Detecção Cooperativa de Intrusões em Redes Carrier Ethernet

Pan Jieke[1], João Redol[1], Miguel Correia[2]

[1] Nokia Siemens Networks Portugal S.A.
Rua Irmãos Siemens 1, 2720 - 093 Amadora, Portugal
{pan.jieke, joao.redol}@siemens.com

[2] Faculdade de Ciências da Universidade de Lisboa
Campo Grande, Edifício C6, 1749-016 Lisboa, Portugal
mpc@di.fc.ul.pt

Resumo

Hoje em dia os elementos de rede (NEs) da camada 2 do modelo OSI, bridges ou switches, são componentes complexos, com centenas de milhares de linhas de software, que podem ser vulneráveis a ataques, permitindo até a execução remota de código no seu CPU interno. Este trabalho apresenta um esquema para proteger infra-estruturas de rede Carrier Ethernet de ataques lançados por NEs maliciosos contra o Spanning Tree Protocol. O artigo propõe um esquema de detecção de intrusões baseada em especificação, estendido com anotações de padrões temporais, de modo a detectar desvios do protocolo por parte dos NEs. A informação sobre ataques é cooperativamente trocada entre os NEs para decidirem se um NE é malicioso e se deve ser desligado da infraestrutura.

1 Introdução

Contexto e motivação Em resposta aos requisitos crescentes de uma economia globalizada, a indústria das telecomunicações tem contribuído para uma maior produtividade interligando comunidades de todo o mundo virtualmente em todos os segmentos de mercado. Os standards internacionais que garantem a eficiência das redes actuais, abrem também caminho para as redes da próxima geração. Por outro lado, a utilização acrescida de protocolos abertos, a multiplicidade de novos componentes, a diversidade de aplicações e plataformas, e os sistemas nunca suficientemente testados, tudo isto tem aumentado a oportunidade para actividades maliciosas em redes de telecomunicações como a Internet. Nos últimos anos, para referir um exemplo, os ataques de negação de serviço têm causado períodos longos de interrupção de serviços, envolvendo inúmeros provedores e utilizadores, com um grande impacto em termos de custos [28]. A questão que se põe é como disponibilizar uma comunicação aberta sem ter que deixar a rede vulnerável a esses ataques. A resposta é complexa, mas um requisito importante é claramente a protecção da infraestrutura de rede em si.

O objectivo do trabalho é apresentar um esquema de protecção para redes *Carrier Ethernet* usando *um sistema distribuído e cooperativo de detecção e resposta a intrusões*. A motivação para este trabalho é o facto deste tipo de rede ser actualmente adoptada em larga escala pelos fornecedores de serviço Internet e por muitas outras empresas [1]. Mais especificamente o trabalho foca essencialmente na camada de ligação de dados do modelo OSI (nível 2), os respectivos protocolos e elementos de rede (NEs), isto é, switches ou bridges.

Actualmente, os NEs são dispositivos de hardware complexos, que executam software que realiza as operações de um sistema operativo, o próprio serviço do NE, e protocolos

[1] A adopção da Carrier Ethernet é promovida pelo Metro Ethernet Forum. O site do forum encontra-se em http://www.metroethernetforum.org

de administração como o SNMP. Portanto, os NEs podem ter vulnerabilidades, que podem ser exploráveis tal como as vulnerabilidades que são exploradas todos os dias na Internet, por exemplo, em aplicações Web ou servidores do email [28]. Para citar um caso real, os NEs da família Cisco Catalyst correm software Cisco IOS, e recentemente foram reportadas várias vulnerabilidades de *buffer overflow* na *heap* existentes em algumas versões do IOS, que permitiriam a execução remota de código dentro desses NEs [5].

Este trabalho incide essencialmente sobre o problema de NEs do nível 2 serem atacados e controlados por piratas informáticos (*hackers*), lançando depois ataques contra a infraestrutura de rede. Mais concretamente, o trabalho considera a existência de um conjunto de NEs dos quais alguns podem ser *maliciosos*, ou seja, podem desviar-se do comportamento correcto de modo a quebrarem propriedades de segurança da rede, como a sua disponibilidade. Para lidar com este problema, é proposto que os NEs sejam equipados com um componente de detecção de intrusões. Cada um dos detectores inspecciona o comportamento dos outros NEs através da análise das mensagens recebidas, e pela cooperação com os detectores nos outros NEs para diagnosticar ataques e levar à desconexão (lógica) dos NEs maliciosos.

Contribuição A contribuição deste trabalho consiste na concepção do primeiro esquema de detecção de intrusões distribuída para redes Carrier Ethernet. Como caso de estudo, o trabalho centra-se no protocolo de gestão de ligações da Ethernet, *Spanning Tree Protocol (STP)*, e as suas variantes, *Rapid Spanning Tree Protocol (RSTP)* e *Multiple Spanning Tree Protocol (MSTP)* [4, 13, 9, 8, 14].

A detecção segue uma aproximação designada por *detecção de intrusões baseada em especificação*, a qual consiste em usar a especificação do protocolo para detectar os desvios do mesmo. No entanto, esta forma de detecção é normalmente usada para detectar desvios de uma sequência de mensagens ou de um conjunto de transições de estados. Relativamente aos ataques contra o STP, é preciso lidar com certos padrões de comportamento em termos temporais. Devido a esta limitação, o esquema da detecção de intrusões foi estendido com *anotações de padrões temporais*, de modo a detectar todos os ataques conhecidos contra o STP e suas variantes.

O esquema de detecção de intrusões funciona da seguinte forma. Os detectores nos NEs inspeccionam as mensagens do protocolo STP recebidas dos outros NEs em tempo-real e sem interferirem com as operações da rede. Independentemente do instante em que a mensagem é recebida num NE, o detector verifica o comportamento do NE envolvido com o seu comportamento esperado. O comportamento correcto dos NEs é descrito tendo em conta a especificação standard do protocolo STP [13]. Se existir um desvio entre um comportamento esperado e o actual, o NE é suspeito de ser malicioso.

Os resultados da detecção local nos NEs são enviados para os outros, para que todos possam correlacionar a informação da detecção, diagnosticar quais são os NEs maliciosos e removê-los da rede, desligando todas as portas a eles ligadas. Uma rede de gestão para Carrier Ethernet frequentemente usada hoje em dia pelos fornecedores de serviços – a Data Communication Network (DCN) – é utilizada para suporte à verificação da integridade e autenticação na comunicação de informação de detecção.

Para além de ataques contra o STP, outros ataques de NEs maliciosos poderiam ser considerados, tais como o descarte de mensagens ou o envio de mensagens para portas/ligações erradas. No entanto, estes tipos de ataques já foram considerados em estudos anteriores para protocolos da camada 3, e existe um conjunto significativo de trabalhos que descrevem como lidar com eles [24, 22].

Organização do artigo O artigo encontra-se organizado da seguinte maneira. A secção 2 descreve alguns conceitos de redes Carrier Ethernet. A secção 3 introduz o protocolo STP, os ataques contra o protocolo e os mecanismos existentes neste momento contra este tipo de ataques. A secção 4 propõe o esquema de detecção de intrusões cooperativa. A secção 5 apresenta resultados experimentais. Finalmente, a secção 6 conclui o trabalho.

2 Carrier Ethernet

A área de telecomunicações abrange hoje em dia uma vasta gama de tecnologias. As diferentes organizações têm diferentes requisitos e as tecnologias estão em constante evolução, exigindo uma constante adaptação. Por isso, é importante ter um modelo de referência que defina regras que permitam lidar com essa complexidade. O modelo standard de redes é designado por *modelo OSI*, de *Open Systems Interconnection* [16]. Este modelo fornece flexibilidade para a construção da infraestrutura da rede e separação de funcionalidades. Basicamente, as funcionalidades da rede são separadas em camadas, em que cada camada é tão independente das outras camadas quanto possível.

Existem 7 camadas no modelo OSI. A camada de baixo fornece serviços para as camadas mais altas sem ter de revelar como é que os serviços são implementados. Esta separação dá uma grande flexibilidade ao modelo, permitindo mudanças nas camadas de baixo (geralmente) sem afectar os utilizadores e aplicações finais. Neste tipo de divisão, no entanto, se a segurança de uma camada de baixo for comprometida, poderá ficar comprometida a segurança de todas as camadas acima dela.

A segunda camada do modelo OSI – a camada de ligação de dados – situa-se entre a camada de rede (camada 3) e a camada física (camada 1). Esta camada é responsável pela transferência de dados entre os nós adjacentes de uma rede de larga escala (WAN), ou entre os nós do mesmo segmentos de uma rede local (LAN), detectando e possivelmente corrigindo os erros que ocorrem na camada física.

A Ethernet é actualmente a tecnologia dominante na camada 2 para LANs, devido a razões como o seu baixo custo, a simplicidade de administração e a constante evolução em termos de largura de banda fornecida [21]. Devido ao seu custo competitivo por Mbps e à elevada largura de banda fornecida, muitas empresas de telecomunicações estão a instalar redes baseadas em tecnologia Ethernet como alternativa a redes ATM e/ou SDH, sobretudo na área de acesso. Originalmente a Ethernet não era uma tecnologia da classe *carrier*, ou seja, para transporte de dados em redes metropolitanas (MANs) ou de larga escala. Pelo contrário, era uma tecnologia para LANs. No entanto, está em curso o esforço de normalização, em organizações como o IEEE, para adaptar a Ethernet para transporte.

2.1 O problema

Os administradores de sistema e programadores normalmente assumem que a camada de ligação de dados é fiável ou, pelo menos, não é habitual que se preocupem com a sua segurança. Existem várias razões para este facto: as bridges/switches da camada 2 não fornecem interfaces pessoa-máquina, são fisicamente controladas por uma organização, os seus protocolos e software são razoavelmente simples quando comparados por exemplo com os protocolos da Web ou servidores e, mais importante, um típico utilizador de Internet não tem acesso directo às redes de camada 2 a que não esteja directamente ligado. Como a Ethernet está a tornar-se numa solução para redes da classe *carrier*, há cada vez mais fornecedores de serviços que disponibilizam acesso à rede ao nível 2, por isso é cada vez mais viável que um pirata informático faça ataques contra este nível. Como um dos protocolos mais comuns de gestão de ligações da Ethernet, o Spanning Tree Protocol é um alvo óbvio para esses ataques, podendo ficar comprometida a segurança de todas as camadas acima.

3 A Família de Protocolos Spanning Tree

As bridges/switches apareceram em meados dos anos 80 como dispositivos de comutação para o nível 2, suportando redes em estrela em vez da arquitectura clássica em *bus* da Ethernet. Nessa altura, havia a preocupação de que ligações redundantes entre duas ou mais bridges/switches gerassem ciclos, causando a replicação das mensagens que circulam na rede. O *Spanning Tree Protocol (STP)* foi desenhado com o objectivo de gerir essas ligações,

removendo os ciclos existentes sem intervenção humana (figura 1) [13]. Actualmente, o STP é utilizado com um objectivo adicional: a tolerância a faltas. Ligações redundantes são introduzidas de propósito na rede de modo a que se uma ligação falhar (ex., por um cabo ou uma interface se avariarem), o STP reconfigura a rede usando uma ligação redundante, garantindo a continuidade do serviço de transporte de dados.

Uma topologia em forma de árvore é sempre livre de ciclos pois existe um único caminho de uma folha da árvore para todas as outras folhas. O objectivo do STP é precisamente organizar a rede numa topologia em árvore sem deixar nenhum segmento desligado do resto da árvore.

Uma árvore tem uma raiz a partir da qual saem os seus ramos. O NE na raiz de uma árvore é chamado *root bridge*. A *root bridge* é o centro lógico de uma rede e existe apenas uma *root bridge* em cada instância da árvore. Dependendo da configuração, qualquer NE pode ser a *root bridge*. A *root bridge* pode mudar ao longo do tempo se a topologia sofrer alterações, como por exemplo, caso um NE seja removido ou adicionado à rede.

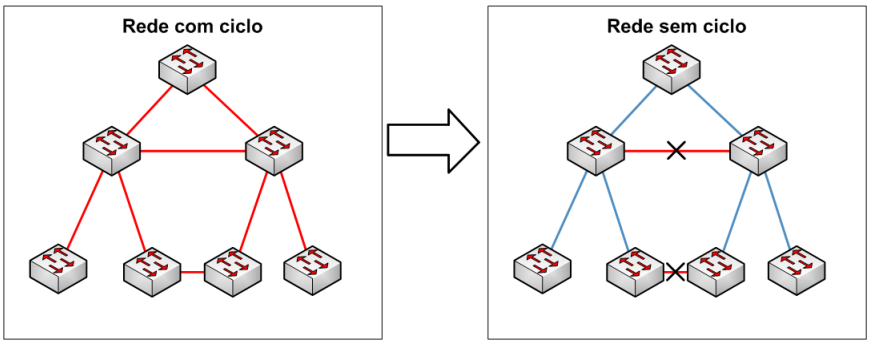

Figura 1: Spanning Tree Protocol

A prevenção de ciclos na rede permite garantir a existência de um e apenas um caminho de um NE para o outro. A forma mais simples de obter isso é garantindo que apenas uma *designated bridge* é responsável pela propagação de tráfego na direcção da *root bridge* para uma dada ligação. Se existir apenas um caminho activo da *root bridge* para uma ligação, então por definição não há ciclos na rede. Cada ligação tem exactamente uma *designated bridge*, que é um dos vários NEs que estão directamente ligados a essa ligação. A *root bridge* é sempre a *designated bridge* para todas as ligações que saem directamente dela. Os NEs não *root* podem ser *designated bridge*s para zero, uma ou mais do que uma ligação. Por exemplo, no lado direito da figura 1, a *root bridge* é a *designated bridge* para as duas ligações à qual está ligada; cada um dos dois NEs abaixo é a *designated bridge* para as duas ligações abaixo deles; os quatro NEs no fim não são *designated bridge* para qualquer ligação.

Para ser possível configurar, calcular e manter a árvore, cada NE na rede tem uma identificação única entre todos os NEs, designada por *bridge ID*. O bridge ID é um valor único de 64-bits, composto por um valor de prioridade (os 16 bits mais significativos) e o endereço MAC do próprio NE (os 48 bit menos significativos).

As mensagens trocada no protocolo STP são designadas por *Bridge Protocol Data Units (BPDUs)*, e contêm o bridge ID do emissor. As BPDUs são usados para a aprendizagem da existência de outros NEs e também para a obtenção de informações necessárias para cálculo e manutenção da árvore. Existem quatro tipos de BPDUs:

- Configuration BPDU
- Topology Change Notification BPDU
- Topology Change Notification Acknowledge BPDU

- Topology Change BPDU

Os NEs usam as interfaces físicas da rede – as *portas* – para ligação com os outros NEs. Cada porta no NE possui um identificador, designado por *port ID*. O port ID é uma concatenação do número da porta (8-bits) com um campo de prioridade (configurável). Os números das portas são únicos localmente por NE. Cada porta liga-se a uma ligação, que por sua vez é ligada a o outro NE na mesma LAN, numa rede de larga escala ou a um computador.

O STP tenta configurar a rede de tal modo que a partir de uma folha da árvore se possa chegar à raiz através do caminho com menor custo. Por isso, a topologia da árvore para um dado conjunto de ligações e NEs é determinada pelos bridge IDs, custos das ligações e port IDs. O STP executa três operações:

- Determinar a *root bridge*, que é o NE com menor bridge ID
- Determinar a *designated bridge* para cada ligação
- Manter a topologia estável ao longo do tempo

O STP funciona essencialmente da seguinte forma. Inicialmente cada NE assume que ele próprio é a *root bridge* e transmite Configuration BPDUs para a rede. O NE com menor bridge ID é eleito a *root bridge*. Depois os outros NEs na rede calculam as distâncias mais curtas para a *root bridge* usando a informação sobre a largura de banda de uma ligação, produzindo uma topologia livre de ciclos.

Cada NE continua a enviar periodicamente Configuration BPDUs, de modo a manter a estabilidade da topologia. Se um NE parar de enviar Configuration BPDUs, significa que houve uma falha no NE, e o protocolo entra na fase de modificação da topologia. O NE que detecta a falta de BPDUs envia um Topology Change Notification BPDU para notificar os outros sobre o evento. Depois a *root bridge* irá enviar Topology Change BPDUs para cada NE e, se for necessário, a topologia será reconfigurada. Estes procedimentos são automáticos e não envolvem qualquer intervenção humana.

3.1 Ataques ao STP

Algumas características do STP deixam-no vulnerável a diversos tipos de ataques por parte de piratas informáticos que tenham acesso directo aos equipamentos da rede. Estes ataques utilizam as ligações legítimas com os NEs de modo a injectar ataques na rede. Convém recordar que neste trabalho um NE é dito *malicioso* se é controlado por um pirata, podendo por essa razão executar ataques arbitrários contra o STP.

Uma análise cuidada do protocolo e da literatura sobre o assunto [20, 2] permitiu concluir que os ataques possíveis contra o STP são essencialmente os que se encontram abaixo. A maior parte dos ataques causam reconfigurações da topologia rede, o que feito repetidamente pode levar à sua indisponibilidade. Os ataques são:

- *Modificação de ID*
    1. *Modificação de prioridade* – Tipicamente, a eleição das entidades que participam no STP (a *root bridge* ou as *designated bridge*s) é baseada nos bridge IDs, que são compostos pela prioridade e pelo endereço MAC do NE. Um NE malicioso pode forçar a sua eleição como *root* ou *designated bridge* através da modificação (redução) do valor da prioridade de modo a conseguir ter a menor bridge ID de todos os NEs.
    2. *Falsificação de MAC* – Embora o ataque de falsificação de MAC (ou *MAC spoofing*) não permita a um NE forçar a sua eleição como *root*, uma vez que o valor de prioridade tem maior peso no bridge ID, este tipo de ataque pode causar reconfigurações da topologia.

- *Silêncio* – Como o STP é um protocolo autogerido, quando uma topologia está estável, cada NE continua a enviar Configuration BPDUs para os outros a indicar que está activo. Um NE malicioso pode omitir o envios desses BPDUs e causar uma reconfiguração da rede.
- *Falha falsa* – No STP, quando um NE detecta a falha de um seu vizinho envia um Topology Change BPDU a indicar esse evento. Um NE malicioso pode gerar Topology Change BPDUs falsos de modo a causar a reconfiguração da rede.
- *Inundação de BPDUs*
  1. *Inundação de Topology Change BPDUs* – Um conjunto significativo de Topology Change BPDUs de diferentes endereços são injectados na rede, podendo causar reconfiguração da rede.
  2. *Inundação de Topology Change Notification BPDUs* – Um conjunto significativo de Topology Change Notification BPDUs de diferentes endereços são injectados na rede, podendo causar a reconfiguração da rede.
  3. *Inundação de Configuration BPDUs reclamando papel de root* – Um NE malicioso injecta Configuration BPDUs com bridge IDs inexistentes, causando a sua eleição como nova *root*.
- *BPDU inválida* – Envio de uma BPDU mal-formada ou que não pode ser enviada no estado actual do NE. Este tipo de ataque poderá ser uma tentativa de encontrar vulnerabilidades no NE alvo, que poderá causar a paragem ou até a execução remota de código no NE.

3.2 Prevenção de ataques ao STP

Vários mecanismos podem ser usados para mitigar estes ataques contra o STP. Marro propôs uma extensão do STP com um campo adicional de autenticação em cada BPDU [20]. Esse trabalho assume que os ataques são gerados por clientes, ou seja, por computadores ligados à rede que simulam ser NEs legítimos; o caso de existirem NEs maliciosos não foi considerado. A solução apresentada consiste na adição de um *message authentication code* (MAC) à BPDU, utilizando chaves simétricas partilhadas entre os NEs. Quando um NE legítimo envia uma BPDU, esta deve levar um MAC legítimo, sendo aceite pelos outros NEs. Se a BPDU é enviada por um computador que não tem chaves partilhadas com os verdadeiros NEs, este será considerado como um MAC inválido, sendo a BPDU recusada pelos NEs. No presente trabalho é assumido que os NEs podem ser maliciosos, logo a utilização de MACs não resolve o problema pois os NEs podem gerar esses MACs.

A Cisco propôs dois mecanismos designados por *BPDU Guard* [6] e *ROOT Guard* [7] que bloqueiam a recepção de BPDUs nas portas que não estão ligadas a NEs, impedindo assim ataques vindos de clientes. O BPDU Guard simplesmente desactiva a porta para a recepção de BPDUs, enquanto o ROOT Guard permite ao equipamento ligado à porta participar no STP desde que não tente tornar-se *root*.

O standard 802.1x especifica mecanismos de autenticação de utilizadores para LANs IEEE 802, por exemplo, para ligações *dial-up* [12]. O utilizador precisa de efectuar o pedido para o gateway que controla os acessos à rede e propaga os pedidos para o servidor de autenticação, por exemplo, um servidor RADIUS. Este pode prevenir os ataques vindos de clientes não autenticados que simulam ser NEs, de forma semelhante ao mecanismo proposto por Marro.

Todas as soluções acima consideram apenas que os clientes ligados aos NEs podem ser maliciosos e assumem que os NEs são sempre correctos, ao contrário da solução descrita neste artigo. Por outro lado, exibem mais algumas limitações. A aproximação de Marro tem como limitações (a) a dificuldade em ser concretizada em sistemas reais, uma vez que requer algumas modificações ao protocolo de STP em si e (b) a necessidade de distribuir chaves secretas por todos os NEs. As soluções da Cisco necessitam de uma configuração

inicial, complicando a administração da rede. O standard 802.1x tem potencial um custo elevado em termos de recursos e sobrecarga administrativa pela utilização da autenticação baseada em certificados.

3.3 Detecção de intrusões e STP

Os mecanismos de segurança como o controlo de acesso do standard 802.1x [12] e firewalls desempenham um papel fundamental na segurança de redes, mas não podem prevenir todos os ataques possíveis contra uma rede. Por exemplo, utilizadores legítimos são capazes de efectuar ataques mesmo que esses mecanismos de protecção sejam correctamente utilizados. Os *sistemas de detecção de intrusões (IDSs)* surgem como uma segunda linha de defesa [11, 25, 18]. A ideia é recolher e analisar dados sobre o que se passa nos computadores e/ou redes de modo a descobrir se estão (ou foram) realizados ataques. Existem duas estratégias de detecção clássicas: *detecção baseada em assinaturas* e *detecção baseada em anomalia*.

Detecção baseada em assinaturas: Este tipo de IDSs é equipado com uma base de dados que contém as assinaturas dos ataques conhecidos [15, 19]. Os dados obtidos acerca do sistema são comparados com o conteúdo da base de dados. Se houver uma correspondência é detectado um ataque e é gerado um alarme; caso contrário, nada é feito. Este tipo de detecção normalmente gera poucos falsos alarmes, mas não detecta ataques que não estejam descritos na base de dados.

Detecção baseada em anomalia: Este tipo de IDSs baseia-se na hipótese de que as actividades fora do normal são maliciosas [18]. O IDS tem de começar por aprender o que é o "comportamento normal" do sistema. Quando uma actividade se desvia desse comportamento normal, é considerada uma intrusão. Esta forma de detecção de intrusões tem tendência a gerar muitos falsos alarmes mas não detecta apenas ataques conhecidos.

Detecção baseada em especificação: esta solução híbrida tenta combinar as vantagens das duas técnicas anteriores [3, 26, 29]. Em vez de usar técnicas da aprendizagem, como a detecção baseada em anomalia, a aproximação baseada em especificação utiliza uma especificação manualmente desenvolvida que descreve o comportamento correcto de um sistema/protocolo. Um desvio da especificação é considerado como uma intrusão. Esta solução tem sido aplicada em alguns protocolos e aplicações [27, 23], mas nunca tinha sido usada para os protocolos da Ethernet ou STP.

4 Detecção Cooperativa de Intrusões

4.1 Arquitectura básica

Este trabalho propõe uma solução distribuída de detecção de intrusões para redes Carrier Ethernet (figura 2). Tipicamente, os NEs da classe carrier são usados para fornecer conectividade entre os utilizadores finais e os fornecedores de serviços (ex., ISPs). Esta proposta envolve estender cada NE com um componente de software de detecção de intrusões de rede. A extensão pode ser concretizada de duas formas:

- inserção do componente de software no próprio NE, caso os seus recursos internos (CPU, memória) o permitirem; ou
- ligação de hardware com o detector ao NE usando uma porta de rede e configurando o NE para propagar todas as BPDUs recebidos pelo NE para o componente (uma funcionalidade normalmente disponível para administração nos bridges/switches actuais).

Assume-se que um subconjunto de NEs de uma rede podem ser comprometidos por piratas informáticos e lançar ataques contra a infraestrutura da rede. Para tornar a hipótese mais fraca, se um NE é malicioso então o seu componente de detecção de intrusões também

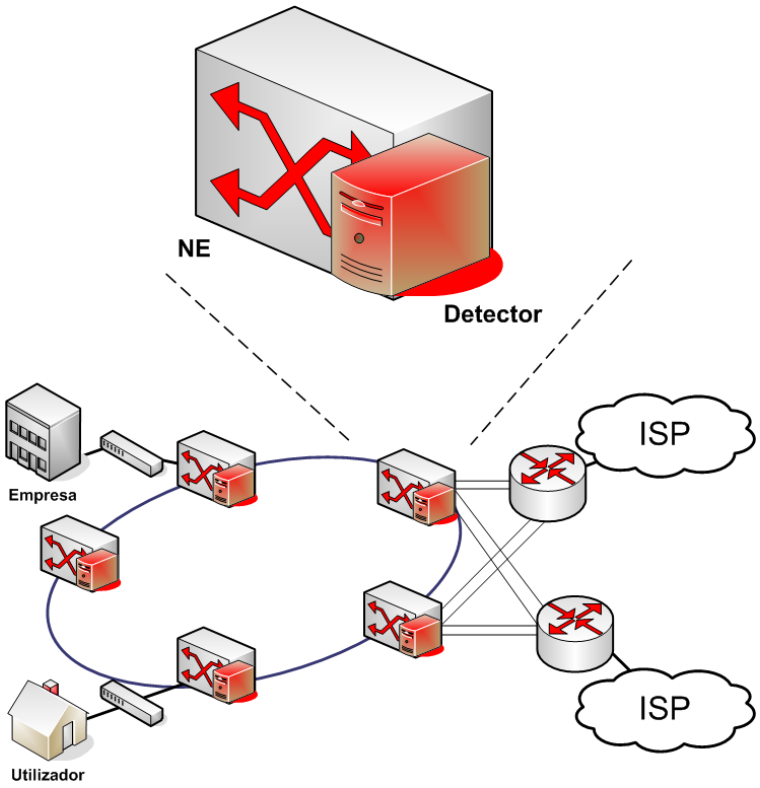

Figura 2: Arquitectura básica

pode ser comprometido e pode fornecer informações erradas de detecção. Assim sendo, um subconjunto de NEs podem não seguir a especificação do STP, e este comportamento incorrecto tem que ser detectado pelos NEs/detectores correctos.

4.2 Detecção de intrusões baseada em especificação estendida

Este artigo utiliza a detecção baseada em especificação, que detecta ataques como desvios do comportamento normal. Ao especificar os comportamentos correctos, os outros comportamentos podem ser classificados como anomalias.

O detector executa a detecção de intrusões em tempo-real. A especificação do protocolo STP é modelada usando uma *máquina de estados*. Os estados da máquina são os estados do protocolo, e as transições dos estados são causadas pela recepção de BPDUs ou expiração de *timeouts*.

Em vez de desenhar a descrição detalhada do protocolo, uma especificação abstracta do STP é utilizada. Desenvolver uma especificação mais precisa requer mais trabalho e foi considerado que pode trazer um impacto negativo na detecção, ao contrário do impacto positivo que se poderia esperar [26]. A razão é que as concretizações de um protocolo tendem a ter diferenças entre si e a desviarem-se da especificação do standard ou RFC. Uma especificação mais abstracta tende a omitir os detalhes que contém estas inconsistências.

A figura 3 apresenta a máquina de estados do STP, resultante da especificação em [13]. Cada nó do diagrama representa um estado em que um NE pode estar num determinado instante. Cada seta representa a transição entre dois estados. Cada seta tem uma etiqueta da seguinte forma: o topo é o evento que causa a transição (recepção de mensagem, *timeout*); o fundo é o evento gerado (mensagem enviada). Existem 5 estados no STP:

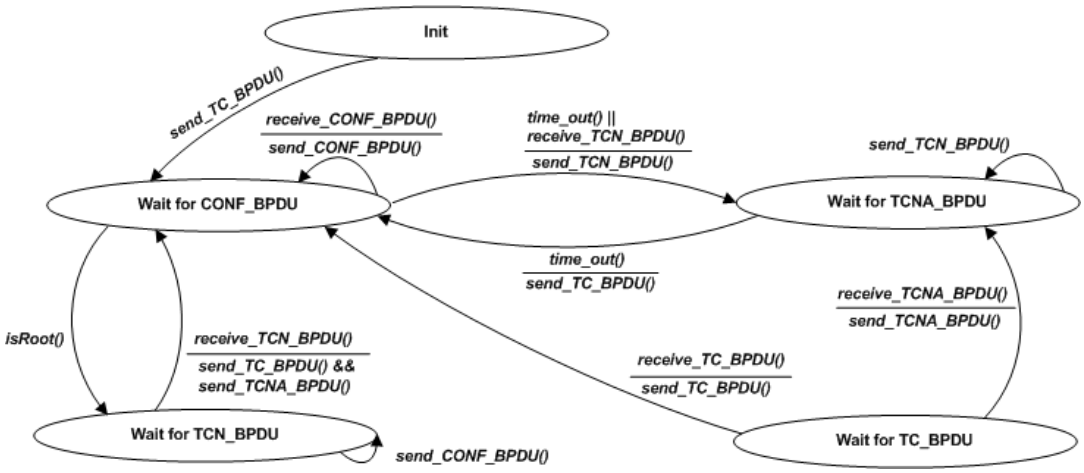

Figura 3: Máquina de estados do STP

- *Init:* o estado inicial quando um NE está ligado à rede.
- *Wait_for_ CONF_ BPDU:* quando uma topologia é estável, um NE não *root* poderá manter-se neste estado e mantém a rede disponível.
- *Wait_for_ TCN_ BPDU:* quando uma topologia é estável, um NE *root* irá manter-se à espera dos Topology Change Notification BPDUs se houverem alterações na rede.
- *Wait_for_ TCNA_ BPDU:* depois de um NE enviar um Topology Change Notification BPDU, mantém-se neste estado à espera da BPDU de confirmação de recepção.
- *Wait_for_ TC_ BPDU:* depois de receber um Topology Change Notification Acknowledge BPDU, o NE irá esperar pelo Topology Change BPDU para reconfigurar a topologia da rede.

Os eventos representados no diagrama são os seguintes:

- *send_ CONF_ BPDU() or receive_ CONF_ BPDU():* para manter a topologia estável, os NEs enviam e recebem periodicamente Configuration BPDUs.
- *time_ out():* é gerado quando um NE não recebe Configuration BPDUs de outro NE durante um certo intervalo de tempo.
- *send_ TC_ BPDU() or receive_ TC_ BPDU():* depois de enviar um Topology Change Notification BPDU, se um NE detectar uma modificação que requer a mudança na topologia, irá enviar um Topology Change BPDU para os outros NEs.
- *send_ TCN_ BPDU():* quando um NE detectar modificações na rede, irá enviar um Topology Change Notification BPDU.
- *receive_ TCN_ BPDU():* o NE recebe um Topology Change Notification BPDU de outro NE.
- *send_ TCNA_ BPDU():* depois de receber um Topology Change Notification BPDU dos outros NEs, o NE irá devolver um Topology Change Notification Acknowledge BPDU.
- *receive_ TCNA_ BPDU():* depois de enviar um Topology Change BPDU, o NE irá receber um Topology Change Notification Acknowledge BPDU.

4.2.1 Anotações

A máquina de estados na figura 3 é uma especificação incompleta do protocolo STP para efeito de detecção de intrusões, uma vez que permite comportamentos que não são aceitáveis. Exemplos desses comportamentos são um NE não enviar Configuration BPDUs embora esteja activo, e um NE enviar Topology Change BPDUs falsos.

Distinguir o comportamento correcto destes desvios requer estender a máquina de estados com anotações[2]. Cada evento de *timeout* ou *send BPDU* tem que ser anotado com um padrão temporal aceitável para esse evento. O formato das anotações pode ser relativamente complexo, mas as experiências com o STP têm demonstrado bons resultados usando um formato simples. No trabalho, considera-se que para um dado evento e, o padrão temporal (ou a anotação) é definido em termos do número máximo de repetições de e por unidade de tempo, $Rmax_e$. Estes valores podem ser definidos para qualquer evento. Exemplos são:

- $Rmax_{timeout}$ – número máximo de repetições da transição entre o estado Wait_for_CONF_BPDU e o estado Wait_for_TCNA_BPDU;
- $Rmax_{tcn}$ – número máximo de repetições para os eventos send_TCN_BPDU no estado Wait_for_TCNA_BPDU.

Estes valores têm que ser cuidadosamente definidos de modo a evitar falsos positivos e falso negativos.

4.3 Detecção local

Considere-se mais uma vez a máquina de estados da figura 3. Cada NE armazena a representação da máquina de estados e o estado em que se encontra cada um dos seu vizinhos (ou seja, dos NEs que estão directamente ligados a ele). O estado de cada vizinho consiste numa variável com o número do estado em que se encontra e em outra com o tempo há que se encontra nesse estado. A máquina de estados para cada NE começa no estado *Init*. O algoritmo da detecção executado pelo *NE/detector* é seguinte:

1. Quando o NE recebe uma BPDU: passá-lo para o detector;
2. Quando o detector obtém a BPDU: verifica se é uma BPDU esperada no estado actual e se satisfaz o padrão definido para esse evento (ex., o seu $Rmax_e$);
3. Se a BPDU não for esperada, então suspeita do NE que enviou a BPDU.
4. Actualizar o estado.

4.4 Detecção cooperativa de intrusões

A detecção local de intrusões tem duas limitações. A primeira é a de que as componentes de detecção de intrusões dos NEs maliciosos não são de confiança, ou seja, podem mentir sobre o que detectam ou não detectam (ver secção 4.1). A segunda é a de que alguns ataques, como silêncio e falha falsa, não podem ser detectados localmente, ou mais especificamente, diagnosticados localmente. No caso do ataque de falha falsa, um NE malicioso reclama (incorrectamente) que um dos seus vizinhos falhou. Embora alguns NEs possam detectar que isso é falso, mais informação é necessária para que os outros NEs tomem conhecimento de que o NE não falhou. Por essas duas razões é necessária uma solução cooperativa na qual é feita correlação entre a informação da detecção local dos diversos NEs. A correlação é feita em cada NE.

[2]Esta limitação dos detectores baseados em especificação não detectarem ataques que envolvem certos padrões temporais foi identificada em [26] que, no entanto, estende o esquema básico com detecção baseada em anomalia, o que é um desvio da ideia básica da detecção baseada em especificação.

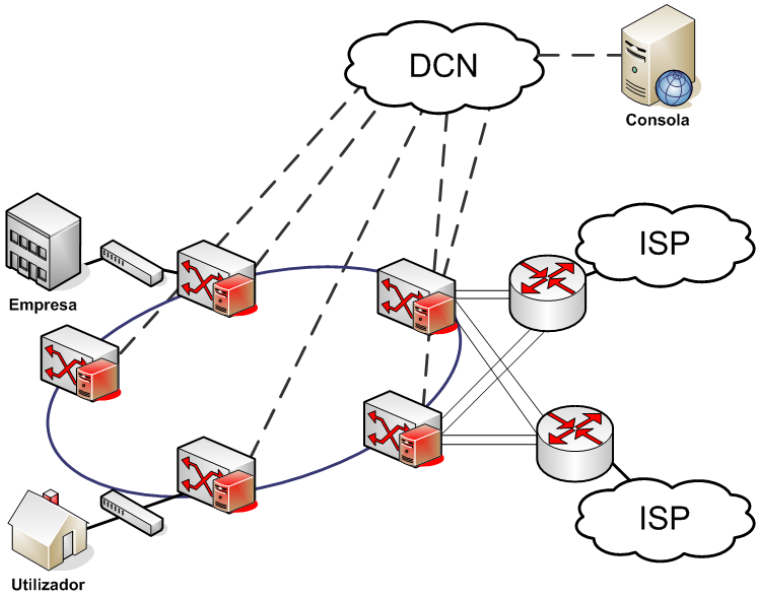

Figura 4: Arquitectura da Detecção Cooperativa de Intrusões

A correlação de detecção de intrusões tem sido estudada desde há vários anos [18, 10]. A ideia é enviar informação sobre os ataques e intrusões detectados por um detector (ou seja, os alarmes) para um motor de correlação, como por exemplo uma consola de gestão. Este esquema simples não pode ser aplicado directamente no ambiente considerado no trabalho, porque os ataques contra o STP podem na prática cortar o canal de comunicação, evitando que os alarmes cheguem ao motor de correlação, que neste caso está em todos os NEs. Por isso, para que a detecção cooperativa seja possível é necessário estender a arquitectura considerada.

A figura 4 apresenta a arquitectura completa que contém os componentes adicionais que ajudam à correlação. Nesta arquitectura, os NEs continuam a estar ligados normalmente, mas adicionalmente todos os NEs e a ferramenta de gestão de rede estão ligados por uma rede alternativa designada por *Data Communication Network (DCN)*. Note que a DCN e a consola não são adições extra à infraestrutura da rede e não envolvem custos adicionais. A DCN e a consola estão normalmente disponíveis na Carrier Ethernet. A consola é usada para configurar e gerir os NEs e a topologia de rede. Normalmente a DCN é uma rede fechada e apenas o administrador da rede tem acesso a ela, a comunicação envolvendo a consola de gestão e os NEs é unicast, isto é, um NE não pode obter acesso às trocas de dados dos outros NEs.

No esquema aqui apresentado, a consola desempenha apenas a função de *autenticador*. A correlação é feita em cada NE mas o NE malicioso tem capacidade de desligar a rede, por exemplo através do bloqueio da comunicação entre dois NEs vizinhos (relembrar que o STP desliga todas as ligações redundantes). O NE malicioso tem também a capacidade de modificar ou forjar mensagens com informação de detecção, uma vez que as mensagens do STP não são autenticadas. O objectivo da DCN e do autenticador é permitir que os NEs descubram se um NE malicioso descarta ou corrompe informação de detecção. A DCN é usada para difundir uma confirmação (ACK) quando uma mensagem é enviada. O algoritmo executado é o seguinte:

1. Quando um NE detecta um ataque, envia as informação de detecção para os outros NEs através da rede carrier;
2. Depois de enviar a informação de detecção, o NE envia um ACK para o autenticador

com o seu bridge ID e a síntese criptográfica da mensagem (*hash*) que este enviou para os outros NEs (usando um algoritmo de síntese criptográfica como o SHA-1);

3. Periodicamente cada NE pede uma lista dos ACKs do período ao autenticador. Depois verifica se algumas mensagens foram perdidas, modificadas ou forjadas (as modificações são verificadas comparando a síntese da mensagem com o *hash* que vem no ACK). Ao correlacionar estas informações, o NE diagnostica qual é o NE malicioso.
4. Se algum NE é detectado como sendo malicioso, os NEs que o detectaram desligam todas as ligações relacionadas com ele e reconfiguram a rede. No fim, o NE envia o relatório da detecção à consola de gestão.

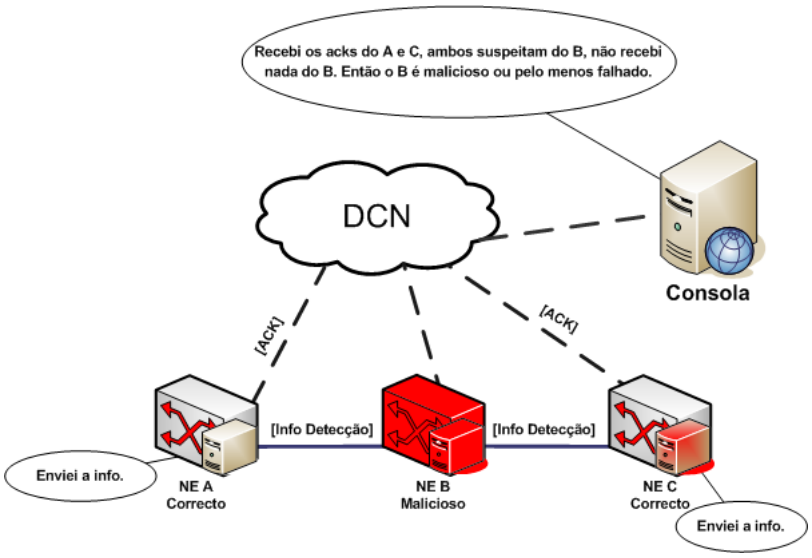

Figura 5: Exemplo de detecção com correlação

A figura 5 mostra um exemplo de detecção e correlação. O NE A e o NE C detectam um ataque do NE B. Depois enviam a informação de detecção para a rede carrier, incluindo o bridge ID do NE B e informação sobre como B se desviou da especificação. A seguir o NE A e o NE C enviam também o ACK de informação de detecção para o autenticador através da DCN. Como o NE B é um NE malicioso, este poderá descartar a informação da detecção enviada do NE A e do NE C, tentando prevenir a correlação. Assim, os NEs A e C não irão receber a informação enviada pelo outro, mas através dos ACKs do autenticador irão concluir que o NE B é malicioso e desligam-no da rede desligando todas as portas a ele ligadas.

5 Resultados Experimentais

As experiências foram feitas não com bridges/switches reais, mas com um emulador. As razões contra o uso de equipamento físico são várias: (1) utilizando um emulador é possível testar redes arbitrariamente complexas, com tantos NEs quanto for necessário; (2) apesar de acreditarmos que a inserção de um detector dentro de um NE é simples para o seu fabricante, para um investigador a tarefa não é trivial.

5.1 Ambiente de emulação

O emulador usado foi o *RSTP simulator* [1]. Embora seja designado por "simulador", na realidade trata-se de um emulador, uma vez que tem componentes (processos) que emulam

os NEs, e não modelos matemáticos que simulam os NEs e a rede. A razão pela qual foi usado um emulador de RSTP e não STP foi a indisponibilidade do último. No entanto, os dois protocolos são semelhantes.

O RSTP simulator é uma concretização completa da norma 802.1s feita em linguagem C, que fornece um conjunto de bibliotecas e APIs. Existem dois tipos de processos: *bridge* e *mngr*. O primeiro faz a emulação de um bridge RSTP e o segundo é o ambiente, isto é, é a componente que faz a gestão da topologia, da comunicação entre os NEs e que permite o envio das mensagens para a rede.

Os dois programas utilizam algumas bibliotecas. O processo *bridge* baseia-se na biblioteca *librstp.a* que contém a implementação do RSTP. A biblioteca *libuid.a* contém as funções para processar as BPDUs, como por exemplo, extrair/inserir a BPDU de/na rede. A biblioteca *libcli.a* fornece os comandos para gerir a rede, por exemplo, comandos de *connect* e *disconnect* que permitem aos utilizadores instanciar uma rede. Existem também ferramentas para registar as transições da máquina de estados e obter informações sobre as modificações da rede. Os dois programas mostram estampilhas temporais para permitir que o utilizador se aperceba de quando ocorreram os eventos e em que ordem.

O código do emulador foi modificado de modo a permitir a injecção de todos os ataques descritos na secção 3.1. Por exemplo, para executar um a ataque silencioso foi criado o comando *sleep*.

A emulação pode ser lançada num ou mais computadores. Cada instância do NE é lançada como um único processo do sistema operativo e os processos comunicam entre eles utilizando UDP/IP. Nas experiências efectuadas todos os processos correram na mesma máquina. A máquina de teste foi um portátil com processador Centrino a 2.1Ghz e Suse Linux 10.2.

5.2 Detecção de ataques

Esta secção apresenta um exemplo de detecção de ataques usando o emulador. O cenário emulado continha só 3 NEs, de modo a que os ecrãns apresentados sejam o mais legíveis possível. No entanto, os ecrãns ainda tiveram de ser limpos para melhorar a legibilidade.

```
17:29:16 Mngr > Bridge B4323 hello :)
Bridge B4324 hello :)
Bridge B4325 hello :)

17:30:13 Mngr > link B4323 1 B4324 2
connect B4323 port p01 to B4324 port p02
17:30:24 Mngr > link B4324 3 B4325 4
connect B4324 port p03 to B4325 port p04
17:30:51 Mngr > link B4323 5 B4325 6
connect B4323 port p05 to B4325 port p06
Sorry, p06 invalid
17:31:06 Mngr > link B4323 3 B4325 5
connect B4323 port p03 to B4325 port p05
```

```
Antes do ataque
17:31:32 B4323 > show bridge
Bridge:          B4323                State:enabled
BridgeId:        8000-00e310000001    Bridge Proirity: 32768
(0x8000)
Designated Root: 8000-00e310000001
Root Port:       none
Time Since Topology Change: 9
Max Age:         20   Bridge Max Age:       20
Hello Time:       2   Bridge Hello Time:    2
Forward Delay:   15   Bridge Forward Delay: 15
Hold Time:        3

Começar o ataque
17:31:36 B4323 > sleep
```

Figura 6: *mngr* (esquerda) e bridge maliciosa (direita)

A figura 6 (esquerda) apresenta o ecrãn do processo *mngr*. Quando este é lançado, descobre os 3 NEs que estavam a correr, que são internamente numerados B4323, B4324 e B4325 (não existe relação entre este número e o bridge ID de cada um). Depois são executados três comandos de *link* para interligar os 3 pares de portas dos NEs, formando uma rede totalmente ligada com um ciclo.

No lado direito da figura está o ecrãn do NE que foi seleccionado para ser o malicioso. O comando *show bridge* mostra algumas informações sobre o NE. Este é o NE com o menor bridge ID, logo foi eleito como a *root bridge* pelo RSTP. No entanto, algum tempo depois, no instante 17:31.36 o comando *sleep* é executado para lançar o ataque silencioso. A partir

desse instante o NE malicioso deixa de executar o RSTP ou propagar qualquer mensagem.

A figura 7 mostra os ecrãns dos dois NEs não maliciosos. O *timeout* para enviarem a Configuration BPDU é definido por omissão como 2 segundos, que é o valor recomendado pela norma. O NE malicioso tornou-se silencioso no instante 17:31:36, por isso aproximadamente 2 segundos depois (17:33:24) ambos os NEs correctos detectaram que o NE malicioso não enviou a BPDU, ambos decidiram que são a nova *root bridge*, trocaram algumas BPDUs e acabaram por eleger como nova *root* o NE B4323.

Quando o *timeout* expira, os detectores de intrusões em ambos os NEs suspeitam que o NE malicioso é isso mesmo, malicioso. No entanto, os ataques nesta fase são indistinguíveis das falhas reais do NE, como uma paragem por falha de energia. Logo, os dois NEs correctos não irão fazer nada sobre isso. No entanto, o NE malicioso voltou novamente a estar em execução e depois de tornar-se a *root bridge*, voltou a manter-se em silêncio, causando reconfigurações frequentes da rede. Relembremos que a transição *timeout* é anotada com o número máximo de repetições $Rmax_{timeout}$. Depois de $Rmax_{timeout}$ repetições deste ciclo, ambos os NEs correctos decidem que o NE malicioso é de facto malicioso, desligam-no da rede e enviam informação sobre esse facto para a consola de gestão.

```
Antes do ataque do NE B4323
17:31:39 B4324 > show bridge
Bridge:          B4324                  State:enabled
BridgeId:        8000-00e410000001      Bridge Proirity:
32768 (0x8000)
Designated Root: 8000-00e310000001
Root Port:       8002 (p02), Root Cost:      20
Time Since Topology Change: 10
Max Age:         20   Bridge Max Age:        20
Hello Time:       2   Bridge Hello Time:     2
Forward Delay:   15   Bridge Forward Delay: 15
Hold Time:        3

Durante o ataque do NE B4323
17:31:41 B4324 > 17:33:24:
brige B4324 became root
17:33:24:
brige B4324 new root port: p03
17:33:24:sttrans (B4324-p02): FORWARDING=>DISCARDING
17:33:25:
brige B4324 became root

show bridge
Bridge:          B4324                  State:enabled
BridgeId:        8000-00e410000001      Bridge Proirity:
32768 (0x8000)
Designated Root: 8000-00e410000001
Root Port:       none
Time Since Topology Change: 5
Max Age:         20   Bridge Max Age:        20
Hello Time:       2   Bridge Hello Time:     2
Forward Delay:   15   Bridge Forward Delay: 15
Hold Time:        3
```

```
Antes do ataque do NE B4323
17:31:44 B4325 > show bridge
Bridge:          B4325                  State:enabled
BridgeId:        8000-00e510000001      Bridge Proirity:
32768 (0x8000)
Designated Root: 8000-00e310000001
Root Port:       8005 (p05), Root Cost:      20000
Time Since Topology Change: 30
Max Age:         20   Bridge Max Age:        20
Hello Time:       2   Bridge Hello Time:     2
Forward Delay:   15   Bridge Forward Delay: 15
Hold Time:        3

Durante o ataque do NE B4323
17:31:56 B4325 > 17:33:24:sttrans (B4325-p04):
DISCARDING=>LEARNING
17:33:24:sttrans (B4325-p04): LEARNING=>FORWARDING
17:33:25:
brige B4325 became root
17:33:25:
brige B4325 new root port: p04
show bridge
Bridge:          B4325                  State:enabled
BridgeId:        8000-00e510000001      Bridge Proirity:
32768 (0x8000)
Designated Root: 8000-00e410000001
Root Port:       8004 (p04), Root Cost:      20000
Time Since Topology Change: 4
Max Age:         20   Bridge Max Age:        20
Hello Time:       2   Bridge Hello Time:     2
Forward Delay:   15   Bridge Forward Delay: 15
Hold Time:        3
```

Figura 7: Os NEs correctos

5.3 Severidade dos ataques

Esta avalia a severidade dos ataques usando o ambiente de emulação. Os dois cenários de injecção de ataques, agora com mais NEs, são apresentados na figura 8. No cenário 1, existe apenas um NE malicioso, no meio da topologia, e no cenário 2 foram dispostos 5 NE maliciosos.

Todos os ataques identificados foram testados nos dois cenários. As tabelas 1 e 2 mostram o resultado de avaliação de severidade dos ataques contra o RSTP. Os ataques são caracterizados em cinco categorias [17]: CT (catastrophic: >0.95), CR (critical: >0.75), MG (marginal: >0.5) e MN (minor: >0.25). O valor mais alto representa a maior severidade ou impacto. Os valores são atribuídos manualmente, analisando as consequências de cada ataque na rede.

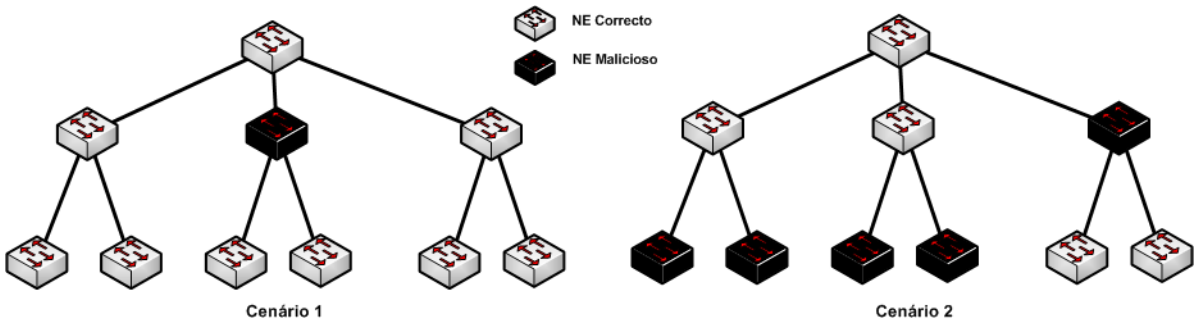

Figura 8: Cenários de ataques

Ataque	**Resultado**	**Severidade**	**Efeitos**
Modificação de ID	Sucesso	CT	rede completamente controlada
Silêncio	Sucesso	CT	rede completamente indisponível
Falha falsa	Sucesso	CR	reconfiguração parcial da rede
Inundação de BPDUs	Sucesso	CR	pequena inundação de tráfego
BPDU inválida	Falhado	MN	sem efeito

Tabela 1: Avaliação dos ataques no cenário 1 (1 NE malicioso)

As experiências permitiram concluir que os ataques de modificação de ID e silencioso são os dois tipos de ameaças mais perigosas ao protocolo. Para conseguir uma negação de serviço catastrófica na rede, um NE malicioso tem que estar o mais perto possível da raiz da topologia. No entanto, se o atacante estiver longe da raiz, este pode executar em primeiro lugar um ataque de modificação de ID para se tornar a *root bridge* ou, no mínimo, uma *designated bridge* que tenha acesso a várias ligações. Depois deste passo, os outros ataques tornam-se muito mais fáceis de concretizar. O ataque silencioso é mais crítico, uma vez que um único NE malicioso – cenário 1 – pode forçar a reconfiguração periódica da rede.

No caso dos ataques de falha falsa e inundação de BPDUs, os efeitos não foram tão fortes. Estes ataques são muito mais críticos quando são executados por um conjunto grande de NEs maliciosos (cenário 2). Esta é a razão principal pela qual eles são classificados com CT na Tabela 2 e apenas CR na tabela 1.

Em ambos os cenários os ataques de BPDU inválida falharam sempre, porque os NEs descartaram as BPDUs com campos inválidos. Isto aconteceu para o NE fornecido com o emulador e para um conjunto limitado de mensagens mal formadas de teste, mas não é necessariamente verdade para todos os NEs actualmente disponíveis e para todos os tipos de mensagens mal formadas.

Ataque	**Resultado**	**Severidade**	**Efeito**
Modificação de ID	Sucesso	CT	rede completamente controlada
Silêncio	Sucesso	CT	rede completamente indisponível
Falha falsa	Sucesso	CT	reconfiguração frequente da rede
Inundação de BPDUs	Sucesso	CT	rede completamente indisponível
BPDU inválida	Falhado	MN	sem efeito

Tabela 2: Avaliação dos ataques no cenário 2 (5 NEs maliciosos)

6 Discussão e Trabalho Futuro

O trabalho apresenta um novo esquema para proteger redes Carrier Ethernet de ataques executados por NEs. O esquema é baseado na detecção de intrusões baseada em especificação para detectar ataques contra o STP. Além disso, é usada correlação de alarmes para obter uma decisão sobre que NEs são maliciosos e têm que ser removidos da rede.

Uma emulação de redes Carrier Ethernet foi usada para os teste de detecção do ataque silencioso e para a avaliação de todos os ataques conhecidos contra o STP. Uma avaliação mais completa do esquema da detecção está em curso.

Existe um conjunto de ataques executados por NEs que são muito similares aos ataques feitos por *routers* maliciosos. Alguns exemplos são a entrega de mensagens a ligações erradas ou o simples descarte de mensagens. Futuramente pretende-se investigar a maneira de lidar com este tipo de ataques. Nesta fase ainda não foram considerados porque as soluções são similares com as disponíveis para *routers* maliciosos, uma área na qual existem vários trabalhos, nomeadamente o trabalho seminal de Perlman [24]. A outra linha do trabalho futuro será a melhor compreensão do diagnóstico e remoção dos NEs maliciosos no caso de existir mais do que um.

Referências

[1] *Rapid Spanning Tree 802.1w Simulator*. http://sourceforge.net/projects/rtsplib.

[2] O. K. Artemjev and V. V. Myasnyankin. Fun with the Spanning Tree Protocol. *Phrack*, 11(61), Aug. 2003.

[3] I. Balepin, S. Maltsev, J. Rowe, and K. N. Levitt. Using specification-based intrusion detection for automated response. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection*, pages 136–154, 2003.

[4] Cisco. *Using VlanDirector, Appendix C - Understanding Spanning-Tree Protocol*. Cisco Systems Inc., 1997.

[5] Cisco. *Cisco Security Advisory: IOS Heap-based Overflow Vulnerability in System Timers*. Cisco Systems Inc., 2005. Document ID 68064, Revision 1.2.

[6] Cisco. *Spanning Tree PortFast BPDU Guard Enhancement*. Cisco Systems Inc., 2005. Document ID 10586.

[7] Cisco. *Spanning Tree Protocol Root Guard Enhancement*. Cisco Systems Inc., 2005. Document ID 10588.

[8] Cisco. *Understanding Multiple Spanning-Tree Protocol (802.1s)*. Cisco Systems Inc., 2006. Document ID 24248.

[9] Cisco. *Understanding Rapid Spanning-Tree Protocol (802.1w)*. Cisco Systems Inc., 2006. Document ID 24062.

[10] H. Debar and A. Wespi. Aggregation and correlation of intrusion detection alerts. In *Proceedings of the 4th Workshop on Recent Advances in Intrusion Detection*, volume 2212 of *Lecture Notes in Computer Science*, pages 85–103. Springer-Verlag, 2001.

[11] D. E. Denning and P. G. Neumann. Requirements and model for IDES - a real-time intrusion detection expert system. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, 1985.

[12] IEEE. *802.1X - Port Based Network Access Control*, 1998.

[13] IEEE. *ANSI/IEEE 802.1D-2004 standard - Part 3: Media Access Control (MAC) Bridges*, 1998.

[14] IEEE. *ANSI/IEEE 802.1Q-2003 standard - Part 3: Media Access Control (MAC) Bridges*, 1998.

[15] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *Software Engineering*, 21(3):181–199, 1995.

[16] ISO. *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*, 1994.

[17] ISO. *Risk management vocabulary guidelines for use in standards*, 2002. ISO Copyright Office.

[18] C. Kruegel, F. Valeur, and G. Vigna. *Intrusion Detection and Correlation: Challenges and Solutions*, volume 14 of *Advances in Information Security*. Springer-Verlag, 2005.

[19] U. Lindqvist and P. A. Porras. Detecting computer and network misuse through the production-based expert system toolset (p-BEST). In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 146–161, 1999.

[20] G. M. Marro. Attacks at the data link layer. Master's thesis, University of California, 2003.

[21] R. M. Metcalfe and D. R. Boggs. Ethernet: distributed packet switching for local computer networks. In *Innovations in Internetworking*, pages 25–34. Artech House, 1988.

[22] A. T. Mizrak, Y.-C. Cheng, K. Marzullo, and S. Savage. Detecting and isolating malicious routers. *IEEE Transactions on Dependable and Secure Computing*, 3(3), 2006.

[23] J.-M. Orset, B. Alcalde, and A. R. Cavalli. An EFSM-based intrusion detection system for ad hoc networks. In *Proceedings of the 3rd International Symposium on Automated Technology for Verification and Analysis*, pages 400–413, 2005.

[24] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, Massachussets Institute of Technology, 1988.

[25] P. A. Porras, D. Schnackenberg, S. Staniford-Chen, M. Stillman, and F. Wu. The common intrusion detection framework architecture. 1998. http://www.isi.edu/gost/cidf/drafts/architecture.txt.

[26] R. Sekar, A. Gupta, J. Frullo, T. Shanbhag, A. Tiwari, H. Yang, and S. Zhou. Specification-based anomaly detection: a new approach for detecting network intrusions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 265–274, 2002.

[27] C.-Y. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, and K. Levitt. A specification-based intrusion detection system for aodv. In *SASN '03: Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 125–134, 2003.

[28] D. Turner(editor). Symantec Internet security threat report. Trends for January 06–June 06. Volume X, Sept. 2006.

[29] P. Uppuluri and R. Sekar. Experiences with specification-based intrusion detection. *Lecture Notes in Computer Science*, 2212:172–189, 2001.