DESCRIÇÃO, GERAÇÃO E DIFUSÃO DE POLÍTICAS DE SEGURANÇA

Filipe Caldeira[1], Edmundo Monteiro[2]
caldeira@di.estv.ipv.pt, edmundo@dei.uc.pt

Laboratório de Comunicações e Telemática
Centro de Informática e Sistemas da Universidade de Coimbra
Pólo II, Pinhal de Marrocos, 3030 Coimbra

Sumário

Neste artigo é feita a apresentação de uma ferramenta de descrição de Políticas de Segurança, baseada na linguagem SPSL (*Security Policy Specification Language*) [1]. A ferramenta desenvolvida permite efectuar a descrição de políticas de segurança que serão posteriormente utilizadas na criação e difusão de regras reconhecidas por vários tipos de equipamentos na implementação das políticas de segurança das organizações.

1. Introdução

Uma linguagem de especificação de políticas de segurança pretende formalizar e normalizar as regras usadas em vários sistemas que implementam algum tipo de segurança numa comunicação. A heterogeneidade dos sistemas utilizados para garantir a segurança numa rede de computadores de uma organização leva a um aumento da dificuldade em definir uma política de segurança global, pois, para além da sintaxe usada por cada equipamento ser diferente, também o modo de "pensar" a segurança muda entre cada equipamento ou pelo menos entre cada marca de equipamento.

Neste contexto, uma linguagem de definição de políticas de segurança (SPSL) pode ser usada para definir uma única política de segurança para toda a organização, incluindo todas as suas entidades, os atributos e as acções que podem ou não ser efectuadas. Cada equipamento, mesmo usando uma linguagem proprietária para definir as suas regras de segurança, deverá conseguir interpretar a política de segurança global e identificar regras que terá que pôr em prática na sua zona de acção. Na figura seguinte é apresentada um modelo de organização usando um servidor de políticas de segurança para configuração de duas *firewalls*.

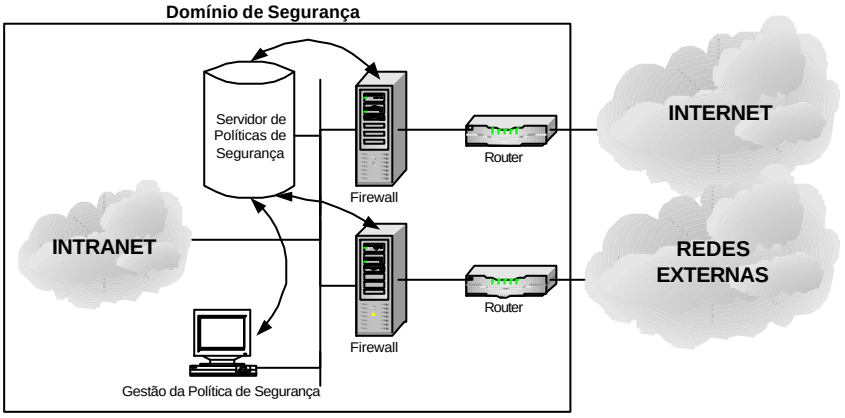

Figura 1 – Configuração de *Firewalls* usando a linguagem SPSL

2. A linguagem SPSL

Patrocinada pelo IETF, a linguagem SPSL[1] foi definida inicialmente para ser usada com ligações *IPSec* e com sistemas de gestão de chaves *IKE*. Contudo é possível usar esta mesma linguagem para outros fins, expandindo a sua sintaxe. A sintaxe desta linguagem foi derivada da linguagem de especificação de políticas de routing (RPSL) [3], pelo que está bastante direccionada para lidar com regras referentes à comunicação de dados, como por exemplo a especificação de regras para *Firewalls*.

---

[1] Departamento de Informática - Escola Superior de Tecnologia – Instituto Politécnico de Viseu
[2] Departamento de Engenharia Informática – Universidade de Coimbra

A linguagem SPSL baseia-se em objectos, sem contudo ser uma linguagem orientada a objectos. Está definido um conjunto de classes que podem servir para instanciar outros objectos. Cada objecto tem associado um conjunto de atributos que podem ser opcionais ou obrigatórios e que contêm os dados relevantes para a definição das políticas. Os objectos não contêm métodos e não é possível nesta especificação criar novos objectos.
Cada objecto é identificado univocamente pelo seu atributo chave, normalmente usado como nome do objecto.

Estão definidas 4 categorias principais de objectos:

- Dados primitivos – contendo dados atómicos usados na definição dos objectos, por exemplo o nome do objecto, um endereço IP, etc;
- Agentes de Manutenção – pretendem guardar informação referente à pessoa que pode modificar a informação contida nos objectos. Integra também um conjunto de informações sobre certificados que irão servir para autenticar cada objecto;
- Entidades da Rede - entidades relevantes para a especificação de uma política global de segurança: domínios, nós de rede, etc;
- Políticas – contém a especificação pormenorizada de cada política de segurança. Existem dois tipos de políticas já definidas: "*class policy*" que define filtragem de pacotes e "*class ipsec-policy*" que define acções *IPSec*.

De modo a permitir uma política de segurança global, a SPSL pode lidar com políticas baseadas em nós ou em domínios. Para servir de suporte a estes conceitos estão definidas as classes, *"node"*, *"gateway"*, *"polserv"* e *"domain"*. Um *"node"* é uma representação de uma entidade de rede que tenha pelo menos um interface e um nome. Um *"gateway"* é uma entidade que implementa as políticas de segurança. Um *"polserv"* define um servidor de políticas que é capaz de lidar com regras SPSL e futuramente com recurso a esta aplicação, também deverá ser capaz de as traduzir para a linguagem proprietária de cada equipamento.
Um *"domain"* engloba um conjunto de *"nodes"*, podendo ter mais do que um *"gateway"* que implemente as políticas para esse domínio. As políticas são representadas usando a classe *"policy*", estando cada uma associada a um ou mais "*node*" ou "*domain*".
Os objectos *"mnter"* guardam informação sobre a pessoa responsável pela política de segurança, como nome, telefone, email, certificados, etc.
Cada objecto deve ser assinado por um certificado digital de modo a garantir a sua genuinidade.

2.1. Estrutura da Linguagem

Na linguagem SPSL, os objectos são escritos num único ficheiro de texto, podendo existir ficheiros secundários incluídos usando a instrução *$INCLUDE*, sendo que, neste caso, os objectos existentes no ficheiro secundário são lidos como se fossem escritos na posição onde se encontra a instrução que os inclui.
A ordem segundo a qual os objectos são escritos é de extrema importância, pois no caso da filtragem de pacotes, a primeira regra que satisfaça as condições é aplicada, pelo que a política a aplicar por defeito a um dado domínio deverá ser sempre a última política para esse mesmo domínio.
Cada objecto é escrito em linhas consecutivas contendo cada uma o par atributo-valor. Se o valor do atributo continua para a linha seguinte é utilizado o caracter "\". O nome do atributo é separado do seu valor pelo caracter ":". Uma linha em branco serve de separador para os objectos.
O ficheiro deverá definir as entidades relevantes existentes na rede usando os objectos *"node"*, *"gateway"* ou *"polserv"*. Estes objectos têm entre outros os atributos *"name"* e *"ifaddr"*, respectivamente o nome DNS e endereço(s) IP da entidade modelada.
De modo a facilitar a administração da política de segurança, as entidades podem ser agrupadas em conjuntos, usando os atributos *"node-sets"* e *"gateway-sets"*.
O objecto *"domain"* define um conjunto de entidades na rede às quais será aplicado o mesmo conjunto de políticas de segurança. O seu atributo *"coverage"* lista as entidades que pertencem ao domínio, podendo ter como valores objectos *"node"*, *"node-sets"* ou ainda uma lista de endereços IP.
Cada política deverá ter o atributo *"association"* que define o domínio ou entidades a que esta se aplica.

A classe *"policy"* poderá ter um ou mais atributos do tipo *"policy"*, sendo possível especificar uma ligação usando o seu destino, origem, porto de destino e origem, protocolo de transporte e direcção. As acções base a executar que estão definidas são *permit*, *deny* e *forward*, tendo sido definido também como extensão à linguagem a acção *reject* e porta para re-direccionamento, pois estas acções podem ser usadas nas regras implementadas na *firewall* do *kernel* do *linux*, recorrendo à aplicação que permite efectuar a manutenção destas regras, *IPChains* [5].

Exemplo de uma politica definida usando uma classe *"policy":*

```
policy-name:  EXEMPLO
policy:       dst 193.197.128.43 \
              src * \
              xport-proto 6 permit
```

Esta política permite todas as ligações para o endereço 193.197.128.43 a partir de qualquer origem, desde que o protocolo seja o TCP.

A mesma política pode ser escrita usando o formato longo da mesma classe, no qual existem mais atributos que podem ser usados, por exemplo se os endereços forem IPv6.

A mesma política especificando individualmente todos os atributos:

```
policy-name:  EXEMPLO
dst:          193.197.128.43
src:          *
xport-proto:  6
tfr-action :  permit
```

Exemplo de um ficheiro contendo uma política de segurança usando SPSL:

```
mntner:         FILIPE
auth:           crypt-pw slijhygsjhsku
address:        IPV - Viseu
phone-number:   +351-232480500
email:          caldeira@di.estv.ipv.pt
mnt-by:         FILIPE
certs:          FILIPE-CERT
changed:        FILIPE 20000829
signature:      FILIPE FILIPE-CERT \
                dsa-sha1 klj498dg

cert:           FILIPE-CERT
certlocation:   x509_sig file filename \
                /home/FILIPE/.x509_cert
mnt-by:         FILIPE
changed:        FILIPE 20000829
signature:      FILIPE FILIPE-CERT \
                dsa-sha1 sli4nflk

node:           WEB
name:           www.ipv.pt
ifaddr:         193.137.7.1
mnt-by:         FILIPE
changed:        FILIPE 20000829
signature:      FILIPE FILIPE-CERT \
                dsa-sha1 gkudodx7

policy-name:    TCP-WEB
association:    WEB
policy:         dst 193.137.7.1 \
                src * \
                xport-proto 6 permit
mnt-by:         FILIPE
changed:        FILIPE 20000829
signature:      FILIPE FILIPE-CERT \
                dsa-sha1 k75fnmkj

policy-name:    DEFAULT
association:    WEB
policy:         dst * src * deny
mnt-by:         FILIPE
changed:        FILIPE 20000829
signature:      FILIPE FILIPE-CERT \
                dsa-sha1 o29fnmkj

#
```

O exemplo anterior contém uma política de segurança que está associada apenas à máquina www.ipv.pt (WEB), declarando que apenas permite ligações TCP de qualquer origem para a máquina WEB.

No que diz respeito à segurança das próprias políticas usando SPSL, esta deverá ser assegurada pelas aplicações que utilizem este ficheiro, existindo para tal os atributos *"mnt-by"* e *"signature"* que permitem verificar a validade da política. Sendo simples alterar o ficheiro de texto, é contudo difícil de escrever uma assinatura válida para o objecto sem a respectiva chave privada, pelo que a integridade dos objectos deverá está garantida. Por outro lado, terá que haver outras preocupações de segurança sobre este ficheiro, pois o acto de apagar uma política não será reconhecido por uma aplicação que apenas verifique as assinaturas dos objectos. Neste caso, deverá ser dada atenção à validação da integridade de todo o ficheiro.

3. Geração de regras a partir da SPSL

Na figura seguinte estão representados os principais módulos da aplicação construída:

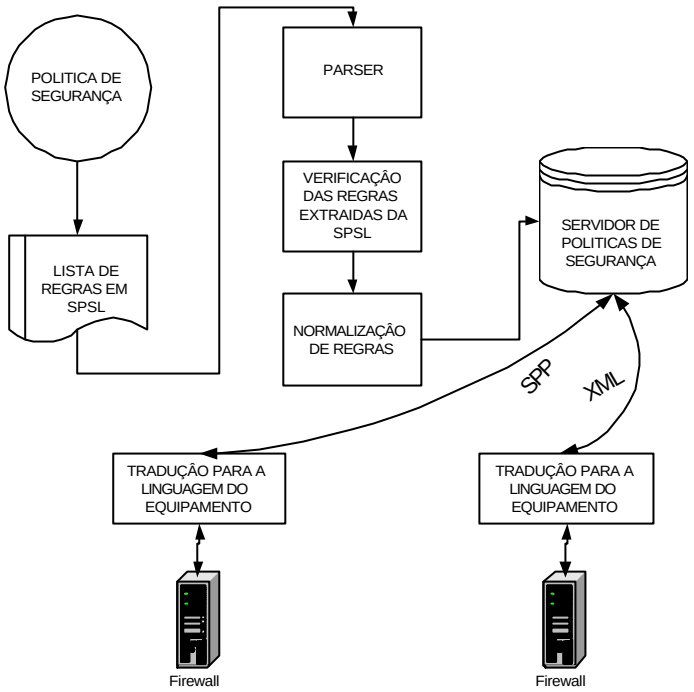

Figura 2 – Principais módulos da aplicação

O primeiro aspecto a ser tratado pela aplicação em desenvolvimento neste trabalho é a verificação da sintaxe do ficheiro que contém as políticas escrito usando linguagem SPSL. Para tal é usado um *parser* gerado usando *bison*[7] e *flex*[8], de modo a verificar a sintaxe definida pela BNF (*Backus-Naur Form Notation*) em que se baseia a linguagem. Ao ser verificada a sintaxe da linguagem é criada também um estrutura de dados que contém todos os objectos válidos. Além disso, são geradas mensagens de erro quando é encontrado algum objecto que não esteja escrito de forma correcta, excluindo apenas esse objecto da estrutura de dados resultante.

Os objectos são tratados individualmente, tendo em consideração o seu tipo, *"domain"*, *"policy"*, etc, de modo a poder ser criada uma nova estrutura que contenha apenas as políticas retiradas dos objectos "*policy*", associando de imediato cada uma destas políticas ao objecto base que lhe deu origem, de modo a conseguir identificar, por exemplo, a que domínio essa política está associada.

Partindo dos seguintes objectos,

```
domain:     IPV-SERVERS
coverage:   SERVER1,SERVER2
mnt-by:     FILIPE
changed:    FILIPE 20000823
signature:  FILIPE FILIPE-CERT \
            KEYEDMD5 ABBA007AB47E

policy-name:POL1
association:IPV-SERVERS
policy:     dst 193.137.7.1  src * \
            xport-proto 6 permit
mnt-by:     FILIPE
changed:    FILIPE 20000823
signature:  FILIPE FILIPE-CERT \
            KEYEDMD5 ABBA007AB47E
```

```
policy-name:POL2
association:IPV-SERVER
policy:     dst 193.137.7.2 \
            src 193.137.6.0/24  \
            xport-proto 6 permit
mnt-by:     FILIPE
changed:    FILIPE 20000823
signature:  FILIPE FILIPE-CERT \
            KEYEDMD5 ABBA007AB47E

policy-name:DEFAULT
association:IPV-SERVER
policy:     dst * src any deny
mnt-by:     FILIPE
changed:    FILIPE 20000823
signature:  FILIPE FILIPE-CERT \
            KEYEDMD5 ABBA007AB47E
#
```

ao executar o *parser* e a criação das estruturas de dados, o resultado será:

```
.......
Nº: 2
Associado a: IPV-SERVERS
Nome: POL1
Accao: Permit
Origem:  *
Destino: 193.137.7.1
Protocolo: 6

Nº: 3
Associado a: IPV-SERVER
Nome: POL2
Accao: Permit
Origem: 193.137.6.0/255.255.255.0
Destino: 193.137.7.2
Protocolo: 6

Nº: 4
Associado a: IPV-SERVER
Nome: DEFAULT
Accao: Deny
Origem:  *
Destino:  *
```

O que é feito neste ponto é a extracção de regras da política de segurança global, referentes a um ou vários domínios.

Depois de garantir que a sintaxe está correcta, o algoritmo utilizado para gerar as regras examina a sua estrutura, gerando mensagens de erro sempre que a lógica das regras seja incorrecta, por exemplo o uso de uma gama de endereços em que o valor mínimo seja menor que o valor máximo, (Ex: 193.134.56.7 - 193.134.55.1) ou mesmo para endereços de rede inválidos (Ex. 193.3.3.5/24).

3.1. Verificação das regras

Uma vez validadas as regras, poderão ainda existir alguns problemas quanto à sua lógica, por exemplo regras cujas gamas de endereços se intersectem.

Uma vez que a aplicação das regras segue a ordem pela qual elas foram escritas, a intersecção de endereços não deverá causar problemas na sua aplicação, contudo como o que se pretende é a definição de uma política global, a existência de regras que nunca irão ser aplicadas leva a um aumento da complexidade dessa mesma política.
Neste caso, pode ser aplicado um algoritmo que tem como objectivo a simplificação de regras:

Exemplo:

```
.......
policy-name:POL1
association:IPV-SERVERS
policy:     dst not 193.137.7.1 \
            src * xport-proto 6 permit
mnt-by:     FILIPE
changed:    FILIPE 20000823
signature:  FILIPE FILIPE-CERT \
            KEYEDMD5 ABBA007AB47E

policy-name:POL2
association:IPV-SERVER
policy:     dst 193.137.7.2  \
            src 187.100.100.0/24 \
            xport-proto 6 permit
mnt-by:     FILIPE
changed:    FILIPE 20000823
signature:  FILIPE FILIPE-CERT \
            KEYEDMD5 ABBA007AB47E
#
```

No exemplo apresentado, a primeira regra permite a ligação TCP, de qualquer endereço, para uma qualquer máquina do nosso domínio, à excepção da 193.137.7.1. A segunda regra permite a ligação TCP para a nossa máquina 193.137.7.2, caso a ligação seja efectuada partindo dos endereços 187.100.100.0/24. Podemos então verificar que a primeira regra generaliza a segunda ao permitir que qualquer máquina se ligue ao nosso endereço 193.137.7.2, pois este endereço é diferente de 193.137.7.1, fazendo com que a segunda regra nunca seja aplicada.
Usando o algoritmo para simplificação das regras, que transforma todos os endereços em gamas de endereços e verifica se existem intersecções, as duas políticas anteriores seriam transformadas apenas na seguinte regra:

```
Nº: 1
Associado a: IPV-SERVER
Nome: POL1-01
Accao: Permit
Origem: *
Destino: 0.0.0.0- 193.137.7.0,
         193.137.7.2-255.255.255.255
Protocolo: 6
```

Neste caso, o resultado final é equivalente à descrição anterior, usando apenas uma regra que permite ligações TCP de qualquer origem para o conjunto de endereços 0.0.0.0 – 193.137.7.0 e 193.137.7.2-255.255.255.255, isto é todas excepto a máquina com endereço 193.137.7.1.
O algoritmo usado também verifica intersecções com outros elementos das regras, como os portos de origem destino e o protocolo usado.

3.2. Conversão de regras em comandos reconhecidos por equipamentos que implementam regras

Uma vez escrita e processada a política de segurança global da empresa, esta deverá ser aplicada a um ou mais equipamentos, como sejam as *Firewall's* ou *Routers*. Pretende-se assim que cada equipamento possa ser instruído através de um conjunto de instruções que este reconheça, geradas automaticamente por um tradutor.

Um dos problemas com que nos deparamos é o da existência de potencialidades diferentes entre cada tipo de equipamento. Neste caso decidimos que seriam geradas mensagens de erro avisando que determinada função não pode ser implementada usando a linguagem pedida.
As regras escritas usando SPSL têm maior flexibilidade do que normalmente encontramos nos equipamentos. Por exemplo, um dos aspectos iniciais da conversão é a passagem de gamas de valores, não suportadas pela maioria dos equipamentos, para valores atómicos, a seguinte regra:

```
.......
policy-name:POL1
association:IPV-SERVERS
policy:     dst 193.137.7.1-193.137.7.2 \
            src 193.1.1.1, 194.1.1.1 xport-proto 6 direction inbound permit
mnt-by:     FILIPE
changed:    FILIPE 20000823
signature:  FILIPE FILIPE-CERT KEYEDMD5 ABBA007AB47E
```

terá que ser convertida em 4 regras diferentes através das combinações de todos os valores que não são atómicos, resultando então as seguintes regras:

```
Nº: 1                                   Nº: 3
Associado a: IPV-SERVERS                Associado a: IPV-SERVERS
Nome: POL1                              Nome: POL1
Accao: Permit                           Accao: Permit
Origem:  193.1.1.1                      Origem:  194.1.1.1
Destino: 193.137.7.1                    Destino: 193.137.7.1
Protocolo: 6                            Protocolo: 6
Direccao: inbound                       Direccao: inbound

Nº: 2                                   Nº: 4
Associado a: IPV-SERVERS                Associado a: IPV-SERVERS
Nome: POL1                              Nome: POL1
Accao: Permit                           Accao: Permit
Origem:  193.1.1.1                      Origem:  194.1.1.1
Destino: 193.137.7.2                    Destino: 193.137.7.2
Protocolo: 6                            Protocolo: 6
Direccao: inbound                       Direccao: inbound
```

Esta acção conjugada com o algoritmo de simplificação de regras, que expande todos os valores para gamas de endereços e as gamas de endereços escritas originalmente, pode levar a um grande aumento do número final de regras. Para minorar este problema, as gamas de valores são convertidas em um ou mais valores atómicos. Por exemplo se a regra simplificada tiver como endereço de destino a gama de valores 193.137.7.0 – 193.137.7.255, deverá ser transformada em 193.137.7.0/24, valor este que já pode ser aplicado na sintaxe da maioria dos equipamentos, não implicando efectuar a expansão da gama de endereços e consequentemente o número de regras finais será menor.

4. SPSL para *IPChains*

Actualmente o sistema desenvolvido suporta a conversão de SPSL para *IPChains* [5], podendo facilmente evoluir para outro tipo linguagem.

Esta conversão é executada usando os mecanismos descritos anteriormente, efectuando de seguida uma comparação dos atributos existentes nas regras com a BNF do *IPChains*, gerando gradualmente regras em cada uma das cadeias *IPChains*. Sempre que não existam dados suficientes na regra, são usados valores por defeito e comunicada essa atribuição ao utilizador.

Exemplo:

```
.......
policy-name:POL1
association:IPV-SERVERS
policy:     dst 193.137.7.1  src * \
            xport-proto 6 \
            direction inbound permit
mnt-by:     FILIPE
changed:    FILIPE 20000823
signature:  FILIPE FILIPE-CERT  \
            KEYEDMD5 ABBA007AB47E

policy-name: POL2
association: IPV-SERVER
policy:     dst 193.137.7.2 port 110 \
            src 193.137.6.0/24 \
            xport-proto 6,17 \
            direction inbound permit
mnt-by:     FILIPE
changed:    FILIPE 20000823
signature:  FILIPE FILIPE-CERT \
            KEYEDMD5 ABBA007AB47E
```

```
policy-name: POL3
association: IPV-SERVER
policy:     dst 193.137.7.3 port 80 \
            src * xport-proto 6 \
            direction inbound deny, \
            forward port 8080
mnt-by:     FILIPE
changed:    FILIPE 20000823
signature:  FILIPE FILIPE-CERT \
            KEYEDMD5 ABBA007AB47E

policy-name: DEFAULT
association: IPV-SERVER
policy:     dst any src any reject
mnt-by:     FILIPE
changed:    FILIPE 20000823
signature:  FILIPE FILIPE-CERT \
            KEYEDMD5 ABBA007AB47E

#
```

O extracto de 4 regras de uma política global escrita em SPSL dá origem ao seguinte conjunto de regras (existem 5 regras, pois os protocolos existentes na política POL2, foram expandidos para formar duas regras diferentes):

```
Nº: 1
Associado a: IPV-SERVERS
Nome: POL1
Accao: Permit
Origem:  *
Destino: 193.137.7.1
Protocolo: 6
Direccao: inbound

Nº: 2
Associado a: IPV-SERVER
Nome: POL2
Accao: Permit
Origem: 193.137.6.0/255.255.255.0
Destino: 193.137.7.2
Porto de destino: 110
Protocolo: 6
Direccao: inbound
```

```
Nº: 3
Associado a: IPV-SERVER
Nome: POL2
Accao: Permit
Origem: 193.137.6.0/255.255.255.0
Destino: 193.137.7.2
Porto de destino: 110
Protocolo: 17
Direccao: inbound

Nº: 4
Associado a: IPV-SERVER
Nome: POL3
Accao: Deny e Forward para porto 8080
Origem:  *
Destino: 193.137.7.3
Porto de destino: 80
Protocolo: 6
Direccao: inbound

Nº: 5
Associado a: IPV-SERVER
Nome: DEFAULT
Accao: Reject
Origem:  *
Destino:  *
```

Partindo deste conjunto de regras é feita a tradução para a linguagem que as vai implementar, neste caso o *IPChains*:

```
Regra Nº 1,POL1, assoc: IPV-SERVERS
ipchains -A input –p 6 -d 193.137.7.1 –j ACCEPT

Regra Nº 2, POL2, assoc: IPV-SERVER
ipchains -A input –p 6 –s 193.137.6.0/255.255.255.0 -d 193.137.7.2 110 –j ACCEPT

Regra Nº 3, POL2, assoc: IPV-SERVER
ipchains -A input –p 17 -s 193.137.6.0/255.255.255.0 -d 193.137.7.2 110 –j ACCEPT

Regra Nº 4, POL3, assoc: IPV-SERVER
ipchains -A input –p 6 -d 193.137.7.3 80 –j REDIRECT 8080

Regra Nº 5, DEFAULT, assoc: IPV-SERVER
ipchains -A input –j REJECT
```

A definição da BNF da linguagem deverá ser ainda melhorada, permitindo incluir *flags* que se pretendem utilizar com uma determinada linguagem. Por exemplo a *flag* –l do *IPChains* [5] poderá ser incluída se quisermos efectuar o *log* dos pacotes que verificam uma determinada regra.

5. Difusão das políticas de segurança

A política de segurança global da empresa deverá estar armazenada num servidor, em que esteja implementado algum tipo de protecção ao seu acesso.
O acesso aos servidores de políticas poderá ser efectuado usando SPP [2] (*Security Policy Protocol*), que define normas para o acesso e transporte de políticas de segurança, com aplicação directa aos objectos SPSL. Este protocolo define o modo como a informação de políticas é transportada, processada e protegida pelos servidores e clientes. O protocolo também define que informação deve ser trocada, bem como qual o formato em que esta deve ser codificada [4]. São definidos seis tipos diferentes de mensagens para a troca de informação, cada uma contendo um *header* seguido de um ou mais *payloads* com a informação pretendida.

Outra abordagem possível para a troca de informação entre o servidor de políticas e os clientes que as implementam poderá ser o uso de um formato de dados como o XML (*eXtended Markup Language)*[6], em que a estrutura do objecto que está a ser enviado está integrado na própria mensagem. Neste caso também deverá ser assegurada a segurança da comunicação com o uso de algum tipo de encriptação. Uma vantagem que pode existir no uso de XML é o facto de este ser já um *standard* criado pelo W3C e ser bastante extensível no que respeita ao formato de dados a ser trocado.

A aplicação desenvolvida permite criar um ficheiro XML com um DTD (*Document Type Definition*) associado, que define o conjunto de regras que são posteriormente recebidas e traduzidas pelo módulo de tradução especifico de cada equipamento. No exemplo seguinte é apresentado um ficheiro XML gerado pela aplicação, onde está descrito o DTD utilizado na versão actual, assim como é apresentada uma regra em formato XML obedecendo ao DTD definido.

```
<?xml version="1.0" encoding="windows-1252"?>
<!DOCTYPE  ListaRegras [
              <!ELEMENT ListaRegras (regra)*>
              <!ELEMENT regra (nome, dominio?, accao, forward?, ipforward?, portforward?,
              protoforward?, src+, dst+, src_port*, dst_port*, proto*, dir?, extras?)>
              <!ATTLIST regra  num  ID #REQUIRED>
              <!ELEMENT nome (#PCDATA)>
              <!ELEMENT dominio (#PCDATA)>
              <!ELEMENT accao (#PCDATA)>
              <!ELEMENT forward (#PCDATA)>
              <!ELEMENT ipforward (#PCDATA)>
              <!ELEMENT portforward (#PCDATA)>
              <!ELEMENT protoforward (#PCDATA)>
              <!ELEMENT src (not?, wild?, mask_min_max, min?, max?)>
              <!ELEMENT dst (not?, wild?, mask_min_max, min?, max?)>
              <!ELEMENT src_port (not?, wild?, mask_min_max, min?, max?)>
              <!ELEMENT dst_port (not?, wild?, mask_min_max, min?, max?)>
              <!ELEMENT proto (not?, wild?, mask_min_max, min?, max?)>
              <!ELEMENT dir (#PCDATA)>
              <!ELEMENT extras (#PCDATA)>
              <!ELEMENT not (#PCDATA)>
              <!ELEMENT wild (#PCDATA)>
              <!ELEMENT mask_min_max (#PCDATA)>
              <!ELEMENT min (#PCDATA)>
              <!ELEMENT max (#PCDATA)> ]>
<ListaRegras>
         <regra num="_1">
                  <nome> POL1 </nome>
                  <dominio> IPV-SERVERS </dominio>
                  <accao> PERMIT </accao>
                  <src>
                           <not> 0 </not>
                           <wild> 0 </wild>
                           <mask_min_max> 0 </mask_min_max>
                           <min> 193.137.6.0 </min>
                           <max> 255.255.255.0 </max>
                  </src>
                  <dst>
                           <not> 0 </not>
                           <wild> 0 </wild>
                           <mask_min_max> -1 </mask_min_max>
                           <min> 193.137.7.2 </min>
                  </dst>
                  <proto>
                           <not> 0 </not>
                           <wild> 0 </wild>
                           <mask_min_max> -1 </mask_min_max>
                           <min> 6 </min>
                  </proto>
                  <dir> inbound </dir>
         </regra>
</ListaRegras>
```

6. Conclusão e direcções futuras

A linguagem utilizada, sendo bastante flexível, deverá ser expandida antes de se poder tornar um *standard*, pois como o seu objectivo inicial foi a utilização de *IPSec* e *IKE*, existem algumas lacunas no que diz respeito à descrição de regras para configuração de *Firewalls*.

Com uma pequena extensão da linguagem a tradução das políticas para *IPChains* fica completa, o que permite facilitar a configuração de *Firewall's* deste tipo numa organização de tamanho considerável.

Para além dos módulos descritos, o sistema proposto deverá prever o desenvolvimento de mecanismos adicionais de tradução para outros equipamentos, como por exemplo as listas de acesso da Cisco, bem como a possibilidade de utilizar nomes DNS ou nomes de nós já definidos como endereços de origem ou destino.

A construção de uma interface gráfica, que permita visualizar apenas o conjunto de regras pretendidas, como por exemplo um só domínio, ou a construção visual e teste de regras poderá também ser importante no sucesso deste projecto.

7. Referências

[1] M. Condell, C. Lynn, J. Zao
"Security Policy Specification Language", Internet Draft: draft-ietf-ipsp-spsl-00.txt, Março 2000

[2] M. Condell, L.A. Sanchez
"Security Policy Protocol", Internet Draft: draft-ietf-ipsp-spp-00.txt, Julho 2000

[3] C. Alaetinouglu, et al.
"Routing Policy Specification Language (RPSL)"., RFC 2280. Janeiro 1998

[4] M. Blaze, A. Keromytis, Sandelman, M. Richardson, Sanchez
"IPSP Requirements", Internet Draft: draft-ietf-ipsp-requirements-00.txt, Julho 2000

[5] Linux IPChains HowTo
http://mirror.ipv.pt/LDP/HOWTO/IPCHAINS-HOWTO.html

[6] Bray, T. et al.
eXtensible Markup Language (XML) 1.0, W3C, Fev. 1998. - http://www.w3c.org/TR/REC-xml

[7] Charles Donnelly, Richard Stallman
Bison , The YACC-compatible Parser Generator, http://www.gnu.org/manual/bison/html_mono/bison.html

[8] Vern Paxson
Flex, A fast scanner generator, http://www.gnu.org/manual/flex-2.5.4/html_mono/flex.html