FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Segurança da Informação

DIAN GIDARO

PROXY COMO CONTROLE DE ACESSO
SERVIDOR DINÂMICO

Americana, SP

2018

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Segurança da Informação

DIAN GIDARO

PROXY COMO CONTROLE DE ACESSO SERVIDOR DINÂMICO

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação, sob a orientação do Prof. Ms. Wladimir da Costa.

Área de concentração: Segurança em Redes

Americana, SP

2018

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

AG885p GIDARO, Dian

Proxy como controle de acesso: servidor dinâmico. / Dian Gidaro. – Americana, 2018.

55f.

Monografia (Curso de Tecnologia em Segurança da Informação) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Ms. Wladimir da Costa

1 Segurança em sistemas de informação 2.Internet – rede de computadores I. COSTA, Wladimir da II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.518.5

681.519

Dian Gidaro

Proxy como controle de acesso servidor dinâmico

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.
Área de concentração: Ciência da Computação

Americana, 25 de Junho de 2018.

Banca Examinadora:

Prof. Ms. Wladimir da Costa (Presidente)
Fatec Americana

Prof. Ms. Clerivaldo José Roccia (Membro)
Fatec Americana

Prof. Ms. Alberto Martins Junior (Membro)
Fatec Americana

DEDICATÓRIA

Dedico este trabalho ao meu pai, Durval, “In Memorian”, a minha mãe, Tânia, pelo apoio, a minha namorada, Isabela por todo amor e carinho sem os quais não teria conseguido chegar até aqui.

RESUMO

Atualmente, as atividades das instituições públicas ficam prejudicadas sem estarem integradas à Internet, porém, o valor da conexão é elevado e os órgãos não dispõem de recursos suficientes para manter um acesso adequado. A ETEC Polivalente de Americana vem enfrentando dificuldades de acesso à Internet com a falta de largura de banda. Para minimizar esta situação é sugerida a implantação de um servidor Proxy com regras de bloqueio dinâmicas implementadas em uma versão do sistema operacional Linux. Por meio do firewall e do Proxy pode-se garantir a conectividade e banda regular para os serviços de maior necessidade, aproveitando da melhor forma possível o link de conexão à Internet. Para a implantação deste servidor, primeiramente, é feita uma análise sobre a segurança da informação visando a implementação do Script de firewall, abordando em seguida a implementação do servidor de Internet com a instalação do servidor Proxy Squid e suas configurações. Em seguida, é abordada a possibilidade de configuração de regras dinâmicas de bloqueio no Proxy, visando melhor aproveitamento da banda disponível e garantir a satisfação dos servidores. Para melhor visualizar o impacto das alterações propostas e acompanhar o uso do link, aborda-se ainda, a ferramenta MRTG, software que gera gráficos referentes ao tráfego em uma rede. Por fim, é feita uma análise dos testes e resultados dos testes. Observou-se por meio do MRTG e de testes de navegação, upload e download, pontos positivos que confirmam a eficácia da implantação do servidor proposto na Unidade de Ensino.

Palavras Chave: Servidor, Proxy, Internet, Rede, Segurança.

ABSTRACT

Currently, some activities in the public sector can not work well without an Internet connection. However, high-capacity Internet links are expensive and some public offices have not enough resources to support this cost. This work has been developed at the ETEC Polivalente de Americana, where it was revealed shortcomings in the Internet access. In order to minimize this problem, this work proposes the use of a proxy server, implementing dynamic block rules upon Linux operating system. The main goal of the proxy server is to guarantee connectivity and sufficient bandwidth for high priority services in the best possible way. To implement this server, initially, a review of the computer security concepts was done envisaging a firewall script implementation. Afterwards, it is implemented the Squid Proxy server to support the dynamic block rules proposals. So, in order to illustrate the impact of this technique in the Internet link usage, it was installed the MRTG tool. So, it was concluded through MRTG and other tests (response time of upload and download) several positive results, showing the efficiency of this server in the school .

Keywords**:** *Server, Proxy, Internet, Network, Security.*

SUMÁRIO

1 INTRODUÇÃO 11
2 REFERENCIAL TEÓRICO 13
2.1 Ativo de Informação 13
3 MECANISMO DE CONTROLE – Segurança Lógica 17
3.1 Cópias de Segurança (*backups*) 17
3.2 Criptografia 17
3.3 *Firewall* 18
3.4 Registro de Eventos (*logs*) 18
3.5 Controle de Acesso 19
4 IMPLEMENTANDO O SERVIDOR 20
4.1 Histórico da ETEC Polivalente de Americana 20
4.2 Infraestrutura de Rede 21
4.3 Implantação do *Firewall* 21
4.3.1 Filtragem 22
4.3.2 *Iptables* 23
4.3.3 *Implementação do firewall* 24
4.4 *Proxy* 27
4.4.1 ACL (*Access Control List*) 29
4.4.2 Configuração implementada 29
4.5 DNS (*Domain Name System*) 34
4.6 *Proxy* com Regras Dinâmicas 35
4.6.1 Regras de Bloqueio 35
4.6.2 Linguagem *Perl* 36
4.6.3 Implementação do programa de dinamicidade 36
4.6.4 Conteúdo das regras de bloqueio 41
5 RESULTADOS E TESTES DE DESEMPENHO 42
5.1 Instalação e uso do MRTG 42
5.1.1 *Script* de leitura dos dados da interface de rede 45
5.2 Análise das regras de bloqueio 46
5.3 Análise dos resultados 48
5.4 Testes de Desempenho 49
5.4.1 *PCPitstop* 50
5.4.2 *WEBPAGETEST* 50
5.4.3 Análise dos Resultados 50

6 CONSIDERAÇÕES FINAIS **52**
REFERÊNCIAS **53**
APÊNDICE A – Autorização do Estudo **55**

ÍNDICE DE FIGURAS

Figura 1 – Controle de Acesso 19
Figura 2 – Modelo de Topologia com base em servidor 28
Figura 3 – HTML gerado pelo MRTG 43
Figura 4 – Gráfico para análise gerado pelo MRTG 48

INDÍCE DE QUADROS

Quadro 1 – Aspectos da segurança da informação 14
Quadro 2 – *Script* do *firewall* 25
Quadro 3 – Configuração do Squid 32
Quadro 4 – Configuração do BIND.....34
Quadro 5 – Código para alteração do comportamento do Squid 37
Quadro 6 – *Log* gerado pelo programa do Proxy dinâmico.....41
Quadro 7 – Configuração do MRTG.....45
Quadro 8 – *Script* LER 45
Quadro 9 – Períodos de monitoria do servidor.....47
Quadro 10 – Testes de desempenho 51

1 INTRODUÇÃO

Nos tempos atuais a Internet é uma realidade vastamente difundida e faz parte do cotidiano das pessoas. Com o desenvolvimento das tecnologias que a envolve, ela está se tornando mais versátil, possibilitando que diversas aplicações se beneficiem dela.

As empresas e órgãos públicos não ficam longe dessa evolução e cada vez mais, se tornam condicionadas ao uso da Internet para funcionar. Atualmente, praticamente nenhum órgão do setor público consegue desenvolver sua rotina sem estar conectados a esse mundo virtual. Desenvolver as atividades pertinentes a uma repartição pública sem uma conexão rápida e segura com a Internet é uma tarefa difícil. Mas, o custo dessa conexão é elevado; quanto maior a banda de dados necessária, maior vai ser o custo para mantê-la. Portanto, tendo em vista que a maioria das repartições públicas não dispõe de recursos suficientes para manter um *link* de acesso adequado, que atenda a demanda dos serviços, além da utilização para fins particulares dos empregados, técnicas para regular e aperfeiçoar este *link* são nitidamente necessárias.

Neste contexto, este Trabalho de Conclusão de Curso foi concebido com a perspectiva de proporcionar uma solução adequada para os problemas que a ETEC Polivalente de Americana vem enfrentando com o acesso à Internet.

Atualmente, a capacidade do link da Unidade de Ensino é suficiente para execução dos serviços básicos de administração escolar, porém, devido a má utilização dos usuários e o número excessivo de computadores que possuem acesso à rede, a execução das atividades básicas fica muitas vezes prejudicada, trazendo consigo vários transtornos como: retrabalhos, funcionários e professores trabalhando fora do horário de expediente, falta de conexão, entre outros.

O presente trabalho propõe como solução a implantação de um servidor que controle os acessos à Internet dinamicamente, o qual implementa basicamente dois serviços: *Proxy (Web cache)* dinâmico e um *firewall*; através deste servidor garante-se a conectividade e banda suficiente para os serviços de maior prioridade e, posteriormente, libera-se o acesso ao restante dos serviços, como acesso a e-mail e sites diversos. Isto será feito de forma dinâmica, através da análise da utilização da rede e das diversas variáveis que a compõe, de forma segura, garantindo a integridade da rede interna através de um *firewall* implementado no referido servidor.

O principal objetivo deste trabalho consiste em corrigir as dificuldades na infraestrutura da rede da ETEC Polivalente de Americana, através da implantação de um servidor *Proxy,* que se adapte à demanda de dados, com a finalidade de manter a disponibilidade de acesso a *sites* e sistemas para a execução dos serviços da unidade de ensino e, consequentemente, melhorar a produtividade dos servidores e a segurança da rede local.

A utilização do servidor de Internet dinâmico pode permitir o acesso irrestrito aos *sites* quando a banda de Internet estiver livre. À medida que esta estiver sendo utilizadas, as regras dinâmicas são aplicadas, fazendo assim, o bloqueio às páginas que não são importantes para o serviço essencial da unidade escolar. A solução apresentada parece ser eficaz, pois assim possibilita que o usuário continue navegando e utilizando a Internet para fins particulares, como exemplo, para ler os *e-mails* pessoais, mas sem prejudicar o trabalho que é desenvolvido na Unidade de Ensino, além de possibilitar auditorias de acessos aos sites visitados, melhorando assim o controle dos dados e informações que entram e saem da rede interna da unidade.

O restante deste trabalho está organizado da seguinte maneira: o segundo e terceiro capítulo são dedicados à revisão de conceitos de segurança em redes de computadores. O quarto capítulo aborda a descrição do funcionamento e configuração do servidor *Proxy* transparente e *firewall*, coleta de dados da rede e configuração do servidor e seus serviços. A implementação das regras dinâmicas de acordo com o tráfego do *link* de dados está descrita no quarto capítulo. O quinto capítulo aborda os testes e resultados obtidos com a implementação do servidor e, finalmente, apresentam-se as considerações finais e as referências bibliográficas.

2 REFERENCIAL TEÓRICO

Com o avanço das tecnologias, o uso da Internet tem se monstrado indispensável nas atividades cotidianas, sendo esse um dos recursos mais utilizados para troca de informações, tanto empresariais quanto pessoais. Devido ao uso dessa ferramenta, cresce também a apreensão com a segurança da informação.

A política de segurança da informação nada mais é que um conjunto de práticas e controles adequados, formada por diretrizes, normas e procedimentos, com objetivo de minimizar os riscos com perdas e violações de qualquer bem. Se aplicada de forma correta, ajudam a proteger as informações que são consideradas como um ativo importante dentro da organização (FERREIRA, 2008).

Tanenbaum (2003) afirma que a grande disseminação de aplicações via Internet é uma das causas da demanda por sistemas seguros. A noção de sistemas seguros é motivo de polêmicas. Garantir a segurança de um sistema é algo impossível de ser sustentado em bases quantitativas. A complexidade cada vez maior dos sistemas atuais e a competitividade do mercado sempre estão definindo novas direções, determinantes, para estas dificuldades.

2.1 Ativo de Informação

De acordo com a ISO/IEC 13335-1/2004 (CERT, [s.d.]) caracteriza-se como ativo qualquer coisa que tenha valor para a organização. Pode-se considerar como ativo de informação todo bem da organização que se relaciona com informação e que tenha valor, podendo ser um componente humano, tecnológico, físico ou lógico que realize processos de negócio dentro da instituição.

Ferreira (2008) afirma que cada informação tem um diferente grau de importância, por esse motivo é necessário classificá-las. Essa classificação norteia-se mediante ao impacto que causaria a sua perda, alteração ou uso sem permissão. Variadas categorias podem causar conflitos e dificultar o processo, elas devem ser de fácil compreensão e descritas para diferenciação entre si. Normalmente três níveis podem ser suficientes para uma boa classificação. Entre os níveis mais utilizados na classificação de informação estão: informação pública, informação interna e informação confidencial.

- Informações públicas: são aquelas de menor relevância, não precisam de sigilo, por isso não é necessários investimentos para torná-las seguras. Exemplo: testes de sistemas e serviços, *folders*.

- Informações internas: são as que devem ser conservadas fora do alcance de acesso externo, mas que, se for acessada de forma imprópria não causarão grandes problemas. Exemplo: agenda de telefones.

- Informações confidenciais: devem ser privadas de acesso externo, pois se manipuladas por pessoas não autorizadas poderão levar a instituição a ter prejuízos financeiros ou perda de competitividade. Exemplo: salários, dados de clientes, senhas.

Devido à importância de cada informação, deve-se mantê-la segura, porém muitas vezes sua importância só é percebida quando é destruída, perdida ou até roubada.

Ferreira (2008) explica que os princípios da segurança da informação abrangem basicamente os seguintes aspectos: confidencialidade, integridade e disponibilidade, e que toda ação que possa comprometer um desses princípios pode ser tratada como ataque a sua segurança.

O Quadro 1 apresenta um resumo dos aspectos da segurança da informação, baseado em autores diversos:

Quadro 1 - Aspectos da segurança da informação

Confidencialidade	É a garantia de que a informação é acessível somente por pessoas autorizadas a terem acesso. Peixoto (2006) diz que a tramitação das informações deve contar com a segurança de que elas cheguem sem que se dissipem para outros meios ou lugares onde não deveriam passar.
Integridade	É a preservação da exatidão da informação e dos métodos de processamento. Segundo Lyra (2008), a informação deve estar correta, ser verdadeira e não estar corrompida.
Disponibilidade	É a garantia de que os usuários autorizados obtenham acesso à informação e aos ativos correspondentes, sempre que necessário.

Fonte: Próprio autor

As informações estão sujeitas a ameaças e riscos devido sua vulnerabilidade. Ameaça é o que provoca um risco, dano ou perda. A ABNT ISO/IEC 27002:2005 (ABNT, 2005) define risco como combinação da probabilidade de um evento e de suas consequências. Portanto a função básica da segurança da informação é minimizar os riscos até que esses estejam em níveis aceitáveis.

Tanembaum (2003) afirma que um ataque a um sistema é uma ação tomada por um intruso malicioso (ou, em alguns casos, um erro de operação por parte um usuário inocente) que envolve a exploração de determinadas vulnerabilidades de modo a concretizar uma ou mais ameaças.

Para Adachi (2004), que analisou a gestão da segurança em Internet *Banking*, os aspectos envolvidos na segurança da informação agrupando-os em três camadas: física, lógica e humana. Sendo essencial que haja segurança em cada uma das três categorias.

Adachi (2004) apresenta que a segurança física tem como objetivo proteger máquinas e informações contra usuários não permitidos, prevenindo o acesso a esses recursos, pode ser trabalhada sob duas formas:

- Segurança de Acesso: trata das medidas de proteção contrata o acesso físico não autorizado;

- Segurança Ambiental: trata da prevenção de danos por causas naturais;

A segurança lógica aplica-se em casos onde um usuário ou processo da rede tenta obter acesso a um objeto que pode ser um arquivo ou outro recurso de rede (estação de trabalho, impressora, etc.) sendo assim um conjunto de diretrizes, adotados com a finalidade de proteger os dados, programas e sistemas contra tentativas de acessos desautorizados, feitas por usuários ou outros sistemas (ADACHI, 2004).

Ferreira (2008, p. 121) afirma que a interferência humana, seja ela acidental ou não, causa a maioria dos incidentes de segurança. Afirma ainda que de nada vale o investimento em recursos de tecnologia da informação, se o fator humano for deixado de lado, tendo em vista que a segurança está relacionada a pessoas e processos antes da tecnologia. Ferreira (2008, p. 121) ainda concluiu que os funcionários, quanto mais bem preparados, mais segura será a organização.

Ferreira (2008) indica que política de segurança define normas, procedimentos, ferramentas e responsabilidades às pessoas (usuários, administradores de redes e sistemas, funcionários, gerentes, etc.) que lidam com

essa informação para garantir o controle e a segurança da informação na empresa e que dita quais são as regras aplicadas dentro da empresa para uso de recursos tecnológicos e descarte de informações. Ainda segundo o mesmo autor, a política de segurança determina o que é ou não permitido em questões de segurança ao longo da utilização de qualquer programa ou material que contenha informações gerenciais, com base na aplicação de regras que definem o acesso às informações. Assim, a base da política de segurança é a definição do comportamento esperado das pessoas que interagem com um sistema.

Para seu efetivo funcionamento a política deve ter certas peculiaridades como: ser verdadeira, ser válida para todos, simples, contar com o comprometimento dos gestores da empresa, etc. De nada adiantaria implantar uma política que não fosse coerente com as ações executadas pela empresa, pois isso impossibilitaria seu cumprimento.

Ferreira (2008) afirma que em pouco tempo pode-se perceber a prevenção de acessos desautorizados, danos ou interferências no andamento do negócio, além de já se conseguir maior segurança nos processos do negócio, em médio prazo surge a uniformização dos procedimentos, a adequação já de forma segura de novos processos e a qualificação e quantificação de respostas a incidentes; e a longo prazo pode-se obter o retorno do que foi investido por meio da redução de situações relacionadas a incidentes de segurança da informação.

3 MECANISMO DE CONTROLE – Segurança Lógica

Segundo Wadlow (2000), a segurança lógica é um aspecto abrangente e complexo, requerendo, consequentemente, um estudo muito mais apurado e detalhado. Devemos estar atentos aos mínimos detalhes que compõem este tipo de segurança. Considerando que a informação é a principal ferramenta do processo decisório das empresas, não podemos arriscar e levar empresas ao fracasso pelo simples fato de não possuir as informações necessárias e em tempo hábil.

Conforme mencionado anteriormente, os mecanismos de controles que visam a segurança lógica podem ser definidos como barreiras que impedem ou limitam acesso a informação, que está em um ambiente controlado, evitando assim que fique haja alteração não autorizada por algum elemento mal-intencionado (WADLOW, 2000).

Desta forma, os mecanismos mais citados são:

3.1 Cópias de Segurança (*backups*)

Backup (cópia de segurança) é a cópia de dados de um dispositivo de armazenamento a outro para que possam ser restaurados em caso de perdas dos dados originais, em consequência de defeitos graves do computador que podem ser causados por falha de hardware, software e erro humano (TAVEIRA *et. al*., 2006).

Fazer *backup* não é apenas importante, é fundamental. O principal objetivo é manter em segurança e disponíveis, para qualquer eventualidade, todas as informações que julgar necessárias, para não correr o risco de perder dados importantes (TAVEIRA et. al., 2006).

Os backups são classificados em físicos e lógicos. Os físicos são os locais de armazenamento das informações. Normalmente essas unidades são chamadas de “fitas de backup” e apresentam uma enorme capacidade de armazenamento físico. Já o backup lógico é apenas o “salvamento” dos dados, contundo não serão armazenados em forma física e sem virtual (PINHEIRO, 2005).

3.2 Criptografia

A criptografia pode ser entendida como um conjunto de métodos e técnicas para cifrar ou codificar informações legíveis por meio de um algoritmo, convertendo

um texto original em um texto ilegível, sendo possível mediante o processo inverso recuperar as informações originais (MORENO, 2005). O autor complementa que na computação, as técnicas mais conhecidas envolvem o conceito de chaves, as chamadas chaves criptográficas. Trata-se de um conjunto de bits baseado em um determinado algoritmo capaz de codificar e de decodificar informações. Se o receptor da mensagem usar uma chave incompatível com a chave do emissor, não conseguirá extrair a informação.

3.3 *Firewall*

Pode-se dizer que *firewall* é um ponto entre duas ou mais redes, que pode ser um componente ou um conjunto de componentes, por onde passa todo o tráfego, permitindo que o controle, a autenticação e os registros de todo o tráfego sejam realizados (NAKAMURA, 2007).

Ainda segundo o autor, o *firewall* também pode ser visto como uma política de segurança, já que este pode controlar o acesso entre duas redes, o *firewall* é um conjunto de componentes e funcionalidades que definem a arquitetura de segurança, utilizando uma ou mais tecnologias de filtragem.

Para Torres (2001), um *firewall* é, na realidade, um poderoso roteador interligando duas redes e possui, pelo menos, duas placas de rede, onde se tem uma rede pública insegura em uma ponta e uma privada segura na outra.

O *firewall* serve, no entanto, como uma barreira de proteção para um dispositivo que está ligado a outras redes, dando acesso apenas aos dados permitidos (de acordo com o cabeçalho do pacote, não com a informação contida neste, já que não é acessada pelo *firewall*) definidos pelo usuário que ultrapassem essa barreira. O controle do firewall é feito por regras e filtros de dados.

3.4 Registro de Eventos (*logs*)

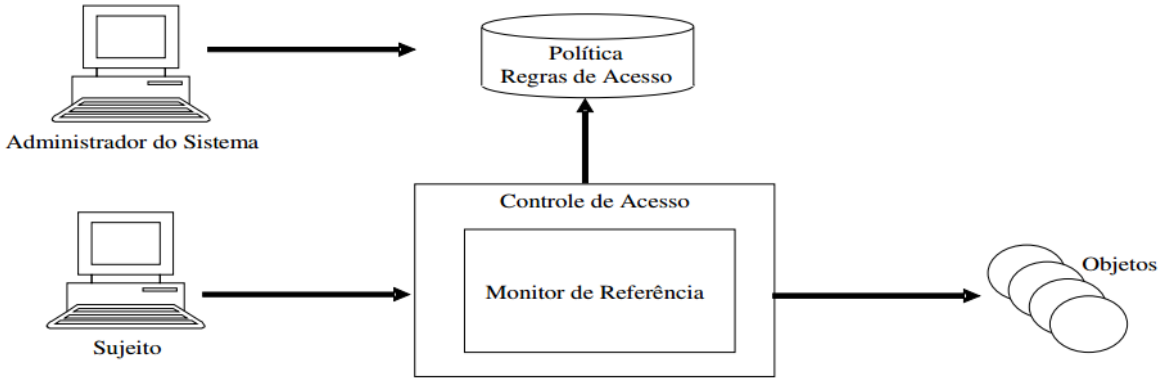
Segundo explicado pelo CERT (2003), *logs* são muito importantes para a administração segura de sistemas, pois registram informações sobre o seu funcionamento e sobre eventos por eles detectados. Muitas vezes, os *logs* são o único recurso que um administrador possui para descobrir as causas de um problema ou comportamento anômalo.

Ainda segundo o CERT (2003), para que os *logs* de um sistema sejam úteis para um administrador, eles devem estar com o horário sincronizado, ser tão detalhados quanto possível, sem, no entanto, gerar dados em excesso. Informações especialmente úteis são aquelas relacionadas a eventos de rede, tais como conexões externas e registros de utilização de serviços (arquivos transferidos via FTP, acessos a páginas *Web*, tentativas de login sem sucesso, avisos de disco cheio, etc.).

3.5 Controle de Acesso

O controle de acesso limita as ações ou operações que um sujeito de um sistema computacional pode executar, restringindo o que ele pode fazer diretamente, como também os programas que podem ser executados em seu nome. A Figura 1 apresenta o esquema básico do controle de acesso exercido através de mecanismos em um sistema computacional.

Figura 1 - Controle de Acesso



Fonte: Próprio autor

Os sistemas de controle de acesso podem ser diferenciados por suas políticas e seus mecanismos de acesso. As políticas de acesso são direcionamentos de alto nível, baseadas nas necessidades dos proprietários dos recursos ou ainda das organizações. A partir destas definições de alto nível devem ser geradas permissões que determinam como os acessos serão controlados, em todos os níveis do sistema.

4 IMPLEMENTANDO O SERVIDOR

Neste capítulo descreve-se um breve histórico da Instituição de Ensino que será usada para estudo de caso, bem como os principais serviços de acesso utilizado. Em seguida, é ilustrada a infraestrutura de rede local, LAN (*Local Area Network*) da escola. Após, é apresentado a implementação do servidor de Internet, iniciando com o *script* de *firewall* e a instalação do servidor *Proxy* e suas configurações, posteriormente, é abordada a instalação do serviço de DNS (*Domain Name System*).

4.1 Histórico da ETEC Polivalente de Americana

Escola Técnica Estadual Polivalente de Americana foi criada através de convênio entre o Governo do estado de São Paulo e a Prefeitura Municipal de Americana, no mês de setembro de 1972, recebendo o nome oficial de Colégio Polivalente.

O Colégio Polivalente iniciou suas atividades em março de 1977 oferecendo cursos de Mecânica, Administração, Secretariado e Edificações. A partir de 1981, foi integrado como Unidade Escolar do Centro Estadual de Educação Tecnológica "Paula Souza", passando a ser denominada Escola Técnica Estadual Americana. Nessa ocasião a Prefeitura Municipal de Americana fez a doação do imóvel ao CEETEPS, através de Lei Municipal.

Tendo em vista a predominância das indústrias têxteis na região, fez-se necessária a implantação do Curso Técnico Têxtil a partir de 1990.

Em dezembro de 1994 alterou-se novamente a denominação da Unidade para Escola Técnica Estadual Polivalente de Americana, em virtude de ser conhecida desde sua criação como "Polivalente". Seguindo as novas tendências no mercado de trabalho e sua diversificação para vários seguimentos produtivos e de serviços, atualmente a escola oferece, além dos cursos já implantados, os cursos Técnicos em: Comunicação Visual, Contabilidade, Design de Interiores, Logística, Segurança do Trabalho e Técnico em Eventos.

Em 1998 e 2001, a Unidade passou a contar, respectivamente, com as Classes Descentralizadas de Hortolândia e Santa Bárbara d’Oeste, que se tornaram Unidades Escolares em 2003. Em 2006, é implantada a Extensão de Nova Odessa

com os Cursos Técnicos em Administração e Segurança do Trabalho, que em 2009, também se tornou uma Unidade de Ensino independente do Polivalente.

Em 2011 foi implantado o primeiro Ensino Médio Integrado ao Técnico em Informática, ampliando em 2012 para os cursos técnicos Integrado ao Ensino Médio em Administração, Edificações, Logística e Mecânica.

4.2 Infraestrutura de Rede

A unidade de ensino conta com aproximadamente 420 computadores, distribuídos entre 20 laboratórios, sala de professores e administração.

A topologia de rede usada será a em estrela, onde toda a informação deve passar obrigatoriamente pelo servidor central, que deve conectar cada estação da rede e distribuir o tráfego para que uma estação não receba, indevidamente, dados destinados às outras.

Tendo em vista as condições financeiras, toda a implantação será com base nos recursos já disponíveis na escola.

Como não há fibra óptica, em alguns pontos os switches serão usados como “repetidores”, como é o caso do enlace que irá até o setor administrativo.

Há dois *links* de acesso geral à Internet para ser distribuído por todos os computadores, cada um com 100 Mbps.

Como cada *link* possui uma cota de 200 GB por mês, o limite é alcançado rapidamente e que esse fato está relacionado com a forma de utilização da Internet, ou seja, a maioria da banda utilizada era para fins particulares e de entretenimento, como: *facebook*, *downloads* de músicas e *softwares*, *streaming* de vídeos e rádios *online* etc; os quais dificultavam o uso de outras aplicações corporativas e afetando os serviços essenciais.

4.3 Implantação do *Firewall*

Para o Firewall, usar-se-á duas ferramentas:

- O *Squid*, além de ser um servidor *Proxy* que suporta HTTP, HTTPS, FTP, entre outros protocolos, também realiza o serviço de *cache;* ou seja, armazena as páginas já acessadas em disco para quando outro cliente realizar a requisição dessa mesma página seja enviada diretamente, aumentando assim a velocidade de

navegação. O *Squid* também é considerado um *firewall* de aplicação. Através dele é possível restringir o acesso a alguns usuários, conteúdos ou tipo de mídia (MORIMOTO, 2008).

- De acordo com a descrição do criador desse pacote (*NetFilter*), o *Iptables* é uma ferramenta para criar e administrar regras e assim filtrar pacotes de redes. O *iptables* pode funcionar baseado no endereço, porta de origem, destino do pacote, prioridade. Ele funciona através da comparação de regras para saber se um pacote tem ou não permissão para passar. Em *firewalls* mais restritivos, o pacote é bloqueado e registrado para que o administrador do sistema tenha conhecimento sobre o que está acontecendo em seu sistema.

A distribuição do *Linux* usada para instalar o *iptables* e o *Squid* é a Debian 9 – Stretch. Para a instalação é utilizada a ferramenta APT (*Advanced Packaging Tool*), a qual executa o *download* da aplicação ou serviço desejado. Depois do *download* realizado, o pacote é instalado automaticamente.

4.3.1 Filtragem

As regras de filtro de um *firewall* estão fundamentadas nas políticas de segurança da empresa. Na base dessa política deve-se ponderar dois fatores: nível de proteção e usabilidade, tentando o gestor buscar o nível de proteção o mais alto possível, porém sem deixar que partes importantes do sistema fique inoperantes. Uma boa maneira é impedir todo o tráfego e ir autorizando os serviços imprescindíveis para a organização (CHESWICK, BELOVIN e RUBIN, 2005).

De acordo com Ferreira (2003), existe dois gêneros de *firewalls*:

- Nível de aplicação: O *firewall* considera o conteúdo do pacote para ambasar suas decisões de filtragem. *Firewalls* deste tipo são mais intrusivos, pois avaliam o conteúdo de tudo que cruza por ele e permitem um controle em relação com o conteúdo do tráfego. Servidores *Proxy*, como o *Squid*, podem ser classificados como um exemplo.

- Nível de pacotes: O *firewall* calcula suas decisões baseado nos parâmetros do pacote, como porta/endereço de origem/destino, situação da conexão e outros parâmetros do pacote. O *firewall*, então, pode negar o pacote (*DROP*) ou aceita-lo (*ACCEPT*). O *iptables* é um *firewall* que se enquadra nesta descrição, o indiciado é que os dois tipos de *firewalls* sejam usados para prover uma camada dupla de segurança no acesso à rede interna.

4.3.2 *Iptables*

O *iptables* é um *firewall* em nível de pacotes que trabalha através da checagem de regras para saber se um pacote tem ou não autorização para continuar. Também pode ser usado para alterar e monitorar o tráfego da rede, fazer NAT (*masquerading, source nat, destination nat*), redirecionar pacotes, marcação de pacotes, alterar a prioridade de pacotes que chegam ou saem do servidor, contagem de bytes, dividir tráfego entre máquinas (ZWICKY; COOPER; CHAPMAN, 2005).

Netfilter (2018) afirma que as possibilidades oferecidas pelos recursos de filtragem *iptables* são vastas, pois pode garantir uma grande flexibilidade na administração de suas regras. Como a maior parte das ferramentas Linux, o *iptables* pode ser executado em máquinas com menor poder de processamento.

Netfilter (2018) afirmar ainda que seu funcionamento é com base em regras, que são informações passadas ao *iptables* para que realize uma alguma ação, como impedir ou autorizar um pacote de acordo com o endereço/porta de origem/destino, interface de origem/destino. Afirma também que as regras são guardadas dentro das *chains* que são usadas na operação do *firewall* e processadas na ordem que são implantadas. As regras das *chains* são armazenadas provisoriamente no *kernel*, o que significa que tudo pode ser perdido ao desligar o computador. Desta maneira, elas devem ser armazenadas em um arquivo para serem carregadas durante a inicialização do sistema.

Como mencionado, o *iptables* funciona com as regras gravadas nas *chains*, que por sua vez fazem parte de tabelas. As tabelas podem ser chamadas com a opção *-t* e existem 3 tabelas disponíveis no *iptables*, sendo que cada uma delas trata uma ação com o pacote recebido. O *iptables* trabalha com uma tabela de regras que é analisada uma a uma até que a última seja verificada. Como padrão, se uma regra tiver algum erro, uma mensagem será exibida e a regra descartada (NETFILTER, 2018).

As regras são lidas na ordem em que surgirem. Sempre que um pacote se encaixar a uma regra, tal regra processa o pacote e a sequência *iptables* é finalizada naquele momento, sem que as regras seguintes sejam verificadas. Isso não se aplica às regras terminadas com *-j LOG*. Nesse caso, a regra com *-j LOG* atua, se for o caso, permitindo a continuidade da sequência, a regra *–j LOG* é

utilizada para gerar logs se a regra é empregada, sendo armazenada no arquivo *kernel.log* (NETFILTER, 2018).

Segundo Netfilter (2018), as regras podem ser colocadas de maneira direta na linha de comando ou através de um *script* simples de *string* e acionado como *bash*. Geralmente as regras são escritas da seguinte maneira: *iptables* [*-t table*] [opção] [*chain*] [dados] *-j* [ação]. Exemplo, *iptables -A FORWARD -p tcp --dport 21 -j ACCEPT*.

4.3.3 *Implementação do firewall*

Para utilização do *script* de *firewall*, são aplicadas regras contra ofensivas já conhecidas. Por padrão, é realizado o bloqueio de todo tipo de diálogo que cruze o *firewall*. Após isso, é autorizada a comunicação necessária. Logo é utilizado o *Squid* que é um *Proxy/Firewall* em nível de aplicação.

Com a finalidade de melhorar o nível de segurança do servidor e da rede interna, com base no *iptables*, é adotado um *script* de *firewall* para o servidor de Internet. O *script* de *firewall* inicializa junto com o sistema operacional. Foi dada a autorização de execução do arquivo com o comando *chmod 700 script*, no qual, por questões de segurança, o responsável do arquivo tem permissão para executar, ler e gravar, o grupo e os outros usuário não têm permissão alguma.

Conforme disposto no quadro 2, o *script* inicia indicando o interpretador a ser utilizado, conforme apresentado na linha 1, após isso, são definidas as variáveis questão utilizadas no pelo código, da linha 5 a 60.

Para o correto funcionamento, todos os módulos do *firewall* devem ser enviados para a memória, conforme linhas 61 a 67. Para se ter certeza que não têm outras regras em memória intervindo no funcionamento do *script*, as regras são zeradas conforme linhas 70 a 77. Nas linhas 80 até 82 são determinadas as regras padrão para bloquear qualquer conexão de entradas, saída ou roteamento.

Nas linhas 84 a 106 estão modificadas algumas opções de operação do *kernel* do sistema operacional para dar uma maior segurança ao servidor e a toda rede. Se houver a necessidade de gerar algum *log* de auditoria ou para aferir o tráfego de informações no servidor é retirado o comentário das linhas 116 até a 119. O restante do *script* está repartido nas cadeias de regras *INPUT*, linhas 128 a 152. *OUTPUT*, linhas 161 a 180 e *FORWARD*, linhas 201 a 291, a função de cada regra disposta no *script* está explanada.

Quadro 2 - *Script* do *firewall*.

```
1  #!/bin/sh
5  IFDMZ=eth0
7  IFLAN=eth1
8  IFEB=eth2
9  GATEWAY=12.11.10.249/32
10 IPREDEDMZ=12.11.10.248/29
11 IPREDELAN=192.168.0.0/24
12 IPREDEEB=10.34.148.0/22
13 IPBRADCASTEB=10.34.151.255/22
14 IPGATEWAYEB=10.34.148.1/32
15 IPDNSEB=10.40.4.2/32
16 IPDMZ=12.11.10.250/32
17 IPLO=127.0.0.1
18 IPADMIN=192.168.0.11/32
19 ##Quem pode fazer acesso ao ftp
20 IPADMINA=192.168.0.11/32
21 #ip local do servidor
22 IPSERV=192.168.0.2/32
23 SERVDNS=12.11.10.249/32
24 SERVDNSB=201.10.1.2/32
25 SERVDNSC=12.11.10.250/32
26 NATFTP=12.11.10.249:20/32
27 NATFTPDATA=12.11.10.249:21/32
28 SERVSIRF=200.140.140.198/32
29 ## PORTAS CONHECIDAS ##
30 HTTP=80
31 ICP=3130
32 DNS=53
33 HTTPS=443
34 FTP=20
35 FTPDATA=21
36 PVPN=4500
37 VPNP=500
38 MSN=1863
39 POP=25
40 SMTP=110
41 FTP=21
42 SSH=22
43 SQUID=3128
44 WEBMIN=10000
45 SIRF=13352
46 RECEITANET=3456
47 SIAFIA=23000
48 SIAFIB=8999
49 REDECMDOEBA=465
50 REDECMDOEBB=995
```

```
51 SEFIP=2631
52 DGP=13353
53 ## IPS CONHECIDOS ##
54 IPSIAFI=161.148.40.200/32
55 iMesh=216.35.208.0/24
56 Napigator=209.25.178.0/24
57 APTDEBIAN = ftp.debian.org
58 APTDEBIANS = security.debian.org
59 SERSEFIP=200.201.174.204/32
60 ### Passo 1: Carrega os Módulos ###
61 modprobe ip_nat_ftp
62 modprobe ip_conntrack_ftp
63 modprobe iptable_nat
64 modprobe ip_tables
65 modprobe iptable_filter
66 modprobe iptable_mangle
67 modprobe ipt_MASQUERADE
68 echo "Carregando Modulos.................[ OK ]"
69 ### Passo 2: Limpa as regras ###
70 iptables -F INPUT
71 iptables -F OUTPUT
72 iptables -F FORWARD
73 iptables -F
74 iptables -Z
75 iptables -X
76 iptables -t nat -F
77 iptables -F -t nat
78 echo "Limpando Todas Regras...............[ OK ]"
79 # Define-se a Politica padrão das strings
80 iptables -P INPUT DROP
81 iptables -P FORWARD DROP
82 iptables -P OUTPUT DROP
83 echo "Definindo Ações Padrão..............[ OK ]"
84 #Parando tráfego da rede para aplicar as regras.
85 echo "0" > /proc/sys/net/ipv4/ip_forward
86 echo "Parando Tráfego da Rede.............[ OK ]"
87 # Configurando a Protecao anti-spoofing
88 for i in /proc/sys/net/ipv4/conf/*/rp_filter; do
89  echo 1 >$i
90 Done
91 echo "Proteção anti-spoofing..............[ OK ]"
92 #Numero de conexoes 64=4096, 128MB = 8192
93 echo "5000" > /proc/sys/net/ipv4/ip_conntrack_max
94 # Impedimos que um atacante possa maliciosamente alterar alguma rota
95 echo 0 > /proc/sys/net/ipv4/conf/all/accept_redirects
96 echo "Proteção anti-redirecionamento......[ OK ]"
   # Utilizado em diversos ataques, isso possibilita que o atacante determine o
97  "caminho" que seu
```

```
# pacote vai percorrer (roteadores) ate seu destino. Junto com spoof, isso se torna
 muito perigoso.
echo 0 > /proc/sys/net/ipv4/conf/all/accept_source_route
echo "Proteção contra pacotes com vida....[ OK ]"
# Protecao contra responses bogus
echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
echo "Proteção anti-bugus_response .......[ OK ]"
# Protecao contra ataques de syn flood (inicio da conexao TCP). Tenta conter ataques
 de DoS.
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
echo "Proteção anti-synflood .............[ OK ]"
## Logs para auditoria ##
#iptables -A INPUT -i $IFLAN -d ! $IPREDELAN -j LOG --log-prefix "INPUT"
#iptables -A FORWARD -j LOG --log-prefix "FORWARD "
#iptables -A OUTPUT -o $IFLAN -j LOG --log-prefix "OUTPUT REDE INTERNA "
#iptables -A OUTPUT -o $IFDMZ -j LOG --log-prefix "OUTPUT DMZ "
### Cadeia de Entrada (INPUT) ###
##################################
# LOCALHOST - ACEITA TODOS OS PACOTES
iptables -A INPUT -i lo -j ACCEPT
### Cadeia de Saida (OUTPUT) ###
#################################
##Conexoes estabilizadas e relatadas
iptables -A OUTPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
### Cadeia de Reenvio (FORWARD). ###
#####################################
# LOCALHOST - ACEITA TODOS OS PACOTES
iptables -A FORWARD -i lo -j ACCEPT
```

Fonte: Próprio autor

Com a implementação acima, podemos aplicar as políticas de segurança da instituição, restringindo acessos limitando possíveis riscos, além de simplificar o gerenciamento da segurança e possibilitar a implementação de serviços de auditorias.

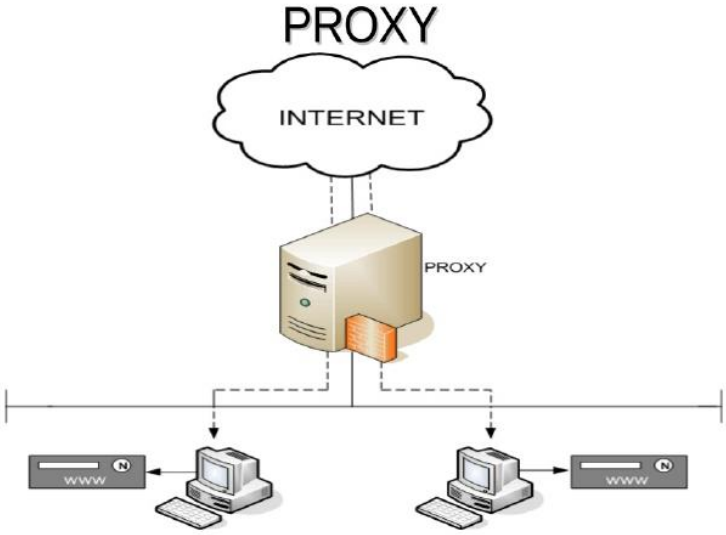
4.4 *Proxy*

Segundo Jucá (2005), o *Proxy* é um servidor que atende a requisições repassando os dados do cliente a frente. Um usuário (cliente) conecta-se a um servidor *Proxy*, requisitando algum serviço, como um arquivo, conexão, *website*, ou outro recurso disponível em outro servidor.

O *Proxy* é um programa que fica entre a rede local e a rede pública (Internet), realizando o controle na comunicação entre os dois lados (UCHÔA, 2003). O *Proxy*

trabalha como um intermediário entre cliente e o servidor, ou seja, ele recebe as requisições e repassa aos servidores. A figura 2 exemplifica uma topologia de rede com um servidor *Proxy.*

Figura 2 – Modelo de Topologia com base em um servidor



Fonte: Uchôa (2003)

De acordo com o CONFIGURATION GUID SQUIDE (2018), um bom servidor *Proxy* para se utilizar é o *Squid*, pois o mesmo apresenta confiabilidade, com inúmeras configurações que se adaptam a diversas situações de trabalho, hardwares e necessidades dos administradores de rede. O recurso que mais atrai no *Squid* é o cache de páginas, que é o recurso de guardar localmente as acessos, de forma a não ter que buscar outra vez na Internet uma página que tenha sido vista a pouco tempo por outro usuário da mesma rede.

O *Squid* fornece suporte na configuração do tempo de armazenagem de cada objeto no cache e o protocolo HTTP (*HyperText Transfer Protocol*) tem recursos suficientes para indicar páginas que não devem ser conservadas no cache, necessitando ser acessadas novamente a cada solicitação (MARCELO, 2003).

Quanto a instalação, é executada com o utilitário APT, através da linha de comando *apt-get install squid*, o mesmo baixa os arquivos necessários para a instalação, logo após, pré-configura o *Squid* com sua configuração padrão, precisando assim ser reconfigurado para melhor atender as necessidades do administrador da rede.

A versão instalada é a 3.5.27. As principais configurações que necessitam ser alteradas e otimizadas estão dispostas no arquivo */etc/squid/squid.conf* que é lido pelo *Squid* cada vez que é iniciado (MARCELO, 2003)

4.4.1 ACL (*Access Control List*)

Segundo Marcelo (2003), as ACLs são regras estáticas de navegação via *Proxy*, sendo interpretadas na ordem em que aparecem. Toda vez que um pacote HTTP atravessa o *Proxy* as regras são aplicadas. Quando uma regra for verdadeira, ele para a leitura da mesma neste ponto, não passando para a próxima. Na configuração do *Squid*, primeiro se definem as ACLs e seu conteúdo e depois as regras.

Marcelo (2003) explica que uma ACL é escrita da seguinte forma: *(acl NOME_DA_ACL TIPO_DA_ACL parâmetro)*, os parâmetros são escritos em seqüência, separados por espaço. Outro exemplo pode ser: *(acl NOME_DA_ACL TIPO_DA_ACL /caminho/completo/regra.conf)*, no qual, o arquivo *regra.conf* estão dispostas as *strings*, uma por linha, geralmente são utilizados para regras com muitas *strings*. Lembrando que todas as ACLs não são tratadas como *case-sensitive* (diferença entre letras maiúsculas e minúsculas). Para definir como *case-insensitive* deve se utilizar a opção *-i* logo após o tipo de ACL.

Depois de definidas as ACLs, deve-se escrever as regras que permitam ou neguem os acessos. Para isso é utilizado o *http_access*, seguido de *allow* ou *deny*. Se a ACL for precedida de um ponto de exclamação (*!*) significa a negação da ACL. A regra *http_access* é escrita da seguinte maneira: (*http_access 'allow/deny' '!' 'nome da acl'*).

Com base nas ACLs acima, segue alguns exemplos de aplicação de regras: *http_access allow DIRETOR* - acesso liberado para o diretor baseado em *login*; *http_access allow SITE_INTEGRAL !HORARIO_ALMOCO* - acesso liberado ao *site integral.inf.br* exceto no horário do almoço; h*ttp_access deny EXTENSAO_PROIBIDA* - extensão proibida baseado em expressão regular.

4.4.2 Configuração implementada

A instalação e configuração do *Squid* são executadas no servidor. O quadro 3 mostra o arquivo *squid.conf* com as configurações implementadas. Cabe ressaltar

que as configurações padrões do *Squid* não estão incluídas, apresentam-se somente as que foram alteradas após os testes preliminares de desempenho (CONFIGURATION GUIDE SQUID, 2018).

A primeira linha do arquivo de configuração indica qual porta o *Squid* está monitorando as requisições. Por padrão, esta porta é a 3128, podendo ser alterada conforme necessidade, a opção *transparent* possibilita que o *Squid* capture as requisições redirecionadas pelo iptables. Na linha 2 é definida a quantidade máxima de memória RAM utilizada para *cache* de objetos. Já, na linha 3, é definido o tamanho máximo do arquivo gravado no *cache* de disco rígido. Na linha 4 configura-se o tamanho máximo do arquivo armazenado nos 85 MB de memória RAM, definidos anteriormente.

Na linha número 6 é definida a política de reposição dos arquivos na memória RAM como GDSF (*Greedy-Dual Size Frequency*) que permite que permaneçam no *cache* os objetos menores e mais referenciados. Esta opção é a que obteve maior performance nos testes efetuados, pois permitem que permaneça no *cache* mais objetos de menor tamanho. Uma vez que a quantidade de memória RAM definida para o *cache* é pouca, caso contrário, permanecem poucos objetos maiores e a possibilidade do mesmo ser referenciado é menor. Na linha 7 é definida a política de reposição dos arquivos no *cache* de disco, neste caso optou-se por utilizar LFUDA (*Least Frequently Used with Dynamic Aging*), que mantém no *cache* os objetos mais referenciados independente do tamanho, porém, a determinação da melhor política de substituição de arquivos não foi explorada neste trabalho. A linha 8 define o caminho no qual são gravados os arquivos de *cache*, o tamanho total em MB e o número de diretórios primários e secundários, neste caso está sendo utilizado 1.2 GB de disco rígido para *cache*, cabe ressaltar que, quanto maior o espaço definido, mais lenta é a procura dos objetos dentro do *cache*.

As linhas 9, 10 e 11 definem os locais em que são gravados os arquivos de log, sendo que *access_log* é o arquivo onde se registram as páginas que os usuários navegam, o arquivo *cache.log* registra as mensagens de operação e possíveis erros do *Squid* e o *store.log* registra informações para uso estatístico sobre o armazenamento dos arquivos no disco e na atual configuração está desabilitado. Continuando, a linha 12 define o número de processos responsáveis pelos direcionamentos dos pacotes de dados. A linha 13 define o *Squid* como *Proxy-caching web server* não precisando da autenticação dos usuários para funcionar.

Da linha número 14 até linha número 32 definem-se o conteúdo das listas de controles de acesso, esse conteúdo pode ser definido na própria linha da ACL ou em arquivo separado.

A linha 18 é um exemplo ilustrando o conteúdo da ACL *Palavras Bloqueadas*, a qual se encontra no arquivo *bloquear* que está gravado no diretório */etc/squid/*. Quando um determinado site não condiz com a finalidade do serviço da unidade, ele vai para a *black list*, ou quando um site que deve estar livre por algum motivo é bloqueado, seu endereço vai para lista de sites desbloqueados.

Depois de definido o conteúdo das ACL é implementam-se as regras de acesso, que estão dispostas no arquivo de configuração da linha 33 até a linha 47. A regra da linha 33 permite o trâmite de protocolo de *cache* de objetos, é necessário para o funcionamento do *Squid* e vem como padrão. A linha 34 permite que todas as máquinas conectadas à rede, com este servidor definido como *gateway*, atualizem o antivírus.

A linha 35 nega o acesso a todas as máquinas que o IP for diferente dos definidos na ACL linha 21, que por sua vez indica o arquivo *iptodos*. A linha 36 permite o acesso às portas seguras.

A linha 37 permite o acesso sem restrição dos números IPs definidos na ACL da linha número 28, pois sem esta regra, alguma expressão da URL do site da UNIPLAC, que na comparação é verdadeira com na regra de bloqueio, não bloqueie, pois está autorizado em uma regra de maior prioridade. Na linha 38 se aplica o mesmo da linha 37, mas autorizando o acesso irrestrito às expressões encontradas nas ACL linha 32. Na linha 39 são bloqueados os downloads dos arquivos com a extensão da ACL linha 22.

Na linha 40 é permitido o acesso do IP definido na ACL na linha número 30 ao endereço da ACL da linha número 31. A linha 41 permite acesso irrestrito a todos endereços que contenham as expressões *sp.gov.br, urh.cps.sp.gov* e *urhsistemas.cps.sp.gov.br.*

Na linha 42 são bloqueados todos os sites do arquivo próprio para os bloqueios. A linha 43 permite o acesso a certas expressões que são bloqueadas, mas não deviam, em sites que estão definidos na respectiva ACL, como exemplo, a palavra game se encontra bloqueada em uma ACL, se o usuário deseja efetuar uma pesquisa no *Google* da palavra pagamento, esta é bloqueada caso esta regra não exista, pois no meio da palavra pagamento se encontra a expressão game que posteriormente é bloqueada.

A linha 44 permite acesso às expressões da ACL da linha 27 pelos usuários possuidores do IP da ACL na linha 26. Na linha 45 é liberado o acesso a diversos sites que por algum motivo entraram no bloqueio, mas não devem. A linha 46 bloqueia qualquer expressão na URL que for igual à lista do arquivo referenciado.

A linha 47 permite o acesso às portas conhecidas usadas pelo navegador e pelo *Squid*. Na linha 48 define que o *Squid* responderá todas as requisições da rede interna. A linha 49 permite a comunicação do *Squid* com outro *cache* através da porta ICP (*Internet Cache Protocol*) definida na linha 57.

Na linha 50 é definido o tamanho máximo dos cabeçalhos HTTP permitidos, os que passam deste limite são descartados, pois podem ser algum tipo de ataque ao servidor. A linha 51 define o nome do *host*. Na linha 52 é definido para aceitar cabeçalhos HTTP fora do padrão. A linha 53 ativa o balanceamento de servidores de nomes. A linha 54 e 55 definem os nome e grupo que o *Squid* vai assumir dentro do sistema operacional. Finalmente, a linha 56 define o tamanho em MB do *cache* de IP de páginas visitadas, diminuindo assim as consultas ao servidor de nomes.

Quadro 3 – Configuração do *Squid*

```
 1  http_port 192.168.0.2:3128 transparent
 2  cache_mem 85 MB
 3  maximum_object_size 13 MB
 4  maximum_object_size_in_memory 7 KB
 5  fqdncache_size 4096
 6  memory_replacement_policy heap GDSF
 7  cache_replacement_policy heap LFUDA
 8  cache_dir ufs /cache/squid 1200 16 256
 9  cache_access_log /cache/logs/access.log
10  cache_log /cache/logs/cache.log
11  cache_store_log none
12  redirect_children 9
13  auth_param basic realm Squid Proxy-caching web server
14  acl manager proto cache_object
15  acl all src 192.168.0.0/255.255.255.0
16  acl SSL_ports port 443 563
17  acl Safe_ports port 20 21 80 3128 10000
18  acl PalavrasBloqueadas url_regex -i "/etc/squid/bloquear"
19  acl HostPermitidosAll dstdomain "/etc/squid/liberar"
20  acl Site_Bloqueado dstdomain "/etc/squid/sitesb"
21  acl IPpermitido src "/etc/squid/iptodos"
    acl Download urlpath_regex -i \.mp3$ \.wav$ \.mov$ \.mpg$ \.mpeg$ \.avi$ \.flv$ \.rpl$
22   \.wmv$ \.wmv?$ \.wmf$
25  acl END_Eb_Gov_Perm url_regex .sp.gov.br
```

```
26 acl IPsecinfo src 192.168.0.09 192.168.0.10 192.168.0.11 192.168.0.3 192.168.0.4
   acl STR_IPSecinfor url_regex download downloads driver vga placa baixaki crack ftp
27  details jsp jpr jsc cfm huffman
28 acl Uniplac dst 200.135.4.222 64.58.88.108 65.55.197.125
29 acl Antivirus url_regex .avast.com .grisoft.cz .grisoft.com
30 acl IPGHearth src 192.168.0.154
31 acl GHearth dstdomain kh.google.com
32 acl SiteExp url_regex login.live.com mail. hotmail .sun.com
33 http_access allow manager
34 http_access allow Antivirus
35 http_access deny !IPpermitido
36 http_access allow SSL_ports
37 http_access allow Uniplac
38 http_access allow SiteExp
39 http_access deny Download
40 http_access allow IPGHearth GHearth
41 http_access allow END_Eb_Gov_Perm
42 http_access deny Site_Bloqueado
43 http_access allow NomesPermitidosHost HostPermitidosNome
44 http_access allow STR_IPSecinfor IPsecinfo
45 http_access allow HostPermitidosAll
46 http_access deny PalavrasBloqueadas
47 http_access allow Safe_ports
48 http_reply_access allow all
49 icp_access allow all
50 reply_header_max_size 20 KB
51 visible_hostname DEBadsl
52 relaxed_header_parser on
53 balance_on_multiple_ip on
54 cache_effective_user Proxy
55 cache_effective_group Proxy
56 ipcache_size 4120
57 icp_port 3130
```

Fonte: Próprio autor

A implementação do *script* acima é importante para segurança dos dados da rede e dos servidores. Buscou-se melhorar o tempo de resposta da navegação de Internet, com uma sintonia fina de seu *cache* e regras de bloqueio estáticas, com acessos direcionados às finalidades não particulares, ocasionando a diminuição do uso do *link*, liberando assim, a banda passante.

4.5 DNS (*Domain Name System*)

Para que funcione a resolução de nomes na rede interna, é necessário ser instalado no servidor, o serviço de DNS, sistema de nomes de domínios. Segundo Ferreira (2003), um dos melhores serviços de resolução de nomes para *Linux* é o BIND 9 (*Berkeley Internet Name Domain*). Para sua instalação no *Debian*, basta digitar na linha de comando *apt-get install bind*, com isso, os arquivos necessários são baixados do repositório e instalados no servidor para posterior configuração.

Por se tratar de um servidor relativamente pequeno, o serviço de DNS é configurado para repassar as requisições de resolução de nomes para outro servidor primário, fazendo apenas o *forward* das requisições. Para isso, o arquivo *named.conf.local* deve ser editado, conforme apresentado no quadro 4.

Quadro 4 – Configuração do BIND

```
 1  zone "10bec" {
 2   type forward;
 3   forwarders {
 4   12.11.10.249 port 53;
 5   201.10.1.2 port 53;
 6   201.10.120.3 port 53;
 7   };
 8   forward first;
 9   check-names ignore;
10   };
```

Fonte: Próprio autor

Na linha 1 do arquivo de configuração do BIND, do quadro 4, define-se o nome da zona como *10bec*. A próxima linha define o modo de trabalho do BIND como *forward*, ou seja, para repassar as requisições de resolução de nomes. A linha 3 define os endereços dos servidores de nomes do provedor de acesso à Internet.

As linhas 4, 5 e 6 definem os IPs dos servidores propriamente ditos, e a porta de acesso do mesmo. Como padrão, o acesso do serviço de resolução de nomes é a porta 53 e o protocolo geralmente é o UDP. Na linha 8 o BIND está configurado para repassar imediatamente todas as requisições DNS, por último, a linha 9 configura o BIND, no caso, ao encontrar *hosts* inválidos (*fail*); ignorar a consulta.

4.6 *Proxy* com Regras Dinâmicas

Aqui há necessidade na abordagem da implementação de regras dinâmicas de bloqueio no *Proxy*, sendo introduzidos alguns conceitos da linguagem de programação *Perl*, essenciais para esta implementação. Explora-se também, outras funcionalidades do *iptables*, indispensáveis para executar a medição do uso do *link* de dados.

Dando continuidade, é apresentado o código do programa implementado que permite as alterações das regras do *Squid*, por fim, após testes de funcionalidades, aborda-se o conteúdo das regras de bloqueio.

4.6.1 Regras de Bloqueio

A implementação do *Proxy Squid*, descrita ao longo do capítulo, melhorou a conectividade da unidade de ensino, desafogando o *link* de dados e aprimorando a segurança da rede interna. Porém, trouxe consigo uma grande insatisfação por parte dos servidores, que estavam acostumados a ter acesso irrestrito aos sites da Internet. Por outro lado, este bloqueio restringiu muito o acesso deixando o *link* com banda livre, ou seja, mal aproveitado.

Segundo Configuration Guide Squid (2018), o *Squid*, por padrão, permite o uso das regras de bloqueio definidas por intervalos de horários, que poderiam ser implementadas no servidor, porém, não há como definir horários, pois o trabalho dentro da ETEC Polivalente de Americana é executado de acordo com aquela necessidade, fazendo, por muitas vezes, que os funcionários trabalhem a noite e não trabalhem durante a tarde. Portanto, inviabilizando a referida implementação.

Pensando no problema descrito, as regras de bloqueio dinâmicas são propostas, as quais são baseadas na informação de utilização do *link* de dados; permitindo assim, a melhor utilização do mesmo para atividades de rotina da unidade escolar, sem comprometer ou comprometendo menos as necessidades particulares dos funcionários.

4.6.2 Linguagem *Perl*

O programa para tornar as regras de bloqueio dinâmicas foi implementado na linguagem *Perl*. Segundo Wall, Christiansen e Orwant (2001), Perl é uma linguagem de programação estável e multi-plataforma, usada em aplicações de missão crítica em todos os setores, usado mais comumente em sistemas Unix e compatíveis.

Criada por Larry Wall em dezembro de 1987, a origem do *Perl* remonta ao *shell scripting, Awk* e linguagem C, estando disponível para praticamente todos os sistemas operacionais. Originalmente, o nome proposto e usado foi descrito por Larry Wall como Practical Extraction and Report Language (WALL; CHRISTIANSEN; ORWANT, 2001).

Perl é uma das linguagens preferidas dos administradores de sistemas e autores de aplicações para a *web*. É especialmente versátil no processamento de cadeias strings, manipulação de texto e no *pattern matching* implementado através de expressões regulares, além de permitir tempos de desenvolvimento curtos. No geral, a sintaxe de um programa em Perl se parece muito com a sintaxe da linguagem C. Os principais exemplos de similaridades são a existência de variáveis, expressões, atribuições, blocos de código delimitados, estruturas de controle e sub-rotinas (WALL; CHRISTIANSEN; ORWANT, 2001).

O Perl é bastante influenciado pelas linguagens de *shell script*: todas as variáveis escalares são precedidas por um cifrão $. Essa marcação permite identificar perfeitamente as variáveis num programa, onde quer que elas estejam. Um dos melhores exemplos da utilidade desse recurso é a interpolação de variáveis diretamente no conteúdo de *strings*. *Perl* também possui muitas funções integradas para tarefas comuns como ordenação e acesso de arquivos em disco (WALL; CHRISTIANSEN; ORWANT, 2001).

4.6.3 Implementação do programa de dinamicidade

Com o apoio didático obtido, o programa que permite a dinamicidade das regras do *Proxy* é implementado conforme ilustrado no quadro 5. Seu funcionamento básico é baseado no total de *Bytes* que trafegam na interface de rede do servidor. Para obter esses totais existem várias maneiras, segundo Ferreira (2003), uma opção a ser utilizada é implementar o serviço de SNMP (*Simple Network Management Protocol*), porém, a utilização do mesmo, traz consigo muitos

problemas como a utilização dos recursos do servidor e rede, possíveis brechas na segurança do servidor. Por estes motivos e por razões de simplicidade, optou-se pela sua não utilização na implementação do programa de dinamicidade.

Buscando uma nova forma de obter os valores totais de *Bytes* trafegados na interface, são utilizadas algumas funcionalidades do *iptables*, que segundo Netfilter (2018), permite a contabilização de pacotes/*Bytes* por regra aplicada. Para liberar a contabilização do tráfego de entrada para rede, *download*, é necessário o seguinte comando: *iptables -t mangle -A PREROUTING -i eth0 -j ACCEPT*, o qual permite a passagem dos pacotes vindos da interface de rede externa pela tabela *mangle*, responsável por alteração no pacote de dados, antes de serem roteados pelas outras tabelas do *iptables*.

A partir do momento em que os pacotes passam pela tabela *mangle*, eles são contabilizados. O comando *iptables -t mangle -L PREROUTING –nvx*, permite verificar esta contabilização que retorna um relatório com as informações da regra especificada. A opção *–L* é utilizada para listar os pacotes que passam pela *chain PREROUTING* da tabela *mangle*, a opção *–nvx* previne que o *iptables* tente resolver os endereços IP, mostra todos os detalhes das regras, como os contadores de pacotes e bytes, com os números impressos inteiros, independente de seu tamanho, omitindo os sufixos K, M ou G para 1.000, 1.000.000 e 1.000.000.000 respectivamente.

Quadro 5 – Código para alteração do comportamento do *Squid*

```
1  #!/usr/bin/perl
2  ##################################################
3  #Programa escrito na linguagem Perl por Dian Gidaro #
4  # Objetivo alterar o estado de funcionamento do squid depen #
5  # dendo do uso do Link de dados #
6  # Deve ser executado a cada 5 minutos #
7  ##################################################
8  @inn=`iptables -t mangle -L PREROUTING -nvx`;
9  @in= map { split ' ' } @inn;
10 $in = @in[18];
11 open (ARQ, "bd.in");
12 read (ARQ, $calc, 13);
13 open (ARQ, ">bd.in");
14 print ARQ $in;
15 $banda = (($in - $calc)/1024);
16 $alta = 32400;
17 $media = $alta/2;
```

```
18  if ($banda >= $alta)
19   {
20   &squid ("/etc/squid/bloquearA", "/etc/squid/sitesbA");
21   &log ("Uso PLENO do Link");
22   }
23  if ( ($banda >= $media) && ($banda <= $alta) )
24   {
25   &squid ("/etc/squid/bloquearM", "/etc/squid/sitesbM");
26   &log ("Uso MODERADO do Link");
27   }
28  if ($banda <= $media)
29   {
30   &squid ("/etc/squid/bloquearB", "/etc/squid/sitesbB");
31   &log ("Uso BAIXO do Link");
32   }
33  sub squid
34   {
35   unlink </etc/squid/bloquear>;
36   unlink </etc/squid/sitesb>;
37   symlink "$_[0]", "/etc/squid/bloquear";
38   symlink "$_[1]", "/etc/squid/sitesb";
39   exec (`squid -k reconfigure`);
40   }
41  sub log
42   {
43   $tráfego = $banda / 300;
     ($segundo,$minuto,$hora,$dia_m,$mes,$ano,$dia_s,$dia_a,$isdst)=lo
44   caltime(time);
45   open (ARQ, ">>status.log");
     print ARQ "$dia_m-$mes / $hora H - $minuto M -- $_[0] [Media: $tráfego
46   KBps]\n";
47   }
```

Fonte: Próprio autor

O código que possibilita a dinamicidade do servidor é implementado em *Perl* conforme mostra no quadro 5. Este código está dividido em 3 partes, a primeira é responsável pela coleta do total de *Bytes* trafegados na interface de rede, a segunda responsável pelo cálculo da banda de dados, por último a parte responsável pela alteração das regras de bloqueio do *Squid*.

A primeira parte do código, responsável pela coleta do total de *Bytes* trafegados, inicia na linha 1, declarando o interpretador do código, no caso o *Perl*, a linha 8 atribui a variável da saída do comando *iptables*. As variáveis iniciadas com @ são *arrays* e as iniciadas com *$* são variáveis simples.

A linha 9 contém a função *map* do *Perl* que separa o conteúdo da variável *@inn* por espaços em branco, e aloca cada palavra em uma posição do *array @in*. As informações são apresentadas conforme ilustrado na figura 6 e o que interessa é somente o valor em baixo da palavra *bytes*. Na linha 10 é atribuída à variável $in o valor total de *Bytes* trafegados na interface de rede que está alocado na posição 18 do *array @in*.

A segunda parte do código, responsável pelo cálculo da banda de dados, funciona lendo o valor de *Bytes* anterior, do arquivo bd.in, alocando o mesmo na variável $*calc*, conforme linhas 11 e 12, respectivamente. Sempre que é executado, o código captura o total de *Bytes* anterior do arquivo mencionado, alocando-o em outra variável, logo após este valor é substituído pelo atual que na próxima execução será o anterior.

A gravação do conteúdo atual de *Bytes* pode ser visualizado na linha 13, que abre o arquivo. A linha 14 grava o conteúdo da variável $*in* no arquivo aberto. O cálculo da banda é executado na linha 15, na qual a variável $banda recebe o valor do total de Bytes atual (–) menos o valor total de Bytes anterior (/) dividido por 1024 para transformar de Bytes para KiloBytes.

O programa irá rodar de cinco em cinco minutos, ou seja, trezentos segundos, o valor máximo de *KiloBytes* é de 300 (*) vezes 108 = 32400 KBp/5min (*Kilobyte* por cinco minutos) a partir deste valor máximo o programa executa suas ações. O valor 108 é obtido nos status do modem ADSL (*Asymmetric Digital Subscriber Line*), onde a taxa de download identificada é de 864 *Kbps* (*Kilobit* por Segundo), convertendo para *KiloBytes* se obtém 108 *KBps* (*Kilobyte* por Segundo). Cabe ressaltar que, a execução do programa consome insignificantes recursos de processamento e memória, e seu tempo de execução é menor que um segundo.

A terceira parte do código é responsável pela alteração das regras de bloqueio do *Squid*, composta por três estruturas condicionais (linhas 18, 23 e 28). Inicialmente é definido qual o valor da banda é alta e média conforme linha 16 e 17, o valor da banda alta é abordado no parágrafo anterior e o da média é a metade do valor da alta.

A instrução *if* da linha 18 executa somente se a banda estiver com seu uso pleno no período dos cinco minutos, já a instrução *if* da linha 23 executa somente se a banda estiver com seu uso além da metade no período dos cinco minutos. A instrução *if* da linha 28 é executada somente se a banda estiver com seu uso menor que a metade.

O conteúdo de cada *if* chama duas funções distintas, passando diferentes parâmetros de acordo com o programado, uma que altera os arquivos da ACL (*Access Control List*) do *Squid* e outra que grava um log para acompanhamento das alterações de estados das ACL.

A função *squid* da linha 33 recebe como parâmetro o local do arquivo com a ACL do respectivo *if* executado, porém, como o *Squid* possui somente um único arquivo de configuração, o qual aponta para um arquivo fixo contendo as palavras e sites que devem ser bloqueados, houve a necessidade alteração desta configuração. Desta forma, o arquivo de configuração do *Squid* aponta para um *link* simbólico */etc/squid/bloquear* e */etc/squid/sitesb*, que por sua vez aponta para o arquivo com o verdadeiro conteúdo que é recebido pela função como parâmetro. A função remove o *link* atual linhas 35 e 36 e cria outro *link* com o mesmo nome, porém, este novo *link* está relacionado com outro arquivo (linhas 37 e 38). Finalmente, executa a função que permite o funcionamento do *Squid* com o parâmetro *–k* reconfigure que recarrega as suas configurações com as devidas alterações.

Ressalta-se ainda, a função *log* da linha 41 que recebe como parâmetro a mensagem referente a situação atual do *link*. Em seguida converte o valor de KBp/5min para KBps conforme linha 43, na linha 44 é feito o uso da função *localtime(time)* do Perl que repassa os valores de data e horas locais para as variáveis dentro do parênteses.

A linha 45 abre o arquivo *status.log*, as setas duplas (>>) significam que deve-se acrescentar o conteúdo no arquivo sem substituir o conteúdo anterior. A linha 46 grava o conteúdo das variáveis no arquivo *status.log*, gerando assim o log com diames / horas-minutos - uso do *link* e média de KBps conforme mostra no quadro 6.

A execução do *script* se da com a utilização do utilitário *cron*, que possibilita agendar a execução do *script* de 5 em 5 minutos, as funcionalidades do *cron* não são abordadas neste trabalho de conclusão de curso.

Quadro 6 – Log gerado pelo programa do *Proxy* dinâmico

```
9-4 / 1 H - 66 M -- Uso PLENO do Link Media de 108.7222070312 KBps
9-4 / 1 H - 71 M -- Uso PLENO do Link Media de 108.2936035156 KBps
9-4 / 1 H - 76 M -- Uso BAIXO do Link Media de 0.02955078125 KBps
9-4 / 1 H - 81 M -- Uso BAIXO do Link Media de 0.290983072916667 KBps
9-4 / 1 H - 86 M -- Uso BAIXO do Link Media de 2.57131184895833 KBps
9-4 / 1 H - 91 M -- Uso BAIXO do Link Media de 5.8275390625 KBps
9-4 / 1 H - 96 M -- Uso MODERADO do Link Media de 70.9327734375 KBps
9-4 / 2 H - 01 M -- Uso BAIXO do Link Media de 0.161725260416667 KBps
9-4 / 2 H - 06 M -- Uso BAIXO do Link [Media: 0.00949544270833333 KBps]
9-4 / 2 H – 11 M -- Uso PLENO do Link [Media: 108.32222i32323 KBps]
```

Fonte: Próprio autor

4.6.4 Conteúdo das regras de bloqueio

Com o programa de controle de regras do *Squid* implementado e rodando no servidor, é definido o conteúdo das ACL do *Proxy Squid*. Este conteúdo é alterado de acordo com a regra em vigor e o uso da banda de dados, para isto, o programa rodou por um período de 14 dias onde periodicamente eram verificados o uso da banda de dados e qual ACL estava em vigor, ou seja, no caso de uma taxa de utilização elevada deve-se bloquear mais sites e palavras, se a taxa e utilização é baixa libera-se uma quantidade maior de sites, de acordo com a respectiva ACL.

As regras foram implementadas a partir do arquivo de ACL que estava implantado e rodando estaticamente, o qual ficou definido no uso moderado do *link*. Após os ajustes, foram definidas uma ACL para cada estado de utilização do *link* de dados: uso baixo, moderado e alto.

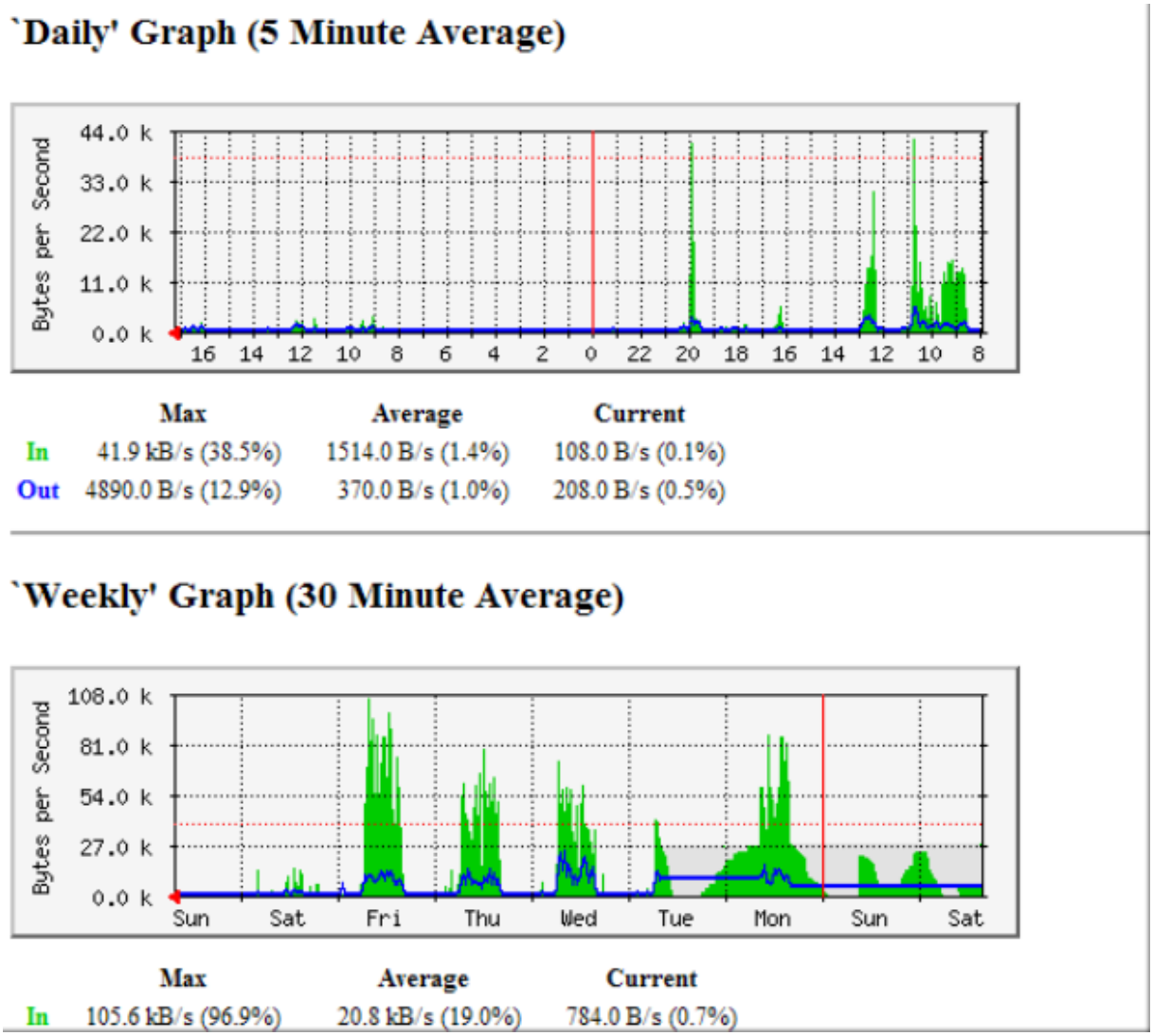
5 RESULTADOS E TESTES DE DESEMPENHO

Neste capítulo aborda-se a instalação e o uso do MRTG (*Multi Router Traffic Grapher*), software livre que gera gráficos do uso do *link* de dados. O objetivo de utilização deste software, no contexto deste trabalho, é permitir uma melhor visualização do real uso da banda de dados da Internet, efetuando uma análise comparativa entre as implementações das regras de bloqueio do *Proxy* e *firewall*. Além disso, apresenta-se neste capítulo alguns testes de navegação, tais como: tempos de *ping*, *upload* e *download* de arquivos. A finalidade destes testes é verificar a velocidade do *link* e as influências do servidor *Proxy* na rede interna da Unidade de Ensino. Maiores informações sobre o MRTG podem ser encontradas na página: http://oss.oetiker.ch.

5.1 Instalação e uso do MRTG

O MRTG é um software livre, distribuído nos termos da GNU, sendo uma ferramenta de monitoração que gera páginas HTML (*HyperText Markup Language*) com gráficos de dados coletados a partir de SNMP (*Simple Network Management Protocol*) ou *scripts* externos que retornem os dados no seu formato padrão. É conhecido principalmente pelo seu uso na monitoração do tráfego de rede, mas pode monitorar qualquer coisa, desde que o *host* forneça os dados via SNMP ou *script*. Foi desenvolvido por Tobias Oetiker e Dave Rand, escrito na linguagem Perl utilizando um módulo na linguagem C para gerar os gráficos (FERREIRA, 2003).

Figura 3 – HTML gerado pelo MRTG



Fonte: Próprio autor

O MRTG sempre mede 2 valores, os quais representam a entrada e saída do tráfego. No processo de monitoração os dados são coletados, por padrão, a cada 5 minutos, porém, este tempo pode ser alterado. O software cria automaticamente uma página HTML com quatro gráficos (figura 3), sendo eles: diário, semanal, mensal e anual. As informações apresentadas na cor verde representam a entrada de dados ou download e a cor azul representa a saída de dados ou upload, logo abaixo dos gráficos, é apresentado o uso máximo do *link*, a média e o uso atual do *link* no período. O MRTG possibilita avisar caso o valor do gráfico atinja um valor pré-estabelecido (*threshold*). Por exemplo, se determinado servidor atinge 95% do uso do *link*, o MRTG manda um e-mail para o administrador informando o ocorrido. Possui também a ferramenta *INDEXMAKER*, utilizada para gerar uma página de índices para os casos em que muitos itens são monitorados (FERREIRA, 2003)

Como citado anteriormente, MRTG faz uso do protocolo SNMP para capturar os dados da interface de rede, porém, o referido serviço não é implementado no servidor em estudo neste TCC, pois a utilização do mesmo acarreta consigo muitos problemas, como a utilização dos recursos do servidor e rede, possíveis brechas na segurança, dentre outros. Para resolver o problema da não utilização do protocolo SNMP é implementado um *script* que efetua a leitura dos dados da interface de rede. Este *script* retorna a quantidade total de *bytes* trafegados na interface sempre que é executado, possibilitando assim, que o MRTG efetue o cálculo da banda e gere os gráficos de utilização do *link*, o funcionamento do *script* é abordado no item 5.1.1 deste trabalho.

Para efetuar o *download* e a instalação do MRTG, autenticado como *root*, é inserido o comando: *apt-get install mrtg,* e sua instalação prosseguirá automaticamente. Por padrão o arquivo de configuração do MRTG fica gravado no */etc/mrtg.cfg,* seu conteúdo é mostrado no quadro 8, no qual, a linha 7 define o local que são gravados os gráficos gerados.

A linha 8 define o uso de meta *tags* de validade nos arquivos HTML gerados, a próxima linha define que o MRTG é executado em segundo plano, *background*, na linha 10 é definido o título da janela do navegador quando abrir o arquivo HTML com os gráficos.

A linha 12 define o título da página e a 13 o sub-título, na linha 14 é definido o local onde o MRTG irá capturar o total *bytes* trafegados na rede, no caso, irá executar o *script* que retorna os valores de *upload* e *download*, por fim, as linhas 15 e 16 definem o valor total de Mbps (*Megabits* por Segundo) que o *link* permite trafegar referentes ao 100% do gráfico, o valor da linha 15 referência o valor de *download*, cor verde do gráfico e o da linha 16 o valor de *upload*, cor azul do gráfico, o valor máximo é representado com uma linha vermelha no gráfico.

Depois de configurado, o MRTG é executado com o comando: *mrtg -P /etc/mrtg.cfg* e deve ser colocado para iniciar no *boot* do Linux, para isso, o comando acima citado é colocado em um arquivo de texto chamado *mrtg*, é dada a permissão de execução do arquivo com o comando *chmod 700 mrtg*, no qual, por motivos de segurança, somente o dono do arquivo tem permissão para executar, ler e gravar. Após, o arquivo é copiado ao diretório */etc/init.d* e adicionado na inicialização do Linux, com o comando *update-rc.d mrtg defaults*. No quadro 7, pode-se observar o arquivo de configuração do MRTG.

Quadro 7 – Configuração do MRTG

```
1   ##################################################################
2   # Multi Router Traffic Grapher -- Sample Configuration File
3   ##################################################################
4   # This file is for use with mrtg-2.5.4c
5   #Local: /etc/mrtg.cfg
6   # Global configuration
7   WorkDir: /var/www/mrtg
8   WriteExpires: Yes
9   RunAsDaemon: Yes
10  Title[^]: Grafico do uso do link ADSL
11  # ----------------
12  Title[leased]: ADSL de 1Mb
13  PageTop[leased]: <H1>Grafico de uso do LINK ADSL de 1Mb </H1>
14  Target[leased]: `/root/mrtg/ler`
15  MaxBytes1[leased]: 109000
16  MaxBytes2[leased]: 40000
```

Fonte: Próprio autor

5.1.1 *Script* de leitura dos dados da interface de rede

O arquivo de configuração, apresentado no quadro 7, do MRTG faz referência ao *script bash* LER conforme linha 14, que retorna os valores de *upload* e *download* para que o MRTG gere os gráficos. Esses valores são capturados do *iptables* utilizando as funcionalidades descritas no item 4.6.3 (Implementação do programa de dinamicidade). O MRTG necessita receber somente os valores totais de *bytes* trafegados na interface de rede e para isso usamos o *script* apresentado no quadro 8.

Quadro 8 – *Script* LER

```
1   #!/bin/bash
2   IN=`iptables -t mangle -L POSTROUTING -nvx | awk '/--/{ print $2}'`
3   OUT=`iptables -t mangle -L PREROUTING -nvx | awk '/--/{ print $2}'`
4   echo "$OUT"
5   echo "$IN"
```

Fonte: Próprio autor

O *script* LER inicia declarando qual será seu interpretador, conforme linha 1, na linha 2 e 3 são atribuídos as variáveis IN *download* e OUT *upload*, a saída do comando *awk* que por sua vez captura a saída do *iptables*, ou seja, por característica da linguagem *bash*, o comando *iptables* é executado, devido ao (`) acento grave, sua saída é repassada para o comando *awk* devido a (|) barra vertical ou *pipe*.

A função básica do *awk* é procurar por linhas ou outras unidades de texto em arquivos que possuem certos padrões especificados no programa, para cada padrão *pattern* deve haver uma ação associada, isto é, quando uma linha corresponde a um dos padrões, *awk* realiza a ação especificada naquela linha; depois, continua processando as linhas de entrada desta maneira, até encontrar o fim do arquivo de entrada. O conjunto de comandos padrão-ação pode aparecer literalmente como um programa ou em um arquivo específico com a opção *-f*, arquivos de entrada são lidos em ordem (MANSSOUR, 2008).

O *awk*, inicialmente, procura a linha que contenha os caracteres entre as duas barras (/--/), no caso “--”. Após encontrar a linha com os caracteres definidos, o *awk* separa cada conjunto de caracteres da linha em campos diferentes. Para retornar o valor de *bytes* esperado, é necessário retornar somente o campo *$2,* isto é programado com o comando *print $2*, que é o valor total de *bytes* trafegados na interface de rede, que o MRTG necessita (MANSSOUR, 2008).

Enfim, as linhas 4 e 5 que com o comando *echo*, imprimem os valores das variáveis referenciadas, no padrão exigido pelo MRTG, o valor de upload seguido do valor de download (MANSSOUR, 2008).

Cabe ainda ressaltar que, para liberar a contabilização do tráfego de rede, *download* e *upload*, é necessário o seguinte comando: *iptables -t mangle -A PREROUTING -i eth0 -j ACCEPT*, que permite a passagem dos pacotes vindos da interface de rede externa pela tabela *mangle*, responsável por alteração no pacote de dados, antes de serem roteados pelas outras tabelas do *iptables* (NETFILTER, 2018).

5.2 Análise das regras de bloqueio

Nesta seção, apresenta-se uma análise dos resultados que esta nova implementação trouxe ao uso do *link* de dados da escola. Para fins comparativos, apresenta-se uma comparação gráfica da implementação original do servidor sem

Proxy e *firewall* versus o servidor executando o *Proxy Squid* estático e dinamicamente.

Desta forma, para melhor visualizar o impacto das implementações, executadas no servidor, sobre o tráfego do *link* de dados, é executado o MRTG com configurações específicas. Optou-se por monitorar a rede durante uma semana (segunda à sexta-feira). Diariamente a rede foi monitorada com uma configuração diferente, conforme quadro a seguir.

Quadro 9 – Períodos de monitoria do servidor

Dia – Representação no gráfico	**Estado de funcionamento do servidor**
Segunda-feira *Mon*	Servidor **sem regras de bloqueio**, *firewall e Proxy*. Configuração idêntica a do inicio deste TCC.
Terça-feira *Tue*	Servidor com **regras de bloqueio estáticas**, *firewall e Proxy* implementados. Configuração implementada no quarto capítulo deste TCC
Quarta-feira *Wed*	Servidor com **regras de bloqueio estáticas**, *firewall e Proxy* implementados. Configuração implementada no quarto capítulo deste TCC.
Quinta-feira *Thu*	Servidor com **regras de bloqueio dinâmicas**, *firewall e Proxy* implementados. Configuração implementada no quarto capítulo deste TCC.
Sexta-feira *Fri*	Servidor com **regras de bloqueio dinâmicas**, *firewall e Proxy* implementados. Configuração implementada no quarto capítulo deste TCC.

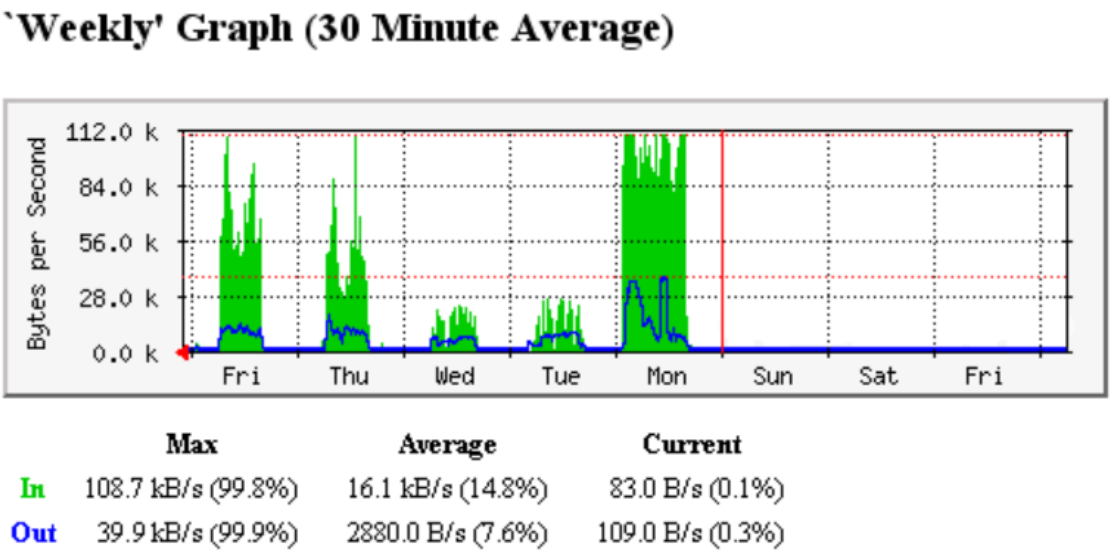
Fonte: Próprio autor

Portanto, o monitoramento dos diferentes estados é executado no mesmo servidor, o qual foi configurado sem regras de bloqueio no domingo e permaneceu neste estado até o fim do expediente da segunda-feira. Posteriormente, o servidor é reconfigurado para operar com as regras de bloqueio estáticas, *firewall e Proxy*,

permanecendo nesta configuração até o fim do expediente de quarta-feira. Neste dia, pela última vez foi reconfigurado para fazer uso das regras dinâmicas, permanecendo assim até o final do expediente de sexta-feira, onde terminou o período do acompanhamento.

5.3 Análise dos resultados

Figura 4 – Gráfico para análise gerado pelo MRTG



Fonte: Próprio autor

No final do período de monitoramento, o MRTG gerou o gráfico apresentado na figura 4. As colunas em verde apresentam os valores de *download* e a linha azul os valores de *upload*. A linha vermelha logo abaixo do meio da área do gráfico representa o valor máximo da taxa de upload. A linha vermelha na extremidade superior da área do gráfico representa o valor máximo da taxa de *download*.

Cada coluna preenchida representa a média do tráfego no período de meia hora. As subdivisões das linhas pretas separam a escala de *bytes* por segundo. As subdivisões das colunas separam os períodos por dia. O gráfico apresenta o tráfego no período total de uma semana e um dia completo, com início da direita e sendo preenchido para a esquerda.

Os valores abaixo do gráfico apresentam o tráfego medido e entre parênteses em percentual. *In* representa entrada e *Out* saída, a coluna *Max* apresenta o maior valor capturado no período, a coluna *Average* apresenta a média dos valores, e a coluna *Current* apresenta o último valor capturado.

Na figura 4 pode-se observar que os valores apresentados na segunda-feira, em certas ocasiões atingem o limite de uso da capacidade do *link*, ou seja, o *link* é inteiramente utilizado tanto para *download* como para *upload*, motivo pelo qual, a navegação se torna lenta. Pois, excesso de informações estão trafegando pela rede, tendo em vista, que não existe nenhum bloqueio implementado aos usuários.

Desta análise, é possível observar que os valores apresentados nos intervalos de terça-feira e quarta-feira, estão abaixo se comparados aos outros períodos, pois as regras de bloqueio fazem com que praticamente não exista navegação, além daquela utilizada para a execução das tarefas de rotina da escola, não atingindo a capacidade máxima do *link* e gerando uma certa insatisfação por parte dos professores e funcionários que fazem uso da Internet para fins particulares. É interessante notar que por um certo período as colunas verdes representando o *download*, permanecem próximas ao valor zero, indicando que não há usuários navegando no momento.

Finalizando a análise, é possível averiguar que os valores apresentados na quinta-feira e sexta-feira continuam no meio da área do gráfico, em poucos instantes de tempo atingindo o máximo na escala, porém, logo em seguida voltam a uma taxa de utilização média, mostrando assim a ação das regras de bloqueio no servidor. Observa-se também que o *link* de dados é melhor aproveitado, as taxas de *upload* não chegam ao seu máximo, mesmo não existindo um controle sob a mesma e os usuários não estão mais limitados quanto ao uso da Internet.

5.4 Testes de Desempenho

Para certificar-se que o servidor em análise apresenta desempenho aceitável, são executados dois testes, o primeiro que mede o tempo de carregamento de sites e o segundo que mede o tempo de resposta de *ping*, taxa de *download* e *upload*. Os testes são aplicados em dois tipos de situações, a primeira com a utilização do servidor *Proxy* dinâmico, e a segunda sem o uso do mesmo, neste caso, o computador de teste é ligado diretamente no modem ADSL (*Asymmetric DigitalSubscriber Line*), com acesso direto à Internet. Cada teste é executado duas vezes, durante horário de expediente, sendo que, após cada um dos testes, os *caches* de página do computador local são limpos e os computadores não são alterados, possuindo a mesma configuração, visando um resultado mais seguro possível.

Na realização destes testes são utilizados dois sites, os quais são apresentados abaixo.

5.4.1 *PCPitstop*

Site baseado em *flash*, que verifica os tempos de *ping* e a taxa de *download* e *upload* do servidor que o usuário escolhe. No teste executado foi selecionado o servidor de *Los Angeles* na *Califórnia*, Estados Unidos. O endereço eletrônico do site é: http://www.pcpitstop.com/internet/bw.asp e os resultados são apresentados no quadro 10, onde os tempos de *ping* são apresentados em ms (Milissegundos) e as taxas de *download* e *upload* em *KBps* (*Kilobyte* por Segundo).

5.4.2 *WEBPAGETEST*

Site desenvolvido em *javascript*, rodando no computador local, cronometra o tempo de carregamento do site especificado pelo usuário, o cronômetro é iniciado no momento que o navegador começa a carregar a página, e para quando esta é carregada. O tempo também é apresentado em ms (milissegundos). O endereço eletrônico do site é: http://www.webpagetest.org/ e os resultados obtidos são apresentados no quadro 10.

5.4.3 Análise dos Resultados

Os resultados para os tempos de respostas de *ping* são mais lentos com o uso do servidor. Pois, com o computador de teste conectado diretamente no modem diminui-se o número de saltos necessários para atingir o computador destino. A diferença dos valores de *ping* não chega a influenciar no desempenho da rede, pois a maior variação do tempo de resposta foi de apenas 8ms, imperceptível ao usuário.

Também apresentado no Quadro 10, os testes de *download* e *upload* apresentam resultados variados. Esta variação deve-se à existência de tráfego externo variado no *link* de dados, porém, os valores estão dentro da taxa de transmissão, não refletindo nenhuma influência negativa do servidor *Proxy*.

Por fim, referente ao tempo de carregamento dos sites, o referido teste mostra a eficiência do *Proxy Squid*, sendo que no segundo acesso ao mesmo site, o

tempo de carregamento é muito menor, devido a página estar armazenada no *Proxy cache*. Consequentemente, diminuindo a utilização do *link* de dados.

Com a execução dos testes foi possível verificar que o servidor tem influência positiva na navegação devido ao uso do *Proxy Squid*, contribuindo assim para um melhor desempenho da Internet na ETEC Polivalente de Americana.

Quadro 10 – Testes de desempenho

Testes	Resultados			
	Com *Proxy*		Ligado diretamente no modem	
	1ª execução	2ª execução	1ª execução	2ª execução
Tempo de ping	258 ms	255 ms	252 ms	250 ms
Velocidade de Download	101.4 KBps	73.7 KBps	98.4 KBps	79.1 KBps
Velocidade de Upload	37.9 KBps	23.5 KBps	37.1 KBps	30.3 KBps
Tempo de carregamento do site: www.uol.com.br	14,157 ms	4,156 ms	14,272 ms	14,126 ms
Tempo de carregamento do site: www.globo.com	5,071 ms	4,855 ms	4,936 ms	5,120 ms

Fonte: Próprio autor

6 CONSIDERAÇÕES FINAIS

As dificuldades de acesso à Internet que ocorriam na ETEC Polivalente de Americana motivaram o desenvolvimento deste trabalho. Após o estudo do problema, chegou-se à conclusão que se deveria acrescentar um servidor de Internet à infraestrutura da rede, para isto foi necessário revisar alguns conceitos da área de segurança, com a finalidade de desenvolver um *script* de *firewall* que atenda as expectativas e necessidades, além da importância para segurança dos dados da rede e do servidor. Portanto, com os conceitos revistos foi possível efetuar a implementação do servidor de uma forma mais segura e completa, minimizando os riscos de invasões e infecções pelas pragas eletrônicas.

A implantação e configuração do *Proxy Squid* possibilitou a melhora do tempo de resposta na navegação de Internet, aplicando no servidor regras dinâmicas de bloqueio a páginas, com acessos priorizados às finalidades escolares, ocasionando a diminuição e melhor aproveitamento do *link* de dados.

De forma a ultrapassar a limitação citada anteriormente, regras de bloqueio dinâmicas, baseadas no uso do *link* de dados, foram criadas com o intuito de harmonizar um melhor uso do mesmo para atividades de rotina da unidade escolar, e ao mesmo tempo, sem comprometer as necessidades particulares dos funcionários e os serviços que são desenvolvidos na ETEC Polivalente de Americana

Para visualizar os resultados da aplicação do servidor e permitir uma abordagem comparativa dos diferentes modos de operação, foi utilizado o *software* livre MRTG, que gera gráficos do uso do *link*. Através dos gráficos verificou-se a eficiência do servidor *Proxy* dinâmico e os benefícios que o mesmo trouxe para a rede desta Unidade de Ensino, minimizando os problemas de acesso à Internet e melhor aproveitamento da capacidade do link.

As dificuldades principais encontradas para o desenvolvimento deste trabalho foram no *iptables*, na elaboração das regras de liberação do acesso, que muitas vezes se tornam complexas e não funcionam, sendo necessárias várias tentativas para corrigi-las. A ambientação com a linguagem de programação Perl e o entendimento do funcionamento da linguagem procedural awk, a qual permitiu a implementação do *script* de leitura do total de bytes trafegados no link.

REFERÊNCIAS

ADACHI, Tomi. **Gestão de segurança em Internet Banking**. São Paulo: FGV, 2004. 121p. Mestrado. Fundação Getúlio Vargas - Administração. Orientador: Eduardo Henrique Diniz.

ABNT. **NBR ISO/IEC 27002**. Tecnologia da informação - técnicas de segurança - código de pratica para gestão da informação. Rio de Janeiro: ABNT, 2005.

CERT, Centro de Estudos e Resposta e Tratamento de Incidentes de Segurança no Brasil. **Práticas de segurança para administradores de redes Internet**. Disponível em: <http://www.cert.br/docs/seg-adm-redes/seg-adm-redes.html#sec4. Acesso em: 26 outubro 2016.

CHESWICK, W. R.; BELOVIN, S. M.; RUBIN, A. D. **Firewalls e segurança na Internet:** Repelindo o hacker ardiloso. 2. ed. Porto Alegre: Bookman, 2005.

CONFIGURATION GUIDE SQUID. **Welcome to SQUID 3.5**, Configuration Guide. Apresenta manual técnico para configuração do Squid, bem como versões para download. Disponível em: <http://www.squid-cache.org/Versions/v3/3.5/cfgman/>. Acessado em: 10 abril 2018.

FERREIRA,F.N.F; ARAÚJO,M.T. **Políticas de segurança da informação**: guia prático para elaboração e implementação. 2 ed. Rio de Janeiro: Ciência Moderna, 2008.

FERREIRA, R. E. **Linux guia do administrador do sistema**. São Paulo: Novatec, 2003. 510 p.

JUCÁ, Humberto L. **Técnicas avançadas de conectividade e *firewall* em *GNU/Linux***. Rio de Janeiro, Brasport, 2005.

LYRA, Maurício R. **Segurança e auditoria em segurança da informação**. Rio de Janeiro: Ciência Moderna, 2008.

MANSSOUR, I. H. Resumo do manual da gnu, gawk.ps. Apresenta manual técnico e exemplos do uso linguagem procedural GAWK . Disponível em: < https://www.inf.pucrs.br/manssour/AWK/index.html >. Acessado em: 23 maio 2018.

MARCELO, A. **Squid, configurando o proxy para Linux**. 2. ed. Rio de Janeiro: Brasport, 2003.

MORENO, E. D., Pereira, F. D., & Chiaramonte, R. B. [s.d.]. **Criptografia em *software* e *hardware***. Novatec Editora, 2008.

MORIMOTO, C. E. **Servidores Linux:** guia prático. Local: GDH Press e Sul - Editores, 2008.

NAKAMURA, Emilio Tissato; GEUS, Paulo Lício de. **Segurança de redes em ambientes cooperativos**. São Paulo: Novatec Editora, 2007.

NETFILTER. **Linux 2.4 Packet filtering HOWTO**. Disponível em: <https://netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.html>. Acessado em: 10 abril 2018.

PEIXOTO, Mauro C.P. **Engenharia social e segurança da informação**. Rio de Janeiro: Brasport, 2006.

PINHEIRO, José Maurício Santos. **Políticas de *backup* corporativo**. Disponível em:<http://www.projetoderedes.com.br/artigos/artigo_politicas_de_backup_corporativo.php>. Acesso em: 07 abril 2018.

TANENBAUM, A. S. **Redes de computadores**. 4. ed. Rio de Janeiro: Campus, 2003.

TAVEIRA, Gilda A.; FERNANDES, Leila Maria P.; BOTINI, Joana, **Elementos do microcomputador**. Rio de Janeiro: Ed. SENAC nacional, 2006.

THE NETFILTER.ORG. **What is iptables?**. Disponível em: <https://netfilter.org/projects/iptables/>. Acesso em: 07 abril 2018.

TORRES, Gabriel. **Redes de computadores:** curso completo. Rio de Janeiro: Axcel Books do Brasil Editora, 2001.

UCHÔA, J. Q.; SIMEONE, L. E.; SICA, F. C. **Administração de redes Linux**. Lavras: UFLA/FAEPE, 2003. (Curso de Pós Graduação “Lato Sensu” (Especialização) a Distância em Administração em Redes Linux).

WADLOW, Thomas A. **Segurança de redes:** projeto e gerenciamento de redes seguras. Tradução: Fábio Freitas da Silva. Rio de Janeiro: Campus, 2000.

WALL, L.; CHRISTIANSEN, T.; ORWANT, J. **Programação Perl**. 3. ed. Rio de Janeiro: Campus, 2001.

ZWICKY, E. D.; COOPER, S.; CHAPMAN, B. **Construindo firewalls para a Internet**. 2. ed. Rio de Janeiro: Campus, 2005.

APÊNDICE A – Autorização do Estudo

AUTORIZAÇÃO DO ESTUDO

Eu, Mary Damiani, na qualidade de Diretora e Representante Legal da ETEC Polivalente de Americana, Unidade de Ensino do Centro Estadual de Educação Tecnológica Paula Souza, **autorizo para fins acadêmicos** o estudo de caso, proposto e desenvolvido nas dependências deste órgão, para o Trabalho de Conclusão de Curso do Sr. Dian Gidaro, funcionário da escola e aluno matriculado junto à Faculdade de Tecnologia de Americana.

Por ser verdade e para que surta os efeitos necessários, firmo a presente.

Mary Damiani

Diretora de Escola Técnica - ETEC