botnet [16][17] sono insiemi di calcolatori controllati direttamente da un *botmaster*, il quale se ne può servire per far partire gli attacchi più disparati,

restando nell'anonimato. Il termine botnet significa letteralmente rete di bot dove per bot (abbreviazione di robot) si intende appunto una macchina infettata da un software malevolo che la controlla. I malware creati per rendere le macchine infettate parte di una botnet, possono essere virus informatici o *trojan*, e non appena assumono il controllo del sistema, forniscono al proprio autore i dati relativi al dispositivo assoggettato.

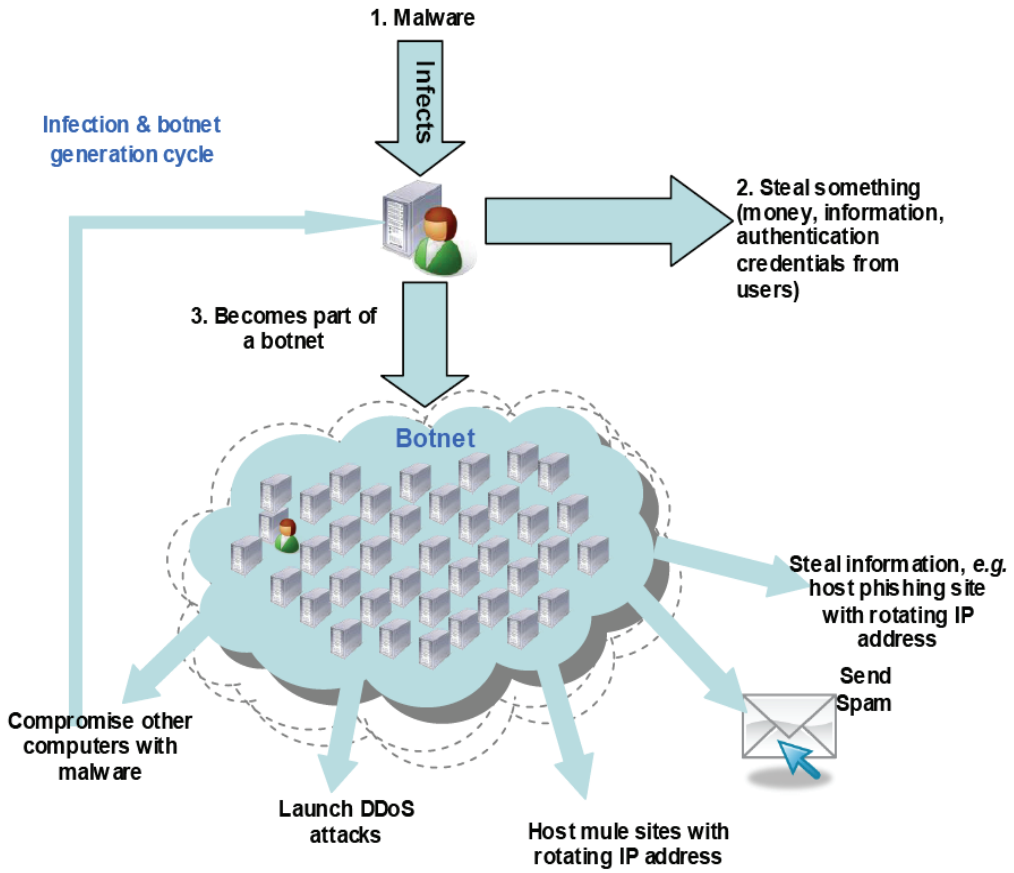


Figura 1.9 – Esempio di una botnet (fonte OECD)

Per fare ciò spesso vengono sfruttati i canali IRC (*Internet Relay Chat*) che consentono di comunicare direttamente con il programma che gestisce l'intera rete, situato su di un particolare server e protetto da una password per dare accesso esclusivo all'autore. Tramite il canale di chat il botmaster è in grado di controllare contemporaneamente tutti i sistemi infettati collegati in quel momento (i quali

possono essere anche decine di migliaia) ed impartire ordini specifici. In particolare mediante questi collegamenti un botmaster può riportare il comando che desidera far eseguire nel *topic* del canale. Ciò consente ai bot, man a mano che si connettono alla chat, di apprender data ed ora in cui sferrare l'attacco nonché il bersaglio da colpire.

Per esempio, con un solo comando un *botmaster* potrebbe far partire un attacco DDoS (*Distribuited Denial Of Service*) verso un sistema a sua scelta. Un'aggressione di questo tipo può bloccare interamente le funzionalità di un sito riempiendolo di richieste malevoli. Le botnet vengono spesso utilizzate anche per altri scopi oltre al DDoS, come punti di partenza da cui inoltrare infinità di messaggi spazzatura, sfruttando addirittura gli account dei proprietari delle macchine. I virus utilizzati sono di sovente programmati anche in modo da spiare il sistema infetto e intercettare password ed altre informazioni utili. Possono inoltre essere adoperati per ridirigere il traffico internet (*mirroring*) garantendo anonimato in rete agli attacker.

1.12. Conclusioni e prospettive future

Riepilogando quanto visto finora, si riporta un esempio (Figura 1.10) [14] in cui si sviluppa un attacco con vari strumenti. In particolare questo può compiersi mediante l'uso di più server internet per distribuire spam. Una volta ricevuta l'email e clickato sul link contenuto, il sistema può essere infettato e quindi compromesso, rubando i dati dell'utente e inviandoli ad altri siti web controllati dall'attacker o ad una sua casella di posta.

Generalmente l'attacker opera sotto vari nomi di dominio ognuno dei quali ha diversi indirizzi IP, che vengono fatti ruotare durante l'azione. Questo serve a rallentare il tempo delle verifiche circa l'origine fraudolenta del sito in questione, da parte di chi subisce l'attacco o di un ISP. Sotto il *domain name server* (DNS)

gli attacker possono cambiare velocemente le loro tabelle degli indirizzi per riassegnarne di nuovi alle macchine contenenti i siti malevoli. Ciò permette, nel caso un indirizzo venga chiuso, di poter comunque raggiungere il sito attraverso un altro presente nella tabella DNS.

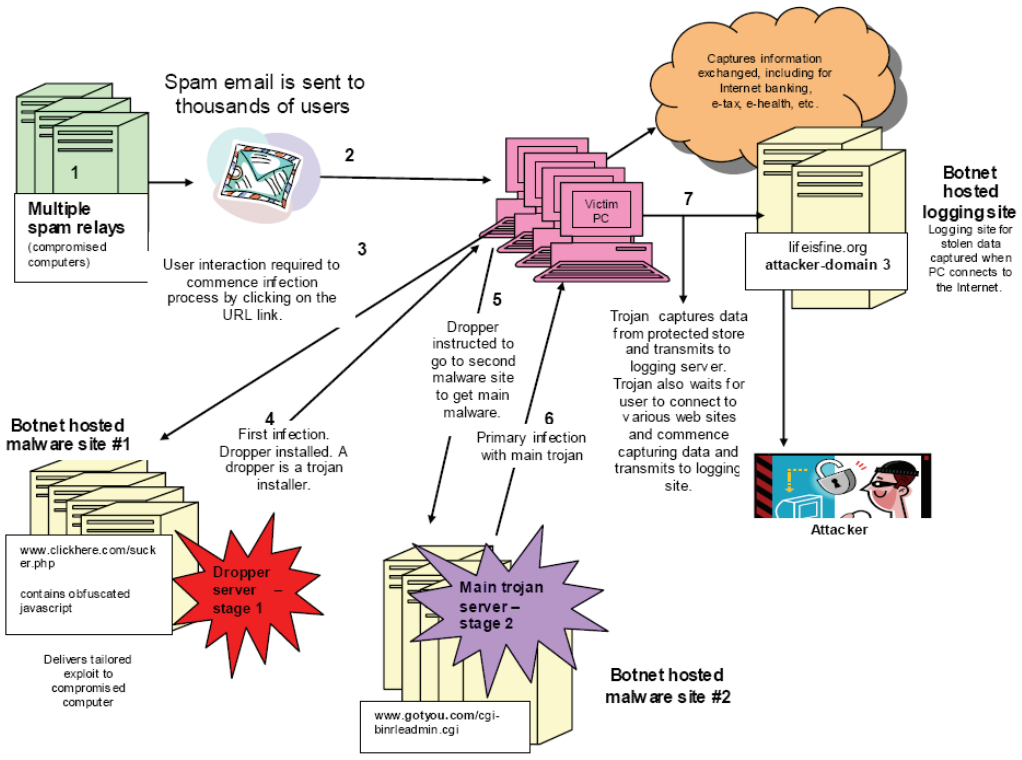


Figura 1.10 – Schema di un attacco internet utilizzando più strumenti (fonte OECD)

La tendenza attuale nel campo degli attacchi informatici su internet è variegata e può essere meglio compresa nella qualitativa rappresentazione della Figura 1.11 [14]. Negli USA, secondo OECD, che si interroga sul futuro di internet, 59 milioni di utenti covano nei propri sistemi spyware, backdoor e malware di vario genere. Considerando che la popolazione internet del paese veniva quantificata in 216 milioni alla fine del 2007, quasi un quarto risulterebbe quindi affetta dal problema. Numeri a cui non si fatica a credere considerando che *Rustock.C*, considerato il rootkit più sofisticato mai creato, è passato inosservato alle società di antivirus e ai software di sicurezza per quasi otto mesi.

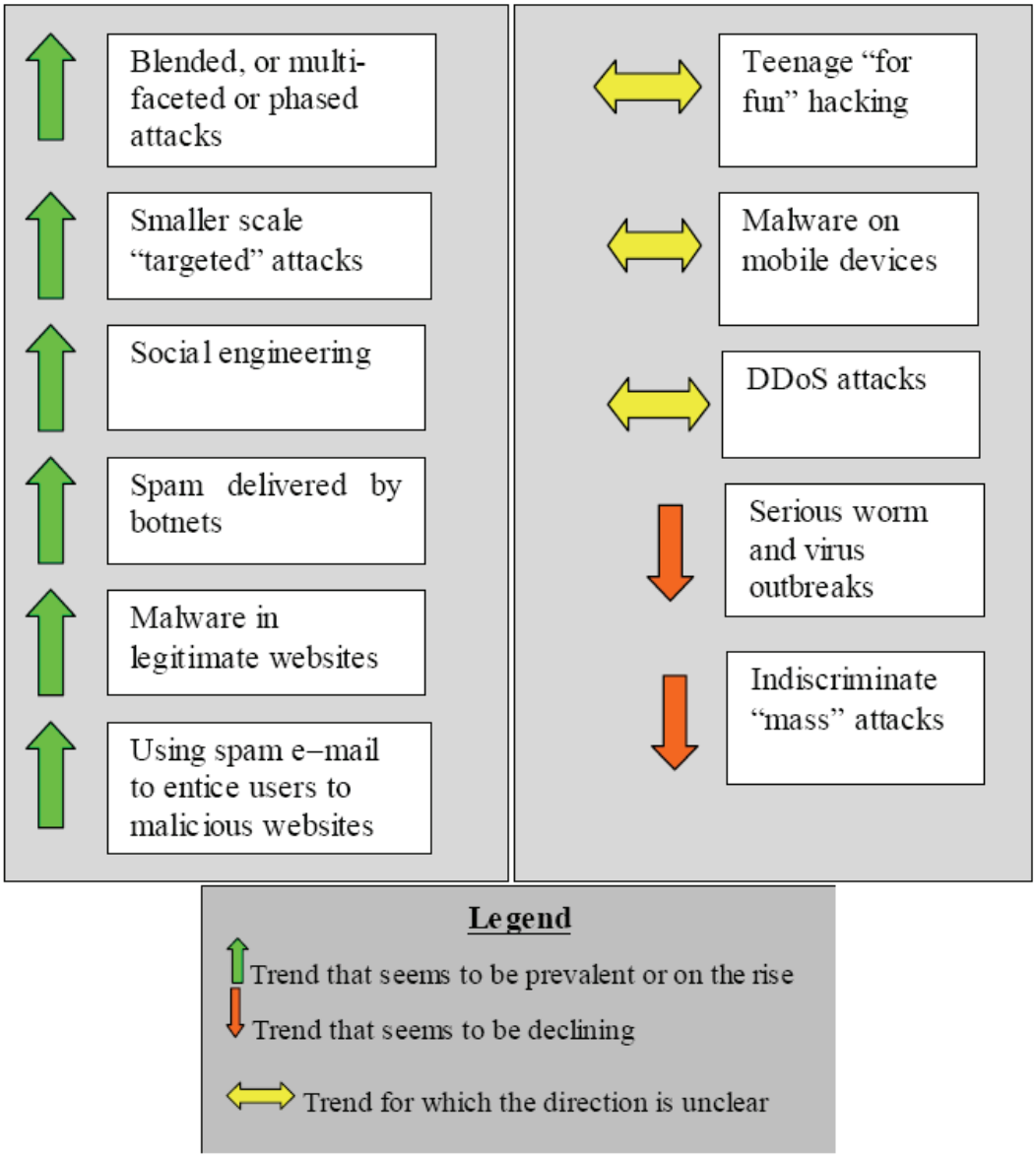


Figura 1.11 – Tendenze odierne relative alle tecniche di attacco (fonte OECD)

Mentre il servizio di posta elettronica è oggi ormai maturo e si presenta come terreno di dure battaglie tra malintenzionati che compiono attacchi sempre più spregiudicati e tecnici che tentano di arginare i problemi, le nuove tecnologie vanno avanti allargando il fronte dello scontro. In particolare con l'avvento di sistemi quali l'*instant messaging*, il *social network* e con tutte le novità portate dal cosiddetto "*web 2.0*" gli spammer hanno trovato terreni vergini sui quali prolificare.

Un esempio è proprio quello dell'instant messagging ossia il servizio di messaggistica istantanea che rende in qualche modo le email come uno strumento antico e poco interattivo. Questa nuova applicazione nel giro di poco tempo è già stata resa vittima di attacchi *SPIM* (*SPam over Internet Messaging*). Uno studio del *Pew Internet and American Life Project* [18] riporta che il 30% degli utenti di IM in USA riceve spim, corrispondenti a 17 milioni di vittime. Il Primo caso di spim si è registrato nel 2005 quando furono spediti più di 1,5 milioni di messaggi su MySpace.com, lo spimmer negoziò con il sito per essere l'unico a poter inoltrare spim ai suoi utenti, minacciando di rendere altrimenti pubblico il metodo utilizzato.

Anche il social network non sfugge al phishing, siti come *Facebook*, *LinkedIn* o *MySpace* nei quali migliaia di utenti inseriscono i propri dati per incontrare altre persone con interessi o conoscenze comuni, sono una vera miniera di informazioni utili per questi truffatori. I ricercatori di *FaceTime* hanno addirittura scoperto un kit "fai da te" che permette a chiunque di confezionare in pochi passaggi email di phishing su misura per questi siti.

Un altro campo in cui sta emergendo un problema del tutto simile al precedente è quello rappresentato dalle comunicazioni cellulari. Per ora da una ricerca industriale risulta che solo l'8% dei possessori di telefonini abbiano ricevuto sms di pubblicità non richiesta, tuttavia l'83% delle industrie di telecomunicazioni vedono in questo un enorme problema per il futuro.

Capitolo 2

Soluzioni agli attacchi email

Come visto nei capitoli precedenti gli attacchi al servizio email possono essere vari, gli unici fattori ad accomunare queste tecniche sono la crescita e l'evoluzione continua. Gli sforzi necessari a combattere o almeno ad arginare questi fenomeni si moltiplicano ogni anno e, seppure con qualche successo, sono in molti a ritenere che mai si riusciranno a debellare totalmente i rischi derivanti dal servizio di posta elettronica com'è oggi strutturato. In questo capitolo si passeranno in rassegna i metodi attualmente più utilizzati per affrontare soprattutto la minaccia derivante dallo spam.

2.1. Il bloccaggio e il filtraggio

Esistono molti servizi e software che i server email e gli utenti possono utilizzare per ridurre il carico di spam sui loro sistemi e caselle di posta. Alcuni contano sul rifiuto dei messaggi provenienti dai server riconosciuti come spammer. Altri analizzano in modo automatico il contenuto dei messaggi email ed eliminano quelli che assomigliano a spam. Questi due approcci al problema sono talvolta definiti come *bloccaggio* e *filtraggio*. Ognuna delle tecniche ha i suoi pregi e difetti, entrambe però riducono l'ammontare dello spam ricevuto dalle caselle postali degli utenti.

Il bloccaggio essendo effettuato più a monte nella comunicazione, permette di ridurre la banda sprecata, rifiutando i messaggi prima che siano trasmessi al server dell'utente. Il filtraggio anche se viene effettuato a valle tende ad essere una soluzione più accurata, poiché può esaminare tutti i dettagli delle lettere. Molti sistemi di filtraggio si avvantaggiano delle tecniche di apprendimento automatico del software , che permette di aumentare l'accuratezza globale del metodo rispetto al sistema manuale. Più verso il ricevitore ci si occupa di spam e più il sistema sarà inefficiente, perché si presuppone che l'email avrà comunque effettuato un determinato cammino dal trasmettitore, ed avrà quindi consumato un certo numero di risorse.

Il primo filtraggio ad essere stato realizzato, e probabilmente anche il meno efficace, fu effettuato sul lato ricevitore. Era basato su di una *blacklist* cioè una lista di indirizzi di possibili spammer dai quali non si voleva ricevere messaggi. Tuttavia mediante cambio di indirizzo mittente ed uso di botnet questo metodo risulta inefficiente. L'opposto del *blacklisting* è il *whitelisting* in cui vengono ricevute solo le email in arrivo da certi indirizzi, la soluzione funziona bene ma non è sempre utilizzabile soprattutto per quegli utenti che vogliono poter ricevere posta anche da nuovi corrispondenti [19].

Un altro tipo di filtraggio può essere effettuato sul corpo del messaggio o sull'argomento ed avviene andando a verificare la presenza di alcune parole chiave che sono spesso presenti nello spam. Tra gli ultimi e più evoluti di questi metodi si trovano i filtri "*Bayesian*" [20][21], che si basano sul calcolo delle probabilità che ogni parola comparendo in un testo ha di essere legata allo spam, questo metodo con le sue varianti sembra dare buoni risultati. Il filtraggio di *Bayes* è reputato da molti uno dei più funzionanti con stime del 99% di successi circa. La tecnica basata sul calcolo delle probabilità per funzionare bene deve conoscere prima dettagliatamente cosa è spam e cosa non lo è. Per farlo ha bisogno di una fase di *training* e successivamente durante il quotidiano funzionamento si potrà sfruttare le capacità di auto-apprendere dell'algoritmo, migliorando quindi la sua efficacia ad ogni utilizzo.

In fase di apprendimento, vengono esaminati i messaggi dividendoli in parole, chiamate *token*. Ogni token viene mantenuto in memoria con accanto il numero delle volte che compare nei messaggi di spam e nei messaggi legittimi. Essendo la memoria spesso limitata, vengono salvate di volta in volta le parole che compaiono più spesso nei messaggi. Quando la fase di apprendimento è terminata l'algoritmo utilizza il teorema di Bayes per effettuare su ogni messaggio un'analisi statistica sulle parole in esso presenti, fornendo una stima della probabilità che ci si trovi davanti a spam.

Tornando ai filtraggi, alcune soluzioni aziendali che lavorano su server compiono queste analisi andando ad interessarsi del numero di indirizzi contenuti nel campo destinatario. Nel caso in cui il campo sia costituito da una quantità elevata di utenti c'è la possibilità che il messaggio sia spam e per questo occorrerà effettuare una verifica in un deposito centralizzato di informazioni sulle email non legittime. Questo tipo di filtri è anche detto *collaborativo*, gli spammer riescono a bypassarli facendo piccole modifiche ad ogni messaggio, tuttavia le ricerche in questo campo sono ancora in piena fase di sviluppo.

Il filtraggio a volte è considerato troppo invadente nei riguardi della privacy, anche se molti amministratori preferiscono bloccare i messaggi che provengono dai server più tolleranti nei confronti degli spammer. Questi metodi di controllo possono aiutare regimi o dittature senza scrupoli ad ottenere filtraggi ad hoc per ridurre la libertà di parola imponendo una censura su alcuni argomenti mirati. Altro problema è quello legato ai "*falsi positivi*" cioè quelle email trattate come spam ma che invece sono messaggi del tutto innocui, se non addirittura molto importanti per il ricevente.

Per molti utenti perdere anche solo un'email legittima è inaccettabile, per questo preferiscono fare a meno dei filtraggi e cancellare anzi manualmente il proprio spam. Un esempio abbastanza eclatante si è verificato in una delle prime implementazioni di sistemi per il filtraggio della posta elettronica di *America Online* quando nel 2002 vennero bloccati 100 messaggi di posta inviati da

Harvard ai candidati che erano stati ammessi [22]. Esiste anche il problema opposto, quello legato ai "*falsi negativi*", cioè lo spam non correttamente riconosciuto. Quest'ultimo fa sentire il suo impatto soprattutto per quanto riguarda i costi legati alla produttività dei singoli utenti.

2.2. Il teergrubbing

Una tecnica diversa dalle precedenti che pure presenta dei punti di similitudine è quella denominata di *Teergrub* (detto anche *Trapit*) [23], dal nome del processo che viene fatto eseguire sul server. Con questo metodo si otterrà di poter ritardare la risposta fornita al server mittente durante l'inoltro di una lettera. La differenza principale nei confronti dei sistemi di filtraggio standard è legata al fatto che non si propone di bloccare veramente lo spam ma bensì di ritardarne la propagazione nella rete.

Il risultato sarà quindi quello di un drastico abbassamento del *rate* di invio dei messaggi da parte degli spammer. Nella versione base questo processo non si interessa del verificare il tipo di email che sta trattando, ritardando indifferentemente sia i messaggi utili che quelli spazzatura. Tutto ciò provoca un ritardo in tutte le comunicazioni via email, spesso rendendo il servizio poco efficiente nel caso lo si utilizzi massicciamente e non in maniera malevola.

Per ovviare a tale problema, sono state effettuate evoluzioni che vanno a verificare prima se l'indirizzo di partenza è tra quelli degli spam server noti e quindi provvede a ritardare solo quelle comunicazioni, esempi di questi tipi di sistema sono il *tarproxy* [24] e *jackpot*. Come visto in precedenza gli spammer si avvalgono spesso di metodi quali botnet per avere un largo numero di indirizzi mittenti non presenti nelle blacklist eludendo questo metodo.

2.3. Il greylisting

Una proposta ingegnosa che cerca di trarre beneficio dai metodi precedentemente illustrati si basa sull'utilizzo di un filtro lato client in grado di stimare la probabilità che quella trattata sia un'email di spam. In base alla probabilità calcolata, viene proporzionalmente ritardata la propagazione del messaggio, attraverso il *TCP damping* [25]. Tale tecnica consiste nel modificare i parametri TCP per rendere la comunicazione più difficoltosa, rallentandola.

In particolare, cambiando i dati legati all'*handshake* TCP tra il ricevitore ed il trasmettitore, si può rendere la comunicazione affidabile. Il ricevitore deve confermare la ricezione di ogni pacchetto e specificare la quantità di dati che potrà ricevere la volta dopo. Il server *receiver* calcola lo *spam score* per un messaggio in arrivo ed automaticamente provvede quindi a ritardare proporzionalmente la conferma di ricezione al *sender*. In alternativa è possibile agire sulla dimensione dei pacchetti che si vuole far inviare la volta dopo, rendendola molto piccola si può generare un alto *overhead* che rende la comunicazione molto inefficiente. Questo metodo è alla base del *greylisting* [26].

Mentre l'impatto su una singola email appare pressoché trascurabile, se invece si va a considerare il caso di uno spammer che invia uno stesso messaggio a molti utenti, ed ognuno di questi avrà identificato il messaggio come spam, la comunicazione risulterà estremamente lenta. Occorre differenziare il tipo di risposte che si riscontrano quando questo metodo agisce nei confronti di un *legittimate server* o invece di uno *spamming server*. Nel primo caso, cioè quando viene contrassegnata un'email come spam anche se in realtà non lo è, la consegna avviene comunque, anche se difficoltosamente. Quando invece il ritardo viene applicato nei confronti di un server di spam questi è più facile che si arrenda a concludere la comunicazione qualora il ritardo sia troppo elevato, perdendo il messaggio.

In questo fenomeno si riassume l'idea generale del *greylisting*, cioè rifiutare

la prima volta la connessione da chiunque non rientri in una whitelist. Lo spammer di solito invia *burst* di email a un gran numero di utenti quindi quando vedrà negarsi una prima volta la connessione è difficile che tornerà a riprovare più tardi avendo comunque tanti altri utenti a cui inoltrare. Gli email server legittimi invece di solito provvedono alla ritrasmissione anche per più di tre giorni non perdendo quindi i loro messaggi.

2.4. Risultati delle tecniche di listing

Si calcola che un uso combinato delle tecniche di greylisting, whitelisting e blacklisting possa ridurre di circa l'88% lo spam [27]. Tuttavia studi più approfonditi hanno rilevato che il greylisting presenta performance discontinue, in particolare ha un elevato numero di falsi positivi. Si stima che il 98% dello spam venga ritardato ma anche il 40-50% delle email legittime [28]. Se il mail server non è stato configurato correttamente è possibile che la connessione, qualora si verifichi ritardo eccessivo, venga persa.

Le tecniche basate sullo *spam score* delle email individuali sono buone quanto il filtro che utilizzano e portano gli spammer a modificare il formato dei loro messaggi. Nel caso si proceda poi a rendere più lente le comunicazioni di un server che invia spam, questo può penalizzare profondamente anche coloro che utilizzano quel server per attività legittime. Infine il TCP dumping richiede di riscrivere il codice dei client email sul lato ricevitore per rendere possibile lo *scoring* dei messaggi.

2.5. Il challenge-response

La tecnica chiamata *challenge response* si basa sull'utilizzo di una whitelist

e di una blacklist che devono essere mantenute dal ricevente. Il mittente non ancora presente in nessuna delle due liste, riceverà una cosiddetta CAPTCHA (*Completely Automated Public Turing test to tell Computer and Human Apart*) [29] che consiste in un test semplice per un umano ma difficile per una macchina che permette appunto di capire chi lo stia eseguendo. Uno spammer che si trova a inviare un vasto numero di email non può rispondere a così tante richieste e preferisce abbandonare la connessione. Gli utenti legittimi invece dovranno rispondere al test una sola volta cioè la prima in cui si stabilisce il nuovo contatto, dopodiché saranno inseriti automaticamente nella whitelist.

Un pregio di questo metodo è sicuramente quello di poter agire più a monte nella comunicazione (sul sender) evitando alla base la propagazione dello spam alleggerendo di molto lo spreco delle risorse di rete. Alcuni studiosi criticano questo modo di agire ponendo il caso in cui due utenti comunichino tra loro per la prima volta, il fenomeno si concluderà in una sorta di duello tra i due sistemi *challenge-response*. Qualcuno giudica questi strumenti come spam vero e proprio per il motivo che demandano il lavoro di capire se si tratti di messaggi utili a colui che invia la prima email.

2.6. Il computational stamp

La cosiddetta tecnica *computational stamp* si basa sull'utilizzo di una firma elettronica che viene generata singolarmente per ogni messaggio inviato. La *signature* viene calcolata utilizzando il testo del messaggio, il destinatario e il timestamp [30]. Il tempo di calcolo e di verifica sono asimmetrici, per il primo si stima nell'ordine della decina di secondi mentre per il secondo è quasi istantaneo. Lo spreco di tempo serve a rendere l'invio di molti messaggi in contemporanea difficoltoso e dispendioso, essendo necessario il calcolo singolo di ogni firma. Una variante di questa tecnica (chiamata *bread pudding protocol* [31]) è utilizzata dal progetto SETI (*Search for ExtraTerrestrial Intelligence*), che la sfrutta

insieme al white e blacklisting.

Spesso un utente legittimo può trovarsi nella necessità di dover inoltrare email da sistemi che non offrono particolari performance sotto il profilo delle capacità di calcolo, come palmari o sistemi di vecchia generazione [32]. In questa situazione la decina di secondi stimata in precedenza per un caso intermedio, si allunga di molto, rendendo l'operazione dispendiosa anche come consumo di batteria nel caso di sistema wireless. Dal punto di vista degli spammer, di sovente le capacità elaborative sono piuttosto ragguardevoli grazie all'utilizzo di *botnet* o *server farm,* per questo l'incidenza della computazione sarà molto minore della stima precedente rendendo comunque possibile una buona quantità di inoltri.

Altro difetto di questo sistema è costituito dall'obbligo del mittente di generare sempre la firma per ciascun messaggio anche quando il ricevente non la richieda. Tale *signature* a sua volta può rappresentare addirittura, per un sistema non correttamente istruito, una sigla di caratteri legati al mondo spam e per questo essere filtrata.

2.7. Tecniche a modifica di protocollo

Gli ultimi due metodi descritti sopra (challenge-response e computational stamp) vengono spesso racchiusi insieme nella categoria dei *sistemi a pagamento* (in termini di tempo). Queste tecniche per essere realizzate fisicamente richiedono modifiche al protocollo di trasmissione email. Per una loro implementazione si renderebbe necessaria una migrazione dal sistema email attuale ad uno più evoluto.

Questo cambio ha dei costi sia in termini finanziari che pratici, ad esempio nel caso in cui alcuni utenti tardassero a cambiare protocollo vedrebbero le loro email legittime non essere più recapitate correttamente. Un cambiamento così

repentino e drastico non sarebbe facilmente attuabile nell'odierno variegato universo di internet e si dovrebbero prevedere particolari sistemi di tolleranza per gestire la transizione.

Ammesso che comunque una modifica al protocollo email sia possibile ed un giorno si rendesse obbligatoria, le possibili variazioni attuabili potrebbero essere molteplici. Un altro esempio prevederebbe di dividere il protocollo in due passi: il mittente emette una richiesta di comunicare al ricevente, mentre quest'ultimo a sua volta risponde alla richiesta accettandola o rifiutandola ed in caso positivo avvierà la comunicazione. Altre modifiche al protocollo sono in fase di studio, un'idea consiste nel prevedere che il messaggio debba essere forzatamente copiato sul server del mittente invece che venir subito ricollocato nel server del destinatario.

2.8. L'information hiding

Si presentano ora le tecniche che non prevedono modifiche al protocollo email e che sono oggi utilizzate per combattere le insidie della posta elettronica non legittima. Una soluzione al problema viene fornita dall'utilizzo dell'*information hiding*. Questa tecnica che in realtà potremmo definire una regola di buon comportamento che ogni utente del servizio di posta elettronica dovrebbe adoperare, si basa sul tenere il proprio indirizzo segreto il più possibile.

Gli utenti che vogliono comunicare per un particolare scopo si possono generare un indirizzo temporaneo utilizzabile per quello scambio di corrispondenza [33]. Quando uno spammer riesce a violare quell'indirizzo, esso viene eliminato e si provvede a sostituirlo con uno nuovo. Tecniche simili sono già disponibili per chiunque ne voglia approfittare e sono offerte ad esempio da servizi quali *spamex.com* o *sneakemail.com*.

2.9. Tecniche di autenticazione

L'idea alla base di un sistema di autenticazione è che se un utente può fornire documentazione sulla propria identità si può essere quasi certi che non sia uno spammer perché potrebbe essere perseguito andando incontro a leggi severe. Sulla base di questo presupposto quindi un sistema può mettere l'utente che riporta una propria autenticazione certificata nella sua whitelist e accettare la sua posta. Il problema è che spesso gli spammer riescono a rubare le identità degli utenti attraverso i metodi visti in precedenza, in particolare un test eseguito su 400.000 mail di spam ha dimostrato che il 16% dei mittenti possedeva una qualche autenticazione [9].

Un accorgimento che si sta diffondendo oggi è quello di affiancare un indice di reputazione ad ogni identità per avere una maggiore sicurezza da questo tipo di filtraggio. Questo campo sembra promettente e grandi aziende lo stanno sperimentando, due esempi su tutti sono il *DKIM* (*Domain Keys Identified Mail*) di Cisco e Yahoo, e il *Microsoft's Sender ID*. I due sistemi proposti lavorano sul server ricevente e danno la priorità alle email di mittenti autenticati. In particolare il server ricevente controlla che il server mittente sia valido e nel dominio dove dovrebbe stare, andando ad analizzare le informazioni in un database DNS distribuito su internet, se il *check* risulta negativo la lettera viene ritardata o rifiutata. La soluzione sembra funzionare tuttavia la sola autenticazione può produrre problemi, come visto prima, ad esempio se un utente legittimo ha la sua registrazione nel DNS non configurata correttamente. Inoltre se il ricevente ha molte lettere quotidianamente nella sua casella è costretto ad effettuare un gran numero di controlli rallentando il suo lavoro.

2.10. Applicativi antispam

Fino ad ora si sono prese in esame le varie tecniche per il controllo dello

spam con i loro pregi e difetti. Le applicazioni antispam presenti sul mercato che attualmente danno i risultati migliori sono quelle usanti combinazioni delle varie tecniche esaminate. Il coordinamento tra di esse spesso è fornito da modelli collaborativi o distribuiti che di solito permettono anche un sistema di controllo remoto. Nella letteratura esistono molti esempi di sistemi collaborativi o distribuiti, per esemplificare meglio ne verrà analizzato uno nel seguito. Il metodo qui presentato è stato descritto in un documento IEEE dal nome: *Research on the Comprehensive Anti-Spam Filter* [34].

Il sistema esaminato è, a detta dei suoi autori "intelligente", cioè permette filtraggi più evoluti delle tecniche fin qui viste implementando inoltre una sorta di autoapprendimento. Alla base dell'idea c'è un filtraggio multistadio, formato da tre blocchi principali in ognuno dei quali si esegue un particolare controllo sui messaggi di posta:

1. *Analisi dell'origine*: Classica analisi dell'indirizzo mittente e scelta in base ad una white o blacklist.

2. *Analisi del contenuto*: I progettisti utilizzano un filtraggio basato sul metodo di *Naive-Bayes* (NB) per classificare in maniera probabilistica i messaggi in base al loro contenuto. Il metodo NB va ad analizzare le probabilità combinate che le parole in un testo assumano un particolare significato. Nel nostro caso andando ad esaminare le parole contestualizzate, invece che indipendentemente tra loro com'è stato fin qui detto, la probabilità di essere un testo legato a spam risulterà più verosimile. La scelta di questo algoritmo è stata anche in parte dettata dalla sua presunta efficienza rispetto ad altri sistemi di complessità esponenziale.

3. *Analisi del comportamento*: basandosi sull'analisi delle tabelle dei file di log sul mail server e sul filtraggio precedente, vengono costruite delle classi in cui suddividere i messaggi ricevuti.

Per smistare le lettere tra queste classi gli autori utilizzano un albero di decisione. Questa struttura è composta da: nodi intermedi che rappresentano test sulle email, rami che rappresentano vero o falso al test e dai nodi foglia che sono le classi vere e proprie. Attualmente esistono anche metodi più precisi per fare quanto detto, come *support vector machines* (SVM), ma sono stati scartati per l'eccessiva complessità computazionale.

Gli autori di questo articolo riportano che sono state eseguite 20.000 prove, di cui 15.000 di *training*, 2.500 di *pruning* e 2.500 di *testing*. Il risultato al termine è stata una precisione di circa il 92.7%.

2.11. Conclusioni

L'opinione oggi più accreditata è che attualmente non esista un metodo un loro aggregato in grado di assicurare un affidabilità del 100% e quindi una completa rimozione del problema. Le tecniche più promettenti sono quelle che usano una combinazione di vari procedimenti e che soprattutto sono in grado di lavorare sul server in particolare quelle che operano su quello del mittente in modo da limitare lo spreco di risorse. Queste tecniche sfortunatamente però richiedono una drastica modifica ai protocolli email già esistenti il che significa spese sia in termini di tempo che di denaro.

Attualmente si cerca soprattutto con mosse legali di ridurre il numero degli *open relays* che sono i centri dove si trasporta e produce il maggior numero di spam su internet si stima circa l'80% del totale. In aggiunta i metodi che hanno dato i maggiori risultati sono quelli basati sull'autenticazione, in particolare, negli email providers Google e Yahoo hanno avuto esiti molto buoni. Una stretta identificazione dell'utente può però portare a preferire altri servizi per mantenere la propria privacy ed il proprio anonimato in rete. Apparentemente oggi la tecnologia non permette di avere i benefici delle email (possibilità di anonimato e

comunicazione pressoché istantanea) e allo stesso tempo la protezione totale da un problema come lo spam.

Capitolo 3

I protocolli SMTP e POP3

3.1. Introduzione all'SMTP

L'SMTP (*Simple Mail Transfer Protocol*) è lo standard di fatto per la trasmissione delle email attraverso internet. Gli RFC più importanti a riguardo sono l'821 e il 1123, successivamente l'SMTP è stato esteso con nuove funzionalità creando lo standard ESMTP (*Enhanced SMTP*) definito nell'RFC 2821. Al momento la maggior parte dei server di posta utilizza l'ESMTP, dove la comunicazione avviene attraverso connessioni TCP utilizzando di solito la porta 25.

Il traffico SMTP si basa su di una serie di comandi che il client può richiedere al server e dall'altra parte su di un certo numero di risposte che quest'ultimo fornisce. Chiunque può effettuare una connessione ad un server SMTP ad esempio utilizzando il programma *telnet* per interloquire direttamente mediante lo standard senza intermediari. In principio la comunicazione avveniva senza la necessità di autorizzare il cliente ad operare sul server, questo perché gli utenti erano pochi e ancor meno i malintenzionati. Successivamente anche grazie all'ESMTP è stata resa disponibile questa funzionalità che altrimenti renderebbe i server preda di attacchi incontrollati da parte di chiunque, congestionando l'intera architettura.

3.2. Comandi SMTP

Si riassume ora quali sono i comandi supportati dal protocollo SMTP (RFC 2821), si ricorda che le risposte del server iniziano con un codice 250 in caso di successo, altrimenti con un altro numero indicante il tipo di errore verificatosi.

- *"HELO" / "EHLO"* : Questi due comandi forniscono risposte simili e sono i primi che un client deve impartire al server quando si connette. Si può scegliere in alternativa tra uno dei due, in particolare il secondo si è reso disponibile dopo la specifica dell'ESMPT mentre l'altro era il comando originale. Dopo uno dei due ordini occorre specificare un nome di dominio, infatti la loro funzione è proprio quella di rendere identificabile un client attraverso il nome indicato. Come risposta positiva, il server fornisce un elenco di comandi a cui è capace di rispondere, ognuno su di una nuova linea e iniziante con il codice "250-", alla fine invece sarà inviato il numero 250 senza trattino. In caso di malfunzionamento verrà ritornato in alternativa un codice di errore.

- *"MAIL FROM:"* : Si usa per iniziare il trasferimento di un'email, consecutivamente al comando occorre riportare l'indirizzo del mittente che sta inviando il messaggio (chiamato *reverse-path*).

- *"RCPT TO:"* : Come per il comando precedente, anche questo deve essere seguito da un indirizzo, in questo caso del destinatario, e può essere ripetuto più volte consecutivamente per specificare tutti coloro ai quali si vuole far giungere il messaggio.

- *"DATA"* : Con questo ordine si può iniziare a trasferire il messaggio email vero e proprio comprensivo di header e body. Normalmente si ottiene come risposta un codice 354 che invita a continuare l'immissione di linee del messaggio. Per terminare la lettera occorre digitare su linea nuova un punto e mandare a capo

(<*CRLF*>.<*CRLF*>), tale ultima riga verrà tralasciata e servirà solo per indicare il termine del comando. Alla fine il messaggio sarà completamente affidato al server che provvederà immediatamente al suo successivo inoltro. Se tutto sarà andato a buon fine verrà ritornato un codice 250, in alternativa un numero di errore indicherà il malfunzionamento.

- *"RSET"* : Permette il reset dei dati forniti al server fino a quel momento, questa direttiva è utile in caso di errore durante l'immissione delle informazioni di un'email.

- *"VRFY"* : Questo comando permette di sapere se l'argomento che lo segue identifica un utente o una mailbox.

- *"EXPN"* : Con quest'ordine si può chiedere conferma circa la possibilità che l'argomento consecutivo identifichi una mailing list.

- *"HELP"* : Con questo comando si possono richiedere al server alcune informazioni di aiuto qualora esse siano supportate.

- *"NOOP"* : Questa direttiva non produce alcun effetto pratico.

- *"QUIT"* : Con questo comando si può chiudere il canale di comunicazione aperto in precedenza verso il server, le transazioni pendenti saranno cancellate. Se va a buon fine produce un codice 221.

I comandi sopraelencati consentono di effettuare tutte le operazioni base per l'invio delle email, più qualche operazione particolare per conoscere la correttezza di alcuni degli argomenti che si vorrebbero utilizzare. Non viene però fatta menzione dei comandi necessari alla autenticazione implementati nell' ESMTP.

3.3. Risposte SMTP

Le risposte ai comandi previste dallo standard sono dei codici, formati da tre cifre, che possono essere seguiti da una o più linee di testo a seconda dei casi. Sull'RFC 2821 si possono trovare tutte le possibili risposte che un server può fornire ad ogni comando, si riporta di seguito le regole generali in base alle quali sono state scelte le tre cifre componenti il codice. Il primo numero fornisce informazioni circa il successo o meno dell'operazione richiesta:

- 1YZ : Risposta positiva preliminare, viene fornita di solito per indicare che finora il comando è andato a buon fine anche se si attende ancora un comando da parte del cliente per continuare o abortire l'azione.

- 2YZ: Risposta positiva di completamento, indica che l'azione è stata completata con successo e si è pronti a trattare una nuova richiesta.

- 3YZ: Risposta positiva intermedia, viene fornita durante i comandi che prevedono sequenze (ad es. DATA), indica che il server sta attendendo altre informazioni prima di poter completare l'azione.

- 4YZ: Risposta negativa temporanea, indica che la richiesta non è andata a buon fine in quel frangente ma che può essere ripetuta di nuovo.

- 5YZ: Risposta negativa permanente, il server non ha accettato l'azione e ne scoraggia la ripetizione successiva.

Si analizza adesso il secondo numero del codice di risposta, esso indicherà una certa categorizzazione di ciò che è avvenuto durante l'elaborazione dell'azione richiesta:

- X0Z: Indica un problema inerente la sintassi del comando.

- X1Z: Questa risposta viene fornita a seguito di una richiesta di informazioni come ad esempio dopo un comando di HELP.

- X2Z: Una risposta con un codice di questo tipo è legata a qualche problema avvenuto a livello di canale di trasmissione.

- X3Z: Non specificato.

- X4Z: Non specificato.

- X5Z: Una risposta di questo tipo indica lo stato del sistema mail ricevente.

La terza cifra fornisce la sottocategoria di risposta specificata dalla seconda e non è precisamente definita dallo standard lasciando libertà di scelta all'implementatore. Tale potere decisionale è comunque limitato alle risposte non scrupolosamente definite dall'RFC.

3.4. Esempi di comunicazioni SMTP ed ESMTP

Si riporta in questa sezione un piccolo esempio allo scopo di rendere più chiaro al lettore quanto descritto nella prima parte del capitolo. Un mittente (Bob) vuole inviare un'email ad un ricevente (Alice), utilizzando il suo client di posta (in questo caso telnet) per connettersi al server smtp (smtp.example.com). Per effettuare la connessione viene usato il comando:

telnet smtp.example.com 25

A questo punto Bob può iniziare a dialogare con il server di posta mediante

lo standard SMTP:

```
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<bob@example.org>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example" <bob@example.org>
C: To: Alice Example <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 Jan 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Alice.
C: This is a test message with 5 headers and 4 lines in the body.
C: Your friend,
C: Bob
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
{The server closes the connection}
```

In alternativa se Bob si trova a comunicare con un server ESMTP potrebbe usare il comando EHLO per farsi identificare, ottenendo in risposta una serie di comandi che il server riconosce:

```
S: 220-smtp2.example.com ESMTP Postfix
C: EHLO bob.example.org
S: 250-smtp2.example.com Hello bob.example.org [192.0.2.201]
S: 250-SIZE 14680064
S: 250-PIPELINING
S: 250 HELP
```

Fin qui non si sono ancora introdotti gli strumenti per permettere a Bob di autenticarsi e quindi per poter utilizzare il servizio che gli viene messo a disposizione da un server ESMTP.

3.5. Autenticazione mediante ESMTP

Per utilizzare i moderni server di posta, come quelli che ad esempio ci vengono forniti dopo una registrazione ad un ISP, occorre un comando specifico definito dallo standard ESMTP che consenta di dimostrare l'effettiva legittimità all'uso di questo servizio. L'operazione di autenticazione viene descritta specificatamente dall'RFC 4954 (che ha reso obsoleto il 2554), si illustra di seguito un breve riassunto:

- *"AUTH"* : Permette di indicare il meccanismo di autenticazione che si vuole adottare, occorrerà riportarlo consecutivamente al comando. I meccanismi di autenticazione che possono essere usati sono quelli definiti dallo standard SASL (*Simple Authentication and Security Layer*) e devono anche essere supportati dal server stesso. Se si tenta di adoperare un meccanismo non supportato si otterrà un codice di errore 504. Una volta autenticati lo si rimane per tutta la sessione, se si ritenta l'operazione, si otterrà un codice di errore 503. Un'autenticazione SASL si basa sul concetto di *challenge* (server) - *response* (client). Per comunicare in questa fase viene usata una codifica *BASE64* e nel caso si presenti un errore verrà fornito un codice 501. I formati SASL utilizzabili sono vari, i più noti sono *PLAIN* e *CRAM-MD5*. Se l'autenticazione va a buon fine verrà ritornato un codice 235.

Si analizzano di seguito due esempi che chiariscono come funzioni il meccanismo di autenticazione ed in particolare l'utilizzo dei formati PLAIN e CRAM-MD5:

- Autorizzazione PLAIN:

```
S: 220-smtp.example.com ESMTP Server
C: EHLO client.example.com
S: 250-smtp.example.com Hello client.example.com
S: 250 AUTH GSSAPI DIGEST-MD5 PLAIN
C: AUTH PLAIN dGVzdAB0ZXN0ADEyMzQ=
S: 235 2.7.0 Authentication successful
```

- Autorizzazione CRAM-MD5:

```
S: 220-smtp.example.com ESMTP Server
C: EHLO client.example.com
S: 250-smtp.example.com Hello client.example.com
S: 250-AUTH DIGEST-MD5 CRAM-MD5
S: 250-ENHANCEDSTATUSCODES
S: 250 STARTTLS
C: AUTH CRAM-MD5
S: 334 PDQxOTI5NDIzNDEuMTI4Mjg0NzJAc291cmNlZm91ci5hbmR
       yZXcuY211LmVkdT4=
C: cmpzMyBlYzNhNTlmZWQzOTVhYmExZWM2MzY3YzRmNGI0MWFjMA==
S: 235 2.7.0 Authentication successful
```

3.6. Introduzione al POP3

Il POP3 (*Post Office Protocol version 3*) nasce come evoluzione del POP1 e del POP2, diventati ormai obsoleti. In particolare rappresenta il protocollo standard a livello applicazione in internet per poter ricevere i messaggi di posta da un server direttamente sul proprio client. Il POP3 lavora su connessioni di tipo TCP/IP generalmente utilizzando la porta 110. Il protocollo è piuttosto flessibile e consente sia di scaricare i messaggi cancellandoli dal server che di lasciarli sulla stessa macchina remota una volta acquisiti.

Un nuovo standard chiamato IMAP (*Internet Message Access Protocol*), arrivato già alla versione 4, cerca oggi di spodestare il solido POP3. La differenza tra i due protocolli è legata alla permanenza dei messaggi già scaricati sul server. IMAP4 quando invia le email al client ne fa una copia mantenendo l'originale sul server, e rimuovendo solo in un secondo momento, qualora appositamente richiesto, l'originale. Il POP3 può lavorare allo stesso modo usando generalmente il comando UIDL (*Unique IDentification Listing*) che produce un elenco dei messaggi presenti nella mailbox identificati da un numero univoco. Senza usare il comando specificato i messaggi vengono identificati da numeri che possono cambiare tra sessioni differenti e quindi rendere problematico capire se sono già stati scaricati dal client o meno.

Il POP3 è ampiamente descritto nell'RFC 1939 in cui vengono elencati anche tutti i comandi supportati. Per connettersi ad un server di posta usando questo standard è necessaria l'autenticazione dell'utente, questa all'inizio era prevista solo attraverso un meccanismo di login non cifrato. Successivamente sono stati ammessi altri tipi di meccanismi più sicuri per rendere più difficili possibili attacchi. Ad esempio il sistema APOP che utilizza una funzione hash MD5 per evitare possibili attacchi di tipo *reply* o il furto del segreto condiviso. Per rendere il tutto ancora più protetto oggi alcuni email client cifrano il loro traffico POP3 usando TLS o SSL oppure altri come il servizio *Gmail* di Google usano il metodo *alternate-port* lavorando sulla porta TCP 995.

3.7. Comandi POP3

In questa sezione si espongono i comandi supportati dal protocollo POP3 come definiti dallo standard riportato sull'RFC 1939. Si anticipa che le risposte possibili a tutti gli ordini saranno sempre di due tipi:

- **+OK** : Se tutto è andato a buon fine.
- **-ERR** : Se si sono verificati errori.

Elenco Comandi con relativi esempi:

- *USER* : Permette di iniziare il processo di autenticazione in chiaro specificando il nome dell'utente che sta cercando di accedere al server.

```
C: USER frated
S: -ERR sorry, no mailbox for frated here
...
C: USER mrose
S: +OK mrose is a real hoopy frood
```

- *PASS* : Permette di completare il processo di autenticazione in chiaro specificando la password dell'utente che vuole accedere al server.

```
C: USER mrose
S: +OK mrose is a real hoopy frood
C: PASS secret
S: -ERR maildrop already locked
...
C: USER mrose
S: +OK mrose is a real hoopy frood
C: PASS secretx
S: +OK mrose's maildrop has 2 messages (320 octets)
```

- *APOP* : Specificando l'identificativo di una mailbox, ed un MD5 digest permette di effettuare una autenticazione alternativa a quella in chiaro.

```
S: +OK POP3 server ready
    <1896.697170952@dbc.mtview.ca.us>
C: APOP mrose c4c9334bac560ecc979e58001b3e22fb
S: +OK maildrop has 1 message (369 octets)

Il segreto condiviso è la stringa 'tanstaaf'.
Applicando l'MD5 alla stringa
'<1896.697170952@dbc.mtview.ca.us>tanstaaf'
Produce un valore di digest:
c4c9334bac560ecc979e58001b3e22fb
```

- *STAT* : Serve a far conoscere al client la *drop listing*, cioè informazioni utili sullo stato della mailbox.

```
C: STAT
S: +OK 2 320
```

- *LIST* : Se viene passato anche un argomento indicante il numero di un messaggio, il server risponderà con informazioni su di esso dette *scan listing*. Nel caso invece in cui non sia passato alcun argomento verrà ritornata una lista dei messaggi disponibili sul server.

```
C: LIST
S: +OK 2 messages (320 octets)
```

```
S: 1 120
S: 2 200
S: .
...
C: LIST 2
S: +OK 2 200
...
C: LIST 3
S: -ERR no such message, only 2 messages in maildrop
```

- *UIDL* : Se viene passato anche un argomento, indicante il numero di un messaggio, il server risponderà con un identificativo unico per esso, detto *unique-id listing*. Nel caso invece in cui non sia passato alcun argomento verrà ritornata una lista dei messaggi disponibili sul server ognuno col proprio identificativo unico.

```
C: UIDL
S: +OK
S: 1 whqtswO00WBw418f9t5JxYwZ
S: 2 QhdPYR:00WBw1Ph7x7
S: .
...
C: UIDL 2
S: +OK 2 QhdPYR:00WBw1Ph7x7
...
C: UIDL 3
S: -ERR no such message, only 2 messages in maildrop
```

- *RETR* : Permette di ricevere un messaggio identificato da un numero passato come argomento, che può essere stato acquisito in precedenza mediante una delle funzioni di listing. Questo comando non prevede la rimozione automatica dell'email dal server.

```
C: RETR 1
S: +OK 120 octets
S: <the POP3 server sends the entire message here>
S: .
```

- *TOP* : Questo comando richiede due parametri il primo serve a specificare il numero di un messaggio mentre il secondo il numero di

righe del suo body che si vogliono visualizzare.

```
C: TOP 1 10
S: +OK
S: <the POP3 server sends the headers of the
message, a blank line, and the first 10 lines
of the body of the message>
S: .
...
C: TOP 100 3
S: -ERR no such message
```

- *DELE* : Serve a specificare il numero di un messaggio da marcare come *deleted* cioè eliminato. L'email non sarà più accessibile per nessun comando, tuttavia non verrà cancellata finché non si passerà allo stato di *update* e cioè non si concluderà la sessione con il comando QUIT.

```
C: DELE 1
S: +OK message 1 deleted
...
C: DELE 2
S: -ERR message 2 already deleted
```

- *NOOP* : Questo comando non compie nessuna azione, il server potrà solo rispondere con un "+OK".

```
C: NOOP
S: +OK
```

- *RSET* : Permette di ripristinare lo stato dei messaggi che durante la stessa sezione erano stati marcati come da cancellare attraverso il comando DELE.

```
C: RSET
S: +OK maildrop has 2 messages (320 octets)
```

- *QUIT* : Questo comando prima di tutto rimuove i messaggi che sono stati marcati dal comando DELE, successivamente chiude la connessione rilasciando il lock esclusivo sulla mailbox. La terminazione della connessione TCP avviene indipendentemente dall'esito della rimozione delle email che comunque verrà riportato come risposta al client.

```
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
...
C: QUIT
S: +OK dewey POP3 server signing off (2 messages left)
```

Come ultima annotazione vale la pena di segnalare che il protocollo prevede la possibilità di risposte su più linee. Un esempio è fornito dalla reazione al comando RETR che propone in risposta l'intera email. In questi casi particolari è previsto che per specificare la fine della risposta venga restituito all'inizio dell'ultima riga un punto e a capo (<*CRLF*>.<*CRLF*>).

3.8. Esempi di comunicazioni POP3

Si riporta in questa sezione un piccolo esempio allo scopo di rendere più chiaro al lettore quanto descritto nella prima parte del capitolo. Un utente vuole accedere alla sua posta contenuta in una mailbox su di un server all'indirizzo *pop.example.com*. Per effettuare la connessione viene quindi usato il comando:

telnet pop.example.com 110

A questo punto l'utente è connesso al server e può autenticarsi, si ponga il caso utilizzando il comando APOP. Successivamente lo stesso vuole scaricare tutti i suoi messaggi cancellandoli dal server:

```
S: +OK POP3 server ready <1896.697170952@pop.example.com>
C: APOP mrose c4c9334bac560ecc979e58001b3e22fb
S: +OK mrose's maildrop has 2 messages (320 octets)
C: STAT
S: +OK 2 320
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: RETR 1
S: +OK 120 octets
S: <the POP3 server sends message 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: RETR 2
S: +OK 200 octets
S: <the POP3 server sends message 2>
S: .
C: DELE 2
S: +OK message 2 deleted
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
{The server closes the connection}
```

La possibile alternativa per quanto riguarda l'autenticazione è quella basata sull'utilizzo di USER e PASS in chiaro, rimpiazzando quindi il comando APOP con:

```
C:    USER mrose
S:     +OK User accepted
C:    PASS mrosepass
S:     +OK Pass accepted
```

Capitolo 4

Il sistema proposto

Bastano poche ricerche sul web per capire quanto i problemi legati alla posta elettronica stiano di giorno in giorno diventando più importanti per la gestione di un sistema informatico moderno che si affacci ad internet. Applicativi software quali firewall, antivirus ed antispyware sembrano non bastare più per risolvere tutti i possibili attacchi provenienti dalla rete. Sempre più spesso anche nelle piccole realtà aziendali il problema della sicurezza informatica risulta essere un punto fondamentale, dedicando grosse risorse finanziarie alla gestione dei sistemi e alla loro salvaguardia.

Da qualche anno anche spam e phishing posseggono degli strumenti per essere combattuti quali i filtri anti-spam. Pur essendo tecniche sviluppate da pochi anni, sul mercato sono già disponibili software piuttosto efficaci che attraverso algoritmi sempre più complessi cercano in diversi modi di arginare la posta elettronica indesiderata e tutti gli altri attacchi che da essa possono derivare. La maggior parte dei programmi tuttavia si occupa soprattutto di porre un freno al martellamento dei messaggi proveniente dalla rete e si disinteressa totalmente del traffico che lo stesso sistema produce in uscita. L'idea alla base di queste realizzazioni resta quella in cui si pensa che l'amministratore sia abile a configurare il proprio firewall in maniera da bloccare qualsiasi connessione uscente non desiderata.

Questa semplice considerazione è stata lo spunto di partenza dal quale si è deciso di proporre un nuovo tipo di applicativo. In particolare anche se un amministratore fosse molto specializzato in questi problemi (cosa del tutto non scontata, soprattutto per sistemi di modeste dimensioni), ed utilizzasse un firewall altamente evoluto, non potrebbe comunque sapere a priori se la posta che sta uscendo dal proprio sistema sia legittima o meno. Da qui l'idea di porre anche in uscita i filtri che si usano per la posta entrante, e creare un nuovo metodo sullo stile del *challenge-response* per garantire autorizzazioni ed autenticazioni. L'applicativo dovrà essere inoltre di facile utilizzo, in modo da poter essere fruibile soprattutto da parte degli utenti medi che, senza avere conoscenze specifiche nel networking, possono comunque trarne vantaggio.

4.1. Obiettivi del sistema proposto

Il sistema proposto si pone l'ambizioso fine sia di fornire un robusto meccanismo **anti-spam** per la posta entrante, sia un innovativo metodo **anti-bot** per il bloccaggio del traffico di posta non desiderata in uscita, che può essere prodotta inconsapevolmente a seguito di un'infezione. Si è cercato in particolare sia di proteggere una sola macchina, sia un'intera rete locale. Per permettere la flessibilità voluta si è pensato di porre il sistema di filtraggio sulla macchina dalla quale si ha l'accesso ad internet, sia questa un *gateway* o un *proxy*.

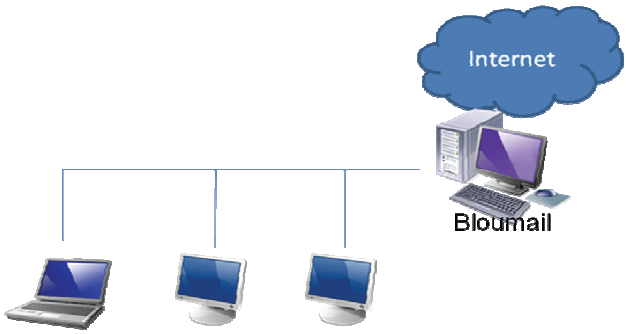


Figura 4.1 - Schema di utilizzo su di una rete LAN

In fase implementativa si è tentato soprattutto di rendere il sistema il più efficiente possibile sia dal punto di vista dell'uso di memoria sia dello spreco di risorse di calcolo, questo per rendere il filtraggio veloce e permetterne l'utilizzo anche su gateway di modeste capacità che si possono trovare nelle reti LAN più piccole. Un altro target progettuale è stato quello di programmare in ottica di successive modifiche che potranno essere svolte al fine di permettere l'utilizzo di questo software anche su veri e propri router. L'uso di macchine con più di due interfacce di rete permette di adoperare il sistema anche su un vero e proprio dominio formato da varie reti LAN che si affacciano tutte ad internet attraverso lo stesso gateway, il quale può funzionare anche come un bridge tra le varie reti.

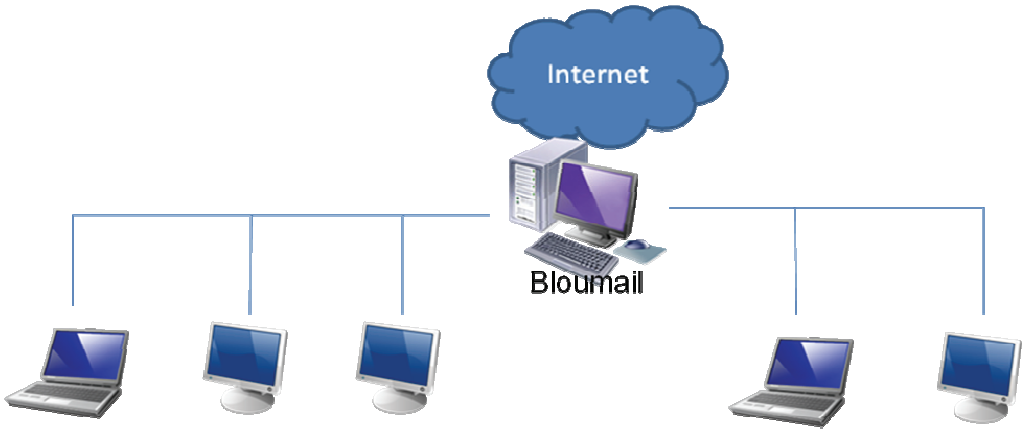


Figura 4.2 – Schema di utilizzo su più reti LAN

Le scelte topologiche appena descritte portano altri vantaggi, ad esempio si può dare per scontato che il gateway non finisca mai vittima di infezioni da parte di software malevolo né possa essere preda di attacchi, risultando ben protetto da appositi firewall settati dall'amministratore. Allo stesso tempo, l'uso ristretto al solo amministratore di sistema di questa macchina permette di operare in maniera sicura anche sui file di configurazione dei filtraggi. Non sempre il gateway è già configurato adeguatamente, un altro obiettivo sarà quindi quello di permettere

l'accesso ad internet alle macchine della LAN, bloccando opportunamente tutte le connessioni di tipo SMTP o POP3 non specificate nel file di configurazione.

Per rendere il sistema pienamente sfruttabile e di facile utilizzo occorre particolare attenzione alle scelte progettuali legate alle interfacce, sia per l'utente che per l'amministratore. Queste devono necessariamente essere il più intuitive possibili e limitare allo stretto indispensabile le operazioni che chi sovrintende al sistema deve gestire. Altro requisito fondamentale è quello di un'elevata personalizzazione del servizio di protezione richiesto. In particolare sarebbe preferibile permettere solo le connessioni di posta segnalate in un file di configurazione e bloccare tutte le altre. Allo stesso tempo però dovrà essere garantito l'accesso multiplo contemporaneo ai servizi email permessi.

Una delle priorità fondamentali è rappresentata dalla compatibilità, questo software deve essere in grado di lavorare con la maggior parte dei server e dei client disponibili oggi in internet. Rendere il sistema compatibile vuol dire crearlo in modo che sia il più "trasparente" possibile agli occhi di tutti coloro che lavorano sulla singola connessione. In particolare né il server di posta né il client utilizzato per l'invio o la ricezione delle email devono accorgersi che la loro comunicazione è sottoposta a controllo. L'unico momento in cui il sistema antispam ed antibot deve agire è quando rivela qualcosa che per lui non è lecito o che l'utente vorrebbe bloccare.

L'obiettivo vero e proprio resta comunque quello del filtraggio e bloccaggio, per essere effettuato nella maniera migliore occorre che tutti gli utenti finali possano segnalare le loro impostazioni, specie riguardo cosa per loro è spam e cosa non lo è. Il sistema dovrà quindi prevedere metodi per consentire le segnalazioni da parte degli utenti e dall'altra parte mezzi per permettere l'apprendimento del software che migliori di conseguenza i propri filtraggi. Errori di cattiva classificazione delle email sono comunque sempre possibili come si è visto nei capitoli precedenti (*falsi positivi* e *falsi negativi*), tuttavia è auspicabile che questi si presentino in quantità esigua rispetto alla totalità delle analisi.

Quando si verifica un errore deve essere sempre garantita la possibilità da parte dell'utente di segnalarlo e di poter in ogni caso ricevere o inoltrare la lettera in questione.

4.2. Strumenti utilizzati

Analizzando il panorama delle macchine utilizzate per fornire l'accesso ad internet con un qualche sistema di protezione, si può notare che la maggioranza di queste si basa su sistemi operativi *Unix like*, come *Linux* o *FreeBSD* con qualche eccezione per i sistemi *Windows*. Spesso viene fatta la scelta di utilizzare sistemi linux per vari fattori come la gratuità essendo software open source, ma al tempo stesso la forte attitudine alla gestione del networking apprezzata dagli amministratori.

Alla luce di quanto visto si è scelto di sviluppare il prototipo del sistema su di una macchina linux, in particolare utilizzando un computer con installato *Ubuntu 7.10*, una delle *relase* più diffuse, basata sulla ormai storica distribuzione *Debian*. Allo stesso tempo si è cercato durante l'implementazione di dare importanza alla portabilità verso i sistemi Windows, mantenendo il più possibile le operazioni lineari per permettere anche successivi adattamenti da realizzare su veri e propri router. Per questo si è optato per l'utilizzo del linguaggio *Perl*.

Perl è un linguaggio di programmazione ad alto livello, dinamico, procedurale ed interpretato, creato nel 1987 da Larry Wall. Perl ha un singolare insieme di funzionalità ereditate da: C, scripting shell Unix (sh), awk, sed ed in diversa misura da molti altri linguaggi, compresi alcuni di quelli funzionali. Questo lo rende molto potente per operare sul testo di cui sono costituiti i messaggi email, e al tempo stesso permette il supporto sia del paradigma procedurale che di quello ad oggetti, senza contare che si basa su di una delle più

ampie comunità di utenti che nel tempo ha consentito di creare una vasta collezione di moduli liberamente utilizzabili.

Perl è comunemente ritenuto un linguaggio interpretato, ossia che per essere eseguito viene tradotto al momento dell'esecuzione. In realtà, la prima cosa che fa l'interprete è di trasformare il codice sorgente in bytecode, un po' come il Java; dal bytecode crea un grafo intermedio sul quale applica delle ottimizzazioni, ed è proprio questo grafo ad essere interpretato. Questo approccio permette di limitare la lentezza tipica dei linguaggi interpretati rendendolo abbastanza agile durante l'esecuzione. Altro punto di forza del Perl è la sua possibilità di interagire con gli altri linguaggi e ambienti di sviluppo, siano essi: dialetti shell, altri linguaggi interpretati, linguaggi specializzati (come l'SQL), o i più comuni linguaggi compilati. Per lo sviluppo del prototipo si è usata una delle ultime versioni disponibili di Perl (v. *5.8.8*) appresa da testi quali *O'Reiliy Perl Cookbook* [35] e *Pro Perl* [36]. Un'importante annotazione riguarda i *thread*, questi particolari processi sono supportati dal Perl solo nelle sue ultime versioni, pertanto si suggerisce di controllare il numero della propria copia dell'interprete.

Per l'amministrazione dei numerosi dati si è reso necessario l'utilizzo di un DBMS. La scelta è ricaduta su *MySQL*, soprattutto vista la possibilità di essere utilizzato con la licenza GNU GPL che lo rende gratuito ma anche la sua alta stabilità, potenza, velocità e soprattutto diffusione ed usabilità. Le funzioni molto importanti di cui dispone MySQL, che sono alla base di un sistema come quello che ci si accinge a descrivere sono: la possibilità di operare in regime di multithreading, permettendo di realizzare accessi esclusivi alle tabelle e anche la possibilità di operare con linguaggi diversi come il PHP utilizzato per l'interfaccia utente. La versione di MySQL adoperata nella realizzazione del programma è la *5.0.51*.

Per consentire l'interazione dell'utente con il sistema si è scelto di utilizzare un sito web dinamico appositamente realizzato. Questo approccio porta il vantaggio di non dover installare nulla sulle macchine della LAN e di poter

dialogare con l'applicativo indipendentemente dal sistema operativo installato sul singolo elaboratore. Un altro vantaggio di questo modo di operare è la maggior sicurezza in quanto la macchina dell'utente potrebbe essere preda di infezioni bot mentre il gateway risulta sempre protetto, come dalle ipotesi fatte a priori.

La realizzazione del server web è piuttosto semplice ed implica l'utilizzo di *Apache* il software più diffuso su internet per svolgere questo compito. Anche questo programma è open source e multipiattaforma, così da rispettare ancora gli obiettivi prefissi. Il server web verrà installato sulla macchina gateway così come MySQL ed il linguaggio che permette di realizzare le pagine web dinamiche ossia *PHP*.

PHP è un linguaggio di scripting interpretato, il suo nome è un acronimo ricorsivo che sta per *PHP: Hypertext Preprocessor* (PHP: preprocessore di ipertesti). Nato da pochi anni, ha visto crescere costantemente la propria popolarità portando nel 2005 la configurazione LAMP (Linux, Apache, MySQL, PHP) a superare il 50% del totale dei server sulla rete mondiale. Questo linguaggio consente la piena interazione con il database MySQL, sfruttato anche dal Perl per la parte del programma principale. PHP inoltre permette di realizzare semplicemente tutte le pagine di interfaccia per l'utente in maniera dinamica, riducendone di molto il numero. La versione usata durante lo sviluppo di questo software è la *5.2.4*.

In questa sezione non si accenna a nessun browser web né client di posta, in quanto il sistema è stato strutturato in modo da funzionare con tutti i client e browser più utilizzati attualmente sul mercato. Sono stati fatti test con vari software e l'obiettivo sembra essere stato rispettato anche se non si garantisce l'assenza di eventuali errori dovuti a particolari implementativi non specificati dagli standard che alcuni software potrebbero utilizzare.

4.3. Bogofilter

Continuando nella lista dei software utilizzati per lo sviluppo del prototipo anti-spam/anti-bot, si riporta la sezione dedicata a *Bogofilter* [37]. Tale programma verrà utilizzato per compiere i calcoli più difficoltosi e complessi, cioè quelli relativi al filtraggio probabilistico sui contenuti delle email. In rete si trovano diversi applicativi simili a questo che funzionano più o meno bene. La scelta è caduta su Bogofilter per diverse ragioni:

1. Tipo di algoritmo usato: E' un filtro bayesiano implementato secondo le linee indicate da *Paul Graham* [20][21]. In particolare Bogofilter usa un'evoluzione del metodo standard basandosi sull'algoritmo per la media geometrica di *Gary Robinson* [38]. Inoltre si avvale della modifica del metodo di *Fisher* e della tecnica del *chi quadro inverso*.

2. Linguaggio di programmazione: Seppure il perl risulti veloce rispetto ad altri linguaggi interpretati, come visto nella sezione precedente, non può essere paragonato in termini di rapidità nei confronti dei linguaggi compilati. Proprio la scelta del C rende questo programma più veloce rispetto alla concorrenza che usa il Perl come *Spamassassin*.

3. L'autoapprendimento: E' possibile istruire l'applicativo su cosa è spam e cosa non lo è ad ogni email analizzata, permettendo di riclassificare anche le lettere erroneamente segnalate.

4. L'analisi: Il software riconosce già tutti i campi da analizzare delle email, identificando anche lo standard MIME, e tralascia tutte le parti dei messaggi non utili alla classificazione.

5. Il database: Bogofilter ha un proprio database dal nome *wordlist.db* organizzato secondo il *Berkeley DB*. Essendo tutto contenuto in un unico file risulta semplice sia la sua eliminazione in caso di corruzione,

sia il suo rimpiazzamento, in più consente ricerche veloci ed ottimizzate.

6. La configurabilità: La configurazione e personalizzazione del programma avviene in modo in maniera piuttosto intuitiva. In particolare si può agire semplicemente facendo seguire al nome del software le specifiche richieste nella riga di comando.

7. Basso numero di errori: Secondo i test riportati dai programmatori di Bogofilter questo garantisce un basso numero di falsi positivi e negativi.

A dimostrazione della validità della scelta fatta si riporta una prova comparativa tra i due maggiori concorrenti nel campo del filtraggio probabilistico dei contenuti delle email, Bogofilter e Spamassassin, dal nome *Spam avoidance techniques* [39]. Il test è stato compiuto ricevendo email per due settimane alla casella *lwn@lwn.net*, 3.000 in tutto, di cui 2.705 di spam. Al termine tutti i messaggi sono stati analizzati una volta con Spamassassin e due con Bogofilter, la prima dopo averlo fatto apprendere con il 15% delle 3.000 lettere e la seconda dopo un training con tutte le email.

Filtro	**Falsi Positivi**	**Falsi Negativi**	**Run Time (sec)**
Spamassassin	2	250	11900
Bogofilter (15%)	0	517	108
Bogofilter (100%)	0	94	134

Tabella 4.1 – Comparazione analisi su 3000 email (fonte *Spam avoidance techniques*)

Successivamente sono stati analizzati con i due filtri 5.000 *post* recenti sulla lista di *linux-kernel*, di cui 12 erano spam. In tabella 4.2 si riportano i risultati:

Filtro	Falsi Positivi	Falsi Negativi	Run Time (sec)
Spamassassin	0	6	19600
Bogofilter	0	4	251

Tabella 4.2 – Comparazione analisi su 5000 post (fonte *Spam avoidance techniques*)

Si Ricorda che i falsi positivi sono le email legittime classificate come spam, sono gli errori peggiori perché implicano perdite di messaggi buoni. I falsi negativi invece rappresentano lo spam identificato come posta legittima. Entrambi i filtraggi tendono ad avere giustamente più falsi negativi rispetto a quelli positivi, in quanto errori meno gravi. In particolare Bogofilter non commette nessun errore di classificazione identificabile come falso positivo durante l'intero test questo è già un ottimo risultato rispetto al concorrente. Seppure all'inizio del confronto il numero di falsi negativi sia più che doppio rispetto a Spamassassin, se viene eseguito un corretto periodo di apprendimento si osserva un enorme miglioramento delle prestazioni riportando meno di 100 falsi negativi ripetto ai 250 del concorrente.

Il periodo di training risulta quindi fondamentale per bogofilter in quanto gli permette di adattarsi al traffico ricevuto dal suo utilizzatore. *Eric Raymond* (l'ideatore di Bogofilter) consiglia di collezionare almeno un migliaio di messaggi di spam e altrettanti legittimi per poter ottenere risultati soddisfacenti durante l'uso. D'altra parte più lettere si usano nel training più il database crescerà e quindi occorrerà maggior tempo per eseguire le ricerche in esso, facendo aumentare il tempo complessivo di analisi.

Riguardo al tempo di analisi si possono verificare le maggiori differenze tra i due applicativi. Spamassassin risulta essere notevolmente più lento del rivale quasi 100 volte! La causa di tale differenza sta in gran parte nel linguaggio di programmazione utilizzato. Si ricorda che Spamassassin è stato implementato in Perl mentre Bogofilter in C. Altro fattore che spiega la netta differenza di prestazioni tra i due software è la scelta dell'algoritmo, l'uso delle ultime teorie in

campo probabilistico e l'elevata ottimizzazione rispetto alla sua velocità computazionale sono il punto di forza di bogofilter.

Capitolo 5

Bloumail

Si è deciso di chiamare l'applicativo *BLOUMAIL* derivante dall'acronimo *BLOck Unwanted MAIL*. Il suo scopo principale come indicato dal nome stesso è quello di bloccare tutta la posta elettronica indesiderata, sia essa entrante o uscente dalla propria rete.

Figura 5.1 – Logo di Bloumail

Bloumail nasce come completamento di *BLOBOT* [40], software realizzato dalla collega *Rossella Candela*, come tesi di laurea. Blobot si dedica in particolare al filtraggio delle pagine web presentando un sistema basato su diverse liste di indirizzi di siti internet. Il suo scopo è quello di prevenire il traffico generato da un bot che potrebbe infettare il sistema durante la normale navigazione dei siti

web. In conclusione gli applicativi Blobot e Bloumail si propongono di lavorare insieme per permettere una piena protezione della propria rete da minacce derivanti dalle botnet. A tal scopo e con l'ambiziosa idea di poter essere utilizzato dal più vasto bacino di utenza possibile si è pensato di realizzare tutto il codice con commenti e testo di output in lingua Inglese.

5.1. Struttura di Bloumail

Bloumail si compone di tre parti fondamentali che interagiscono e collaborano tra di loro al fine di realizzare compiti quali: filtraggio, bloccaggio, autenticazione utente, autorizzazione all'invio, ecc.

1. Programma principale: Questo modulo è realizzato in Perl e viene mandato in esecuzione dall'amministratore di rete sulla macchina gateway trasformandola in un proxy che analizza il traffico email, dopo una opportuna installazione e configurazione.

2. Interfaccia web: Accessibile agli utenti che si trovino nella rete LAN, realizzata in PHP, permette le varie operazioni dopo una opportuna autenticazione.

3. Database: Realizzato in MySQL permette l'immagazzinamento dei vari dati necessari al filtraggio e all'autenticazione, consentendo la gestione della concorrenza e la velocizzazione delle ricerche necessarie sui valori contenuti.

Il filtraggio viene eseguito in maniera asimmetrica per quanto riguarda la posta in uscita rispetto a quella entrante. Le idee alla base delle due realizzazioni sono le stesse e sono il risultato dell'analisi presentata nei capitoli precedenti. Uno

dei filtraggi che fornisce la maggior garanzia di protezione è quello eseguito sul contenuto. Proprio per questo è stato scelto come cuore del sistema il filtraggio di Bayes, aiutato nella sua azione da una blacklist di indirizzi.

La tecnica di blacklisting è stata preferita a quella del whitelisting perché quest'ultima potrebbe rivelarsi un'arma a doppio taglio, infatti nessuno può sapere a priori se gli indirizzi "amici" presenti in tale lista siano realmente insensibili ad un attacco. Ad esempio un'infezione da parte di un virus potrebbe insorgere in qualsiasi momento anche per una piccola distrazione dell'utente di quel sistema. D'altra parte invece si può sapere chi è quasi certamente infettato mediante un riscontro sullo SPAM che precedentemente ha inoltrato.

Con questo sistema, vengono bloccate in uscita le email con indirizzi TO elencati nella lista, mentre per quelle in ingresso gli indirizzi da confrontare sono quelli elencati nel campo FROM. Già da solo questo filtraggio, se tenuto costantemente aggiornato da parte degli utenti riportandovi all'interno ogni indirizzo dal quale si riceve abitualmente spam, può essere un valido rimedio al problema della posta indesiderata. Tuttavia gli *attacker* cambiano spesso i loro indirizzi per questo risulta necessario affiancare al sistema descritto un controllo sui contenuti dei messaggi. Tutto questo è utilizzato per la posta in ingresso, per quella in uscita si è anteposto un altro tipo di filtraggio.

Ogni lettera in uscita viene sottoposta ad un algoritmo di *digest* di tipo *MD5* che fornisce una sorta di impronta digitale dell'email. La scelta di MD5 è dovuta alla sua velocità rispetto ad altre funzioni *hash* ed al suo largo utilizzo in varie implementazioni software sul mercato. Il digest viene calcolato sui campi *header*: *from* e *to,* concatenati alla parte leggibile del corpo del messaggio. Tale decisione è legata al fatto che si vuol far ricadere tutta una certa categoria di messaggi in un unico digest, in particolare tutti quelli che hanno stesso corpo leggibile ed indirizzi. Se si fossero presi tutti i campi header di due messaggi apparentemente identici non avrebbero avuto stesso digest in quanto alcuni di questi campi

vengono modificati dal client ad ogni invio ad esempio l'ora di inoltro talvolta imposta.

Un filtraggio sugli identificativi delle email consente di evitare le altre analisi qualora la lettera stessa, o una simile, fosse già stata bloccata in precedenza. In più semplifica di molto la tabella dei messaggi bloccati rendendola più snella e di facile consultazione. L'espediente del digest è legato al fatto che gli standard di posta elettronica non delegano come compito obbligatorio al client la creazione di un identificativo del messaggio, anche se alcuni di questi tuttavia autonomamente lo generano. Per estrapolare la parte leggibile delle email si è realizzato un algoritmo apposito per la scansione dell'email in formato MIME che in più eliminasse anche la maggior parte dei *tag* HTML presenti nei testi di alcune email prodotte da client particolari come ad esempio *Outlook*.

5.2. Il Database

Il database contiene tutte le informazioni necessarie al funzionamento di Bloumail. In particolare rappresenta anche il ponte di collegamento tra il programma principale ed il sito web di interfaccia utente. Altro compito di questa struttura è quello di garantire la consistenza dei dati durante l'utilizzo concorrente da parte di più thread. Il database di Bloumail è costituito in particolare da tre tabelle senza nessuna particolare reciproca relazione. Di seguito si elencano descrivendo le varie tabelle ed i loro scopi.

1. *Allow*: Questa tabella serve ad immagazzinare tutti i messaggi che sono stati bloccati dal sistema di filtraggio in uscita. L'altra funzione essenziale che svolge è consentire la successiva autorizzazione alla spedizione dei messaggi in essa contenuti.

ALLOW

msgid	auth	captcha	try	expire

Tabella 5.1 – Tabella ALLOW

msgid: Identificativo dell'email, digest MD5 di alcuni campi dell'email.

auth: Può valere 0 se il messaggio non ha l'autorizzazione per essere inviato o 1 viceversa (di default viene messo a 0 ad ogni inserimento di un nuovo msgid).

captcha: Stringa di cinque caratteri random che l'utente deve fornire attraverso l'interfaccia web per autorizzare l'invio della lettera (cambia sia ad ogni visualizzazione che ad ogni tentativo).

try: Numero di tentativi che l'utente può ancora effettuare prima che non sia più possibile autorizzare l'email (ad ogni nuovo inserimento di msgid viene impostato di default a 3).

expire: Data e ora in cui l'intera entry non vale più e deve essere rimossa, sia essa autorizzata o meno (di default si ha un giorno per autorizzare l'email ed inviarla prima che venga rimossa dalla tabella).

2. *Blacklist*: Questa tabella è composta da un solo campo, *address*. Non è altro che un elenco di indirizzi email che sono considerati pericolosi perché possono appartenere a *spammer* o a sistemi bot.

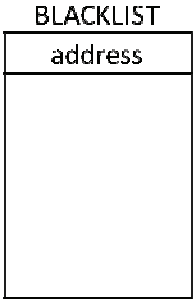


Tabella 5.2 – Tabella BLACKLIST

3. *Login*: Questa tabella mantiene l'elenco di tutti gli utenti che possono autenticarsi, consentendo loro di modificare la blacklist o di riclassificare un'email. L'elenco di user ID e password può essere riempito o variato solo dall'amministratore del sistema. Questo consente di avere sempre sotto controllo il numero di utenti che hanno i privilegi per poter modificare le caratteristiche dei filtraggi, limitando possibili attacchi.

   La riclassificazione è un operazione molto importante perché consente di mantenere sempre aggiornato il database usato dal filtro di Bayes. Ogni messaggio classificato come SPAM o GOOD entra a far parte delle statistiche del filtro, è quindi essenziale che nel caso in cui il filtro operi una classificazione errata venga corretto dall'utente segnalando qual'é la vera famiglia a cui appartiene il messaggio.

   Per compiere questa operazione il sistema provvede a generare su richiesta due *mailbox* cioè due particolari indirizzi email riconosciuti da Bloumail, ai quali è possibile inviare una copia del messaggio per poterlo riclassificare. Di seguito si elencano descrivendo i vari campi ed il loro scopo.

LOGIN					
user	pass	captcha	try	expire	mbox

Tabella 5.3 – Tabella LOGIN

user: Identificativo dell'utente.

pass: Password usata per autenticare l'utente.

captcha: Stringa di cinque caratteri random che l'utente deve fornire attraverso l'interfaccia web per poter compiere un operazione (cambia sia ad ogni visualizzazione che ad ogni tentativo).

try: Numero di tentativi che l'utente può ancora effettuare prima che non sia più possibile eseguire operazioni per un giorno (di default viene reimpostata a 3 dopo ogni captcha risolta correttamente).

expire: Data e ora fino a cui, in caso di 3 tentativi consecutivi sbagliati di risolvere la captcha, per l'utente non è più possibile compiere operazioni. Ogni volta che si inserisce correttamente una captcha o si supera il tempo di expire precedente, il campo viene risettato ad un giorno dall'ora attuale.

Mbox: Identifica due mailbox che consentono di ricevere le email da riclassificare. Ha durata fino allo scadere di expire (di default un giorno). Una volta ricevuto un messaggio su di uno dei due indirizzi vengono entrambi eliminati settando a NULL questo campo.

L'indirizzo avrà la struttura seguente:

USER-ID.STRINGA-RANDOM@BLOUMAIL.SYS
Ad esso saranno anteposte le stringhe SPAM e GOOD per permettere le riclassificazioni, ad esempio per reimpostare un'email che era stata rilevata come spam, nella categoria della posta legittima basta inoltrare una copia della stessa alla casella:

GOOD.USER-ID.STRINGA-RANDOM@BLOUMAIL.SYS.
La scelta di questo particolare formato degli indirizzi è legata al fatto di rendere il più difficile possibile un attacco di *bruteforce* sulle mailbox.

5.3. Il programma principale

Si analizzano ora i dettagli implementativi del programma principale, questo si compone di tre classi, ciascuna con compiti diversi, incluse in un file di *main*:

1. *bloumail*: Corrisponde al vero e proprio cuore dell'intero sistema, contiene il *main* ed include tutti i file seguenti comprese anche altre funzioni di libreria del Perl.

2. *email.pm*: Classe che permette di gestire i messaggi email come veri e propri oggetti con tutti gli attributi ed i metodi utili per le elaborazioni successive.

3. *toclient.pm*: Classe che si occupa della comunicazioni lato client, permette cioè il dialogo tra Bloumail ed i client di posta nella rete locale.

4. *toserver.pm*: Classe che si occupa delle comunicazioni lato server, permette cioè il dialogo tra Bloumail ed i server di posta disponibili su internet.

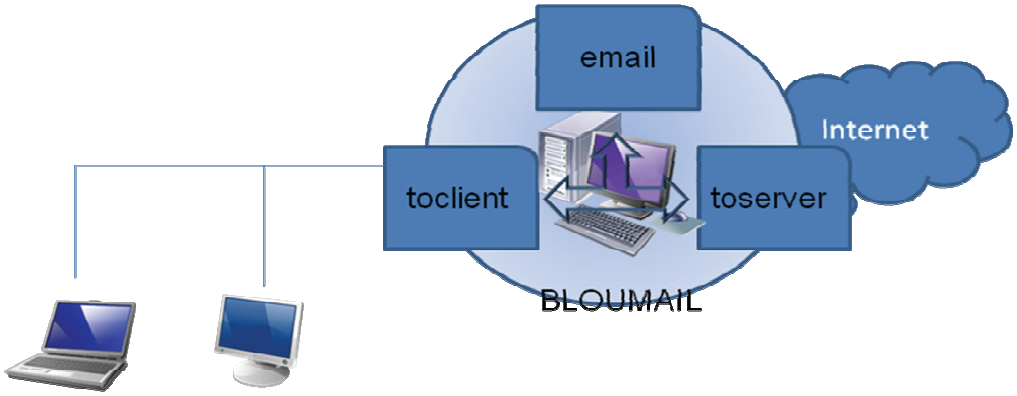


Figura 5.2 – Struttura del programma principale di Bloumail

Di seguito si riportano le funzioni che si sono rese necessarie costruire per gli oggetti appena elencati, insieme agli attributi indispensabili per lavorare con essi.

1. Attributi `email.pm`:

   `from`: Array contenente gli indirizzi mittenti.

   `to`: Array contenente gli indirizzi destinatari.

   `bound`: Array contenente eventuali boundary MIME.

   `ctype`: Conterrà durante l'analisi dei campi MIME i *content type*.

   `encode`: Conterrà durante l'analisi dei campi MIME le codifiche.

   `bodyfile`: File temporaneo contenente il solo corpo leggibile del messaggio.

   `msgfile`: File temporaneo contenente l'intero messaggio.

Metodi `email.pm`:

`new()`: Costruttore degli oggetti email, alloca lo spazio necessario.

`parseadd(stringa)`: Analizza una stringa riconoscendo se contiene un indirizzo email ed eventualmente lo ritorna ripulito dal contesto.

`setfrom(from)`: Inserisce un indirizzo mittente nell'array *from*.

`setto(to)`: Inserisce un indirizzo destinatario nell'array *to*.

`reset()`: Ripulisce gli attributi dell'oggetto.

`getemailfile()`: Ritorna il file contenente il messaggio.

`getbodyfile()`: Ritorna il file contenente il corpo leggibile.

`getfrom()`: Ritorna il vettore dei mittenti.

`getto()`: Ritorna il vettore dei destinatari.

`headerparse()`: Analizza il messaggio contenuto in *msgfile* ed inizializza gli attributi *from* e *to*.

`bodyparse()`: Analizza il messaggio contenuto in *msgfile* ed inizializza l'attributo *bodyfile*, in particolare salta l'header e guarda i singoli campi MIME se contengono testo e nel caso li decodifica opportunamente rimuovendo anche alcuni tag HTML.

`getmsgid()`: Ritorna un identificativo per il messaggio eseguendo un MD5 digest sull'aggregazione di: *from, to, bodyfile.*

`getmime()`: Analizza il singolo campo MIME inizializzando opportunamente gli attributi *bound, ctype, encode.*

`markspam(stringa)`: Provvede ad inserire all'inizio del campo *subject* dell'email originale la stringa passata, in particolare crea il *subject* se non esiste o nel caso dopo la concatenazione superi i 78 caratteri definiti dallo standard, lo frammenta su più linee.

2. Attributi `toclient`:

`sockcon`: Socket in attesa di connessioni dei clienti.

`interface`: Indirizzo IP dell'interfaccia in attesa di connessioni.

`port`: Numero di porta in attesa di connessioni.

`prot`: Tipo di protocollo di comunicazione (SMTP o POP3).

`peeraddr`: Indirizzo IP del cliente connesso.

`sockcom`: Socket per le comunicazioni con il cliente.

Metodi `toclient`:

`new(indirizzo,porta)`: Costruttore oggetto *toclient*, apre il socket in attesa di connessioni con i clienti.

`accept()`: Attende la connessione di un cliente e apre un nuovo socket per comunicare con esso.

`getpeeraddr()`: Ritorna l'indirizzo IP del cliente connesso.

`msgtoclient(messaggio)`: Invia un messaggio al cliente.

`mailtoclient(file)`: Invia un'email contenuta in un file al cliente.

`getcommand()`: Legge un comando del cliente.

`resetcom()`: Resetta gli attributi *peeraddr* e *sockcom*.

`fetchemail(file)`: Salva in un file l'email inviata dal client.

3. Attributi `toserver`:

`sock`: Socket per la connessione col server.

`interface`: Indirizzo IP del server.

`port`: Numero di porta del server.

`prot`: Tipo di protocollo di comunicazione (SMTP o POP3).

`timeout`: Tempo entro il quale è possibile stabilire una connessione.

Metodi `toserver`:

`new(indirizzo,porta)`: Costruttore oggetto *toserver*, apre il socket per comunicare con il server.

`msgtoserver(messaggio)`: Invia un messaggio al server.

`mailtoserver(file)`: Invia un'email contenuta in un file al server.

`getresponse()`: Ascolta la risposta del server e la ritorna.

`fetchemail(file)`: Salva in un file l'email inviata dal server.

Il file principale di tutto il sistema è *bloumail*. Il codice di questa parte è modulare, composto da diverse funzioni, si riportano per punti le operazioni principali compiute nelle varie fasi di esecuzione.

1. Analisi dei parametri riportati in linea di comando ed in base a questi agire di conseguenza, cioè far partire o arrestare l'esecuzione del programma ed utilizzare la modalità specificata. Allo stesso tempo avviene una verifica circa l'esistenza di un'altra istanza di esecuzione dello stesso programma, operazione non permessa.

2. Caricamento del file di configurazione, viene controllata l'esistenza dello stesso e la sua correttezza. In seguito il file viene scandito e si procede ad inizializzare le strutture dati globali che conterranno i parametri utili. Allo stesso tempo si provvede a ripulire la tabella *allow* dalle entry scadute.

3. Salvataggio delle *iptables* in un file di backup e loro sostituzione. In base alla modalità utilizzata le *iptables* possono essere interamente sostituite o alternativamente è possibile semplicemente aggiungere ad esse le direttive di Bloumail. Alla fine di queste operazioni la macchina permetterà di far accedere ad internet i calcolatori delle reti locali specificate (possono essere più di una) e provvederà a ridirigere il traffico email a Bloumail attraverso una operazione di *natting* sulle porte utilizzate dai server SMTP e POP3. Come ultima operazione verranno bloccate tutte le comunicazioni email delle LAN che non sono state specificate nel file di configurazione.

4. Creazione degli oggetti *toclient* richiesti dal file di configurazione. Questi oggetti posseggono ognuno un socket in ascolto su di un indirizzo IP e porta specifici, attendono la connessione dei clienti di posta, che girano sulle macchine della LAN controllata.

5. Creazione dei figli necessari a gestire le connessioni. Il programma viene frammentato in un insieme di figli ognuno dei quali gestirà la comunicazione tra i client ed un particolare server di posta. Possono esserci più figli che si occupano di un unico server di posta consentendo di poter compiere accessi multipli simultanei ad un unico server da parte di più utenti della LAN.

   In principio si era adottato il metodo del sistema multiprocesso compiendo varie *fork* nel padre. Successivamente come evoluzione, grazie soprattutto al supporto al *multithreading* offerto dalle ultime versioni del Perl, si è convertita la struttura rendendola meno onerosa in termini di consumo di memoria e cambi di contesto.

6. Il *thread* padre si mette in attesa della terminazione di uno qualunque dei figli. Il programma è stato strutturato in modo che nessun figlio dovrebbe mai terminare se non quando viene arrestato Bloumail. In caso di errori gravi è comunque possibile che un *thread* finisca la propria esecuzione inaspettatamente, questa eventualità, seppur remota, deve essere gestita in modo da ricreare immediatamente un figlio che si metta in attesa sullo stesso socket del *thread* scomparso.

   Per fare quanto detto è stato usato un tipo particolare di variabile condivisa tra i *thread*, questa consente il sincronismo tra i processi mandando in esecuzione il padre ogni qual volta un figlio segnali la sua terminazione. In più con la stessa variabile è stato possibile associare un indice che serve ad identificare il socket sul quale il thread stava lavorando.

7. I *thread* figli possono provvedere a salvare un file di log di tutte le comunicazioni che avvengono verso un dato server e tutte le operazioni che Bloumail compie in quell'ambito. Se tale opzione è specificata dal file di configurazione il figlio provvede a creare il file in una apposita

cartella. Se più thread operano sulle comunicazioni di uno stesso server è possibile che operino anche sullo stesso file di log.

La mutua esclusione è gestita in automatico su questi file, riga per riga, nel caso in cui si voglia stampare più linee consecutive ed impedire agli altri di scrivere nel mezzo tra una e l'altra si può ricorrere alla funzione *flock* disponibile nelle librerie del Perl. Si è preferito eliminare il *buffering* dell'output sui file di log in modo da avere subito una scrittura su di essi anche qualora si verificassero errori.

8. Quando un *thread* figlio riceve una connessione da un client esce dalla fase di attesa passiva e instaura subito una connessione con il server che gestisce. A questo punto a seconda che il server sia del tipo SMTP o POP3 viene mandata in esecuzione un'apposita funzione, cioè rispettivamente: *chattingsmtp* o *chattingpop*. Queste si occupano di permettere l'interazione tra il client ed il server. Le procedure prevedono il trattamento di tutti i comandi descritti dagli standard SMTP e POP3, e sono state testate con vari client e server di posta accessibili su internet non presentando mai particolari problemi di incompatibilità.

   Quando possibile Bloumail cerca sempre di essere "trasparente", qualora sia possibile, infatti si limita il più delle volte semplicemente a ricopiare ciò che dice uno degli interlocutori all'altro. Quando vengono rilevati comandi o risposte particolari il programma va ad inizializzare le sue strutture dati. Questo approccio consente di limitare al minimo la gestione delle eccezioni agli standard SMTP/POP3 e degli errori. Praticamente sono server e client stessi che vigilano sulla correttezza del dialogo che intrattengono.

   L'unico vero momento in cui il proxy smette di essere trasparente è quando riceve un'email, sia essa proveniente dal server o dal client. A

quel punto la lettera non viene subito propagata all'interlocutore ma si provvede a salvarla in un file temporaneo, e ad eseguire su di essa le analisi richieste nel file di configurazione. Si analizza nel dettaglio cosa avviene nel caso in cui la posta sia in ingresso ed in quello in cui sia in uscita.

- **Posta in ingresso (POP3)**: Di questa parte si occupa in particolare la funzione *chattingpop* che consente di intrattenere il dialogo tra il server ed il client utilizzando il protocollo POP3. Una volta ricevuto tutto il messaggio dal server, Bloumail lo analizza a seconda dei filtri che sono stati abilitati nel file di configurazione.

  Per primo viene eseguito il filtraggio basato sulla blacklist, qualora il messaggio contenga un indirizzo nel campo *from* presente nella tabella *blacklist* questo viene "marcato" inserendo in testa al campo *subject* l'etichetta ***BANNED***. Il filtraggio sul contenuto a questo punto viene saltato e si provvede ad inviare l'email originale modificata al client.

  Se il messaggio non viene segnalato dal blacklisting si passa alla seconda analisi, cioè quella sul contenuto. Per questa verifica si usa l'applicazione *Bogofilter* [37] con delle particolari opzioni: *-t* per segnalare i casi spam/good con sole due lettere Y/N, *-u* per utilizzare il risultato della classificazione per le successive volte, *-d* per specificare il database delle parole, *-o* per assegnare oltre quale probabilità si considera spam, *-I* per leggere l'email da un file. In particolare con queste opzioni è possibile manipolare il filtro in modo che elimini il margine di "insicurezza", così che tutte le lettere possano essere classificate come spam o good senza risultati intermedi ed indeterminati. Allo stesso tempo grazie all'opzione *-u* si può realizzare l'autoapprendimento, così da rendere l'analisi sempre più accurata e mirata sul tipo di posta elettronica che ricevono gli utenti della LAN.

Un'importante annotazione riguarda la modalità con cui viene mandato in esecuzione Bogofilter, anche se in alcuni casi sarebbe possibile avere più istanze del programma che girano in contemporanea, per scelte di progetto si è deciso di limitare ad una singola esecuzione alla volta. Questo modo di agire porta lo svantaggio di un apparente rallentamento dell'analisi in caso di molta posta scaricata in contemporanea. D'altro canto consente di avere sempre sotto controllo la memoria e le risorse di calcolo usate non rendendo mai la macchina proxy sovraccarica. Questo permette di rendere minimo l'impatto di eventuali attacchi di tipo *DoS* sul sistema.

Per effettuare quanto detto si è ricorsi ad un semplice artificio, ovvero porre una sorta di variabile condivisa con le veci di semaforo in modo da gestire la risorsa Bogofilter in mutua esclusione. Questo consente inoltre di mantenere il database delle parole sempre consistente non potendovi accedere più istanze contemporanee dello stesso programma.

Qualora *Bogofilter* rilevi una probabilità che il messaggio sia spam sopra il 90% l'email viene classificata come tale e Bloumail provvede a marcarla come ***SPAM*** nel campo *subject*. Successivamente il messaggio viene inoltrato al client di posta. Se quest'ultimo è stato istruito correttamente potrebbe dirigere automaticamente la posta marcata come *banned* e *spam* in due cartelle appositamente create.

Al termine delle operazioni elencate Bloumail torna nello stato di *chatting* ed è pronto ad esaminare altre richieste del client di posta, questo stato termina quando si riceve un comando di *quit* o si interrompe la comunicazione lato server o lato client. Quando questo accade il thread figlio chiude i socket usati in precedenza e si rimette in attesa di nuove connessioni, lasciando l'esecuzione ai suoi fratelli.

- **Posta in uscita (SMTP)**: Di questa parte si occupa in particolare la funzione *chattingsmtp* che consente di intrattenere il dialogo tra il server ed il client utilizzando il protocollo SMTP. Una volta ricevuto tutto il messaggio dal client, Bloumail lo analizza a seconda dei filtri che sono stati abilitati nel file di configurazione.

  Per l'esattezza i filtraggi e le loro opzioni sono praticamente identici a quelli precedentemente descritti, a cambiare sono solo le operazioni da compiere quando si rileva il messaggio come non desiderato. Altra differenza rispetto alla posta in entrata è la presenza di un filtro anteposto ai due descritti. Questa analisi supplementare lavora sull'identificativo dell'email, descritto nelle sezioni precedenti, ed in modo specifico sulla tabella *allow* del database.

  In effetti quando l'email viene ricevuta dal client, sulla macchina dove gira Bloumail, viene subito passata ad una funzione che ne esegue un *digest* MD5 per poterla identificare. Il *msgid* così costruito viene confrontato con tutti quelli presenti nella tabella *allow* per sapere se si è già tentato di inviare lo stesso messaggio, o uno particolarmente simile, che a suo tempo era già stato bloccato. L'utilità di un approccio simile è quella di limitare la dimensione dell'elenco dei messaggi bloccati da autorizzare e la velocizzazione del filtraggio non dovendo rifare tutti i passaggi da capo.

  Qualora uno dei tre filtri attesti che il messaggio potrebbe essere frutto di una infezione bot, si avvia una procedura allo scopo di segnalare il pericolo all'utente:

  1. Viene inoltrato un codice di errore numero 530, non definito dallo standard ma che provoca il bloccaggio del client di posta dopo una segnalazione.

Figura 5.3 – Avviso proposto dal client di posta Evolution

2. Vengono chiuse le connessioni con il client e con il server di posta.

3. Si provvede a creare una nuova email in formato MIME, ponendo come allegato lo stesso messaggio originale che si è tentato di inviare e riportando un link ad una pagina web che ne permetterà, qualora richiesto, il successivo inoltro.

4. Nel caso di messaggio BANNED, verranno specificati nell'email di notifica anche gli indirizzi che sono stati segnalati dalla blacklist.

5. Viene creata una nuova entry nella tabella ALLOW.

6. Viene aperta una connessione con il server SMTP di default specificato nel file di configurazione.

7. Si procede all'invio del messaggio appena creato all'utente che ha tentato l'inoltro.

Nel caso in cui sia stato il filtro sul *msgid* a segnalare il problema la lettera di notifica non sarà inviata in quanto già inoltrata in precedenza. L'utente ha a disposizione un giorno per autorizzare l'invio delle email presenti nella tabella *allow*,

rispondendo correttamente alla captcha che appare nella pagina il cui link è riportato nel messaggio di notifica.

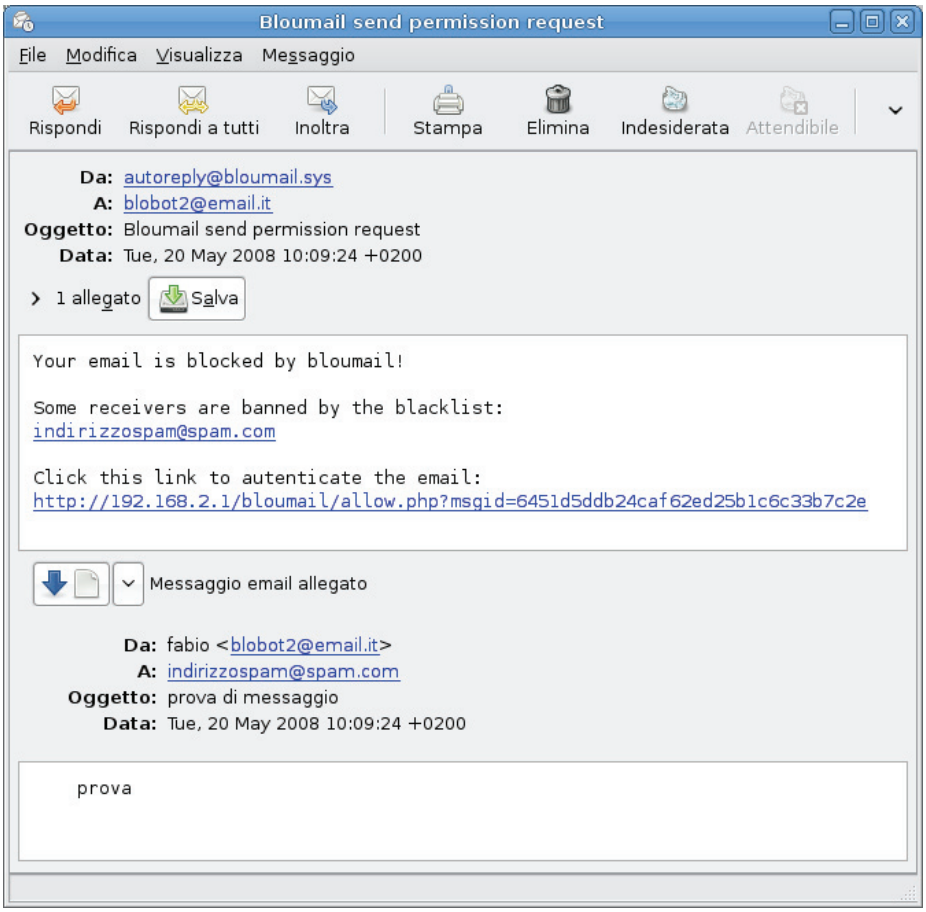


Figura 5.4 – Esempio di email di notifica sul client di posta Evolution

5.4. L'interfaccia web

L'utente che si trova in una rete gestita con Bloumail interagisce in maggior parte con esso attraverso il suo client di posta. Solo quando questo blocca una sua email legittima, o comunque Bloumail effettua una classificazione errata della posta, l'utente si trova nella necessità di doversi interfacciare con il sistema per istruirlo. La scelta di utilizzare il web come interfaccia utente è stata dettata dalla necessità di garantire l'utilizzo su macchine con differenti configurazioni e

potenzialmente unthrusted, quindi senza bisogno di dover installare nulla su di esse.

Le pagine necessarie per eseguire le operazioni di autorizzazione, autenticazione ed istruzione sono molteplici, e quindi alla fine è stato necessario realizzare un vero e proprio sito, accessibile solo dagli utenti della rete LAN.

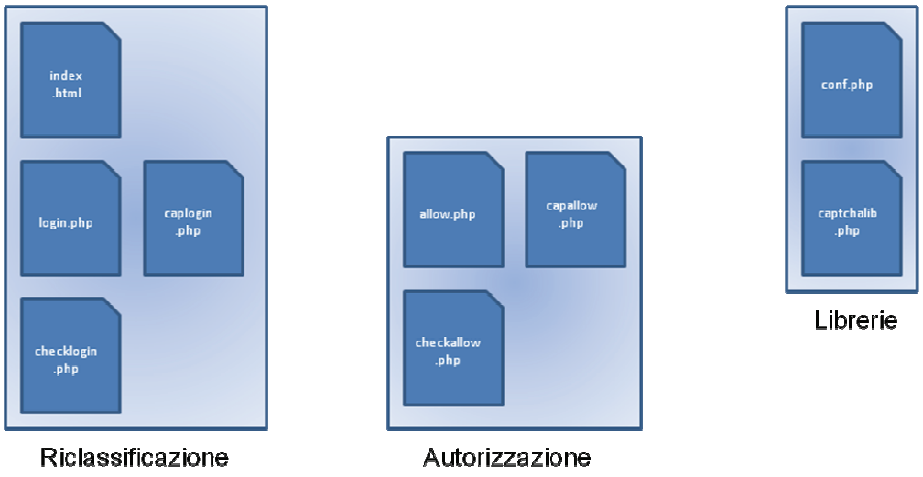


Figura 5.5 – Schema del sito web di interfaccia a Bloumail

Più nel dettaglio, sono stati utilizzati due file PHP allo scopo di fornire le funzioni di libreria necessarie alle altre pagine:

1. *conf.php*: Realizza la connessione al database MySQL di Bloumail.

2. *captchalib.php*: Genera la stringa random che verrà inserita nella CAPTCHA e l'immagine della stessa.

La tecnica CAPTCHA si fonda sull'ovvia considerazione in base alla quale, effettuando un test semplice per un umano ma praticamente impossibile da risolvere per una macchina, si può avere un certo margine di sicurezza sul fatto che un programma non possa autonomamente accedere ad un certo servizio.

L'utilizzo della CAPTCHA è stato determinante proprio per fare in modo che un bot non sia in grado di inoltrare i suoi messaggi di spam, delegando l'eventuale possibilità solo ad un utente consapevole.

Il numero di tentativi messi a disposizione per risolvere il test di autorizzazione all'invio di ogni messaggio è stato ridotto a tre al giorno, ciò dovrebbe limitare le pur già basse probabilità di manomettere il sistema. Altro possibile attacco è quello di bruteforce, per questo si è deciso di cambiare la CAPTHCA ad ogni accesso alle pagina web in cui compaiono. La fase di autorizzazione è formata da tre pagine web. La principale è la *allow.php*, a questa si accede da un link sull'email di notifica bloccaggio, durante il tentativo di inoltro. Alla pagina in questione viene passato con il metodo *GET* il *msgid* dell'email che deve essere autorizzata all'invio.

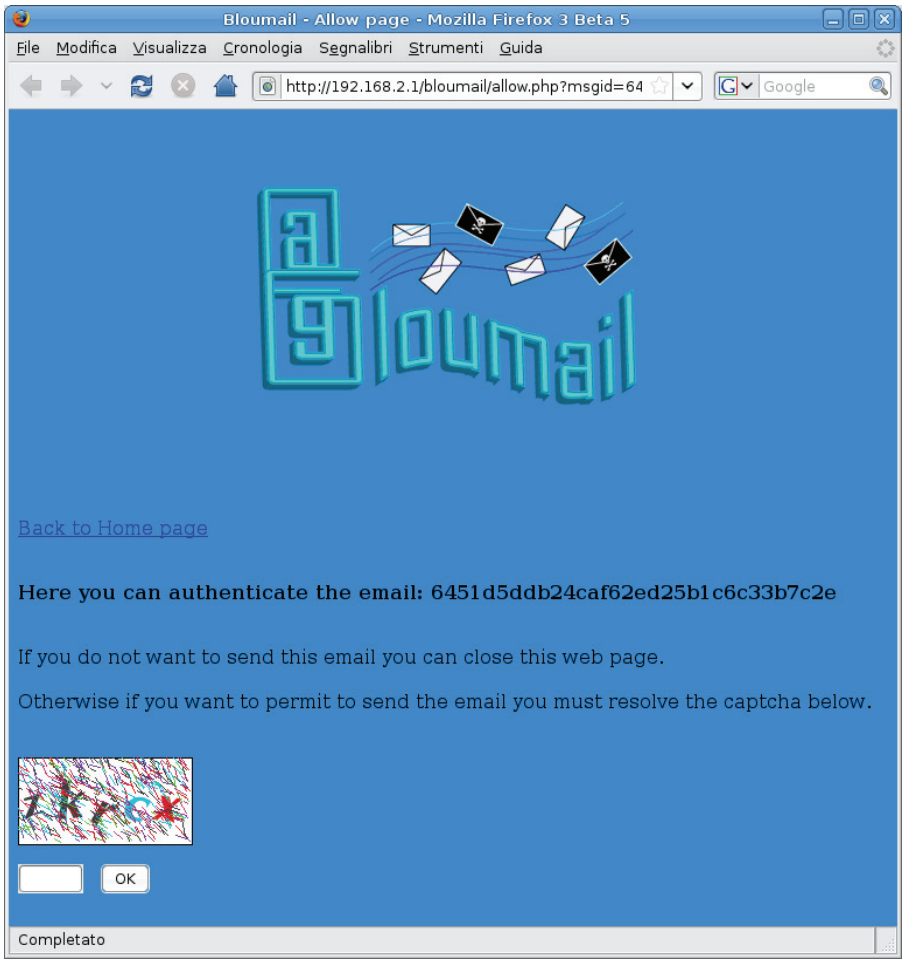


Figura 5.6 – Esempio di pagina per l'autorizzazione all'invio, allow.php

La CAPTCHA viene realizzata dal file *capallow.php* che si occupa anche di modificare nel database la stringa che verrà utilizzata, ed effettua inoltre tutti i controlli e le verifiche necessarie. Per tutte le operazioni di interazione con MySQL e di realizzazione dell'immagine vengono utilizzati i file di libreria. Una nota importante, che interessa tutte le pagine web componenti il sito di Bloumail, riguarda il rispetto della concorrenza e l'attento uso dei lock sulle tabelle sia per quando riguarda la mutua esclusione in lettura che per quella più delicata in scrittura.

Una volta che l'utente risolve correttamente la CAPTCHA viene diretto alla pagina *checkallow.php*. Qui vengono effettuati nuovamente i controlli precedenti circa la possibilità di autorizzare ancora l'email ed in più il confronto con la stringa random. Se tutto va a buon fine viene mostrata una pagina come quella della figura seguente ed è possibile inoltrare nuovamente l'email.

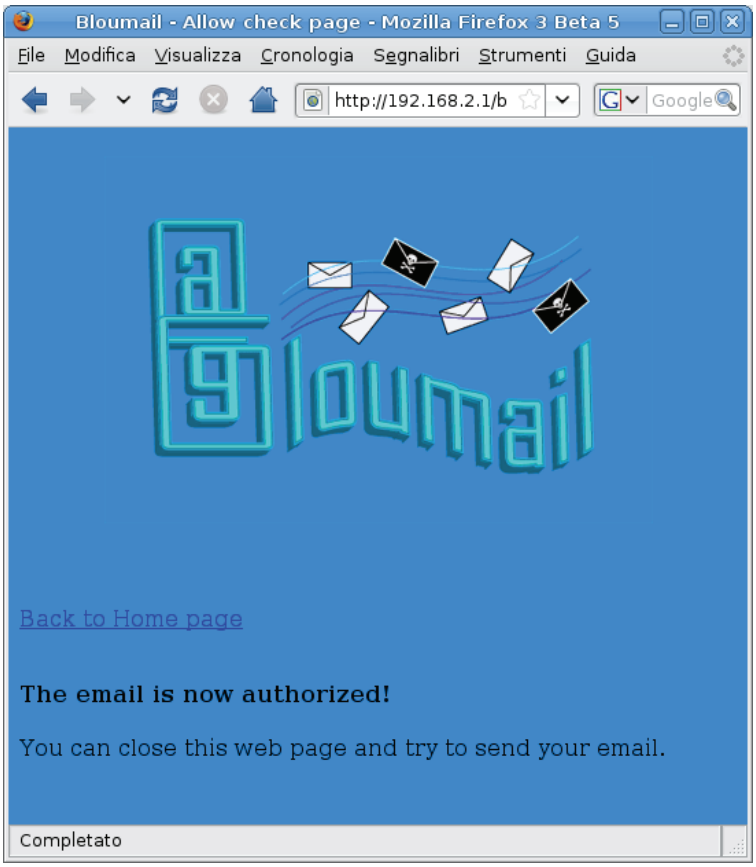


Figura 5.7 – Esempio di corretta autorizzazione, checkallow.php

Se invece la CAPTCHA risulta essere sbagliata, viene decrementato il numero di tentativi che si possono ancora effettuare in quel giorno (*try*) e cambiata la stringa random. Quando il numero di prove raggiunge zero non è più possibile autorizzare l'invio di quell'email per quel giorno ed occorre attendere l'indomani per avere nuovamente tre tentativi utili, dopo aver riprovato ad inoltrare lo stesso messaggio. Le entrate della tabella di allow che sono scadute vengono eliminate sia durante l'avvio di Bloumail che ad ogni test della CAPTCHA garantendo la coerenza dei dati nel database.

L'ultima parte del sito è destinata all'aggiunta o rimozione di indirizzi dalla blacklist ed alla riclassificazione dei messaggi per il filtraggio sul contenuto. A tali settaggi non tutti gli utenti possono accedere, occorre farne richiesta esplicita all'amministratore del sistema che provvederà a fornire in risposta una user ID ed una password.

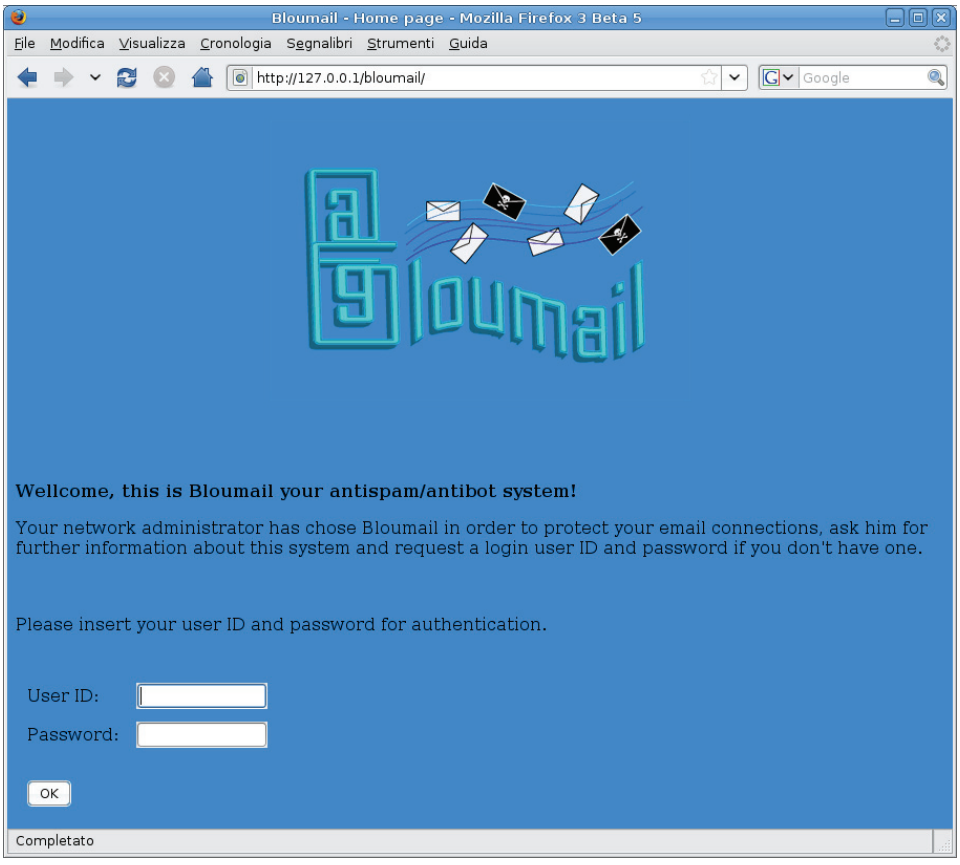


Figura 5.8 – Esempio di richiesta di autenticazione, index.html

In particolare all'autenticazione utente si può accedere attraverso il file *index.html* che corrisponde alla *main page* che appare all'apertura del sito di Bloumail. In tale pagina, l'unica che non contiene codice PHP ma solo HTML, è possibile inserire i propri dati per poter aver accesso alle altre pagine di configurazione. La verifica dei dati inseriti avviene nella pagina *login.php,* i dati sensibili quali ad esempio user ID e password vengono passati tra le pagine che ne hanno bisogno, attraverso il metodo *post* per garantire maggiore sicurezza.

Se l'autenticazione risulta corretta, *login.php* si avvale di *caplogin.php* per costruire una CAPTCHA, compiendo inoltre tutte le operazioni di verifica, simili nella pratica a quanto fatto dalla *capallow.php.* Anche in questo caso si hanno tre possibilità al giorno per risolvere la CAPTCHA, rinnovate ad ogni risoluzione corretta della stessa, la stringa random cambia comunque ad ogni visualizzazione.

L'autenticazione si rende necessaria per poter assegnare un numero finito di tentativi di modifica delle impostazioni dei filtri. Questo perché altrimenti il sistema potrebbe essere facile preda di attacchi continui di *bruteforce* sulla CAPTCHA. Anche se i dati identificativi di un utente venissero in qualche modo letti da un bot che gira sulla sua macchina, questo avrebbe solo tre tentativi al giorno per poter compiere il suo attacco e potrebbe comunque effettuare una sola modifica delle impostazioni dei filtri.

Una volta digitata correttamente la stringa presente nell'immagine, è possibile effettuare una delle seguenti tre azioni:

1. Inserire un indirizzo nella blacklist.

2. Rimuovere un indirizzo dalla blacklist.

3. Richiedere una mailbox per riclassificare un'email.

Figura 5.9 – Esempio di modifica impostazione filtri, login.php

L'ultima pagina del sito di Bloumail, la *checklogin.php*, effettuerà tutti i controlli del caso sul database per verificare definitivamente la correttezza dei dati inseriti e nel caso intraprenderà le azioni richieste. In particolare la modifica degli indirizzi nella blacklist avviene con una semplice query sulla tabella apposita. La generazione della mailbox invece è più problematica in quanto implica la creazione di un indirizzo email fittizio e riconosciuto solo da Bloumail attraverso il quale è possibile riclassificare correttamente un messaggio.

Per rendere le operazioni più semplici si è creato un dominio dal nome BLOUMAIL.SYS. Tutti i messaggi che avranno come dominio del primo indirizzo destinatario tale stringa non saranno trasmessi all'esterno della rete ma

verranno unicamente analizzati da Bloumail. In particolare tali indirizzi per essere validi devono seguire un formato standard del tipo:

1. SPAM.USER-ID.STRINGA-RANDOM@BLOUMAIL.SYS
2. GOOD.USER-ID.STRINGA-RANDOM@BLOUMAIL.SYS

Gli indirizzi del primo tipo consentono di riclassificare un messaggio che Bloumail aveva riconosciuto come buono nella famiglia dello SPAM, mentre quelli del secondo tipo servono per l'operazione inversa. La stringa random consente di rendere particolarmente difficile un attacco di *bruteforce* sugli indirizzi del dominio BLOUMAIL.SYS. In particolare le mailbox hanno una vita limitata ad un giorno, tempo oltre il quale comunque si autodistruggono, anche se non vengono utilizzate.

5.5. Installazione e configurazione

Al fine di semplificare il più possibile le operazioni all'amministratore di rete, è stato creato appositamente un file che consente di installare Bloumail senza dover compiere azioni particolari. Lo script in questione si chiama *setup.sh* ed è possibile lanciarlo da una *shell* linux con i privilegi di *root*. All'avvio vengono compiute le seguenti operazioni:

1. Provvede a verificare se nel sistema esista già un'installazione precedente di Bloumail. In caso positivo chiede all'amministratore se voglia sovrascrivere la copia precedente o abortire l'installazione. Questa utilità consente una facile gestione nel caso futuro in cui si decida di sviluppare nuove versioni dell'applicativo.

2. Installa tutti i pacchetti necessari e le dipendenze mediante *apt-get*. Questo sistema lanciato da *Debian* ed oggi oramai disponibile sulla maggior parte delle distribuzioni linux (come *Ubuntu,* quella qui utilizzata), semplifica molto la gestione delle dipendenze. Ovviamente per essere scaricati i pacchetti necessari c'è bisogno di una connessione ad internet attiva.

3. Decomprime i file necessari all'uso nelle cartelle: */etc/bloumail* e */var/www/bloumail.* Questo passo consente di avere un archivio compresso che occupa molta meno memoria quando non è ancora installato.

4. Decomprime il database delle parole utilizzato da Bogofilter. In particolare richiede all'amministratore se vuole utilizzare un database vergine o uno già preistruito.

5. Costruisce il database MySQL dopo la richiesta della password di root.

6. Riavvia il server web apache per essere da subito pronto all'utilizzo.

Il punto quattro si è reso necessario per permettere una maggior personalizzazione dell'installazione. Si consiglia di utilizzare un database vuoto all'inizio, perché in questo modo il sistema si abitua meglio al traffico email della rete specifica. Tuttavia prima che il sistema diventi abbastanza affidabile in questa modalità sono necessarie diverse migliaia di messaggi legittimi e di spam per far apprendere il filtro di Bayes. A tal scopo si è provveduto ad istruire il sistema con 2 Gb di email di spam (circa 200.000), scaricate da un archivio disponibile in rete.

Seppure a prima vista questa possa sembrare una buona soluzione, occorre dire che non è stato possibile far apprendere il sistema con altrettanti messaggi legittimi, rendendo il database fortemente sbilanciato. In più non è detto che lo spam utilizzato sia lo stesso che si riceverà sulla propria rete, ad esempio si

potrebbe essere colpiti solo da messaggi in lingua italiana mentre il filtro è stato fatto apprendere con testi inglesi. La scelta finale è dunque piuttosto soggettiva, è giusto ricordare che per collezionare qualche migliaia di messaggi su di una rete locale potrebbe volerci poco tempo mentre su di un'altra molto di più. Per tutti questi motivi si è preferito lasciare piena libertà all'amministratore del sistema.

Una volta installato tutto il software necessario si pone per l'amministratore il momento di configurare il proprio sistema. Per iniziare si consiglia di definire quali siano gli utenti che avranno la possibilità di accedere alla modifica dei dati legati ai filtraggi, distribuendo le user ID e le password da usare. Tali informazioni particolarmente sensibili andranno inserite dall'amministratore direttamente nella tabella *login* del database. Si consiglia per tale intervento di avvalersi di uno strumento con una interfaccia *user friendly* quale *phpmyadmin* che può semplificare questa personalizzazione. Il pacchetto necessario viene installato da solo durante la fase di *setup*.

La seconda operazione che si suggerisce di intraprendere è quella legata al riempimento della tabella di blacklist nel caso in cui l'amministratore possieda già un certo numero di indirizzi da bloccare. Come visto in precedenza anche per questa azione è conveniente ricorrere a *phpmyadmin* accedendo direttamente alla tabella *blacklist* del database. Non sempre è necessario agire direttamente sull'elenco, si può lasciare che gli utenti riempiano la tabella di volta in volta quando se ne presenti il bisogno, accedendo alle apposite pagine web dopo essersi autenticati.

A questo punto si può iniziare la configurazione vera e propria, agendo sul file */etc/bloumail/bloumail.conf*. Questo documento testuale si presenta come una sorta di tabella divisa in due parti fondamentali:

1. Tabella che mappa indirizzi e porte locali, ai server usati nella rete.

# TABLE:									
# PROTOCOL	,LOC.INTERFACE	,LOC.ADDRESS	,LOC.PORT	,SERVER.ADDRESS	,SERVER.PORT	,P.THREAD.NUM	,LOG	,BLACKLIST	,BAYES
smtp	,eth0	,192.168.2.1	50000	,smtp.email.it	25	1	1	1	1
pop3	,eth0	,192.168.2.1	51000	,popmail.email.it	110	1	1	1	1
smtp	,eth0	,192.168.2.1	50001	,out.alice.it	25	5	0	0	0
pop3	,eth0	,192.168.2.1	51001	,in.alice.it	110	5	0	0	0

Tabella 5.4 – Esempio della prima parte del file bloumail.conf

Questa tabella permette di impostare le operazioni di *natting* che dovranno essere compiute dalle *iptables* assegnando una ben determinata porta ad ogni server di posta sia SMTP che POP3. Ciò consente di rendere il proxy "trasparente", senza dover introdurre altre modifiche ai singoli client di posta. Si consiglia di utilizzare la porta 50000 e seguenti per mappare i server SMTP e dalla 51000 in poi per i POP3. Tali porte sono lasciate libere dallo standard non essendo assegnate a nessuna applicazione particolare, tuttavia potrebbero essere usate da qualche programma nella rete, pertanto prima è indispensabile assicurarsi della loro fruibilità.

Mentre le impostazioni in nero sono obbligatorie, quelle scritte in rosso sono facoltative, presentano cioè dei valori di default specificati nello stesso file. In particolare: P.THREAD.NUM serve ad indicare il numero di thread che gestiranno quel server consentendo operazioni parallele sullo stesso, LOG è un *flag* per specificare se si desidera salvare tutte le comunicazioni intraprese verso quel server in un file, BLACKLIST permette l'abilitazione del filtraggio omonimo e BAYES consente di utilizzare il filtro sui contenuti.

E' importante segnalare che non è possibile utilizzare coppie <*indirizzo, porta*> uguali, perché non avrebbe significato, ma anzi genererebbe conflitti interni. Tutte le connessioni email (porte 25 e 110) per un certo indirizzo locale non specificate nel file saranno bloccate.

Questo permette di rendere impossibile qualsiasi operazione che punti a bypassare il sistema Bloumail.

2. Riga che fornisce i dati del server di default per l'invio di email di notifica.

# DEFAULT.SMTP.SERVER:	OUT. INTERFACE	,SERVER.ADD	,SERVER.PORT	,EMAIL.ADD	,USER	,PASSWORD
DEFAULT.SMTP.SERVER:	eth1	,smtp.email.it	25	,prova@email.it	prova	prova

Tabella 5.5 – Esempio della seconda parte del file bloumail.conf

Il *default smtp server* ha lo scopo di permettere l'inoltro delle email di notifica all'utente quando il sistema blocca l'invio di un suo messaggio. Questa riga è obbligatoria nel file di configurazione e deve essere unica. I parametri in nero sono indispensabili, mentre quelli in rosso sono facoltativi e non necessari qualora si utilizzi un *open relay* e non sia richiesta una autenticazione.

5.6. Utilizzo ed amministrazione

Una volta terminate tutte le operazioni di installazione e configurazione sia hardware che software, è possibile avviare Bloumail. Innanzitutto è bene sapere che una sola istanza di Bloumail può essere avviata sulla macchina gateway, sia perché non avrebbe senso avere più copie del programma in esecuzione allo stesso momento, sia perché potrebbe generare problemi di cooperazione e di mutua esclusione sulle risorse usate. A tal scopo all'avvio viene innanzitutto scaricato l'elenco dei programmi in esecuzione in quel momento sul sistema e verificato che non esista già un istanza di Bloumail.

A questo punto si può introdurre la sintassi con cui far partire Bloumail sul proprio sistema linux da una normale finestra di *shell*. Per utilizzare questo applicativo occorre avere i privilegi di *root* pertanto ad esempio su Ubuntu si può ricorrere al comando *sudo*. A seguire sarà necessario far partire l'interprete con il comando *perl,* poi è possibile specificare il sorgente *bloumail* e quindi la direttiva di avvio *start*, riassumendo: `sudo perl bloumail start`. Ci sono due opzioni supplementari che si possono utilizzare:

- `-v`: Utilizzando questa sintassi è possibile fare in modo che il programma principale di Bloumail fornisca in output un elenco delle operazioni che sta compiendo durante la sua esecuzione, la cosiddetta modalità *verbose*. Permette da un lato di sapere sempre cosa fa il programma in ogni momento verificandone eventuali stalli, ma soprattutto è fondamentale per la fase di debug, anche qualora si decidesse di implementare nuove funzionalità in versioni future.

- `-f`: Detta anche modalità *flush*, indica a Bloumail che durante le operazioni di modifica delle *iptables* deve sovrascriverle completamente anziché aggiungere le proprie righe in coda come avviene nella modalità *standard.* Questo modo di agire è utile quando la macchina gateway non è stata precedentemente configurata e quindi non fornisce ancora l'accesso ad internet alle macchine della LAN. Bloumail in questo caso si incarica di portare anche questo servizio alla rete locale. E' importante segnalare che in qualunque caso le iptables vengono salvate in un file di backup dal quale è possibile ripristinarle all'uscita da Bloumail.

Per permettere l'esecuzione del programma in *background* è utilizzabile in fase di avvio l'opzione Unix "`&`" da riportare in linea di comando. Ciò consente di non occupare una *shell* inutilmente. Se si volesse utilizzare la modalità *verbose*

durante un'esecuzione in background è possibile dirigere l'ouput su di un file attraverso l'operatore unix ">".

A questo punto l'amministratore non deve eseguire altri comandi particolari, Bloumail lavora esclusivamente sulle connessioni che gli sono state indicate, interfacciandosi unicamente con: server, client ed utenti della LAN. Nel caso siano stati abilitati i log, l'amministratore potrà in qualunque momento verificare cosa sta avvenendo su di una specifica connessione analizzando tali file in cui sono riportati: tutti i comandi, le rispettive risposte e le conseguenti azioni intraprese da Bloumail. Per proteggere la privacy e limitare lo spreco di tempo e di memoria non vengono riportati in questi listati i testi dei messaggi di posta scambiati ma solo gli indirizzi nel caso della posta in uscita.

Poiché non è prevista la rimozione automatica dei file di log si consiglia all'amministratore di cancellarli a intervalli di tempo regolare oppure ove sia necessario di salvarli come copie di backup magari in archivi compressi in modo da ridurre lo spreco di memoria. I file di log sono reperibili nella cartella */bloumail/LOG*, suddivisi in due sottocartelle */SMTP* e */POP*, viene utilizzato come formato standard del nome:

INDIRIZZO#PORTA_ANNO.MESE.GIORNO.ORA.MINUTO.SECONDO.LOG

Una volta che si è deciso di arrestare Bloumail, è necessario aprire una nuova *shell*, e sempre con i privilegi di *root* si può dare questa volta il comando di stop: `sudo perl bloumail stop`. Questa procedura consente di interrompere il servizio completamente, rimuovendo dalla memoria tutti i thread che il processo padre aveva generato per gestire le singole connessioni. Anche per questo comando è possibile l'opzione `-v` per la modalità *verbose*, mentre l'opzione `-f` questa volta indica a Bloumail di non ripristinare le *iptables* originali durante l'arresto ma di ripulirle semplicemente da tutto ciò che contengono. La modalità standard invece prevede, alla chiusura, il ripristino delle *iptables* che si erano salvate durante le operazioni di avvio.

Capitolo 6

Test e conclusioni

6.1. Collaudo e test anti-spam

Durante la realizzazione di Bloumail si sono resi necessari vari test in grado di comprovarne il funzionamento. Il collaudo ha lo scopo primario di certificare la corretta esecuzione del programma, l'assenza di eventuali bug e la compatibilità con gli standard ed i software utilizzabili come *client mail* e *browser*. Innanzitutto si è verificato il processo di installazione su due macchine diverse per configurazione hardware e software, in particolare con due distribuzioni Linux differenti. Il *setup* si è concluso senza riscontrare problemi, quindi si è passato ad esaminare le compatibilità, interfacciando la macchina gateway con un altro computer mediante la seconda scheda di rete.

Dopo la breve fase di configurazione in cui si riportano negli appositi file i dati delle connessioni che si vogliono rendere disponibili e si disabilitano i filtraggi, si è pronti per il primo test di compatibilità, con lo scopo di verificare il pieno funzionamento di tutti i comandi SMTP e POP3. Per eseguire questo esame si è ricorsi all'uso di *telnet* che consente di esaminare il dialogo (comando-risposta) tra il client ed il server. In particolare sono state eseguite connessioni multiple contemporanee atte a comprovare il funzionamento della struttura multithread ed allo stesso tempo anche riscontri su due diversi server di posta sia entrante che uscente.

A questo punto si è pensato di controllare cosa succede nel caso in cui la macchina ad interfacciarsi sia basata su *Linux* o su *Windows*. In particolare sono stati usati anche due diversi client di posta come *Outlook* per Windows ed *Evolution* per Linux, realizzando quindi al tempo stesso anche un altro accertamento.

Una volta collaudata la struttura base del programma si può procedere alla verifica dei filtraggi, settando opportunamente i *flag* nel file di configurazione. Per prima è stata eseguita la verifica sul blacklisting imponendo un indirizzo nella tabella appositamente dedicata. Bloumail segnala correttamente la situazione ed in particolare *Evolution* fa apparire anche una finestra in cui specifica il problema (*Bloumail has blocked your email*). Outlook invece ha un comportamento di più basso profilo presentando un errore comunque ben visibile sia sulla barra di invio, sia nella finestra che si apre ad ogni operazione di inoltro. All'operazione di download dell'email successiva, viene ricevuto correttamente il messaggio indicante il link per l'autorizzazione. Tale pagina web risulta accessibile sia da browser prettamente per Windows come *Internet Explorer*, sia da quelli multipiattaforma come *Mozilla Firefox*, rispettando ancora la piena compatibilità. Dopo l'autorizzazione alla spedizione dell'email viene garantito l'inoltro in entrambi i casi. L'intero sistema marca correttamente come BANNED i messaggi riportati nella blacklist.

Per il filtraggio di Bayes i test risultano essere più complicati, in particolare in relazione all'autoapprendimento del filtro stesso. Occorre inizialmente compiere varie operazioni di training per far acquisire i contenuti delle email sia di spam che legittime. Per eseguire queste verifiche è stato necessario reperire un archivio di email di spam da internet, sul quale far apprendere il sistema. Conseguentemente sono state rispedite a campione alcune di quelle lettere, ottenendo quasi sempre in risposta la classificazione corretta. In pratica i risultati sono gli stessi forniti da *Bogofilter* in quanto è proprio quest'ultimo ad eseguire tali analisi. Nell'ultima fase di collaudo si sono provate le altre pagine del sito

web di interfaccia come quelle di autorizzazione e di variazione delle impostazioni.

6.2. Test anti-bot

Per verificare il corretto funzionamento del sistema in relazione al meccanismo anti-bot, si è infettata una delle macchine collegate a Bloumail con diversi malware. Quando si lavora con dei software malevoli è essenziale operare con la massima prudenza per prevenire contaminazioni ai sistemi adiacenti ed eventuali malfunzionamenti totali ed irreparabili ai propri. Per poter compiere i test in piena sicurezza è stato necessario preparare opportunamente le configurazioni delle macchine utilizzate. In Figura 6.1 viene riportato uno schema di massima della configurazione del sistema di test. Si è scelto di utilizzare solo due macchine in modo da circoscrivere eventuali fughe di malware. Per le interconnessioni sono state create due reti indirizzate staticamente una con indirizzo di rete 192.168.1.0 e l'altra 192.168.2.0.

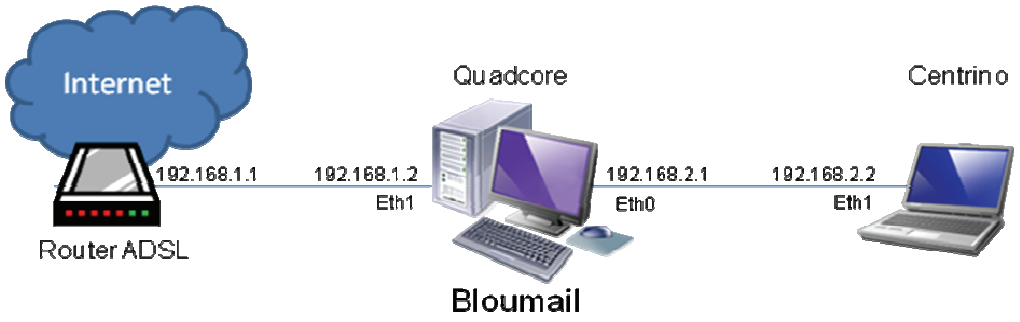


Figura 6.1 – Schema di configurazione del sistema per i test

La configurazione software del client, sul quale si testeranno i virus, sarà quella più delicata e pertanto si è pensato di attrezzarla con una macchina virtuale. Tale espediente si ottiene installando il programma *Vmware* in una partizione governata da Linux. Vmware permette di eseguire una copia di Windows in modalità emulata, lasciando comunque a Linux il pieno controllo del sistema.

Essendo i malware utilizzati avviabili solo in ambiente Windows, il "guscio" Linux circostante garantisce la sicurezza durante i test. Inoltre questa scelta permette nel caso in cui un programma malevolo danneggi i file di configurazione di Windows rendendolo instabile, di ripristinarlo semplicemente avviando una copia di backup del sistema operativo precedentemente salvata. Per le verifiche è stata utilizzata una copia di Windows XP Professional con la Service Pack 1 preinstallata, senza altri programmi aggiunti.

La ricerca di alcuni malware da poter utilizzare per i test non è stata facile, data la pericolosità di questi applicativi ed il reato di diffusione a cui si può andare incontro, in alcuni paesi. Tuttavia dopo una lunga ricerca si è venuti in possesso di un archivio di 2.2 GB corrispondenti a circa 37500 malware eseguibili. La sua composizione è approssimativamente la seguente: 9400 trojan, 23600 virus, 2400 worm e 2100 vari altri programmi malevoli. Si è scelto di analizzare più approfonditamente i worm che sembrano essere tra le maggiori minacce allegate alle email, capaci di indurre una macchina a far parte di una botnet, o di usare la stessa come punto di partenza per lo spam. In particolare si è visto che 1370 di questi presentavano nel nome legami con il servizio di posta elettronica e di questi ultimi 930 interessavano il sistema operativo Windows (tipo *Win32*). Meno della metà, circa 400, sono le famiglie in cui suddividere tali worm, prendendo in esame al massimo uno per ogni famiglia si sono isolati circa 200 worm.

Le informazioni reperibili in rete circa ogni singolo worm sono poche e frammentarie, al più vengono indicate in brevi resoconti di sistemi anti-virus. Questo perché ogni malware sortisce effetti maggiori se può contare sul fattore sorpresa, quindi nessuno di coloro che scrive tali applicativi ne produce anche una qualche documentazione. Allo stesso tempo essendo sempre maggiore il numero di tipi di infezioni prodotte neanche i tecnici anti-virus, che possono contare su enormi risorse economiche, riescono a darne singole complete trattazioni. Non è detto che con le tecniche attualmente disponibili si riesca a conoscere tutte le azioni che un eseguibile di questo tipo compie, senza avere a disposizione i suoi codici sorgenti.

Da queste semplici considerazioni si è preferita alla ricerca di informazioni su ogni singolo malware, una prima fase di test in modo da avere un'idea di ciò che il worm individualmente compie. Questo consente di evitare verifiche più approfondite su tutti quegli applicativi che non lavorano con il servizio email e che quindi non sono rilevanti in questa sede. Per eseguire questo primo tipo di prove si è scelto un approccio molto restrittivo cercando di bloccare il più possibile quanto volesse uscire dalla rete locale. Pertanto si sono resi necessari particolari accorgimenti soprattutto relativamente alla scelta degli account e dei server di posta da utilizzare:

- Uso di un solo account di posta: Ciò consente di avere nel file di configurazione un solo server per la posta entrante ed uno solo per quella uscente. Questo permette di restringere il numero di messaggi eventualmente spediti durante l'infezione e di poter monitorare meglio le connessioni che si apriranno.

- Uso di un solo indirizzo di posta nella rubrica del client: Impostando un solo indirizzo nella rubrica del client di posta (*Outlook express*), si può restringere il numero di eventuali destinatari a cui verranno inoltrati i messaggi. Si ricorda che questo indirizzo e quello dell'account sono stati generati appositamente solo per queste prove.

- Uso di server per la posta in uscita con autenticazione obbligatoria: L'account di posta è stato scelto in base alla presenza del metodo di autenticazione dell'utente mediante user ID e password prima di poter inviare l'email. Questo server inoltre non permette di spedire messaggi che contengano un mittente diverso da quello che si è autenticato. E' bene tener presente che la quasi totalità dei server di posta che consente di registrare un account, oggi è protetto da questa tecnica.

- Uso di server di posta con filtri anti-spam ed anti-virus propri: La scelta di un server che implementi di per sé un filtraggio anti-spam ed anti-

virus permette di avere un ulteriore margine di sicurezza nel caso in cui qualche messaggio dovesse uscire dalla rete.

Oltre a questi piccoli accorgimenti occorre avere degli altri strumenti per poter monitorare le azioni degli worm ed eventualmente reagire di conseguenza. In particolare Bloumail viene avviato in modalità verbose per permettere la segnalazione di malfunzionamenti, inoltre si abilitano i file di log per avere una traccia del traffico email e come ultima cosa, si settano i flag necessari a consentire i filtraggi. Seppure i file di log costituiscano un importante strumento per il controllo del traffico uscente dalla rete, è conveniente prevedere altri metodi più potenti per questo scopo.

Si è ricorsi quindi all'applicativo *Wireshark*, disponibile liberamente per l'ambiente Linux. Questo software consente di avere in tempo reale traccia di qualsiasi traffico dati, sia uscente che entrante dall'interfaccia selezionata, inoltre provvede a classificare le singole connessioni e ad identificare i protocolli utilizzati. Nei test sono state avviate due istanze di questo programma, una nella macchina gateway su cui gira anche Bloumail ed una su quella del client dove è in esecuzione anche la macchina virtuale. In questo modo viene evidenziato sia tutto il traffico della rete locale, monitorato dal client, sia tutto quello destinato ad internet, esaminato dal gateway.

La sperimentazione ha richiesto parecchio tempo in quanto molti worm hanno avuto un comportamento particolarmente violento, rendendo il sistema alquanto instabile e non permettendo di poter procedere se non dopo aver avviato una nuova istanza della macchina virtuale. Questo ha reso necessario, ad intervalli più o meno regolari di tempo, di creare nuove copie del sistema operativo. Considerando la dimensione di ogni copia pari a circa 1GB di memoria, si capisce facilmente quanto, anche solo questa operazione ripetuta molte volte, abbia aumentato il tempo di testing. I comportamenti dei malware esaminati sono stati piuttosto vari, è possibile riassumerli in alcuni punti essenziali:

- Arresto o bloccaggio del sistema: Una parte degli worm implica un completo arresto di tutte le normali funzioni del sistema operativo, oppure ne blocca parzialmente le funzionalità, obbligando l'utente al riavvio.

- Formattazione della partizione attiva: Una parte dei malware ha completamente rimosso qualunque file si trovasse sulla partizione Windows emulata.

- Replicazione incontrollata del worm: Si sono verificati casi di replicazioni dello stesso software malevolo, o in cui sono stati generati, senza soluzione di continuità, vari file di natura incognita.

- Drastico aumento nel carico del processore: Molto spesso si sono verificati picchi di carico nell'uso della CPU riscontrati mediante l'utilità di Windows (*Task Manager*).

- Apertura di finestre: Un gran numero degli worm esaminati ha aperto finestre a volte con dei messaggi apparentemente innocui altre con toni offensivi o di sfida.

- Nessuna operazione evidente: Questo comportamento avrebbe lo scopo di far perdere traccia di cosa possa aver provocato l'infezione. Spesso i risultati del contagio sono visibili solo dopo un successivo riavvio della macchina.

- Apertura di connessioni: Questa è la condotta più interessante, in quando riguarda la generazione di traffico non desiderato uscente dalla rete. I tipi di connessione rilevati sono essenzialmente HTML, IRC, SMTP. La maggior parte di questi worm cerca di lavorare nell'ombra non dando segnali delle operazioni che stanno compiendo, esiste tuttavia un'esigua parte (specie quelli che producono pacchetti

contenenti HTML) che apre alcune finestre del browser accedendo a siti non desiderati.

Rilevati i malware che hanno generato traffico SMTP, si può passare a test più approfonditi. E' utile segnalare che grazie alle misure intraprese nessuna email è stata spedita, fatta eccezione per una che comunque è stata segnalata e neutralizzata dal filtro anti-virus del server di posta. Già questo di per sé rappresenta un buon risultato. Il numero degli worm in questione è pari a 13, rappresentate meno del 7% del campione di 200 software malevoli esaminati.

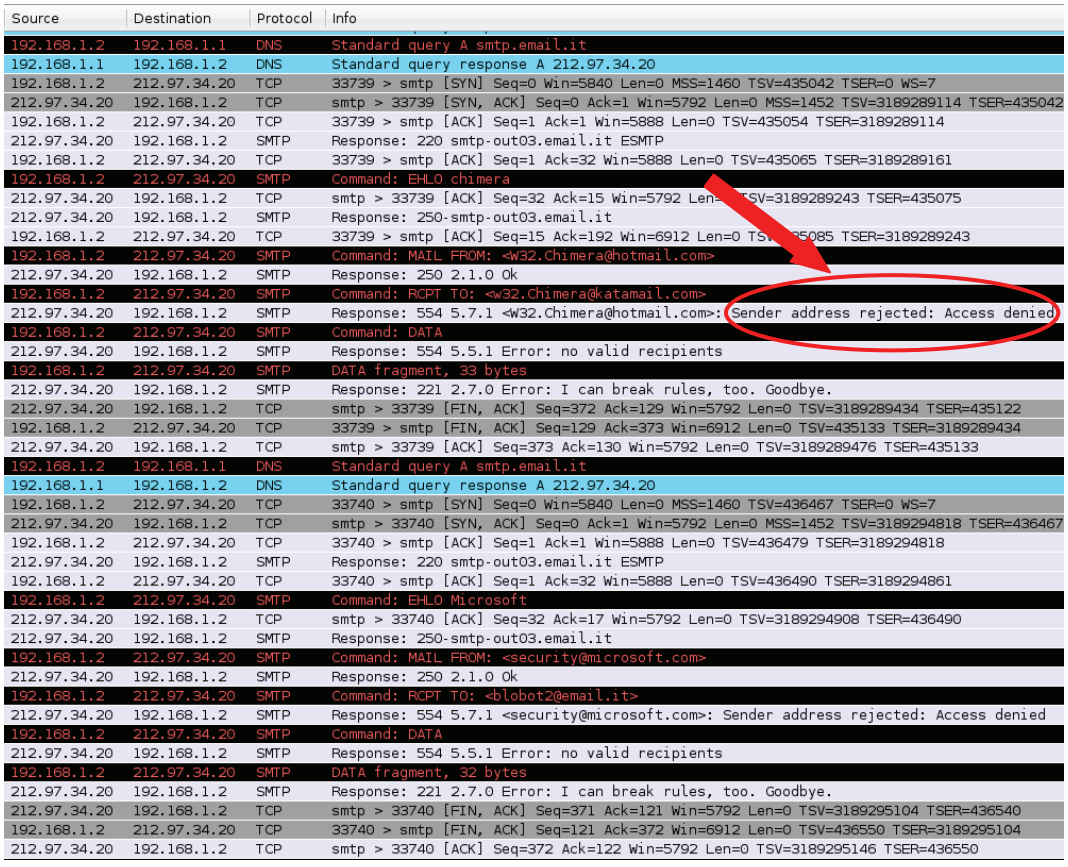
Le verifiche successive si sono concentrare in particolar modo sull'individuazione dei server destinatari delle connessioni SMTP. In seconda analisi ci si è preoccupati di verificare l'eventuale dialogo, prestando maggiore attenzione alla fase di autenticazione e alla scelta degli indirizzi mittenti e destinatari. Gli worm studiati possono essere suddivisi in tre categorie in base alla scelta dei server SMTP da cui inviare le loro email:

1. Utilizzanti server propri: Questi worm cercano di inoltrare i messaggi avvalendosi dei server di posta di loro conoscenza. Il programma Wireshark evidenzia questo comportamento presentando numerose richieste DNS atte a conoscere gli indirizzi IP di tali server. Spesso non sono vengono fornite risposte a queste invocazioni, a significare come alcuni dei server interpellati siano stati ormai chiusi, perché ad esempio di tipo *open relay*. Sono stati rilevati sei worm di questo genere: *anset.a, avron.a, donghe.a, dumaru.a, lingon, mapson.a.*

2. Utilizzanti il server di default: Questi worm cercano di inoltrare le email usando il server di posta specificato nei file di configurazione del client usato. I malware in questione sembrano più evoluti dei precedenti in quanto riescono a conoscere dati sensibili accedendo ai registri di Windows ed alla rubrica di Outlook. A questo genere appartengono: *clepa, ganda, mydoom.i, swen, wallon.b, zircon.a.*

3. Utilizzanti sia server propri che di default: Questi worm permettono connessioni dei due tipi precedentemente analizzati. Si pongono come un anello di congiunzione, in quanto nell'unico caso trovato, l'identificazione del server di default non è avvenuta correttamente, sfruttando quindi l'altra modalità disponibile. Sembra quindi che la lettura delle informazioni nei registri non sia ben implementata, il malware in questione è *bridex.a*.

Analizzando il funzionamento di Bloumail con il primo tipo di worm ci si accorge che l'utilità atta a bloccare le connessioni ai server di posta non specificati nel file di configurazione, permette di non far instaurare le connessioni con i server dei malware. Ciò consente quindi di ottenere una protezione da più del 50% degli worm esaminati solo con questa piccola funzionalità (comprendendo *bridex.a* nella prima categoria).



Source	Destination	Protocol	Info
192.168.1.2	192.168.1.1	DNS	Standard query A smtp.email.it
192.168.1.1	192.168.1.2	DNS	Standard query response A 212.97.34.20
192.168.1.2	212.97.34.20	TCP	33739 > smtp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=435042 TSER=0 WS=7
212.97.34.20	192.168.1.2	TCP	smtp > 33739 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1452 TSV=3189289114 TSER=435042
192.168.1.2	212.97.34.20	TCP	33739 > smtp [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=435054 TSER=3189289114
212.97.34.20	192.168.1.2	SMTP	Response: 220 smtp-out03.email.it ESMTP
192.168.1.2	212.97.34.20	TCP	33739 > smtp [ACK] Seq=1 Ack=32 Win=5888 Len=0 TSV=435065 TSER=3189289161
192.168.1.2	212.97.34.20	SMTP	Command: EHLO chimera
212.97.34.20	192.168.1.2	TCP	smtp > 33739 [ACK] Seq=32 Ack=15 Win=5792 Len=0 TSV=3189289243 TSER=435075
212.97.34.20	192.168.1.2	SMTP	Response: 250-smtp-out03.email.it
192.168.1.2	212.97.34.20	TCP	33739 > smtp [ACK] Seq=15 Ack=192 Win=6912 Len=0 TSV=435085 TSER=3189289243
192.168.1.2	212.97.34.20	SMTP	Command: MAIL FROM: <W32.Chimera@hotmail.com>
212.97.34.20	192.168.1.2	SMTP	Response: 250 2.1.0 Ok
192.168.1.2	212.97.34.20	SMTP	Command: RCPT TO: <w32.Chimera@katamail.com>
212.97.34.20	192.168.1.2	SMTP	Response: 554 5.7.1 <W32.Chimera@hotmail.com>: Sender address rejected: Access denied
192.168.1.2	212.97.34.20	SMTP	Command: DATA
212.97.34.20	192.168.1.2	SMTP	Response: 554 5.5.1 Error: no valid recipients
192.168.1.2	212.97.34.20	SMTP	DATA fragment, 33 bytes
212.97.34.20	192.168.1.2	SMTP	Response: 221 2.7.0 Error: I can break rules, too. Goodbye.
212.97.34.20	192.168.1.2	TCP	smtp > 33739 [FIN, ACK] Seq=372 Ack=129 Win=5792 Len=0 TSV=3189289434 TSER=435122
192.168.1.2	212.97.34.20	TCP	33739 > smtp [FIN, ACK] Seq=129 Ack=373 Win=6912 Len=0 TSV=435133 TSER=3189289434
212.97.34.20	192.168.1.2	TCP	smtp > 33739 [ACK] Seq=373 Ack=130 Win=5792 Len=0 TSV=3189289476 TSER=435133
192.168.1.2	192.168.1.1	DNS	Standard query A smtp.email.it
192.168.1.1	192.168.1.2	DNS	Standard query response A 212.97.34.20
192.168.1.2	212.97.34.20	TCP	33740 > smtp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=436467 TSER=0 WS=7
212.97.34.20	192.168.1.2	TCP	smtp > 33740 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1452 TSV=3189294818 TSER=436467
192.168.1.2	212.97.34.20	TCP	33740 > smtp [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=436479 TSER=3189294818
212.97.34.20	192.168.1.2	SMTP	Response: 220 smtp-out03.email.it ESMTP
192.168.1.2	212.97.34.20	TCP	33740 > smtp [ACK] Seq=1 Ack=32 Win=5888 Len=0 TSV=436490 TSER=3189294861
192.168.1.2	212.97.34.20	SMTP	Command: EHLO Microsoft
212.97.34.20	192.168.1.2	TCP	smtp > 33740 [ACK] Seq=32 Ack=17 Win=5792 Len=0 TSV=3189294908 TSER=436490
212.97.34.20	192.168.1.2	SMTP	Response: 250-smtp-out03.email.it
192.168.1.2	212.97.34.20	SMTP	Command: MAIL FROM: <security@microsoft.com>
212.97.34.20	192.168.1.2	SMTP	Response: 250 2.1.0 Ok
192.168.1.2	212.97.34.20	SMTP	Command: RCPT TO: <blobot2@email.it>
212.97.34.20	192.168.1.2	SMTP	Response: 554 5.7.1 <security@microsoft.com>: Sender address rejected: Access denied
192.168.1.2	212.97.34.20	SMTP	Command: DATA
212.97.34.20	192.168.1.2	SMTP	Response: 554 5.5.1 Error: no valid recipients
192.168.1.2	212.97.34.20	SMTP	DATA fragment, 32 bytes
212.97.34.20	192.168.1.2	SMTP	Response: 221 2.7.0 Error: I can break rules, too. Goodbye.
212.97.34.20	192.168.1.2	TCP	smtp > 33740 [FIN, ACK] Seq=371 Ack=121 Win=5792 Len=0 TSV=3189295104 TSER=436540
192.168.1.2	212.97.34.20	TCP	33740 > smtp [FIN, ACK] Seq=121 Ack=372 Win=6912 Len=0 TSV=436550 TSER=3189295104
212.97.34.20	192.168.1.2	TCP	smtp > 33740 [ACK] Seq=372 Ack=122 Win=5792 Len=0 TSV=3189295146 TSER=436550

Figura 6.2 – Schermata di Wireshark durante il test del worm Clepa

Procedendo con l'analisi del secondo sottoinsieme di worm, si sono riscontrati tre comportamenti poco dissimili tra loro. Nel primo caso, costituito da *clepa* e *ganda*, seppure il default server sia stato correttamente identificato, gli worm non sono riusciti ad inoltrare correttamente i loro messaggi, sia perché non hanno eseguito l'autenticazione al server, sia perché hanno utilizzato un campo mittente (FROM) diverso da quello dell'account che volevano sfruttare (Figura 6.2).

Swen, wallon.b e zircon.a, si differenziano dai precedenti per l'individuazione e l'utilizzo corretto del campo mittente, tuttavia ancora una volta l'autenticazione con il server non viene effettuata, non potendo quindi inoltrare le email (Figura 6.3).

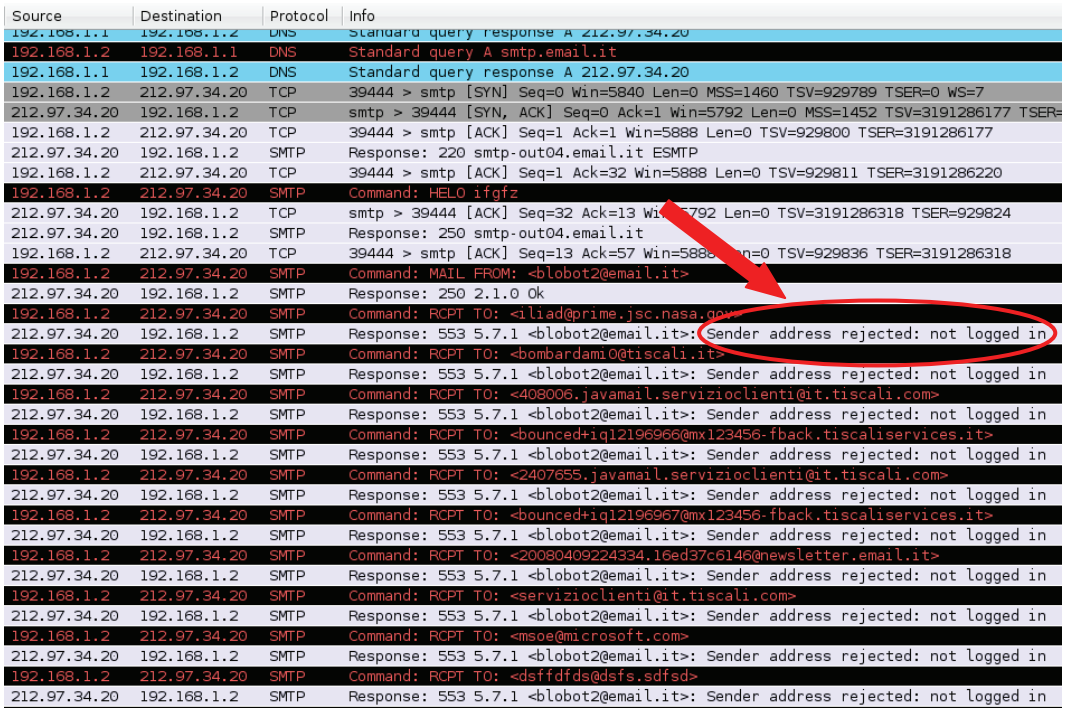
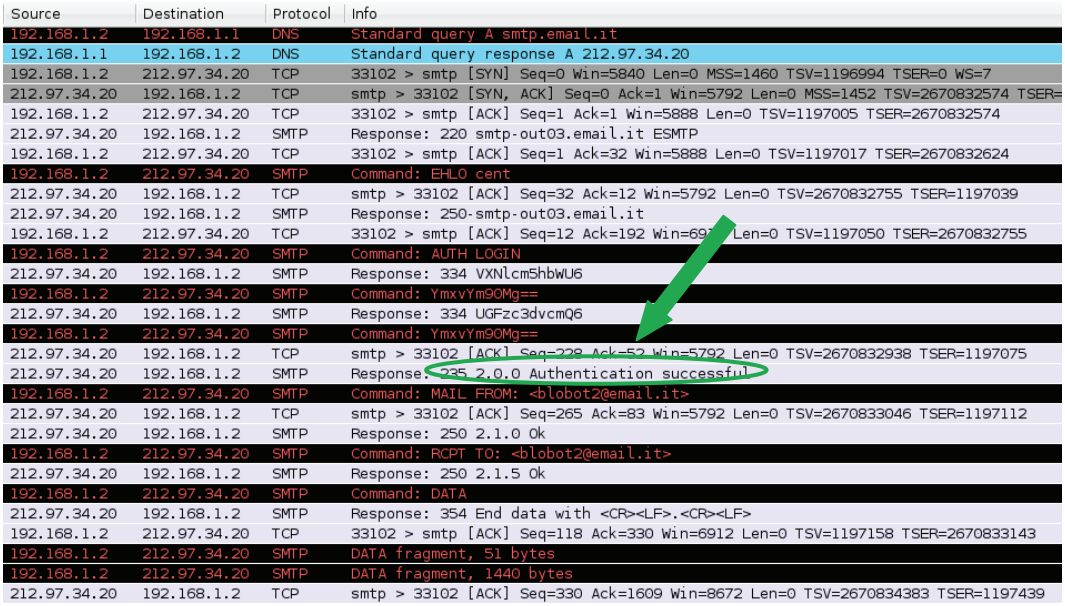


Figura 6.3 – Schermata di Wireshark durante il test del worm Swen

Durante tutti i test l'unico worm ad essere riuscito ad inviare una email correttamente utilizzando il server di default ed autorizzandosi correttamente è stato *Mydoom.i* (Figura 6.4).



Source	Destination	Protocol	Info
192.168.1.2	192.168.1.1	DNS	Standard query A smtp.email.it
192.168.1.1	192.168.1.2	DNS	Standard query response A 212.97.34.20
192.168.1.2	212.97.34.20	TCP	33102 > smtp [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=1196994 TSER=0 WS=7
212.97.34.20	192.168.1.2	TCP	smtp > 33102 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1452 TSV=2670832574 TSER=
192.168.1.2	212.97.34.20	TCP	33102 > smtp [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSV=1197005 TSER=2670832574
212.97.34.20	192.168.1.2	SMTP	Response: 220 smtp-out03.email.it ESMTP
192.168.1.2	212.97.34.20	TCP	33102 > smtp [ACK] Seq=1 Ack=32 Win=5888 Len=0 TSV=1197017 TSER=2670832624
192.168.1.2	212.97.34.20	SMTP	Command: EHLO cent
212.97.34.20	192.168.1.2	TCP	smtp > 33102 [ACK] Seq=32 Ack=12 Win=5792 Len=0 TSV=2670832755 TSER=1197039
212.97.34.20	192.168.1.2	SMTP	Response: 250-smtp-out03.email.it
192.168.1.2	212.97.34.20	TCP	33102 > smtp [ACK] Seq=12 Ack=192 Win=69[illegible] Len=0 TSV=1197050 TSER=2670832755
192.168.1.2	212.97.34.20	SMTP	Command: AUTH LOGIN
212.97.34.20	192.168.1.2	SMTP	Response: 334 VXNlcm5hbWU6
192.168.1.2	212.97.34.20	SMTP	Command: YmxvYm90Mg==
212.97.34.20	192.168.1.2	SMTP	Response: 334 UGFzc3dvcmQ6
192.168.1.2	212.97.34.20	SMTP	Command: YmxvYm90Mg==
212.97.34.20	192.168.1.2	TCP	smtp > 33102 [ACK] Seq=228 Ack=52 Win=5792 Len=0 TSV=2670832938 TSER=1197075
212.97.34.20	192.168.1.2	SMTP	Response: 235 2.0.0 Authentication successful
192.168.1.2	212.97.34.20	SMTP	Command: MAIL FROM: <blobot2@email.it>
212.97.34.20	192.168.1.2	TCP	smtp > 33102 [ACK] Seq=265 Ack=83 Win=5792 Len=0 TSV=2670833046 TSER=1197112
212.97.34.20	192.168.1.2	SMTP	Response: 250 2.1.0 Ok
192.168.1.2	212.97.34.20	SMTP	Command: RCPT TO: <blobot2@email.it>
212.97.34.20	192.168.1.2	SMTP	Response: 250 2.1.5 Ok
192.168.1.2	212.97.34.20	SMTP	Command: DATA
212.97.34.20	192.168.1.2	SMTP	Response: 354 End data with <CR><LF>.<CR><LF>
192.168.1.2	212.97.34.20	TCP	33102 > smtp [ACK] Seq=118 Ack=330 Win=6912 Len=0 TSV=1197158 TSER=2670833143
192.168.1.2	212.97.34.20	SMTP	DATA fragment, 51 bytes
192.168.1.2	212.97.34.20	SMTP	DATA fragment, 1440 bytes
212.97.34.20	192.168.1.2	TCP	smtp > 33102 [ACK] Seq=330 Ack=1609 Win=8672 Len=0 TSV=2670834383 TSER=1197439

Figura 6.4 – Schermata di Wireshark durante il test del worm Mydoom.i

Mydoom è stato segnalato per la prima volta nel 2004 quando prolificò generando una vera epidemia su scala mondiale, oggi ne esistono molte varianti. Oltre a generare traffico email alcune permettono di realizzare attacchi *DOS* mediante le *backdoor* che lo stesso worm apre nel sistema infetto. *Symantec* riporta un'analisi di questo malware, indicando che oltre a carpire informazioni dal registro di Windows e dalla rubrica di Outlook, riesce a rintracciare altri indirizzi mediante la scansione di alcuni particolari tipi di file contenuti nel sistema, come ad esempio *.txt*, *.wab*, *.php* ecc.

Le email generate da *Mydoom* sono piuttosto standard, presentando in maniera casuale un certo numero di soggetti e di corpi dei messaggi. Allo stesso modo, ogni email è corredata di un allegato anch'esso dal nome variabile ma scelto tra una ristretta cerchia. Questo metodo gli consente di replicarsi e diffondersi in rete. Di seguito sono riportate alcune delle possibili alternative, tutte sembrano voler indicare al destinatario un possibile guasto con la richiesta di aprire l'allegato al fine di ulteriori specificazioni.

- Soggetti: *test, hi, hello, mail delivery system, mail transaction failed, server report, status, error*.

- Corpi dei messaggi:

  Mail transaction failed. Partial message is available.
  The message contains Unicode characters and has been sent as a binary attachment.
  The message cannot be represented in 7-bit ASCII encoding and has been sent as a binary attachment.

- Allegati: *document, readme, doc, text, file, data, test, message, body*.

Quanto sopra esposto consente di far apprendere opportunamente il filtro di Bayes in modo da rilevare i messaggi generati da un worm simile a questo, limitando quindi le già esigue possibilità di infezione. Alcuni dei malware esaminati nella prima fase generano traffico email sfruttando Outlook, e soltanto dopo aver fatto accettare all'utente l'inoltro, da una apposita finestra del client di posta. Tali worm non sono stati presi in considerazione in quanto la sicurezza del sistema è già garantita da questa semplice interfaccia resa disponibile da Outlook.

6.3. Conclusioni

Al termine di questa tesi, si possono trarre delle conclusioni su quanto visto, innanzitutto il servizio email basato sui protocolli SMTP e POP3 si presenta piuttosto fragile, specialmente nelle sue prime implementazioni. Con il passare degli anni, si è cercato di porre rimedio almeno in parte, alla necessità di maggior sicurezza specie grazie alle estensioni ESMTP, tuttavia queste da sole non bastano a garantire la protezione dagli attacchi esterni. Allo stesso tempo gli strumenti per

abusare del servizio di posta elettronica si sono moltiplicati, cercando di sfruttarne ogni vulnerabilità.

Ricercatori e s*oftware house* si sono occupati più volte dei problemi correlati all'uso delle email, presentando tecniche sempre più raffinate ma spesso scendendo troppo in profondità e cercando di modificare i protocolli stessi. Seppure una revisione degli standard utilizzati sia auspicabile in futuro, già attualmente non potrebbe essere di così facile attuazione, considerando lo sviluppo e la diffusione di tale servizio. Inoltre non tutte le problematiche sono state affrontate con il medesimo vigore, un esempio è fornito dalle *botnet* che malgrado rappresentino un'insidia piuttosto recente e siano già considerate tra i mezzi più devastanti di cui dispongono gli hacker, non trovano però sul mercato strumenti abbastanza efficaci atti a contrastarle.

Bloumail è stato sviluppato con il fine di trovare una soluzione ai problemi appena descritti. In particolare un requisito alla base di questo applicativo è la duttilità nell'uso, su sistemi di diversa natura, mediante un'unica installazione sul punto di accesso della LAN ad internet. La scalabilità è un altro importante obiettivo, ottenuto secondo una tecnica multithreading, sviluppata in modo da essere poco onerosa in termini di risorse utilizzate, potendo quindi prestarsi a successive modifiche in grado di far funzionare Bloumail su di un router. Allo stesso tempo si è cercato di rendere l'intero sistema di facile utilizzo per gli utenti medi, presentando interfacce basate sul web in modo da superare i problemi di sicurezza legati alla singola macchina come virus o bug. Anche gli amministratori possono gestire semplicemente il sistema modificando opportunamente l'unico file di configurazione disponibile, strutturato in maniera tabulare.

Da questi presupposti si è sviluppato Bloumail, implementando tecniche di filtraggio quali blacklisting e metodi probabilistici di Bayes. Queste consentono di poter istruire il sistema a bloccare ciò che desidera l'utente, mediante una continua fase di apprendimento, sia passivo, che attivo correggendo Bloumail quando esegue classificazioni errate. Allo stesso tempo per verificare che sia solo un

umano a poter operare sui parametri di configurazione, sono state utilizzate CAPTCHA con un ristretto numero di tentativi disponibili. L'accesso all'area di riclassificazione messaggi, nella famiglia dello spam ed in quella delle email legittime, è stata garantita mediante un'autenticazione basata su user ID e password, che ne protegge ancor più l'utilizzo.

I test svolti su alcuni worm responsabili di generare traffico SMTP hanno dimostrato che più del 50% di questi è stato bloccato dalle restrizioni imposte nel file di configurazione circa le connessioni utilizzabili. Dell'altra metà che invece si è avvalsa del server smtp di default, impostato nel client di posta, solo un malware è riuscito ad autenticarsi correttamente ottenendo di poter spedire un messaggio. Ciò si è verificato grazie all'utilizzo di server di posta uscente che necessitano dell'identificazione dell'utente prima di poter inviare email. Negli ultimi anni il numero degli account di posta che utilizzano *open relay* (server senza autenticazione), per permettere ai propri utenti di spedire messaggi, è drasticamente diminuito. Comunque è sempre buona norma, se si è amministratori, richiedere agli utenti di adoperare solo su server con autenticazione, al fine di prevenire eventuali abusi.

Nel caso del worm *Mydoom*, l'unico ad essere riuscito a compiere un inoltro, le tecniche bayesiane forniscono risultati più che soddisfacenti, essendo molto ristretto il numero di tipi di messaggi generati da esso. Bloumail si è dimostrato piuttosto stabile e senza evidenti bug, il carico di lavoro su processore e memoria non sono troppo eccessivi anche durante l'utilizzo multiutente, grazie all'uso del filtraggio di Bayes (il più oneroso) in modalità esclusiva. Ciò consente di prevenire attacchi di tipo *Denial of Service*, a tutto vantaggio della sicurezza globale.

Concludendo Bloumail può crearsi una nicchia nel variegato mondo dei software anti-spam, presentando una struttura che si pone a metà tra le implementazioni legate ai singoli client di posta e quelle che trovano sede sui server. Al tempo stesso è attrezzato con un metodo anti-bot piuttosto innovativo,

che gli permette di garantire un salto di qualità rispetto alle comparazioni con i software attualmente reperibili. Inoltre consente di liberare l'amministratore dalla necessità di aggiungere regole particolari nel firewall della rete per quanto riguarda il traffico di posta. Anche l'aspetto dell'interfaccia *user friendly* è da considerarsi come un punto di forza, garantendo all'utente finale di poter configurare facilmente i filtraggi e di renderli adatti alla particolare realtà della rete in cui lavora. Infine Bloumail può essere installato insieme a Blobot, quest'ultimo garantisce la protezione della rete anche dai bot che operano sul web, integrando così il pacchetto applicativo e rendendolo completo anche sotto questo aspetto.

Bibliografia

[1] Prepared Statement of the Federal Trade Commission on "Unsolicited Commercial Email" before the Committee on Commerce, Science and Transportation, U.S. Senate, Washington, DC, May 21, 2003; http://www.ftc.gov/os/2003/05/spamtestimony.pdf

[2] Statistical data on the Open Relay Database web site; http://ordb.org/statistics/

[3] Bogdan Hoanca, "How Good Are Our Weapons in the Spam Wars?", IEEE Technology and Society Magazine, Spring 2006

[4] M. Krochmal, "Spammer says "Uncle" to AOL"; http://content.techweb.com/wire/story/TWB19971218S0007

[5] "Increasing spam threat from proxy hijackers"; http://spamhous.org/news.lasso?article=156

[6] B. Withworth and E. Withworth, "Spam and the social technical gap", IEEE Computer, vol. 37, no. 10, pp. 38-45, Oct. 2004

[7] Spam: The serial ROI killer, Res. Note E50, June 2004; http://www.nucleusresearch.com/research/e50.pdf

[8] Quick FAQ, Coalition against Unsolicited Commercial Email; http://www.cauce.org/about/faq.shtml

[9] Cynthia Dhinakaran and Jae Kwang Lee, "Characterizing Spam traffic and Spammers", International Conference on Convergence Information Technology 2007

[10] Messaging and Anti-Abuse Working Group (MAAWG), 2007

[11] Global Internet Security Threat Report, Symantec 2008

[12] Global Phishing Report, Symantec 2008

[13] Security Threat Report, SOPHOS 2007; http://www.sophos.com/security/whitepapers/

[14] Malicious Software (Malware): A Security Threat to the Internet Economy, Organisation for Economic Co-operation and Development 2008

[15] Whittaker, Colin, APACS, APEC-OECD Malware Workshop presentation; http://www.oecd.org/dataoecd/33/53/38652807.pdf

[16] Nicholas Ianelli and Aaron Hackworth, "Botnets as a vehicle for online crime", Dec. 2005

[17] Gadi Evron, "Alternative Botnet C&Cs", Syngress

[18] Pew Internet Report, "The advent of spim"; http://www.pewinternet.org/PPF/p/1052/pipcomments.asp

[19] J. Graham-Cumming, "The Spammer's Compendium", Apr. 2005

[20] Paul Graham, "A plan for spam", Aug. 2002

[21] Paul Graham, "Better Bayesian Filter", 2003

[22] L. Walker, “Wedding in the garden of good e-mail”, Washington Post, pp. E.01, Jan. 31 2002

[23] L. Donnerhacke, Teergrubbing FAQ; http://www.iks-jena.de/mitrab/lutz/usenet/teergrube.en.html

[24] Open Source Technology Group, 2006; http://sourceforge.net/projects/tarproxy

[25] K. Li, C. Pu, and M. Ahamad, “Resisting SPAM delivery by TCP damping”, Proc. First Conf. on Email and Anti-Spam (CEAS 2004)

[26] E. Harris, “The Next step in the spam control war: Greylisting”; http://projectspuremagic.com/greylisting/withepaper.html

[27] A. Jones, **“**Greylisting performance”; http://users.aber.ac.uk/auj/spam/greyperf.shtml

[28] R. D. Twining, M. M. Williamson, M. Mowbray and M. Rahmouni, “Email prioritization: Reducing delays on legitimate mail caused by junk mail”, Hewlett-Packard 2004

[29] School of Computer Science, Carnegie-Mellon University, 2000-2005

[30] C. Dwork and M. Naor, “Pricing via processing or combating junk mail”, CRYPTO ‘92

[31] M. Jackobsson and A. Jules, “Proof of work and bread pudding protocols”, Proc. IFIP TC6 and TC 11 Joint working Conf. on Communications and Multimedia Security 1999

[32] B. Laurie and R. Clayton, “Proof-of-Wok’ proves not work”, V. 0.2, Sep. 2004

[33] J.-M. Seigneur and C.D. Jensen, “Privacy recovery with disposable email addresses”, IEEE Security &Privacy Mag., Vol. 1, No. 6, pp.35-39, 2003

[34] Yanhui Guo and Yaolong Zhang, “Research on the Comprehensive Anti-Spam Filter”, IEEE 2006

[35] Tom Christiansen and Nathan Torkington, “O’Reilly Perl Cookbook”, First Edition, Aug. 1998

[36] Peter Wainwright, “Pro Perl”, 2005

[37] http://bogofilter.sourceforge.net/

[38] Gary Robinson, “A statistical approach to the spam problem”, Mar. 1 2003

[39] Corbet, “Spam avoidance techniques”, Sep. 11 2003, LWN.net

[40] Rossella Candela, “Progettazione e realizzazione di un sistema per l’analisi e il filtraggio del traffico generato da un bot”, A.A. 2006-2007