Relazione Progetto Simulazione DDoS

Corso di Simulazione di Sistemi
anno accademico 2014-2015

Giovanni Cattani, Mauro Belgiovine

Informatica Magistrale Bologna
giovanni.cattani@studio.unibo.it
mauro.belgiovine@studio.unibo.it

Abstract. Lo scopo di questo progetto consiste nel modellare una rete di calcolatori e studiarne il comportamento in differenti condizioni, caratterizzate dall'esecuzione di un attacco DDoS, dalla sua intensità e dalla presenza di un maccanismo di difesa basato sul filtraggio del traffico. Dopo una panomarica incentrata su questa tipologia di attacchi ed uno studio del problema, si prosegue con l'implementazione della rete tramite il simulatore software Omnet++. Successivamente vengono svolte molteplici simulazioni, differenziate dalla variazione di alcuni parametri chiave, e ne vengono raccolti i dati. Questi vengono poi analizzati con particolare attenzione per alcune metriche di valutazione. Infine vengono presentate le considerazioni sui risultati ottenuti, le limitazioni riscontrate e possibili sviluppi futuri.

1 DDoS - Distributed Denial of Service

Gli attacchi DDoS sono al giorno d'oggi una delle tipologie di attacchi informatici più diffuse, in quanto facilmente attuabili e molto difficili da riconoscere e da gestire. Lo scopo di questi attacchi è quello di rendere indisponibile un servizio web saturandone le risorse, ritardando o impedendo quindi ai client legittimi l'accesso a tale servizio.
Un generico DDoS viene realizzato come segue: l'attaccante infetta un insieme di calcolatori, creando delle backdoor al fine di poterli controllare all'insaputa dei proprietari; i computer che compongono questa particolare tipologia di rete, definita *botnet*, vengono chiamati *zombie*. A questo punto l'attaccante istruisce tali *zombie* in modo che inviino contemporaneamente un'ingente quantità di richieste ad uno specifico target. Il successo di un DDoS dipende chiaramente dalla dimensione della *botnet* e dall'intensità di richieste che quest'ultima è in grado di inviare, nonché da eventuali meccanismi di difesa instaurati da parte della vittima.
Esistono diversi tipi di attacchi DDoS, ognuno dei quali sfrutta diverse caratteristiche dei sevizi web e della rete stessa. La simulazione di questo tipo di attacchi risulta utile sia per la parte attaccante, sia per la controparte che si occupa di valutare diversi meccanismi di difesa, stimando l'impatto dell'attacco sul sistema vittima e studiandone le caratteristiche.

2 Obiettivo

Lo scopo di questa simulazione consiste nel configurare una rete composta da un server (la vittima), alcuni client *legittimi* che inviano richieste al server, alcuni *zombie* (la parte attaccante) che inviano un'ingente quantità di richieste ping al server (realizzando un attacco di tipo *ICMP-flood*), una serie di nodi intermedi, ed infine un *defender*, ovvero una componente del server che si occupa di filtrare le richieste degli zombie tramite l'analisi del traffico in arrivo. Una volta modellato il sistema, l'obiettivo della simulazione è quello di valutare l'impatto dell'attacco sulle prestazioni del server vittima e dei nodi intermedi, al variare dell'intensità dell'attacco e della capacità delle risorse di rete.

3 Implementazione

3.1 Modello

Questo progetto è stato realizzato tramite il framework di simulazione Omnet++ 4.6, che grazie alla libreria INET permette di modellare una rete reale basata su applicazioni e protocolli tipicamente utilizzati dai servizi web. Gli elementi del nostro modello sono quindi basati su oggetti INET:

- **server**: oggetto *StandardHost* che utilizza l'applicazione *TCPSinkApp* per ricevere dati in arrivo da connessioni TCP, e *PingApp* per rispondere alle richieste ping;
- **client**: oggetto *StandardHost* che utilizza l'applicazione *TCPSessionApp* per inviare dati mediante il protocollo TCP;
- **zombie**: oggetto *StandardHost* che utlizza l'applicazione *PingApp* per inviare ping alla vittima;
- **router**: oggetto *Router* che si occupa di inoltrare le richieste dei nodi alle rispettive destinazioni;
- **defender**: estensione dell'oggetto *NodeBase*, creato da noi per analizzare e filtrare il traffico in ingresso al server.

Come mostrato in Fig.1, la topologia è composta da quattro gruppi di calcolatori, sia client che zombie, ognuno collegato ad un router intermedio. Ogni router intermedio è collegato a sua volta ad un router centrale, al quale sono collegati anche un gruppo di soli client e il server vittima. Questa configurazione è stata scelta per convogliare le richieste dell'intera rete su un unico canale diretto al server, in modo da poter analizzare accuratamente il traffico in entrata e valutare l'impatto dell'attacco sull'infrastuttura di rete che collega il server alla rete "esterna".

3.2 Attacco

Per realizzare il DDoS abbiamo scelto di simulare l'attacco ICMP-flood, che consiste nell'invio simultaneo di un altissimo numero di richieste ping da parte degli

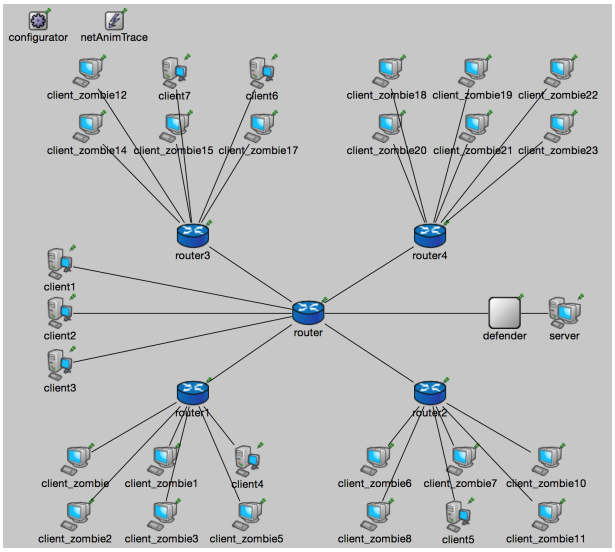

Fig. 1. Topologia della simulazione

zombie: l'obbiettivo è quello di saturare le risorse fisiche della macchina server e, soprattutto, la bandwidth in ingresso, mediante l'utilizzo di una bandwidth d'attacco maggiore di quella del target. Questo provoca un notevole rallentamento nella comunicazione tra client legittmi e server e, in caso di attacchi molto intensi, oltre al canale vengono saturati anche i buffer di rete dei nodi intermedi, provocando lo scarto dei pacchetti che non possono essere accodati quando il canale è occupato ed impedendo quindi l'arrivo delle richieste legittime a destinazione. Nel nostro modello gli zombie utilizzano l'applicazione *PingApp*, i cui parametri ci permettono di specificare le caratteristiche dell'attacco:

- `sendInterval`: intervallo che intercorre tra l'invio di un ping e il successivo; nel nostro caso caratterizza l'intensità dell'attacco;
- `startTime/stopTime`: tempo di inizio e fine dell'invio dei ping; caratterizzano la durata dell'attacco;
- `destAddr`: specifica la destinazione dei ping (il server vittima).

3.3 Difesa

Il meccanismo di difesa consiste nel monitoraggio del traffico in ingresso al server mediante l'utilizzo di un oggetto creato da noi a tale scopo, il *defender*. Quest'ultimo è un'estensione dell'oggetto INET *NodeBase*, provvisto quindi delle interfacce necessarie per connettersi ad una rete fisica e progettato come un generico nodo di rete.
Il defender si comporta come uno sniffer, analizzando tutti i pacchetti che transitano dalla rete esterna verso quella del server e differenziando la tipologia di controllo in due fasi distinte, seguite eventualmente dalla fase difensiva:

– **learning mode**: in questa fase lo scopo è quello di analizzare l'utilizzo del servizio in condizioni di traffico considerato legittimo, ovvero in assenza di alcun tipo di attacco. Viene quindi monitorato il traffico indirizzato verso il server ad intervalli di tempo regolari, registrandone il picco massimo.

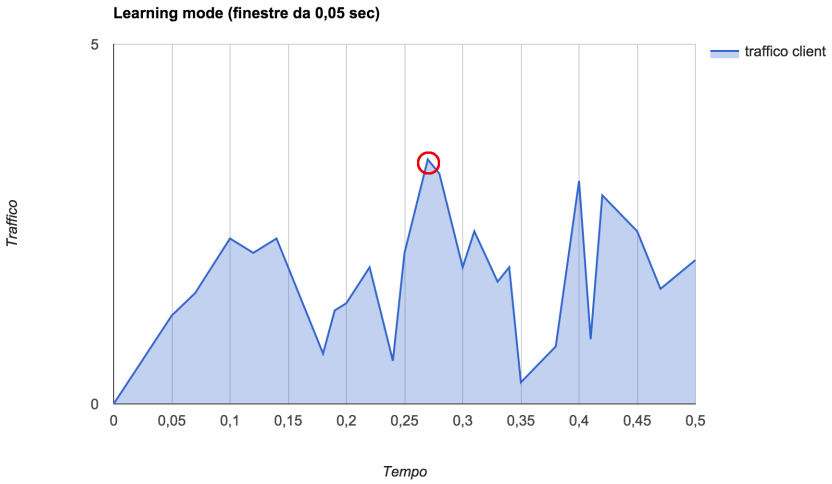

Fig. 2. Esempio di learning mode

In Fig.2 viene mostrato un esempio di learning mode: gli intervalli utilizzati sono di 0,05 secondi, ed il picco massimo registrati è indicato in rosso.

– **monitor mode**: seconda fase, in cui il traffico viene costantemente esaminato, con due obiettivi distinti:

  - **confronto**: il traffico in entrata viene confrontato con i dati raccolti nella learning mode. L'analisi viene suddivisa in finestre temporali e, tramite un contatore, viene tenuta traccia di quante finestre consecutive superino il picco massimo. Durante l'incremento del contatore si entra in uno stato d'allerta e vengono salvati in una tabella tutti gli indirizzi IP attivi, accumulando la rispettiva intensità di traffico[1] generata in tale fase; viene inoltre impostato un threshold facendo un calcolo tra il picco minimo e il valore medio del traffico di tutti gli IP.

    In Fig.3 viene mostrata graficamente questa politica di configurazione del threshold, il quale verrà eventualmente utilizzato nella *defense mode*

[1] Nel nostro caso, l'intensità del traffico è relativa al numero di pacchetti ricevuti, piuttosto che alla dimensione in bytes, in quanto più efficiente per l'attacco ICMP-flood.

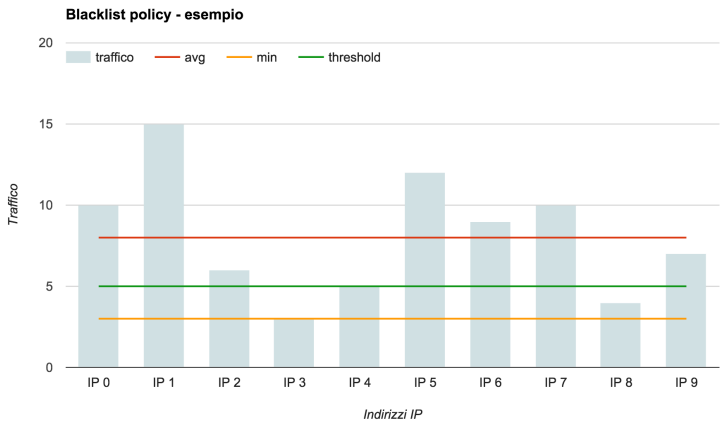

Fig. 3. Impostazione del threshold.

per stabilire quali siano gli IP che devono essere bloccati.

Riguardo all'analisi delle finestre temporali, quando il contatore supera una certa soglia, l'eccessiva intensità di richieste viene considerata anomala e sintomo di un potenziale attacco DDoS; in questo caso si entra in uno stato di allarme e viene azionato il meccanismo di difesa.

In Fig.4 viene mostrato un esempio di confronto: abbiamo un picco massimo di 2,5 registrato nella learning mode ed una soglia di 10 finestre consecuitive di allarme prima di azionare la difesa. Ad ogni finestra di allerta il contatore, indicato in rosso nella figura, viene incrementato; come mostrato, tale contatore viene eventualmente decrementato in presenza di successive finestre in cui il traffico scende sotto il valore critico. In questo esempio l'ultima finestra di confronto fa scattare la fase di allarme, con conseguente attivazione del meccansimo di difesa.

- **filtraggio**: per ogni pacchetto in arrivo, viene fatto un controllo sull'indirizzo IP di provenienza; se tale indirizzo appartiene alla black list il pacchetto viene direttamente scartato. In questo modo il server rimane protetto, ricevendo solamente le richieste ritenute legittime.

– **defense mode**: in questa terza fase viene interrogata la tabella degli IP allo scopo di inserire nella black list gli indirizzi ritenuti malevoli sulla base del threshold preventivamente impostato. Per ogni IP inserito nella black list viene calcolato un tempo di attesa sulla base dell'eccesso di traffico generato rispetto al threshold. In questo modo ogni indirizzo, prima di poter inviare nuovamente richieste al server, dovrà attendere una quantità di tempo proporzionata alla percentuale di traffico in eccesso che stava generando. Inoltre

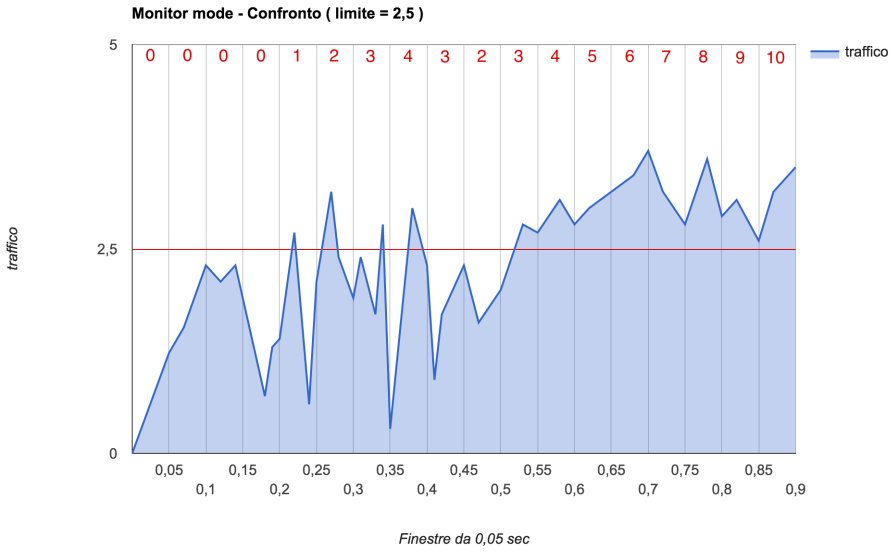

Fig. 4. Esempio di fase di allerta e successiva fase di allarme, quando in contatore raggiunge il valore 10.

eventuali IP legittimi, inseriti nella black list come conseguenza di un casuale picco di richieste, avranno probabilisticamente un timeout minore rispetto agli zombie.

3.4 Defender

In questa sezione presentiamo alcuni dettagli relativi all'implementazione del modulo *defender*.
I parametri che ne regolano il comportamento, configurabili dal NED file, sono i seguenti:

- `sample_time`, intervallo di tempo utilizzato per il conteggio bit/pacchetti, sia durante la fase di learning, sia durante il monitoraggio del traffico;
- `sups_w_limit`, limite di finestre di allerta; se superato, fa scattare il meccanismo di difesa;
- `learning_time`, stabilisce la durata di *learning mode*;
- `defense_mode`, variabile booleana per abilitare disabilitare il filtraggio dei pacchetti.

Per quanto riguarda le funzioni, implementate in `defender.cc`, abbiamo:

- `handleMessage`: si occupa di due compiti distinti, in base al tipo di messaggio ricevuto:
  - richiama la funzione `Update_black_list` e schedula il successivo aggiornamento della *black list*;

    - controlla la variabile `defense_mode` ed in caso affermativo abilita il meccanismo di difesa, occupandosi inoltre di :
        * scartare eventuali pacchetti provenienti da indirizzi IP contenuti nella *black list* (e quindi considerati malevoli);
        * chiamare `IncomingTrafficMonitor` per i pacchetti provenienti dagli altri IP;
- `Update_black_list`: controlla ed elimina dalla *black list* eventuali IP il cui tempo di attesa è scaduto;

- `insertBlackList`: preso in input un IP, lo inserisce nella *black list* e ne imposta un tempo di scadenza, stabilito in base al quantità di pacchetti in accesso rispetto al threshold impostato in *monitor mode*;

- `Update_ip_table`: stabilisce innanzitutto la metodologia di monitoraggio (numero di pacchetti o numero di bit). Successivamente, preso in input un IP lo aggiunge alla `ip_table` (se non già presente) e ne aggiorna in contatore di traffico generato. Infine aggiorna la variabile `threshold`;

- `IncomingTrafficMonitor`: si tratta della funzione principale, responsabile del monitoraggio del traffico e della gestione del meccanismo di difesa. Prende in input ogni pacchetto ricevuto dal defender ed effettua tutti i controlli già descritti. Inoltre, quando il traffico si stabilizza, ovvero non vengono rilevate finestre di allerta per una quantità di tempo prestabilita (nel nostro caso 500 finestre di controllo), la tabella degli indirizzi IP attivi viene resettata.

4 Esperimenti e analisi

I client sono configurati in modo da generare traffico sulla rete trasferendo sul server una somma di byte generata in modo random (uniforme), per un numero fisso di trasfermenti TCP schedulati ad intervalli di 0,5 secondi l'uno. Il server riceve all'inizio della simulazione il numero totale in byte delle richieste dei client (al momento della loro generazione), conta il numero di byte ricevuti e, quando i client finiscono di trasferire, registra il tempo di termine delle richieste. Il server riceve i trasferimenti TCP dei client e risponde immediatamente ai ping ricevuti, senza ritardi.
L'attacco è stato simulato configurando l'applicazione *PingApp* sugli zombie in modo che inviino richieste ping alla vittima con una frequenza di 100 ping/s, 250 ping/s, 500 ping/s e 1000 ping/s. L'invio del ping ha il 20% di probabilità di fallire, in modo da introdurre un errore casuale nell'invio dei ping dovuto a possibili malfunzionamenti della rete o degli zombie stessi.
Gli esperimenti sono configurati in modo da variare l'intensità dell'attacco come descritto e la dimensione dei buffer delle interfacce di rete, variando la capacità delle code di 50, 100 e 150 pacchetti. Ogni configurazione viene ripetuta per 30 volte, ognuna delle quali corrispondenti ad una serie di 20 trasferimenti di un numero random di byte da parte dei client. Il range di dimensioni generate per

le richieste varia tra 2KB e gli 8KB durante la learning mode, in modo da non settare un throughput massimo troppo alto, e tra 2KB e 30KB durante l'attacco, in modo da rendere meno netta la differenza tra client e zombie nella quantità di pacchetti generati.
La durata dell'attacco è impostata a 16s (da T=4s fino a T=20s), la learning mode a 3s e l'intervallo della finestra di monitor del traffico, `sample_time`, a 0,05 secondi. Il livello critico per l'avvio della difesa, `susp_w_limit`, è di 10 sample consecutivi. Infine, la bandwidth dei canali di comunicazione è impostata a 10Mbps.
Per valutare l'impatto dell'attacco sul server vittima e l'efficienza della strategia di difesa implementata, sono state definite le seguenti metriche:

- **Drop Quality**: percentuale dei pacchetti scartati, provenienti dagli zombie identificati dal defender;
- **Server Throughput**: somma dei bit inoltrati al server, dopo essere stati processati e filtrati dal defender;
- **Link Utilization**: percentuale di saturazione del canale in ingresso alla vittima;
- **Termine trasmissione client**: tempo di trasferimento dei client, al variare della dimensione dei buffer di rete.

Dai risultati ottenuti per le configurazioni degli esperimenti appena descritte, una prima considerazione riguarda l'entità dell'attacco in corso, condizionata dalla frequenza con cui gli zombie inviano i ping al server: per frequenze "basse" di invio delle richieste (100 ping/s), i canali di comunicazione e i nodi intermedi non vengono stressati abbastanza da produrre ritardi o alterazioni sul traffico generato dai client. Inoltre, durante l'attacco, il defender, e di conseguenza il server, non riceve una quantità di traffico sufficiente a far intervenire il meccanismo di difesa (tranne che per sporadiche eccezioni, in caso di trasferimenti dei client particolarmente intensi).

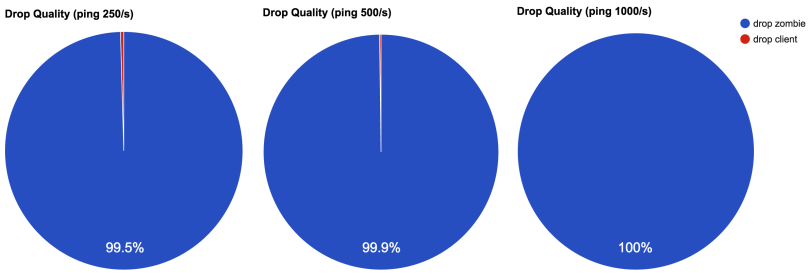

Fig. 5. Drop Quality misurata su diverse intensità di attacco

4.1 Drop Quality

Nella Fig.5 vengono mostrati i risultati relativi alla Drop Quality, calcolata sulla media delle run di ogni configurazione. La percentuale di pacchetti scartati provenienti dagli zombie è generalmente molto alta (oltre il 99%) e incrementa all'aumentare della frequenza di invio dei ping. Ciò accade in quanto il numero di pacchetti inviati dai singoli zombie è, in proporzione, molto più alto rispetto a quello generato dalle richieste dei client: l'incremento dell'intensità di attacco facilita l'identificazione degli indirizzi che stanno trasmettendo un'eccessiva quantità di richieste.
Riguardo ai client, se il numero di pacchetti inviati durante lo stato di allarme risulta simile a quello degli zombie (ad esempio a causa di un picco di richieste) può capitare che questi vengano erroneamente inseriti nella blacklist, come si può vedere nel caso in cui la frequenza di attacco è più bassa. Questa situazione di incertezza sarebbe più frequente in un caso reale, in cui il numero degli zombie è tipicamente più alto e la frequenza di invio dei ping è solitamente più bassa; questo influirebbe negativamente sull'accuratezza di scarto.

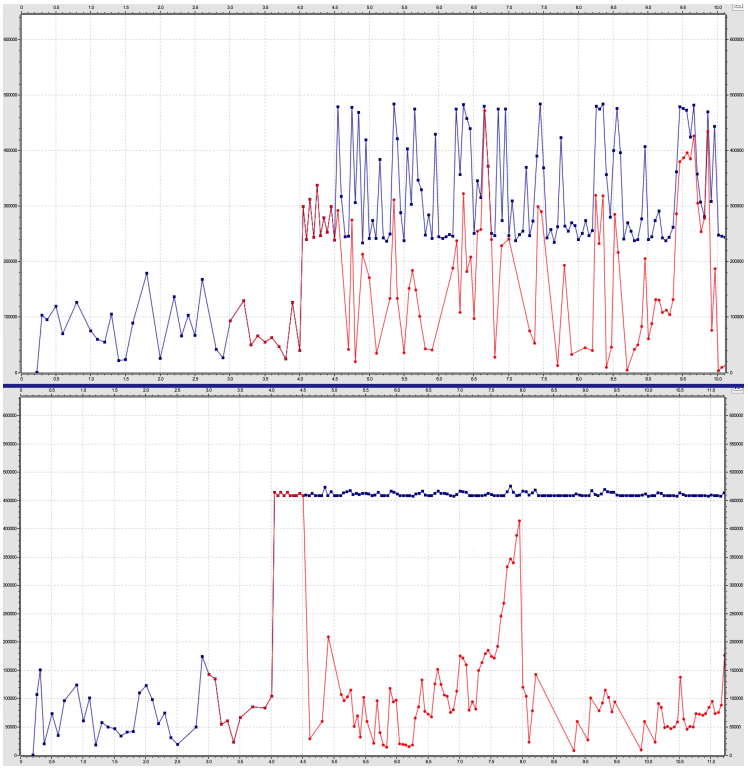

Fig. 6. Throughput con e senza filtro, nei due casi di intensità 500 ping/sec e 1000 ping/sec

4.2 Throughput

A titolo di esempio, la Fig.6 mostra il throughput in arrivo al server, prima e dopo l'applicazione del filtro del defender, nei due casi di attacco più intenso (500 e 1000 ping/s) e riferite ad una singola run. Mediante il meccanismo di difesa descritto, solo le richieste provenienti da indirizzi considerati "legittimi" vengono inoltrate al server, evitando quindi di stressare eccessivamente le sue risorse computazionali e vanificando il potenziale attacco DDoS.
Dai grafici si può notare che per un certo periodo, a partire dal termine della learning mode, il throughput in ingresso e quello filtrato coincidono. Dopo l'inizio della fase di attacco (dal secondo 4 in poi), questi valori iniziano a discostarsi nel momento in cui il defender entra nello stato di allarme e viene innescato il meccanismo di difesa.
Man mano che gli IP filtrati vengono liberati, il throughput in arrivo al server ricomincia a crescere, generando un nuovo periodo di allerta in caso l'attacco sia ancora in corso e riapplicando di conseguenza il filtro sul traffico in ingresso allo scattare di un nuovo allarme.

4.3 Link Utilization

La Fig.7 mostra la percentuale di utilizzazione del canale, prima e dopo l'applicazione del filtro del defender, nei due casi di attacco più intenso (500 e 1000 ping/s) e riferite ad una singola run.
Naturalmente, più è alto il traffico generato dagli zombie, maggiore sarà la saturazione del canale in ingresso al server prima del defender; nei casi in cui tale saturazione raggiunga il livello massimo, potrebbe verificarsi un considerevole rallentamento (o addirittura lo scarto) delle richieste dei client. Come mostrato in Fig.7, l'utilizzazione effettiva del canale dal defender al server risulta molto più bassa grazie al filtro applicato sui pacchetti provenienti da IP considerati malevoli.

La Fig.8 presenta la saturazione media del canale in ingresso tra il defender e il server e gli intervalli di confidenza al 95%, calcolati su ogni secondo di simulazione, a partire dall'inizio dell'attacco ICMP-flood. Come mostrato, in conseguenza del filtraggio del defender, l'occupazione è più variabile per l'intensità di 500 ping/s rispetto a 1000 ping/s; questo è dovuto ad una maggior incertezza della politica di filtraggio nel primo caso, provocando una variazione nell'utilizzazione del canale molto più ampia.

4.4 Termine trasmissione dei client

In Fig.9 viene mostrato il termine della trasmissione dei client in tre condizioni differenti, caratterizzate dalla variazione del buffer delle interfacce di rete, impostata a 50, 100 e 150 pacchetti, rispettivamente. Per questa particolare metrica

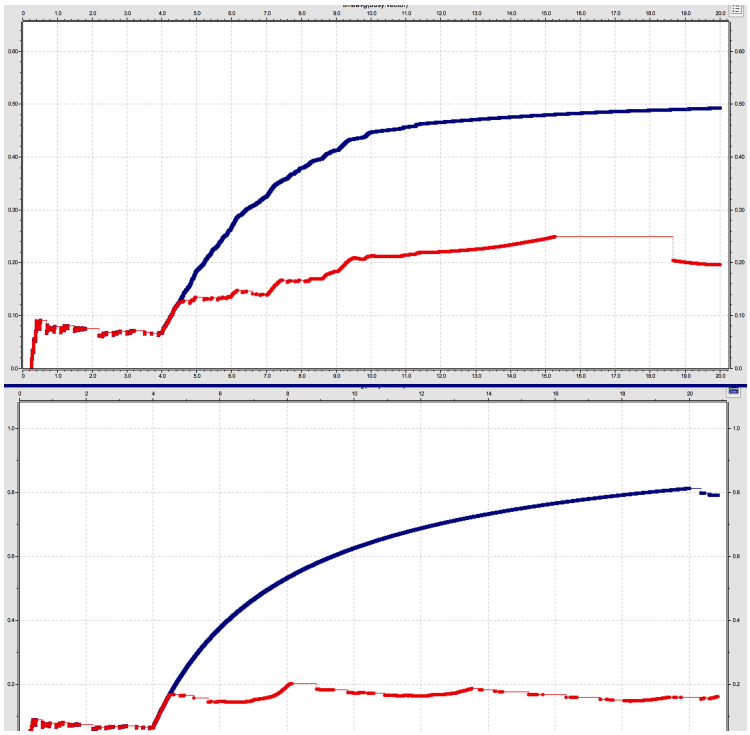

Fig. 7. Utilizzazione con e senza filtro, nei due casi di intensità 500 ping/sec e 1000 ping/sec

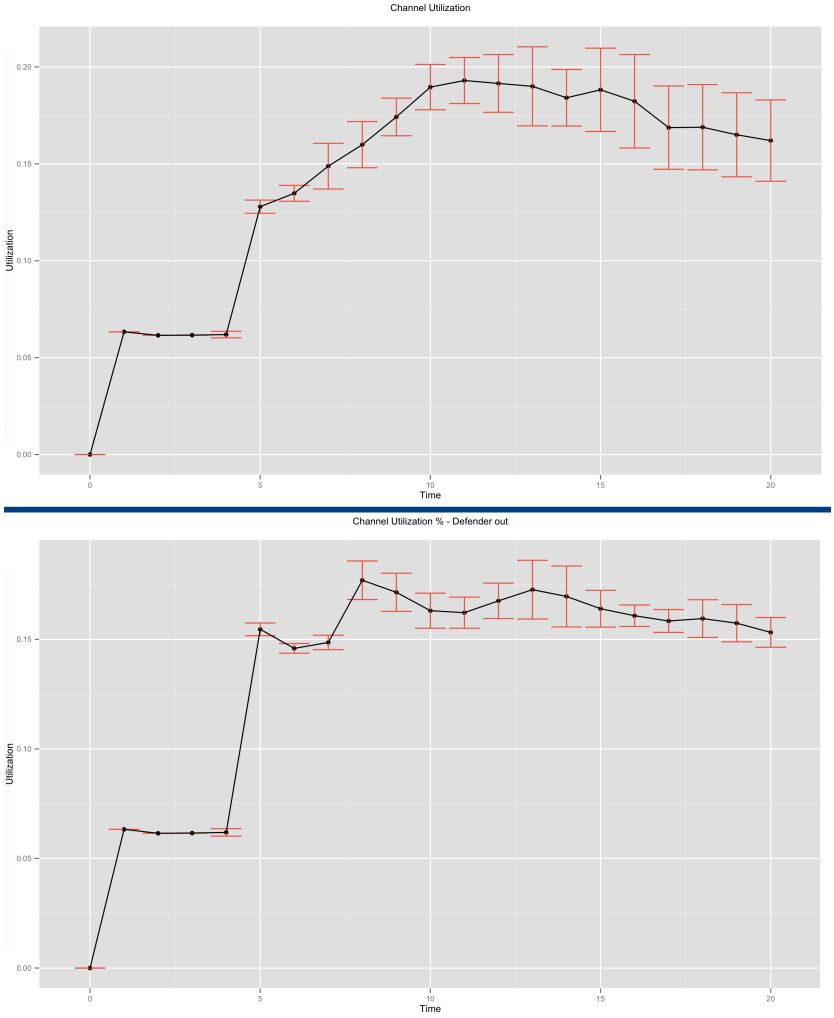

Fig. 8. Utilizzazione media, nei due casi di intensità 500 ping/sec e 1000 ping/sec

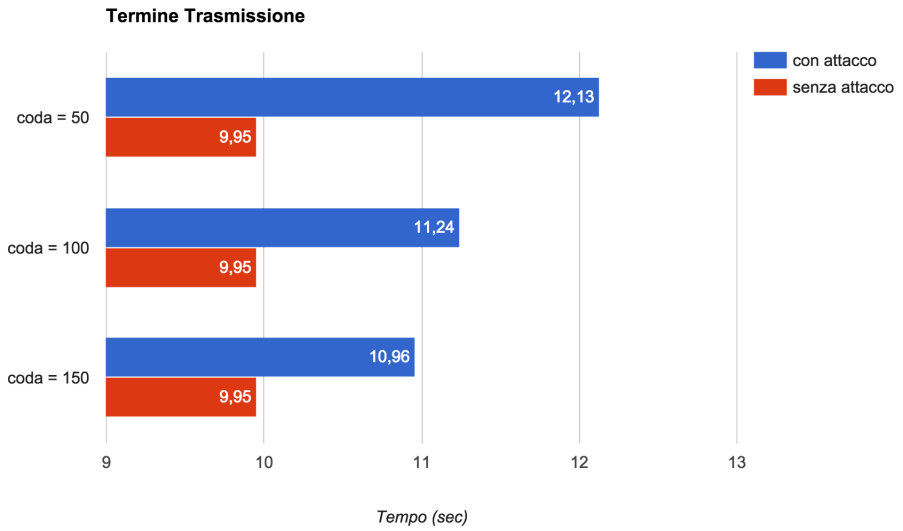

Fig. 9. Tempo di trasmissione dei client, al variare del buffer dei router, con intensità fissata a 500 ping/sec

il traffico generato dai client è stato fissato, in modo da poter confrontare correttamente la durata dei trasferimenti nei tre casi.
Come si evince dai risultati, la durata dei trasferimenti diminusice all'aumentare della dimensione dei buffer, in quanto questi vengono saturati dall'ingente quantità di richieste ping, provocando lo scarto dei pacchetti che non possono essere accodati. Di conseguenza, viene ritardato notevolemente il termine dei trasferimenti dei client, che sono costretti a reinviare parte delle richieste.

5 Conclusioni

Il meccanismo di difesa implementato funziona in modo corretto ed è sufficientemente accurato. Questo approccio risulta maggiormente efficiente qualora vengano considerati diversi pattern di traffico corrispondenti all'utilizzo del servizio web in differenti fasce orarie e periodi (ad esempio, nei periodi festivi l'utilizzo del servizio potrebbe risultare maggiore rispetto alla media annuale). A tale scopo si potrebbe eseguire una *learning mode* per ogni fascia oraria e periodo di riferimento, in modo da creare una serie storica dell'utilizzo del servizio e monitorare il traffico con differenti criteri in base ad una specifica circostanza.
Va però sottolineato che il meccanismo di difesa è stato studiato appositamente per gli attacchi ICMP-flood e potrebbe non risultare altrettanto efficace contro altre tipologie di DDoS. I principi alla base del defender sono in ogni caso da considerarsi validi, in particolare riguardo al funzionamento della *learning mode*. In un caso reale, infatti, un approccio basato su tale tecnica risulta utile contro la

maggior parte delle tipologie di attacco, a patto di utilizzarla in modo corretto. Un secondo aspetto che è risultato particolarmente interessante riguarda la politica di impostazione del tempo di permanaza nella blacklist. Ricordando che tale tempo viene stabilito in base all'eccesso di traffico generato da ogni IP rispetto ad un threshold, abbiamo potuto verificare come un qualsiasi client, inserito erroneamente in blacklist a causa di un particolare picco di richieste, torni ad essere attivo prima di qualsiasi zombie. Questo perché il traffico in eccesso generato dagli zombie risulta molto più intenso rispetto ai client e quindi il ritardo applicato risulta proporzionalmente più alto.
Il meccanismo di difesa implementato si occupa di limitare i danni al server (*victim-end protection*), filtrando i pacchetti provenienti dai client che stanno trasmettendo un numero eccessivo di richieste, ma non ha alcun effetto sulla congestione dei nodi intermedi e dei canali di comunicazione. L'approccio di filtraggio sui router (*intermediate-router protection*) è quello migliore per identificare alla fonte gli IP che stanno inviando le richieste e proteggere le risorse di rete da attacchi di tipo flood [2]. Come possibile raffinamento, in un caso reale, abbiamo considerato la possibilità di diffondere la blacklist ai router di rete esterni, in modo da poter filtrare il traffico anomalo più efficacemente; un tale approccio risulterebbe valido solamente nel caso in cui la blacklist sia effettivamente accurata.
Questo caso di studio ci ha permesso di verificare ulteriormente, e personalmente, quanto gli attacchi DDoS siano difficili da riconoscere e allo stesso tempo facili da realizzare, una volta costruita una botnet. Come già accennato, ogni metodo di difesa efficace sembra essere strettamente legato alla tipologia di attacco e risulta quindi estremamente arduo implementare un meccanismo generico e flessibile, per garatire un buon livello di affidabilità contro diversi tipi di DDoS.

References

[1] Kaur, Lal Sangal, Kumar (2014). "Modeling and Simulation of DDoS Attack using Omnet++".

[2] Wang, Sun (April 2014). "An IP-Traceback-based Filtering scheme for eliminating DDoS attacks".

[3] Kotenko, Ulanov (2006). "Simulation of Internet DDoS Attacks and Defense".

[4] Omnet Manual, https://omnetpp.org/doc/omnetpp/manual/usman.html