Elaborato del corso

SICUREZZA DELLE RETI E DEI SISTEMI SOFTWARE

A. A. 2015/2016

Studenti:		Docente:
Esposito Alessandro	399/92	Prof. Corrado Aaron Visaggio
Ocone Luciano	399/96	Collaboratore:
Pino Luigi	399/88	Ing. Antonio Pirozzi

1. Introduzione

Il tool *s2ipt,* realizzato nel Maggio del 2016 e distribuito sotto licenza GNU, permette di effettuare la traduzione delle regole Snort in regole Iptables, dal contenuto semantico equivalente.

Snort è un *Intrusion Detection System* (IDS) open-source, attualmente sotto licenza GNU GPLv2, che permette l'analisi del traffico di una determinata rete IP e quindi l'individuazione di potenziali minacce ed intrusioni provenienti dall'esterno.

Creato da Martin Roesh nel 1998 e attualmente sviluppato da Cisco Systems, Snort realizza un vero e proprio firewall a livello applicativo, in grado di effettuare analisi real-time dei pacchetti che fluiscono all'interno della rete (attraverso un'interfaccia specificata), offrendo all'utente tre diverse modalità di funzionamento: **sniffer**, **packet logger** e **network intrusion detection**. In quest'ultima modalità il tool è in grado di filtrare i pacchetti osservati in base ad alcune valutazioni effettuate sul loro contenuto, tramite opportune regole definite dall'utente poste in input al sistema. Il grande successo di Snort ha fatto sì che nel corso degli anni si formasse una community che si occupa di raccogliere e rilasciare le regole inerenti alle più recenti e diffuse minacce nell'ambito delle reti IP.

Con Iptables si fa riferimento ad un programma applicativo a livello utente, presente all'interno delle distribuzioni Linux, che permette di configurare opportunamente NetFilter, un componente del kernel dello stesso sistema operativo. NetFilter permette di intercettare e filtrare il traffico che attraversa la macchina ospitante, realizzando un vero e proprio firewall a livello kernel. Il sistema è basato su regole organizzate in catene (*chain*), a loro volta raggruppate in tabelle, dove ogni tabella definisce un diverso tipo di operazione che è possibile effettuare sui pacchetti. Sono previste quattro possibili tabelle (*filter*, *nat*, *mangle* e *raw*), ognuna delle quali prevede delle catene predefinite; le regole che realizzano le funzionalità di filtraggio, e che consentono quindi di implementare il firewall, sono quelle incluse all'interno della tabella *filter*, raggruppabili in tre diverse catene di default:

- INPUT: tutti i pacchetti in ingresso alle varie interfacce di rete della macchina
- OUTPUT: tutti i pacchetti creati dalla macchina e destinati verso l'esterno
- FORWARD: tutti i pacchetti instradati dalla macchina, cioè né creati né destinati ad essa.

Iptables è quindi da intendersi come uno strumento per la configurazione del firewall di Linux, permettendo all'utente di specificare le regole da inserire nelle varie tabelle gestite da NetFilter.

Sebbene le regole Snort e quelle Iptables affrontano tematiche comuni all'interno dello stesso contesto, non è sempre possibile definire un mapping di tipo 1:1 tra le componenti sintattiche dei due formati. Per questo motivo, *s2ipt* effettua la traduzione delle regole Snort in formato Iptables, utilizzando un approccio "*best-effort*", cercando cioè di tradurre una regola, in assenza di corrispondenza diretta, al massimo delle funzionalità offerte dalla semantica Iptables.

2. Obiettivi e vantaggi del tool

La funzionalità di traduzione da regole Snort ad Iptables offerta da *s2ipt* ha come obiettivo principale quello di permettere ad un utente di poter configurare in modo semplice ed automatico un firewall Linux dall'elevata robustezza. In questo senso, quindi, *s2ipt* può essere visto come uno strumento che permette di configurare Iptables come IDS o IPS, a seconda delle necessità dell'utente.

Uno dei principali vantaggi derivanti dall'utilizzo del tool è quello infatti di poter permettere anche ad un utente inesperto nell'ambito della sicurezza delle reti IP di ottenere "a costo zero" una protezione verso le più recenti e diffuse minacce presenti in questo contesto, le cui informazioni sono racchiuse all'interno delle regole Snort. A differenza di Snort, infatti, NetFilter (che può essere configurato tramite Iptables) è un modulo presente in tutte le distribuzioni Linux, quindi non richiede installazione di software aggiuntivo, né di hardware particolarmente performante. Per questo motivo, è possibile pensare di utilizzare come IDS/IPS qualsiasi dispositivo, anche low-cost (ad esempio Raspberry Pi) o qualsiasi altro sistema embedded, rendendo così *s2ipt* uno strumento altamente poiché si procede sempre più nella direzione dell'*Internet of Things*.

S2ipt è un tool da riga di comando, sviluppato in python, che permette di effettuare questo tipo di traduzione in ambiente Linux, specificando una serie di opzioni:

- Specifica dell'*interfaccia di rete*: tramite l'opzione `--iface IFACE` bisogna specificare l'interfaccia di rete sulla quale applicare le regole Iptables.
- *Log mode:* tramite l'opzione `--log` è possibile specificare la possibilità di aggiungere alle regole tradotte in Iptables l'opzione `-j LOG`, in questa modalità di esecuzione, le regole Iptables vengono aggiunte alla custom chain *IDS*.
- *Drop mode*: tramite l'opzione `--drop` è possibile specificare la possibilità di aggiungere alle regole tradotte in Iptables l'opzione `-j DROP`, in questa modalità di esecuzione, le regole Iptables vengono aggiunte alla custom chain *IPS*.

- *Reject mode*: tramite l'opzione `--reject` è possibile specificare la possibilità di aggiungere alle regole tradotte in Iptables l'opzione `-j REJECT`, in questa modalità di esecuzione, le regole Iptables vengono aggiunte alla custom chain *IPS*.
- *Revert mode*: tramite l'opzione `--revert` è possibile eliminare le regole Iptables generate precedentemente da *s2ipt*.

Delle precedenti opzioni, `--iface IFACE` deve essere sempre esplicitata (a meno di utilizzare `--revert`), mentre solo una tra `--log`, `--drop` e `--reject` può essere indicata in fase di esecuzione. L'opzione `--revert`, invece, ha precedenza sulle altre: se specificata, vengono eliminate le regole tradotte, ignorando le altre opzioni.

Il tool prevede inoltre di effettuare periodicamente l'aggiornamento delle regole Snort dalla relativa community, con l'obiettivo quindi di permettere all'utente di poter avere a disposizione le regole più recenti. In questo modo, *s2ipt* permette consente di avere a disposizione un firewall a livello kernel sempre aggiornato.

Quindi, *s2ipt* può essere sia catalogato come un sistema di IDS se utilizzato come semplice strumento per monitorare il traffico di rete (*log mode*), sia come un IPS (*Intrusion Prevention System*) nel caso in cui viene utilizzato per configurare il NetFilter (tramite l'applicazione delle regole Iptables) ad applicare dei filtri sui pacchetti (*drop* o *reject mode*).

In generale, trattandosi di un sistema che consente all'utente di aumentarne le operazioni di controllo tramite l'applicazione di regole che facciano riferimento alle caratteristiche degli end-point della rete, è catalogabile come un *target-based intrusion detection*.

S2ipt permette quindi di combinare la potenza della sintassi Snort con la velocità del kernel Linux e la semplicità dei comandi Iptables, in uno strumento che nel complesso è in grado di abbattere tutto l'overhead presente in soluzioni che operano a livello applicativo. Oltre ad un netto miglioramento in termini di prestazioni, il fatto che il tool operi a livello kernel consente di ottenere altri benefici legati al filtraggio dei pacchetti.

Un problema classico che si presenta in un contesto applicativo come quello di Snort è legato alla frammentazione dei pacchetti. Per evitare attacchi implementati tramite segmentazione Snort offre un layer di deframmentazione all'interno del pre-processore *frag3*, che mette a disposizione del sistema diversi algoritmi utili allo scopo. In tali contesti, è possibile riscontrare effettivamente l'attacco subìto utilizzando lo stesso algoritmo di deframmentazione della macchina bersaglio: se ciò non accade, il comportamento malevolo potrebbe essere mal interpretato o addirittura non riscontrato. Tale problema invece non si presenta operando a livello kernel in

quanto l'unico algoritmo utilizzato per la deframmentazione è quello previsto dal comune stack IP.

3. Approccio utilizzato

Come già anticipato in precedenza, non è sempre possibile stabile una corrispondenza diretta tra una regola Snort ed una espressa con una sintassi Iptables. Per questo motivo, quando ciò si verifica, *s2ipt* effettua una traduzione in "best-effort" delle regole, al massimo delle funzionalità offerte dalla sintassi Iptables. Il motivo principale di questo disallineamento tra le due sintassi risiede nel fatto che il sistema Snort mette a disposizione dell'utente dei meccanismi di alto livello (come ad esempio un motore di espressioni regolari) che non sono invece disponibili in Iptables.

In generale una regola Snort presenta la seguente struttura, formata da due componenti principali:

header (options)

dove la prima parte ha una struttura fissa mentre la seconda è variabile a seconda del numero di opzioni presenti nella regola.

L'*header* nello specifico presenta, da sinistra a destra, i seguenti campi:

- *Action:* viene indicata l'azione da eseguire in presenza del pacchetto specificato dalla regola. Le regole prese in considerazione dal tool sono quelle che presentano la parola chiave *alert* che forza, in presenza di un pacchetto che matcha la regola, la generazione di un messaggio di alert verso l'output specificato.
- *Protocol:* il tipo di protocollo indicato nel pacchetto (Snort supporta TCP, UDP, ICMP e IP).
- *Source IP:* indirizzo IP sorgente
- *Source port:* numero di porta sorgente
- *Operator:* freccia che indica la direzione del pacchetto; sono consentite operatori in entrata (→, indicano che il pacchetto proviene dalla *source* ed è diretto verso la *destination*) oppure bidirezionali (↔ in entrambe le direzioni).
- *Destionation IP:* indirizzo IP destinazione
- *Destination port:* numero di porta destinazione

Le opzioni supportate da Snort sono molteplici e combinano facilità di utilizzo con potenza e flessibilità. Tutte le opzioni inserite all'interno di una regola sono comprese tra parentesi tonde ed ognuna di essa viene espressa nella seguente forma:

keyword: value;

dove *keyword* rappresenta il tipo di opzione specificata e con *value* il suo valore corrispondente.

Tali opzioni possono essere classificate in quattro macro categorie:

- *General*: forniscono informazioni circa la regola in questione (ad esempio il codice identificativo della stessa) ma non hanno effetti circa la sua applicazione.
- *Payload*: permettono di ricercare dati all'interno del payload di un pacchetto e più regole di questo tipo possono essere correlate tra loro all'interno di una stessa regola.
- *Non-payload*: permettono di ricercare informazioni al di fuori del payload del pacchetto.
- *Post-detection*: permettono di specificare dei meccanismi che vanno ad innescarsi dopo il matching della regola.

Per quanto riguarda le regole Iptables, possono essere definite da riga di comando in un sistema operativo Linux based a partire dall'omonimo comando, in una struttura complessiva del tipo:

*iptables [-t <table-name>] <command> <chain-name> *
*<parameter-1> <option-1> *
<parameter-n> <option-n>

dove:

- *<table-name>* specifica a quale tabella applicare le regole; se omesso la tabella di default sarà *filter*.
- *<command>* specifica l'azione da compiere, come ad esempio aggiungere o cancellare una regola.
- *<chain-name>* specifica la catena da modificare, creare o rimuovere.
- Coppie *<parameter>-<option>* rappresentano i parametri e i valori delle opzioni associate che specificano come elaborare un pacchetto che fa scattare una determinata regola.

L'elevata flessibilità permessa dalla sintassi Iptables fa in modo che la lunghezza e la complessità di una singola regola possa variare significativamente in relazione al suo obiettivo. Quando viene costruita una regola di questo tipo è tuttavia necessario tener

ben presente che alcuni parametri e relative opzioni ne richiedono espressamente altre ad esse correlate affinché l'intera regola sia valida.

In generale, al fine di creare una qualsiasi regola Iptables valida, è necessario fissare almeno i tre aspetti chiave:

- *Packet type* – il tipo di pacchetto da filtrare/monitorare.
- *Packet source/destination* – il tipo di pacchetti da filtrare sulla base dei campi di origine e destinazione.
- *Target* – che tipo di azione eseguire quando la regola in questione viene matchata.

3.1 Algoritmo di traduzione

L'approccio utilizzato per la traduzione si basa come detto su una filosofia di tipo "best-effort", dovuta dalla mancanza di corrispondenza diretta tra la sintassi Snort e quella Iptables. In questo modo l'algoritmo definito cerca, in presenza di mapping 1:1 di tradurre "fedelmente" la regola Snort analizzata, mentre negli altri casi si cerca per quanto possibile di riformulare la stessa semantica nella sintassi consentita da Iptables. Se quest'ultimo tentativo non può essere realizzato, la traduzione dell'intera regola fallisce.

In generale l'intero meccanismo di traduzione si basa sull'analisi della regola Snort in relazione alla sua struttura. Secondo tale principio, l'algoritmo si occupa prima di andare a tradurre la componente *header* della regola di partenza per poi concentrarsi sulle singole opzioni Snort presenti.

Di seguito vengono riportate le strategie di traduzione implementate per ogni singolo campo dell'*header* di una regola Snort:

- *Action:* la sintassi Snort supporta otto diverse tipi di action, tuttavia *s2ipt* prende in considerazione solamente le regole con il tipo *alert* (corrispondente all'azione Snort di logging della regola).
- *Protocol:* in questo caso abbiamo un tipo di mapping 1:1 con l'opzione di Iptables `-p protocolname` dove è possibile specificare tutti i protocolli supportati da Snort.
- *Source/destionation IP:* per specificare gli indirizzi IP, sorgente e destinazione, si utilizzano rispettivamente i comandi `-s IPaddress` e `-d IPaddress`. Tuttavia la sintassi Snort prevede per questi campi, in alcuni casi, l'utilizzo di variabili statiche etichettate come *$HOME_NET* e *$EXTERNAL_NET* (rappresentanti la rete locale della macchina e una rete esterna) che non sono supportante da Iptables e per questo non tradotte

dall'algoritmo. Snort permette inoltre di specificare indirizzi IP multipli (separati da virgole), classi di indirizzi IP (nella notazione *CIDR*) e la negazione (tramite il carattere *'!'* specificato prima di `-s` o `-d`) di un determinato indirizzo IP. Queste opzioni vengono supportate da Iptables (e per questo tradotte da *s2ipt*) tranne nel caso particolare in cui viene definita la negazione di indirizzi multipli, caso ammesso dalla sintassi Snort. In particolare Iptables utilizza la stessa notazione per descrivere questi casi particolare, con la sola eccezione dell'utilizzo di parentesi quadre nella definizione di indirizzi multipli, sintassi non ammessa in Iptables.

- *Source/destination port:* Snort supporta tre diversi valori per i campi relativi ai numeri di porta sorgente/destinazione. È possibile infatti specificare un particolare numero di porta o, in alternativa, evitare questa possibilità utilizzando l'etichetta `any` o definirne un intervallo tramite la definizione di un range. In Iptables per specificare il numero di porta sorgente e destinazione vengono utilizzate le opzioni `--sport value` e `--dport value` dove con *value* si indica il valore associato all'opzione. In questo caso non vi è una corrispondenza diretta tra le etichette utilizzabili in Snort ma è comunque possibile effettuare la traduzione seguendo opportuni accorgimenti. Oltre a permettere la specifica di un particolare numero di porta (come in Snort), Iptables permette anche di definirne un intervallo, indicandone gli estremi divisi dal carattere ':'. In questo modo, per definire qualsiasi numero di porta (corrispondete `any` di Snort) è sufficiente non specificare alcuna porta in particolare; per definire un particolare intervallo vengono specificati nello stesso formato gli estremi (se uno dei due è omesso si considera il valore minimo di default per l'estremo inferiore o massimo per quello superiore). Snort permette anche la negazione di particolari numeri di porta e la loro definizione multipla (tramite il separatore virgola), opzioni che posso anche essere opportunamente combinate. In Iptables, nel primo caso, come previsto anche da Snort, bisogna anteporre un carattere *'!'* prima della relativa opzione. Per definire un insieme di numeri di porta, Iptables richiede di specificare il modulo `-m multiport` prima dei normali comandi previsti per la specifica dei numeri di porta. Nel caso in cui si vanno a combinare le opzioni di indirizzi multipli e di specifica di un range, ancora una volta bisogna includere il modulo multiport ma l'opzione da aggiungere per specificare i numeri di porta diventa `--sports` o `--dports`.
- *Operator:* l'etichetta di direzione espressa in Snort non trova corrispondenza diretta in Iptables, dove la stessa informazione viene rappresentata tramite il concetto di *chain*; nel caso di *s2ipt*, viene creata una

custom chain, che è consultata dall'engine di Iptables dopo quelle di default (vedi paragrafo 3.1.1 per approfondimenti).

Una volta tradotto l'*header* di una regola Snort, l'algoritmo passa in rassegna le opzioni presenti, analizzandole una per una e provando ad effettuarne la traduzione. Non tutte le opzioni previste in Snort possono essere tradotte in Iptables perché non supportate; mentre è possibile effettuare un mapping tra quelle più comunemente utilizzate.

Di seguito vengono riportate tutte le opzioni Snort che il tool *s2ipt* riesce a tradurre nel formato Iptables e, per ciascuna di esse, vengono riportate le assunzioni e le strategie definite in fase di progettazione:

- *Content:* permette di cercare all'interno del payload a livello applicativo un determinata sequenza di byte, applicando l'algoritmo di Boyer-Moore. L'estensione di Iptables che consente di effettuare lo "string matching" utilizza una versione a livello kernel dello stesso algoritmo ed è selezionabile tramite il comando `–m string --string "string" --algo bm,` dove *string* è la stringa che si vuole ricercare nel payload.
  Snort permette anche di indicare all'interno del campo *content* dei valori codificati in notazione esadecimale, compresi all'interno di due caratteri '|'. *S2ipt* in questi casi traduce l'opzione tramite l'opzione Iptables `–m string --–hex-string` seguito dal valore specifico da ricercare.
- *Uricontent:* permette di gestire i dati a livello applicativo codificati in URL, trasferiti su HTTP. *S2ipt* gestisce questa opzione allo stesso modo della precedente.
- *Offset:* se presente indica di ricercare il contenuto del payload (indicato dal campo content) a partire dal valore indicato dall'opzione. In Iptables è possibile specificare la stessa opzione mediante il comando `–-from` a cui si fa seguire il numero di byte desiderato.
- *Depth:* impone che tutti i tentativi di matching su un pacchetto non possano eccedere un determinato numero di byte successivi all'inizio del payload. In Iptables è possibile specificare la stessa opzione tramite il comando `--to` seguito dal numero di byte desiderato. È importante sottolineare che questa opzione e la precedente possono trovarsi in più di un'istanza all'interno di una regola Snort, in quanto possono far riferimento a diverse stringhe da ricercare all'interno di un pacchetto. Questo tuttavia non è consentito in Iptables, che invece consente la presenza al massimo di un un'unica istanza dei rispettivi comandi, con il vincolo che il numero di byte indicati con `--from` sia minore o uguale di quello specificato con `–-to`. Nel caso in cui viene presa in considerazione una regola Snort con più occorrenze di queste

opzioni, *s2ipt* effettua la somma dei valori dei campi *depth* (`--to`) considerando il valore minimo del campo *offset* (`--from`).

- *TTL:* permette di matchare un pacchetto in base al valore TTL presente all'interno dell'header IP. Si tratta di un comando abbastanza flessibile che permette di confrontare il valore TTL con un valore specificato tramite gli operandi, *less than* (minore di), *equals to* (uguale a) e *greather than* (maggiore di), specificabili tramite i seguenti comandi:
  - `ttl:<value`
  - `ttl:value`
  - `ttl:>value`

  In Iptables è possibile specificare le stesse opzioni, nel modo seguente:
  - `--ttl-lt value`
  - `--ttl-eq value`
  - `--ttl-gt value`
- *Tos:* indica di analizzare i bit presenti all'interno del campo *Type Of Service (TOS)* dell'header IP. Permette di accettare un valore numerico specificato, o di negarlo tramite il carattere '*!*'. Questa opzione è supportata da Iptables tramite il comando `–m tos --tos value`, aggiungendo, in presenza di negazione, il carattere '`!`' in testa al comando.
- *Flow:* permette di applicare criteri circa lo stato e la direzione nei confronti di un flusso di dati TCP. Snort in particolare mette a disposizione diversi costanti per indicare un particolare stato di una connessione; per esempio la combinazione di *from_client* ed *established* permette di analizzare solo i pacchetti dal lato client di una connessione TCP dopo che il three-way handshake è stato completato. In Iptables è possibile filtrare lo stato di una connessione TCP tramite il comando `–m state --state CONNECTION_STATE`. Tuttavia Iptables, al contrario di Snort, mette a disposizione solo cinque possibili stati da poter osservare, tramite le costanti INVALID (pacchetti che non possono essere correlati a nessuna particolare connessione), ESTABLISHED (connessione stabilita con traffico in entrambe le direzioni), NEW (pacchetti che contengono il flag TCP SYN), RELATED (pacchetti apparentemente appartenenti ad una diversa connessione ma correlati ad una già stabilita) e UNTRACKED (non è possibile correlare il pacchetto a nessuna connessione). *S2ipt* prende in considerazione solo regole Snort che fanno riferimento allo stato ESTABLISHED, uno dei pochi che trova correlazione diretta anche in Iptables.
- *Ip_proto:* consente di restringere il raggio d'azione di una regola in base al valore del campo *protocol* dell'header IP. Iptables permette di specificare la

stessa opzione tramite il comando `-p` seguito dal tipo di protocollo desiderato.

- *Pcre:* acronimo che sta per *Perl Compatible Regular Expression*; permette a Snort di applicare espressioni regolari dall'elevata complessità. Iptables invece non supporta alcun tipo di espressione regolare, motivo per cui questa è l'opzione su cui il tool deve utilizzare maggiormente l'approccio in "best-effort". A causa infatti della limitazione in tal senso imposta da Iptables, *s2ipt* riesce a tradurre solo un piccolo sottoinsieme di tutte le possibili varianti dell'opzione *pcre* di Snort. In particolare, l'algoritmo di traduzione, riesce a convertire in Iptables solo le regole la cui opzione `pcre` contiene dei valori costanti (ad esempio semplici stringhe), riconducendo tale opzione al caso della già citata *content*. Oltre a questo, altro caso in cui è possibile ricondursi ad una notazione Iptables, è quello in cui con tale opzione si descrive un pattern statico a cui si va a specificare un valore numero indicato tra parantesi graffe. Tale valore rappresenta il numero di byte, a partire dall'inizio del payload, dove il pattern indicato può presentarsi; tale scenario viene ricondotto in Iptables al comando `string` (per definire il pattern) a cui va aggiunto il comando `-m length --length`, specificando un valore numerico pari a quello indicato nella regola Snort più i corrispondenti valori dei vari header dei livelli inferiori.

Infine Snort prevede anche una classe di opzioni contenenti metadati che fanno riferimento alla singola regola come `classtype`, `sid` e `reference` (ed altre ancora…) che però vengono ignorate da *s2ipt* perché non utili ai fini della traduzione.

Sulla base delle considerazioni mostrate in precedenza, l'algoritmo complessivo di traduzione utilizzato da *s2ipt* si compone di quattro fasi principali:

- *Pre-processing:* il tool effettua una prima valutazione preliminare delle regole per filtrare quelle che sicuramente non possono essere tradotte a causa di una mancata corrispondenza con la sintassi Iptables. Le regole che possono essere tradotte vengono opportunamente organizzate in una rappresentazione interna al tool che va a discriminare la parte costituente l'header da quelle contenente le opzioni.
- *Header translation:* vengono tradotti opportunamente tutti i campi dell'header della regola Snort oggetto della traduzione.
- *Options translation*: vengono tradotti opportunamente tutte le opzioni della regola Snort oggetto della traduzione.
- *Synthesis:* il risultato della traduzione, a partire dalla rappresentazione interna al tool, viene trasformato in una stringa opportunamente utilizzata come comando Iptables.

3.2 Politica di best-effort

La diversità dei due strumenti oggetti di interesse (Snort ed Iptables) rende impossibile, come citato più volte in precedenza, la traduzione perfetta di tutte le regole Snort in input. Questa considerazione ci ha portato alla definizione di una politica di "best-effort" secondo la quale, se non è possibile tradurre perfettamente la regola, il tool tenta di fare tradurla in maniera simile, per quanto possibile.

Entrando nei dettagli della politica:

- se la regola Snort contiene una *action* diversa da `alert`, la regola non è tradotta;
- se il numero di opzioni Snort presenti nella regola in input, è maggiore del numero di opzioni che il tool riesce a tradurre (tralasciando quelle di tipo *general*, come sid, metadata, rev…), la regola viene considerata tradotta in "best effort";
- se la regola contiene l'opzione `pcre`, si verifica la presenza di pattern non traducibili; in tal caso la regola scartata, altrimenti la regola è considerata tradotta in "best effort";
- se la regola contiene, come indirizzo IP sorgente/destinazione, la negazione di un range, la regola non è tradotta;
- se la regola non contiene un opzione di tipo *content*, non è tradotta in quanto considerata troppo generica (al netto delle opzioni che è possibile tradurre).

Per ogni regola, quindi, vengono effettuati una serie di controlli al fine di verificare se rientra in uno dei casi sopra citati.

3.3 Configurazione del tool nel sistema

Uno degli obiettivi del tool è certamente quello di rappresentare una soluzione *lightweight* per l'applicazione di regole al firewall Iptables dei sistemi Linux. Allo stesso tempo, *s2ipt* ha l'intento di impattare il meno possibile sull'ambiente in cui viene eseguito, lasciando invariato lo stato del firewall e le regole precedentemente (e successivamente) inserite in esso.

Per raggiungere lo scopo, è stata prevista una catena separata su cui andare ad inserire le regole Iptables, che assume dei nomi diversi a seconda che la politica scelta sia quella di loggare semplicemente i messaggi (il nome della catena sarà IDS), oppure quella di effettuare il drop o il reject dei pacchetti (la catena sarà nominata IPS). Tutte le regole, quindi, saranno aggiunte a tale catena. Di norma, però, le uniche catene

consultate da Iptables alla ricezione/invio di un pacchetto, sono quelle di default: INPUT, OUTPUT, FORWARD. Per tale motivo, allo scopo di rendere visibile anche la catena custom creata dal tool, è prevista l'aggiunta di una regola del tipo:

```
iptables -A DEFAULT_CHAIN -j CUSTOM_CHAIN
```

dove DEFAULT_CHAIN sarà il nome della catena a cui dovrà essere aggiunta la regola, mentre CUSTOMC_CHAIN rappresenta il nome assegnato alla catena custom, cioè IDS o IPS. Tale regola viene quindi aggiunta a tutte e 3 le tabelle di default (INPUT, OUTPUT, FORWARD), per permettere la consultazione delle nuove regole generate dal tool.

Ciò però non basta per garantire la "consistenza" dell'ambiente: infatti, nel momento in cui l'engine di Iptables passa dalla consultazione di una tabella all'altra (es. da INPUT ad IDS a causa dell'istruzione di *jump*), se il pacchetto in questione non fa scattare nessuna regola, non viene ripresa la consultazione la vecchia tabella. Per abilitare un comportamento del genere (nel caso in cui l'amministratore del sistema aggiunga nuove regole Iptables alle catene di default, dopo l'esecuzione del tool), è necessaria l'aggiunta di un'ulteriore regola alla custom chain, del tipo:

```
iptables -A CUSTOM_CHAIN -j RETURN
```

In questo modo, se un pacchetto non dovesse far scattare nessuna regola né all'interno della catena di default (es. INPUT), né all'interno della catena custom (es. IDS), l'engine di Iptables riprenderà la consultazione della tabella originaria (INPUT nell'esempio specifico), rendendo così minimale l'impatto del tool sull'intero sistema. Ancora una volta, essendo NetFilter un modulo del kernel di Linux, il nuovo sistema IDS/IPS non occupa risorse né di memoria aggiuntive (RAM o Storage - il tool *s2ipt* di per sé occupa pochi KB), né richiede elevate prestazioni a livello computazionale (il fatto di agire a livello kernel fa sì che i pacchetti analizzati non hanno bisogno di essere copiati in "user-space" come sarebbe avvenuto su sistemi applicativi, abbattendo così gran parte dell'overhead).

4. Architettura di s2ipt

S2ipt è un tool sviluppato interamente in linguaggio python la cui struttura si compone di tre package:

- domain_classes: contiene i moduli con le definizioni delle strutture utilizzare per gestire la rappresentazione del dominio del problema. In particolare s2ipt utilizza due strutture principali, una per gestire opportunamente il contenuto di una regola Snort e un'altra per conservare in maniera strutturata il risultato della traduzione di una singola regola.

- engine: rappresenta il core del sistema contenente tutta la logica di traduzione dell'applicazione.

- utils: contiene un modulo dove vengono definite le costanti utilizzate in fase di traduzione.

Per maggiori dettagli è possibile osservare il diagramma seguente dove viene mostrata nello specifico l'architettura di s2ipt.

Insieme al codice python, il tool è rilasciato con 2 script bash:

- install.sh, presente nella root directort;

- s2ipt-daemon.sh, collocato nella directory "daemon".

Il primo script ha lo scopo di verificare i prerequisiti del sistema, ossia una versione dell'interprete python ≥ 2.7.3, di Iptables ≥ 1.4.12 e del kernel di Linux ≥ 2.6.12, necessaria per alcune opzioni delle regole Iptables. Dopo questa operazione preliminare, lo script:

- effettua il deployment del tool (creazione della gerarchia delle directory e copia dei file);

- esegue un backup delle regole Iptables pre-esistenti;

- configura l'esecuzione del demone s2ipt-daemon.sh per gli aggiornamenti;

- scarica ed estrae le ultime regole della community di Snort.

Se al momento dell'installazione, per qualsiasi motivo, fosse impossibile reperire le ultime regole dal sito di Snort, il tool avvisa l'utente, ed in un secondo momento sarà possibile eseguire tale opzione separatamente specificando l'opzione da riga di comando '-d': install.sh -d.

Lo script s2ipt-daemon.sh, invece, viene eseguito ad ogni riavvio del sistema ed allo scattare della mezzanotte di ogni giorno (configurazione eseguita al momento dell'installazione, sfruttando l'utility crontab dei sistemi Linux). Ad ogni esecuzione, lo script va a leggere in opportuni file di configurazione la data dell'ultimo update e l'intervallo di aggiornamento indicato dall'utente (rispettivamente nei file "daemon.config" e "s2ipt-update.config"), e verifica se la giornata in questione è quella designata per l'aggiornamento (o eventualmente una successiva). In caso positivo, esegue nuovamente lo script di installazione con l'opzione '-d' specificata, altrimenti lascia tutto invariato.

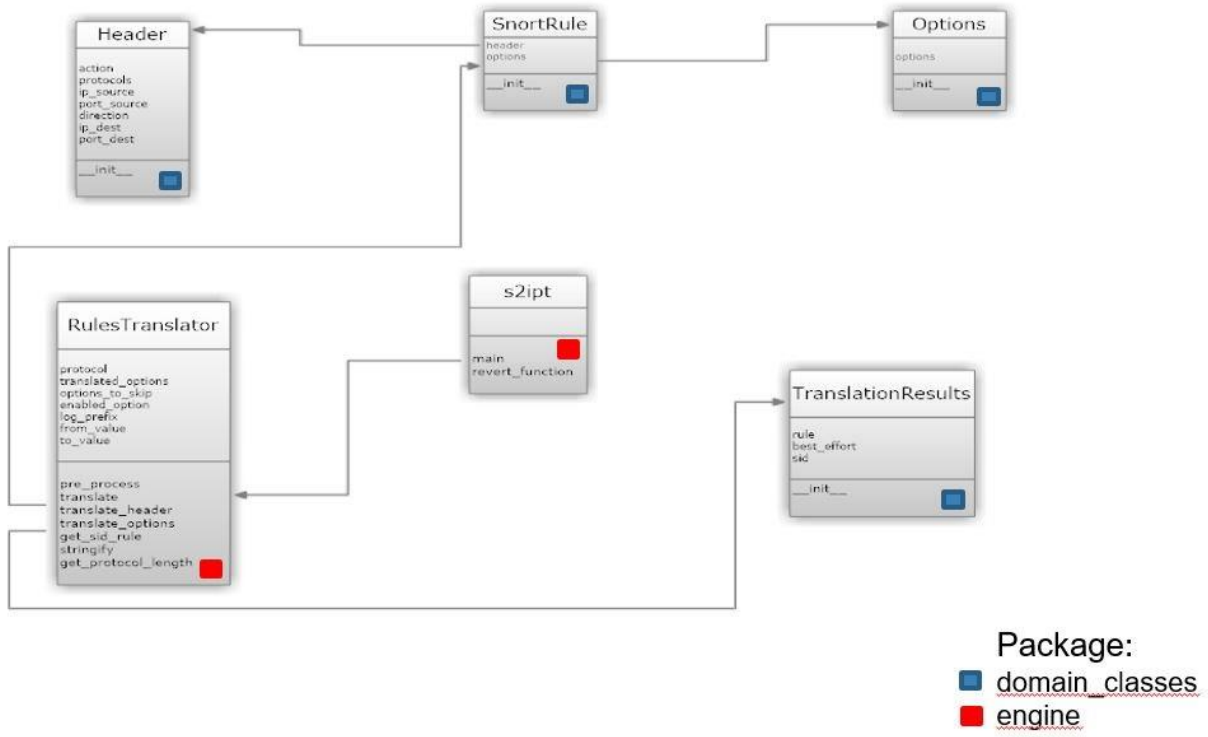

Figura 1. Architettura s2ipt

5. Esempi di traduzione

In questo paragrafo sono riportanti alcuni esempi di traduzione di particolari regole Snort, sulla base dell'approccio descritto in precedenza, al fine di mostrare praticamente il risultato del processo di traduzione attuato da *s2ipt*.

Come primo esempio prendiamo in considerazione una semplice regola Snort del tipo:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"MALWARE-
BACKDOOR Infector.1.x"; flow:established,to_client;
content:"WHATISIT"; metadata:ruleset community;
reference:nessus,11157; classtype:misc-activity; sid:117;
rev:16;)
```

Tale regola prevede di tracciare tutti i pacchetti che fanno riferimento ad una connessione TCP stabilita (dopo il three-way handshake) diretti verso l'applicazione client, il cui contenuto a livello applicativo presenti la stringa "WHATISIT". Un pacchetto avente queste caratteristiche verrà riportato da Snort in un messaggio di log etichettato come "MALWARE-BACKDOOR Infector.1.x". Una regola del genere,

non contiene nessuna opzione non supportata da Iptables, motivo per cui, verrà normalmente tradotta da *s2ipt*. In particolare il processo di traduzione analizzerà prima il contenuto dell'header, per poi passare in rassegna le varie opzioni della regola comprese tra parentesi tonde. Il risultato del processo di traduzione sarà il seguente:

```
iptables -I CUSTOM_CHAIN 1 –i IFACE –p tcp –m state --state
ESTABLISHED –m string --string "WHATISIT" --algo bm -m comment
--comment "MALWARE-BACKDOOR Infector.1.x" –j LOG --log-prefix
[IDS-117]
```

dove, come è facile osservare, le opzioni riguardanti i metadati sono stati ignorati (metadata, reference, classtype, sid e rev), mentre le restanti sono state tratte così come esplicitato nel paragrafo successivo. In particolare, CUSTOM_CHAIN indica la catena su cui inserire la regola (IDS o IPS) ed IFACE rappresenta l'interfaccia interessata (eth0, wlan0, …). In aggiunta a quanto indicato in precedenza, la regola Iptables prevede l'inserimento di un commento alla regola (corrispondente al campo *msg* in Snort) e la specifica di un prefisso di LOG, del tipo "`IDS/IPS-sid`".

Come secondo esempio viene presa in considerazione un'altra regola Snort molto semplice che presenta, come unica opzione presa in considerazione, il vincolo sul campo TTL dell'header IP:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"PROTOCOL-
ICMP digital island bandwidth query";
content:"mailto|3A|ops@digisle.com"; depth:22;
metadata:ruleset community; classtype:misc-activity; sid:1813;
rev:9;)
```

Questa regola impone al sistema Snort di andare a filtrare tutti i pacchetti del protocollo ICMP un contenuto di tipo `mailto|3A|ops@digisle.com`, dove `|3A|` indica un carattere esadecimale, ad una profondità di 22 Byte nel pacchetto. Applicando le osservazioni descritte in precedenza otteniamo l'equivalente regola nella sintassi Iptables:

```
iptables -I CUSTOM_CHAIN 1 -i IFACE -p icmp -m string --string
"mailto" -m string --string "ops@digisle.com" -m string --hex-
string "|3A|" --from 22 -m comment --comment "PROTOCOL-ICMP
digital island bandwidth query" -j LOG --log-prefix [IDS-1813]
```

Infine, come ultimo esempio, viene presa in considerazione una regola Snort leggermente più complessa che comprende al suo interno diverse opzioni tra le quali anche la *pcre* che come già descritto, rappresenta un caso particolare di traduzione.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"FTP SITE
CHOWN overflowattempt"; flow:to_server,established;
content:"SITE";nocase; content:"CHOWN"; distance:0; nocase;
isdataat:100,relative; pcre:"/^SITE\s+CHOWN\s[^\n]{100}/smi";
reference:bugtraq,2120; reference:cve,2001-0065;
classtype:attempted-admin; sid:1562; rev:11;)
```

La regola in questione impone a Snort di tenere traccia di tutti i pacchetti transitanti all'interno di una connessione TCP già avviata (dopo che il three-way handshake è terminato), indirizzati al processo client. Questi pacchetti, inoltre, devono contenere le parole "SITE" e "CHOWN" all'interno del payload, a partire da 100 byte dopo l'inizio del contenuto di livello applicativo. La regola Iptables successiva mostra il risultato dell'algoritmo utilizzato da *s2ipt* per la traduzione della regola Snort; in questo caso l'opzione *pcre* è traducibile in quanto contiene come valore una stringa dal contenuto fisso. L'informazione relativa al numero di byte viene trasferita all'interno della sintassi Iptables attraverso il comando `-m length --length 140:` (valore dato dai 100 byte del payload a cui si sommano i 20 dell'header IP e i 20 di quello TCP, seguito dai ':' poiché indica che 140 sarà la lunghezza minima del pacchetto):

```
iptables -I CUSTOM_CHAIN 1 –i IFACE -p tcp -m state --state
ESTABLISHED -m string --string "SITE" --algo bm -m string --
string "CHOWN" --algo bm -m comment --comment "FTP SITE CHOWN
overflowattempt" -m length --length 140: -j LOG --log-prefix
[IDS-1562]
```

6. Copertura delle regole

Per avere una prima valutazione circa il livello di copertura delle regole e quindi della capacità di traduzione dell'algoritmo sviluppato, è stato lanciato *s2ipt* con dei dati di input costituiti dalle regole della community Snort del 16/06/2016. In particolare, a fronte di 3419 regole prese in considerazione, il tool è riuscito a tradurne 2860, cioè circa l'84%. Di quelle tradotte circa l'80% è stato frutto di un approccio in best-effort, 2268 regole, mentre le restanti 592 sono state tradotte perfettamente dal tool.

Una percentuale così elevata di regole in best-effort è giustificabile essenzialmente da due aspetti principali. Per prima cosa, è opportuno considerare che molte opzioni non vengono considerate da parte dell'algoritmo di traduzione perché non avevano una corrispondente opzione in Iptables. Inoltre poiché la sintassi Iptables non prevede l'utilizzo di pattern di espressioni regolari, ogniqualvolta l'algoritmo incontra l'opzione *pcre* in una regola Snort e riesce a tradurla, si effettua l'ipotesi conservativa per cui tale traduzione sia stata fatta in best-effort.

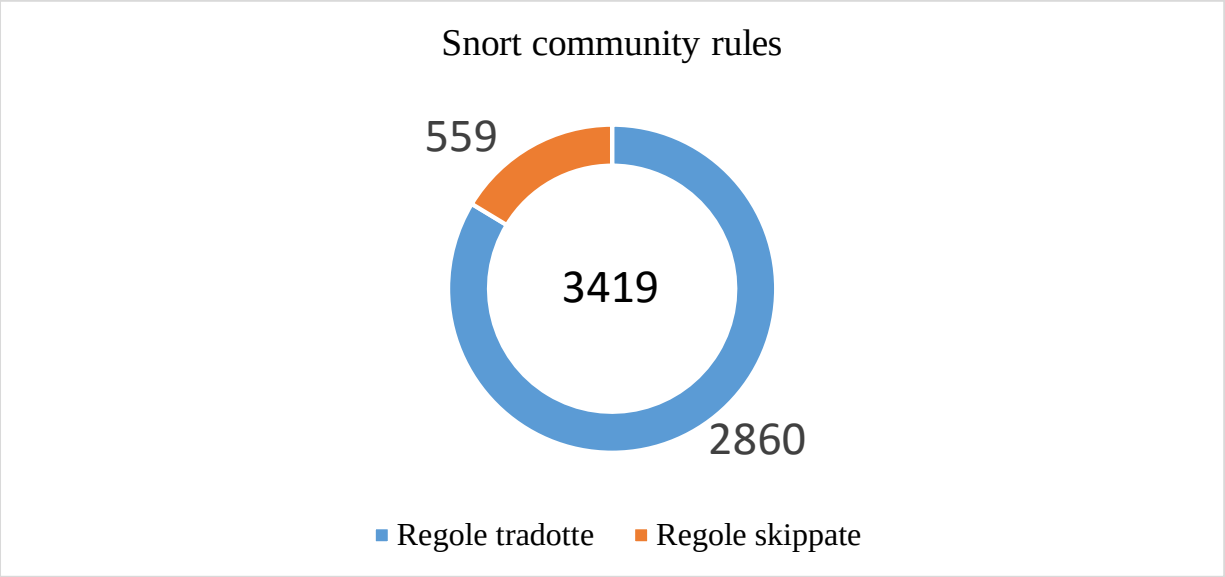

Figura 2. Risultati s2ipt Snort community rules 16/06/2016

Figura 3. Coverage traduzione s2ipt Snort community rules 16/06/2016

7. Testing

L'obiettivo della fase di testing è rivolto a validare il risultato a valle dell'esecuzione di *s2ipt,* ovvero l'accuratezza delle regole Iptables prodotte. Poiché tali regole vengono generate a partire da regole Snort semanticamente equivalenti, il testing si è focalizzato sull'analisi dei comportamenti osservati tra l'IDS Snort e il firewall Iptables (configurato con *s2ipt*) a fronte di pattern di traffico identici. Chiaramente i due sistemi sono stati testati con una configurazione che si basa sullo stesso insieme di regole Snort.

In particolare l'attività di testing è composta di due fasi principali: una rivolta all'analisi di Snort e l'altra di Iptables configurata con *s2ipt.* Infine sono stati confrontati i comportamenti dei due tool con lo scopo di verificare il grado di similitudine esistente tra essi.

Per ogni sistema sono stati considerati due differenti scenari:

- Pattern di traffico malevolo: sono stati generati dei pacchetti sulla base delle regole Snort prese in considerazione. In questo modo, il risultato di Snort è stato considerato l'oracolo per lo stesso test effettuato con Iptables, al fine di determinare le differenze riguardi il matching dei pacchetti effettuati tra i due sistemi.
- Pattern di traffico normale: sono stati riprodotti dei pattern di traffico contenenti pacchetti di uso "comune" al fine di verificare la presenza di eventuali falsi positivi.

Entrambi i casi i test sono stati effettuati a partire dalle 2860 regole Snort pubblicate dalla stessa community il 16/06/2016, che *s2ipt* è stato in grado di tradurre.

Di seguito vengono riportati in dettaglio le due fasi di test, contenenti i due scenari effettuati per ciascun IDS, con i relativi risultati.

7.1 Testing Snort

Il testing di Snort è stato effettuato su una macchina stand-alone su cui è stato configurato opportunamente tale IDS con le regole della stessa community. Ciò è stato possibile dal momento che tale sistema agisce a livello applicativo.

Per quanto riguarda il primo scenario di test, è stato utilizzato il tool *rule2alert* per generare i pacchetti a partire dalle regole Snort indicate nel file di configurazione. Successivamente tali pacchetti sono stati inoltrati verso la stessa macchina in modo da osservare il comportamento dell'IDS. In particolare il testing si è soffermato sul numero di regole che hanno generato degli alert da parte del sistema, in modo da utilizzare tale valore come oracolo per la stesso scenario applicato ad Iptables.

A partire dalle 2860 regole della community Snort prese in considerazione, *rule2alert* ha prodotto 2181 file *.pcap*. Successivamente questi pacchetti sono stati analizzati direttamente da Snort, tale analisi ha prodotto 1671 alert, riuscendo cioè a riscontrare un comportamento malevolo in circa il 75% dei casi. Tale valore costituisce l'oracolo per lo stesso test effettuato su Iptables e rappresenta di fatto il valore ideale di pacchetti matchati dopo aver lanciato il tool *s2ipt*.

Per quanto riguarda il secondo scenario, è stata effettuata la stessa analisi per pattern di traffico "comuni" inerenti a diversi protocolli: BITTORENT, HTTP, LDAP, MYSQL, SIP, SNMP, PPPOE e TELNET.

I risultati di tale test vengono riportati in Tabella 1, dove è possibile osservare i protocolli che hanno prodotto alert da parte di Snort e il numero di pacchetti utilizzati per simulare i corrispondenti pattern di traffico.

PROTOCOL	ALERT	PACKETS
BITTORENT	1	53
HTTP	0	43
LDAP	0	13
MYSQL	4	57
SIP & RTP	3	32
SNMP	30	89
PPPOE	0	28
TELNET	0	92

Tabella 1. Risultati testing traffico normale Snort

7.2 Testing Iptables

Il testing di Iptables è stato effettuato in un ambiente composto da due macchine, una opportunamente configurata con Iptables tramite *s2ipt* mentre l'altra è stata utilizzata per lanciare i pacchetti destinati alla prima. In particolare, tali pacchetti sono stati gli stessi generati per il caso precedente tramite il tool *rule2alert* e sono stati inoltrati verso la macchina ospitante Iptables.

Tale configurazione è stata necessaria in quanto il modulo NetFilter agisce a livello kernel e quindi per poter rendere visibili i pacchetti ad esso, era indispensabile farli attraversare l'interno stack protocollare.

Inoltre, per lanciare i pacchetti è stato utilizzato il tool *tcpreplay* che utilizza delle socket particolari per l'istaurazione della connessione verso la macchina target (RAW_SOCKET). Iptables tuttavia non è in grado di gestire opportunamente i pacchetti che fluiscono all'interno di socket di questo tipo, motivo per cui è stato necessario rilassare i vincoli delle regole testate ed utilizzare opportuni accorgimenti.

In particolare, per tali motivi, le regole Iptables testate sono state inserite all'interno della tabella *raw* nella catena di PREROUTING, target di tutti i pacchetti in ingresso alla macchina provenienti da RAW_SOCKET. Inoltre sono stati rimosse da tutte le regole le opzioni inerenti i numeri di porta e lo stato delle connessioni, poiché non è stato possibile avere pieno controllo in questa fase su tali aspetti.

A partire dai 2181 pacchetti inviati, sono stati riscontrati nel log del kernel della macchina target 883 log univoci generati da Iptables. Un risultato del genere è parzialmente giustificabile per due motivi particolari. Per prima cosa bisogna tener conto dei diversi accorgimenti attuati per ottenere un test case significativo, ed inoltre va considerata la struttura dei pacchetti generata dal tool *rule2alert* che non si è dimostrata essere perfettamente aderente al contenuto delle regole Snort di input visti i risultati del caso precedente.

Per quanto riguarda il secondo scenario, anche qui sono stati utilizzati gli stessi pattern di traffico del caso precedente, che hanno prodotto i risultati riportati in Tabella 2.

PROTOCOL	ALERT	PACKETS
BITTORENT	46	53
HTTP	68	43
LDAP	13	13
MYSQL	29	57
SIP & RTP	6	32
SNMP	16	89
PPPOE	0	28
TELNET	41	92

Tabella 2. Risultati test traffico normale iptables

8. Conclusioni e sviluppi futuri

S2ipt è un tool che consente di configurare in maniera semplice e trasparente un firewall Iptables sulla base delle regole Snort della rispettiva community. In questo modo, anche un utente poco esperto può avere a disposizione un firewall avanzato, senza osservare sostanziali cambiamenti al proprio ambiente.

La potenza di *s2ipt* risiede nella combinazione degli aspetti positivi di due diversi tipi di strumenti: da un lato la potenza della sintassi Snort ed il supporto apportato dalla sua community, dall'altro l'efficienza derivante dall'utilizzo del kernel di Linux e la semplicità dei comandi Iptables, che si traducono in un impatto minimale sulle risorse di sistema.

Durante l'utilizzo di *s2ipt* per configurare NetFilter alla stregua di un IDS/IPS, il resto del sistema non viene impattato da questi cambiamenti:

- utilizzo di risorse di memorizzazione o computazionali aggiuntive pari a 0;
- le regole Iptables pre-esistenti ed inserite successivamente vengono preservate;
- è sempre possibile ripristinare lo stato precedente del sistema tramite la funzionalità di *revert.*

In eventuali sviluppi futuri, anziché utilizzare catene dedicate per il tool del tipo IDS o IPS, si potrebbe prevedere di riconoscere già in fase di traduzione la direzione del particolare pacchetto (in entrata o in uscita), ed effettuare l'inserimento della corrispettiva regola deve avvenire su una delle chain di default (INPUT, OUTPUT o FORWARD).

Un'altra particolare funzionalità da poter implementare, potrebbe essere legata alla traduzione di una singola regola Snort o di una breve lista, potendone specificare i *sid* da riga di comando:

```
s2ipt --iface IFACE --log --sid SID_1,SID_2,…,SID_N
```

In ultima analisi, un miglioramento al processo di traduzione potrebbe essere apportato raffinando ulteriormente il processo di traduzione, tramite anche il supporto ad altre opzioni Snort che non è possibile tradurre "as-is" in Iptables.