UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Implementando Segurança e Controle em Redes de Computadores

por

LEANDRO MÁRCIO BERTHOLDO

Dissertação submetida como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Profa. Dra. Liane M. R. Tarouco
Orientador

Porto Alegre, janeiro 1997.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Bertholdo, Leandro Márcio

Implementando Segurança e Controle em Rede de Computadores / Leandro Márcio Bertholdo.—Porto Alegre: CPGCC da UFRGS, 1996.
136 p.: il.

Dissertação (mestrado)—Universidade Federal do Rio Grande do Sul, Curso de Pós-graduação em Ciência da Computação, Porto Alegre, BR-RS, 1996. Orientador: Tarouco, Liane M. R..

1.Redes de Computadores. 2.Segurança 3.Gerência de Segurança 4.Gerência de Redes de Computadores 5.Firewalls 6.SNMPv2 7.Unix. I.Tarouco, Liane M.R. II.Título

OFERECIMENTO

À vida,
que infelizmente passa
enquanto nós não estamos olhando!

AGRADECIMENTOS

Aos colegas do Curso de Pós-graduação que proporcionaram além da amizade, produtivas discussões sobre os mais variados temas durante os encontros nos bares da universidade.

Aos funcionários do CGPCC, principalmente à Eliane, que sempre foi compreensiva em esticar os horários de funcionamento dos laboratórios. Algo de extrema importância quando às vésperas das entregas de trabalhos.

Ao GOR / POP-RS, um grupo que reuniu, e ainda reúne, pessoas bastante especiais. Em especial a Cleber, Guga, Fernando, Liane e Marcelo, pela paciência que tiveram em todas as vezes que tentaram executar alguma operação e receberam um “access denied”, ou que receberam chuvas de mail do tipo “quem foi que alterou...”.

E, sinceramente, à professora Liane Margarida Rockenbach Tarouco, uma pessoa através da qual consegui aprender muito durante todos estes anos. Tanto em conhecimento técnico como pessoal. À senhora, meu eterno respeito.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ANSI - *American National Standards Institute*
CERT - *Computer Emergence Response Team*
CIAC - *Computer Incident Advisory Capability*
CRC - *Cyclic Redundancy Check*
ISO - *Data Encription Stantard*
IP - *Internet Protocol*
ISO - *International Standards Organization*
MD5 - *Message Digest Algorithm*
NSA - *National Security Agency*
PGP - *Pretty Good Privacy Algorithm*
RFC - *Request For Comments*
TCP - *Transmission Control Protocol*

RESUMO

O crescimento e proliferação da Internet nos últimos anos tem trazido à tona vários problemas relativos à segurança e operacionabilidade das máquinas de universidades e empresas. Inúmeras invasões são realizadas anualmente. Entretanto, a grande maioria delas não possui registro algum, sendo muitas vezes de total desconhecimento do administrador local.

Para prover soluções para estes problemas foi realizado um estudo, aqui apresentado, que tem como principal objetivo propor uma filosofia de gerência de segurança. São utilizados para isso conceitos de gerenciamento de redes como SNMPv2, aliado à implementação de um conjunto de ferramentas que garantam a integridade dos vários sistemas envolvidos. O resultado foi um sistema denominado CUCO[1], que alerta sobre tentativas de ataque e situações de risco.

CUCO foi projetado para permitir a um administrador, protegido ou não por uma *firewall,* dispor de um controle maior e melhor sobre acessos e tentativas de acessos indevidos à sua rede. O sistema usa uma estratégia de monitoração de eventos em diferentes níveis e aplicações, tentando com isto detectar e alertar a ocorrência de ataques tradicionais. Também está incorporado um bloco de funções que visam identificar um agressor situado em algum lugar da Internet, e obter maiores informações sobre ele e o domínio onde esta localizado.

PALAVRAS-CHAVES: Redes de Computadores, Gerência de Redes, Segurança de Dados, Segurança de Sistemas, Gerência de Segurança, Ferramentas.

[1]Analogia ao mecanismo de relógios de parede, que notifica um evento: as mudanças de hora. Pode também ser um acrônimo de: “CUidado, ou Chamo o Operador”

ABSTRACT

Title: "Implementing Security and Control in Computer Networks"

The Internet increase and proliferation in the last years has brought a lot of problems related to the security and handling of hosts in universities and corporations. Many break-ins are done each year, without any record or knowledge by the site's administrator.

To give solutions to this problems was made up a study, here presented, has as the main goal the proposal of a security management philosophy. Are used network management concepts, joined with a toolkit to ensure the integrity of many systems envolved. The result was a system named CUCO[2], that alerts about attacks and risks situations.

CUCO was designed to allow an administrator, protected or not by firewall, to have a bigger and better access control in his network. The system uses an event monitor strategy in different levels and applications, trying to detect and alert the occurrence of common attacks. Moreover, it is also incorporated by a set of functions that attempt to identify aggressor's location in any place in the Internet, and get information about him and the domain where he is located.

Keywords: Computer Networks, Network Management, Data Security, System Security, Security Management, Tools.

[2] [1]Analogy to clock mechanism, that notify an event: hour changes. Furthermore, can be an acronym of: "CaUtion, or I'll Call Operator "

1 INTRODUÇÃO

Após alguns anos estando conectada à Internet, a universidade começou a perceber que, juntamente com os benefícios de estar diretamente ligada ao meio científico mundial, também estava se expondo a um ambiente que se tornava a cada dia mais hostil. Esta abertura têm colocado a universidade em uma posição delicada, onde freqüentes problemas relativos a segurança de informações e operacionabilidade de máquinas tem mostrado a fragilidade ante as investidas realizadas contra a instituição.

A expansão da Internet brasileira tem tornado inúmeros estudantes e curiosos portadores de informações esparsas sobre vulnerabilidades em máquinas e protocolos. Estas informações são obtidas junto a rede mundial[3], diponibilizadas por anarquistas que desejam ocultar suspeitas sobre si próprios, distribuindo parte deste conhecimentos a aprendizes devotos, que povoam todo um submundo bastante bem estruturado, e com intenções quase nunca lícitas.

A comunidade mundial vem se defendendo como pode. Inúmeras ferramentas são construídas anualmente, como SATAN [CER95a] [VEN95] [BER96], ISS [CER93] [ISS96], PINGWARE [BEL96] e AutoHack [MUF95] cujo intuito de realizar uma auditoria sobre a segurança de máquinas ou subredes, outras visam detectar uma invasão ocorrida no passado e ainda outras tentam garantir a segurança através de filtros, como por exemplo o TCPWrapper [VEN92], mais uma das ferramentas consagradas de Wietse Venema. Uma outra forma de manter a segurança de um site, que também vem sendo bastante usada, é o próprio isolamento das máquinas da Internet, através de técnicas de *Firewalling* [CHE94] [CHA95].

[3] Existem hoje inúmeras fontes de informações onde são trocadas diariamente informações sobre como explorar vulnerabilidades, e divulgados endereços de sites que são considerados presas fáceis. Vide grupos de News como alt.hackers e o canal #hackers do IRC, além de várias listas que tratam de assuntos relacionados a segurança. Um exemplo destas mensagens pode ser encontrado no anexo A-1.

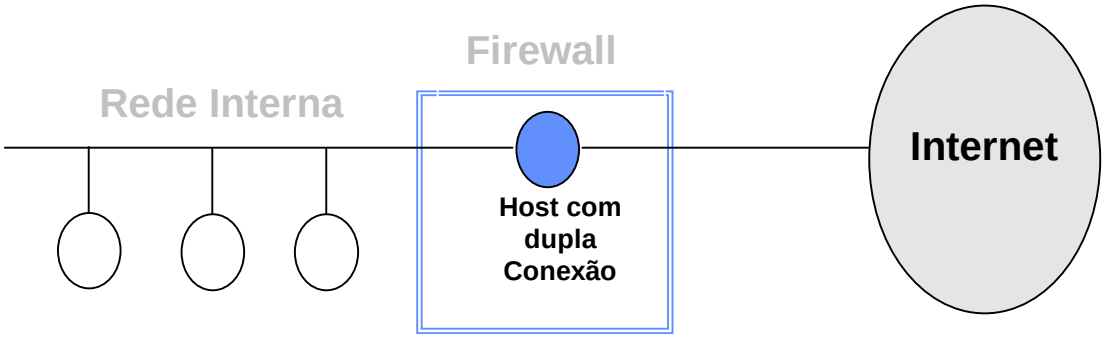

Figura 1.1: Uma *firewall* isolando a rede interna da Internet

Técnicas utilizando o conceito de *firewalls* são hoje consideradas as mais confiáveis para proteger uma corporação de um ataque externo. Contudo, não serão abordadas em profundidade neste trabalho, principalmente por não serem consideradas como a melhor solução para o problema de ambientes que devam disponibilizar acesso de forma não restritiva, como é o caso das universidades. Isto será demonstrado nos capítulos seguintes.

1.2 Motivações

Após exaustivos estudos e acompanhamentos sobre as vulnerabilidades existentes hoje em dia e as formas de explorá-las, pressente-se dentre tudo o que foi divulgado no meio científico mundial, a falta de um dispositivo que consiga captar o momento exato de uma invasão. Esta informação é extremamente útil para que o administrador possa tomar as medidas necessárias de forma eficiente em um momento crucial. Em situações práticas, tem-se por várias vezes notado que um reconhecimento tardio de uma invasão (após uma ou duas horas), pode tornar a situação fora de controle e comprometer inúmeros outros *sites* de forma irremediável e não rastreável.

Um bom exemplo disso, é um dos casos ocorridos durante o estudo realizado, onde uma invasão foi detectada praticamente ao acaso em uma estação de trabalho Sun, com um antigo sistema operacional SunOs 4.1.3. Estação esta, estrategicamente situada no *backbone* principal de uma das instituições alvo do

estudo. A partir deste momento teve-se um primeiro contato com uma das situações que revelavam a fragilidade em que se encontrava grande parte dos ambientes conectados à Internet. Até então haviam sido tomadas tão somente algumas medidas básicas de segurança nas máquinas envolvidas, como indisponibilização de serviços como *finger, tftp, rex, rsh, rexec, rlogin,* e configuração de serviços como ftp anônimo seguindo algumas recomendações existentes, vide [CER93] [LIU94].

Estas medidas básicas foram consideradas como insuficientes, devido a um sistema operacional ultrapassado, e com vários *patches* de segurança em atraso. Uma verificação completa no sistema foi realizada naquela máquina, utilizando-se para isto diversos documentos e informações sobre segurança divulgados por diversos órgãos, como os *Computer Emergence Response Team* (CERT) regionais e o *Computer Incident Advisory Capabilities* (CIAC), além de informações obtidas em grupos de discussão, listas de correio eletrônico e publicações de outras instituições [SMI93] [MUF92] [BLG93] [KLE90] [CUR90].

A busca pela segurança principalmente das máquinas da universidade a partir deste momento tornou-se praticamente uma obsessão. Todos os setores da universidade foram inspecionados utilizando-se o conhecimento adquirido e o auxílio de ferramentas de auditoria, como SATAN [CER95a], ISS [CER93], COPS [FAR91]. Esta verificação gerou um relatório expondo diversas vulnerabilidades as quais a universidade estava exposta [BER95], e obteu-se assim uma imagem da real situação em que se encontrava a segurança das máquinas.

A necessidade de manter-se a disponibilidade dos recursos computacionais além da integridade, confiabilidade e sigilo das informações é a meta final da segurança em qualquer instituição. Levando-se em conta que a universidade possui hoje mais de 100 estações de trabalho utilizando diferentes versões de sistema operacional, SunOs 4.1.1, SunOs 4.1.3, Solaris 2.3, Solaris 2.4, algumas versões de HP-UX, além de vários PCs utilizando Linux, espalhados em vários pontos da rede, torna-se impossível uma implementação de segurança baseada em hosts da forma como nós a conhecemos. Sendo assim, cada host deveria ter como requisito básico o de possuir:

- A última versão do sistema operacional.
- A última versão de programas como httpd, ftpd, innd e outros servidores de aplicações.
- Aplicações de todos os *patches* divulgados para cada sistema operacional ou ferramentas como sendmail, em toda uma centena de máquinas.
- Verificação constante dos logs gerados em cada uma destas máquinas (estimado em aproximadamente 470 Kb/host/dia[4]).
- Monitoração constante de tentativas das acesso indevido de todos os possíveis usuários. Levando-se em conta que alunos podem estar se ligando a partir de várias universidades em todo o mundo.
- Verificação constante das configurações básicas do sistema operacional e de rede, principalmente a respeito de integridade e serviços disponibilizados,

Como pode-se notar, a adoção desta estratégia é inviável na prática, levando-se em consideração as necessidades de segurança e os custos de tempo e pessoal envolvidos. Entretanto, algo deveria ser concretizado para pelo menos ter-se um maior controle sobre o estado das máquinas, de forma a implementar um conceito de gerenciamento de segurança sobre diversos hosts da rede, possuindo sempre a opção de centralizá-lo ou não. Este controle deveria prover algumas informações básicas sobre o estado de segurança das estações, tais como: se um dos nós da rede esta sob algum tipo de ataque; se alguma ferramenta do tipo *sniffer* foi instalada ou não em algum ponto da rede; se algum sistema está comprometido, isto é, teve sua configuração alterada para permitir acesso indevido, sendo ele privilegiado ou não. A partir da constatação destas necessidades, surgiu o planejamento do sistema CUCO.

[4] O cálculo foi realizado levando-se em consideração o log gerado por duas estações de trabalho durante 40 dias. Ele não inclui os logs gerados pelos servidores News e WWW que as estações possuem

1.3 Objetivos

O sistema CUCO tem por principal objetivo alertar o administrador local sobre a identificação de alterações que possam de alguma forma comprometer a segurança do sistema. Ele tem como meta possibilitar ao administrador tomar conhecimento, através do sistema, das possíveis tentativas de violação de segurança, além de agrupar informações que apontem qual foi a técnica usada em uma determinada invasão e possibilitar algumas atitudes emergenciais de forma a impedir que o intruso tome posse do sistema. Isto deve fornecer o tempo necessário para que uma determinada falha na segurança seja sanada.

Inicialmente, o sistema deve ser capaz de identificar e alertar a entidade usuário responsável pela segurança da rede, dos casos de comprometimento ou suspeita de invasão em uma das máquinas. Entre os problemas a serem alertados cita-se:

- Alguma das máquinas teve seus sistema operacional ou aplicativos comprometidos por ferramentas com *rootkit*[5] [ROO96].
- Algum host está coletando senhas na rede com algum programa do tipo *sniffer*.
- Algum host registrou acesso a partir de algum local desconhecido.
- Houve algum registro considerado “atípico” nos logs do sistema.
- Algum arquivo de configuração do sistema foi alterado como por exemplo o próprio arquivo de senhas.

Além dos itens expostos acima, o sistema necessita ser ágil e de fácil utilização, sem que ele necessite despender maiores cuidados do administrador para o conjunto das máquinas monitoradas. A aplicação também deve possibilitar ao seu usuário obter informações de forma rápida e consistente no caso da ocorrência de uma

[5] O *RootKit* é um conjunto de ferramentas utilizados por hackers constituida de diversos aplicativos como z2, es, fix, sl, ic, ps, ns, ls e outros de forma a facilitar a entrada no sistema e de esconder os traços das invasões.

invasão. O objetivo desta informação é tentar identificar a origem e o responsável por um possível ataque.

1.4 Organização deste trabalho

Este trabalho apresenta o estudo realizado e a experiência adquirida durante a concepção e criação do sistema CUCO. O estudo considera procedimentos mais comuns de ataques verificados tanto na bibliografia existente, como em [LIT96] e [CHE92], quanto em testes realizados na própria rede da universidade [BER95]. Na seqüência desta introdução serão apresentados os resultados destas pesquisas, espelhadas no protótipo desenvolvido. O texto se encontra assim disposto:

O capítulo 2 descreve o cenário atual de segurança nas máquinas da universidade e na Internet em geral, e a dificuldade em manter um nível médio de segurança. Também são demonstrados alguns dos diferentes tipos de ataques aos quais estão sujeitas quaisquer máquinas que tenham um ponto de acesso a Internet. Neste capítulo também serão mostradas tecnologias e ferramentas que podem ser usadas para reforçar a proteção dos sistemas e dados, dificultando assim que isto aconteça.

No capítulo 3 é proposta a solução “CUCO”, que vem ao encontro das necessidades citadas no capítulo anterior, e que não é suprida pelos conjuntos de ferramentas hoje disponíveis. Aqui são descritos em detalhes as várias características do sistema, assim como a base na qual ele foi construído.

O Capítulo 4 é uma descrição dos resultados obtidos, ambiente no qual o sistema age atualmente, além de detalhes da concepção do protótipo, como dificuldades de portabilidade e execução do sistema para o ambiente para o qual ele foi proposto inicialmente.

E por fim, o capítulo 5 traz as conclusões obtidas com a pesquisa realizada e o protótipo construído, avaliando a sua importância e propondo expansões futuras.

2 O PANORAMA GERAL DE SEGURANÇA NA INTERNET

Mundialmente o problema de abuso de computadores é tido como danoso. No próprio berço da Internet, os Estados Unidos, existem estimativas de perdas superiores a 2 bilhões de dólares anuais causados por hackers high-tech [LYO94]. Estes mesmos hackers[6] ou crackers[7] costumam tirar proveito da fragilidade de universidades. Atualmente elas são usadas como uma forma de despistamento, ou seja, elas são invadidas com o único objetivo de esconder a real origem do ataque. O que não significa que estejam por isto livre de quaisquer danos, já que apagar os vestígios é a melhor forma de despistamento; e a forma mais rápida é: "rm -rf /"[8], que os usuários de unix sabem o que significa, caso não haja um backup bastante recente.

---

[6] Hacker: 1. Pessoa que se diverte aprendendo detalhes de sistemas de computação e como obter o máximo de suas capacidades - o oposto a maioria dos usuários de computadores, que preferem aprender somente o mínimo necessário. 2. Alguém que programa entusiasticamente, ou que se diverte programando ao invés de simplesmente teorizar sobre programação[GAR94].

[7] Embora a mídia freqüentemente refira-se a pessoas que quebrem a segurança de computadores como *hackers*, isto é uma perversão da palavra original. Alguns profissionais de segurança, muitos deles *hackers*, costumas chamar as pessoas que invadem computadores como *crackers*, uma palavra mais adequada para identificar este tipo de pessoa. Outros nomes também aplicados a estas pessoas são "vândalos" ou mesmo "criminosos"[GAR94].

[8] Caso o usuário tenha o nível de privilégio adequado, este comando força (-f) a remoção (rm) de todo o conteúdo do diretório raiz "/", de forma recursiva (-r), tornando desta forma a máquina não operacional.

2.1 Algumas Formas de Ataque

Desconsiderando-se falhas de implementações (bugs), existentes nas várias camadas do sistema operacional, sistema de rede e aplicativos, a seguir serão apresentadas algumas formas de ataque que são freqüentemente realizadas. O alvo principal destes ataques são problemas intrínsecos à definição e funcionamento dos protocolos TCP/IP. Estas vulnerabilidades estão aqui demonstradas para que o leitor tenha uma idéia das dificuldades que existem para que uma rede seja mantida a salvo. Mesmo estando ela atrás de filtros e esquemas mais sofisticados utilizando soluções como f*irewalls.*

2.1.1 IP Spoofing

Ataques do tipo IP spoofing podem ser considerados tecnicamente como uma forma de ataque composta de vários componentes (ip-spoofing + sequence guessing + SYN flooding) [LIT96]. Na realidade IP-spoofing não é um ataque, mas sim um passo de um ataque, utilizado para explorar um relacionamento de confiança entre duas máquinas, e é assim que ele será tratado neste capítulo.

No mundo UNIX, teias de confiança podem ser facilmente criadas entre as diversas máquinas. Digamos que o usuário X tenha uma conta na máquina A, e outra na máquina B. Para facilitar a movimentação entre as duas com o mínimo de perturbação no que tange a autenticação do usuário, pode-se criar uma relação de confiança mútua entre elas. No diretório home do usuário X em A cria-se um arquivo ".rhosts" (echo "B X" > ~/.rhosts), e no seu diretório home em B cria-se um outro arquivo ".rhosts" (echo "A X" > ~/.rhosts). De uma outra forma, o superusuário pode criar regras equivalentes utilizando para isto o arquivo "/etc/hosts.equiv", a diferença está nas regras, que são mais amplas, relativas a hosts ao invés de bases individuais. Tomado qualquer um dos dois procedimentos, o usuário X pode a partir de agora utilizar os comando r* (rlogin, rsh, ...), sem necessitar prover nenhuma informação de autenticação, que anteriormente permitiria ou negaria o acesso ao sistema. Os comandos (r*), permitem uma autenticação baseada no endereço IP do requerente do serviço.

A técnica de spoofing [BEL89] consiste simplesmente na falsificação do cabeçalho do pacote IP. Mais especificamente no campo onde se encontra o endereço origem da conexão para a qual será retornada a resposta. Isto é possível por não haver nenhuma verificação que autentique o host originador da conexão, ou que valide os dados inseridos no pacote enviado. Observe a seguir a figura com um datagrama IP (figura 2.1). O campo de cheksum utilizado para verificar a integridade do datagrama não abrange os campos de IP origem e IP destino. Isto facilita ainda mais a geração de múltiplos pacotes com vários endereços de origem falsificados, utilizando para isto somente um único host origem.

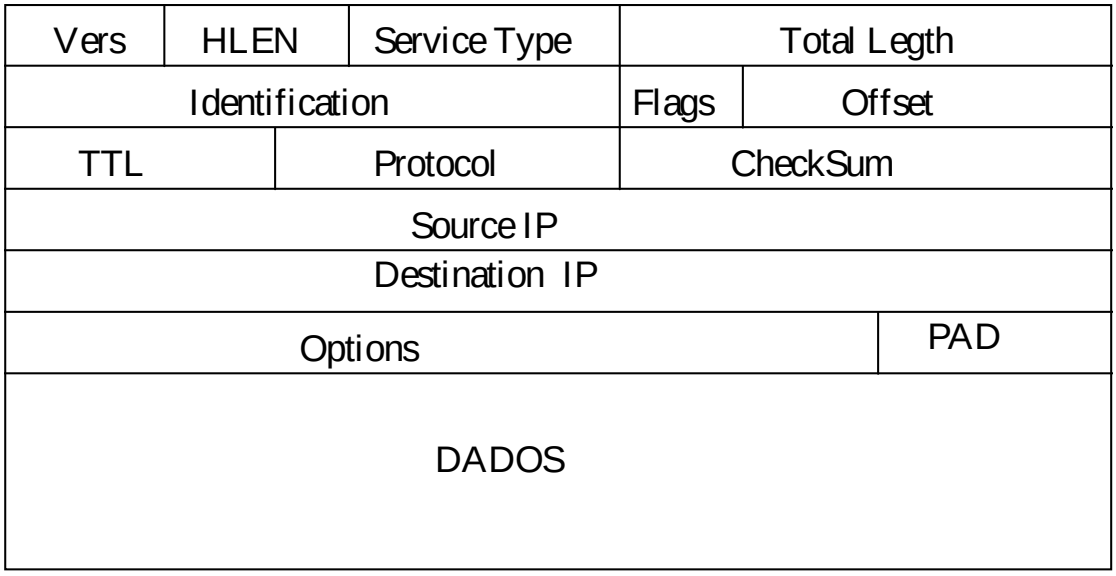

Figura 2.1: Formato de um datagrama IP

Uma solução simples para se prevenir este tipo de ataque é não confiar em autenticações baseadas em endereços. Desabilitar todos os comando r*, remover todos os arquivos ~/.rhosts e o arquivo /etc/hosts.equiv. Isto força os usuários a utilizar em outras formas de acesso que exigem uma autenticação (telnet, ssh, skey, etc.).

Um outro incremento para a solução anterior é o uso de regras de filtragem nos roteadores da instituição, bloqueando a entrada de endereços que utilizem o mesmo intervalo da rede interna e filtrando a saída de endereços que não

façam parte da rede interna, ou sejam endereços reservados, diminuindo assim os ataques de origem externa e impedindo que um ataque seja originado em máquinas internas à rede.

2.1.2 Sequence Number Attack

Ataques baseados em falsificações de números de sequenciamento [BEL89] de mensagens, tem se tornado uma séria ameaça na Internet [CER95]. Primeiramente descrito por Morris [MOR85], este tipo de ataque tem se popularizado, e se mostrado eficiente. Aqui ele será descrito utilizando um detalhamento suficiente para registrar do que trata esta ameaça.

Brevemente, esta técnica usa uma predição do número de seqüência usado para gerar o sequenciamento dos pacotes de uma conexão, de forma a construir uma conexão TCP, sem contudo receber qualquer resposta do servidor. Isto permite enganar um host considerado confiável por uma máquina na rede interna, e dessa forma ganhar acesso à rede da organização.

O estabelecimento normal de uma conexão TCP envolve um diálogo entre as partes envolvidas que ocorre em três fases. Esta troca de mensagens é melhor conhecida como *3-way handshake* [COM91a]. Em um primeiro passo o cliente seleciona e transmite um número de seqüência inicial (SYN). Em um segundo passo o servidor envia o seu próprio número de seqüência (SYN ACK). No terceiro passo o cliente acata o número de seqüência do servidor e envia a sua resposta (ACK). A partir deste momento se inicia o tráfego de dados entre cliente e servidor. O log abaixo relata toda uma conexão, do início (o cliente recebe o login da máquina) até o seu fim (exit do cliente). O reset no final da conexão significa geralmente um encerramento da conexão devido a ocorrência de um erro na troca de mensagens. Nesse caso o pacote de RST é considerado normal. Observou-se que a implementação TCP/IP que foi observada (PC) sempre terminava suas conexões com um RST após a troca de mensagens (FIN ACK) (ACK).

```
state = syn_sent
```

```
 331.0 1026->23 seq 00000000 SYN  wind 4096 opt 020405B4
 331.6 23->1026 seq 3E63AE00 ack 00000001 SYN ACK  wind 4096
state = established
 332.0 1026->23 seq 00000001 ack 00000001 ACK  wind 4096
 333.0 23->1026 seq 00000001 ack 00000001 PSH ACK  wind 4096 data 3
 333.2 1026->23 seq 00000001 ack 00000004 ACK  wind 4093
 333.4 1026->23 seq 00000001 ack 00000004 PSH ACK  wind 4096 data 3
 334.0 23->1026 seq 00000004 ack 00000004 PSH ACK  wind 4096 data 3
 334.1 1026->23 seq 00000004 ack 00000007 PSH ACK  wind 4093 data 12
 334.3 1026->23 seq 00000010 ack 00000007 ACK  wind 4093
 334.8 23->1026 seq 00000007 ack 00000010 PSH ACK  wind 4096 data 15
 335.0 1026->23 seq 00000010 ack 00000016 PSH ACK  wind 4081 data 3
 335.2 1026->23 seq 00000013 ack 00000016 ACK  wind 4081
 335.8 23->1026 seq 00000016 ack 00000013 ACK  wind 4096
 336.0 1026->23 seq 00000013 ack 00000016 PSH ACK  wind 4096 data 24
 336.9 23->1026 seq 00000016 ack 0000002B PSH ACK  wind 4096 data 39
 337.0 1026->23 seq 0000002B ack 0000003D ACK  wind 4057
 337.2 1026->23 seq 0000002B ack 0000003D PSH ACK  wind 4096 data 3
 337.6 23->1026 seq 0000003D ack 0000002B PSH ACK  wind 4096 data 7
 337.8 23->1026 seq 00000044 ack 0000002E ACK  wind 4096
 338.0 1026->23 seq 0000002E ack 00000044 PSH ACK  wind 4089 data 6
 338.1 1026->23 seq 00000034 ack 00000044 ACK  wind 4089
 338.6 23->1026 seq 00000044 ack 00000034 PSH ACK  wind 4096 data 6
 338.8 1026->23 seq 00000034 ack 0000004A ACK  wind 4090
 338.9 1026->23 seq 00000034 ack 0000004A PSH ACK  wind 4096 data 3
 339.6 23->1026 seq 0000004A ack 00000037 ACK  wind 4096
 339.8 1026->23 seq 00000037 ack 0000004A PSH ACK  wind 4096 data 3
 340.4 23->1026 seq 0000004A ack 0000003A ACK  wind 4096
 344.1 1026->23 seq 0000003A ack 0000004A FIN ACK  wind 4096
state = fin_wait_1
state = closed
 344.7 1026->23 seq 0000003B RST  wind 0
345.5 23->1026 seq 3E63AE4A ack CBBF003B ACK  wind 4096
Unknown TCP - sending reset
 346.3 1026->23 seq CBBF003B RST  wind 0
```

```
 346.7 23->1026 seq 3E63AE4A ack CBBF003B FIN ACK  wind 4096
Unknown TCP - sending reset
 347.4 1026->23 seq CBBF003B RST  wind 0
```

Figura 2.2: Monitoração de uma conexão TCP

Resumindo, como pode ser observado no log acima, o procedimento correto é o seguinte:

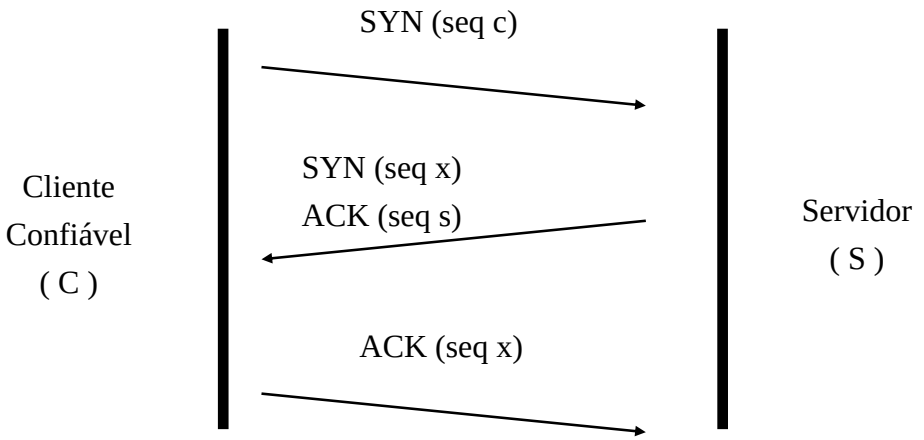

Figura 2.3: 3-way handshake

Contudo, suponhamos que exista uma forma de um intruso prever o número de seqüência (seq x). Neste caso, ele pode enviar a seguinte seqüência e impersonalizar o host Cliente (C) tido como confiável pelo servidor (S) :

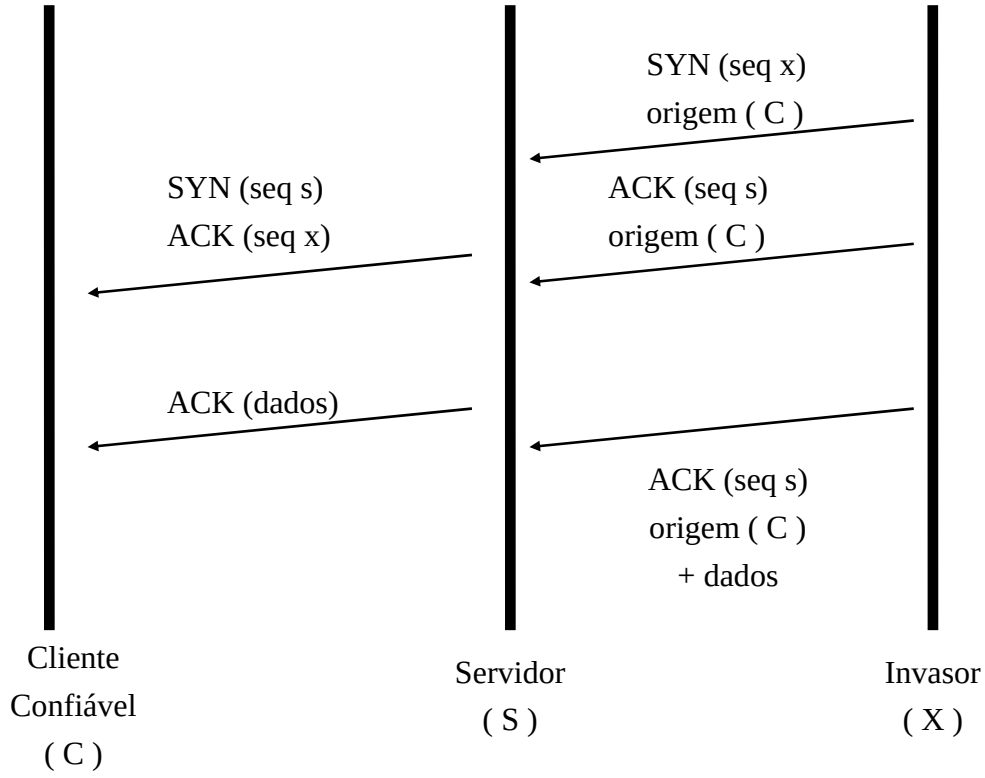

Figura 2.4: Simulando o Cliente (C)

Mesmo que as mensagens enviadas pelo servidor (S) estejam destinadas ao Cliente (C) e não ao intruso (X), o intruso é capaz de saber o seu conteúdo, e portanto pode enviar os dados. Caso o cliente (C) possua privilégios como executar comando shell, através de comandos r* BSD, ele poderá executar quaisquer comandos na máquina servidora (S) [JON95]. Demonstrado isto, o único problema ainda não esclarecido é: como prever o número de seqüência ?

O número inicial de seqüência em sistemas BSD é gerado a partir de um incremento constante realizado uma vez a cada segundo, e por metade deste valor a cada vez que uma seqüência é iniciada. Assim, se alguém iniciar uma conexão legítima e observar o número inicial de seqüência usado, ele pode calcular com um alto grau de confiança, o próximo número de seqüência a ser gerado pelo servidor (S), no próximo pedido de conexão.

Uma variante deste ataque, pode explorar o serviço netstat. Neste ataque, o invasor pode obter todas a informações necessárias sobre as conexões que estão ativas e quais os seus números de seqüência, podendo assim obter o próximo valor a ser usado, sem a necessidade de adivinhá-lo.

2.1.3 Fragmentação de pacotes IP

A fragmentação de pacotes IP pode ser utilizada para dissimular pacotes TCP através dos filtros IP hoje implementados em roteadores e hosts. A exploração da fragmentação de pacotes no nível de rede é hoje considerada uma das mais sutis formas de se tirar proveito das vulnerabilidades nos protocolos TCP/IP [REE95] [BEL93].

A existência de fragmentação torna difícil a filtragem de pacotes. Exceto pelo primeiro, os fragmentos não contém número de portas; existindo assim pouca informação na qual basear uma decisão de filtragem. Se a principal ameaça é uma tentativa de penetração a partir do lado externo de uma firewall, os fragmentos podem passam sem maiores problemas. O fragmento inicial terá a informação sobre o número de porta e pode ser processado apropriadamente. Se ele for rejeitado, o pacote estará incompleto, e os fragmentos restantes serão eventualmente descartados pelo host destino. Contudo, os fragmentos podem, por exemplo, ser usados para vazar informações entre a rede interna e externa, caso não sejam devidamente descartados. Nada impede alguém de construir fragmentos de pacotes sem o pedaço inicial e convertê-los novamente em pacotes em alguma outra máquina.

Em um caso ideal, um datagrama IP deve corresponde a um frame do nível físico, tornando dessa maneira a transmissão através da rede física mais eficiente. Para alcançar este nível máximo de eficiência, o protocolo IP foi projetado de forma a ter um tamanho de datagrama máximo, selecionável pelo usuário, para assim melhor adequar um datagrama a um frame [COM91a].

Para melhor entender o problema, deve-se levar em conta o *hardware* de rede: cada tecnologia de transporte de pacotes costuma possuir um limite máximo

de dados que podem ser transferidos em um único frame. Por exemplo, a Ethernet limita a transferência em 1500 octetos de dados, enquanto a proNET-10 permite 2044 octetos por frame. Este limite é conhecido como MTU - *Maximum Transfer Unit* da rede. O tamanho do MTU também pode ser bastante pequeno, algumas tecnologias o limitam em 128 octetos ou menos [COM91a].

O ataque ocorre da seguinte forma: em muitas implementações IP é possível impor um tamanho não usual de fragmentos nos pacotes de saída. Se o tamanho do fragmento é suficientemente pequeno para forçar alguns dos campos do cabeçalho TCP de um pacote TCP sejam transportados para um segundo fragmento, as regras de filtragem que especificam padrões para aqueles campos não serão verificadas. Se as implementações de filtragem não obrigam um tamanho mínimo para o tamanho do fragmento, um pacote que deveria ser descartado pode ser passado devido a não haver nele o campo no pacote verificado pelo filtro.

Segundo a RFC 791 [POS81]:

“cada módulo internet deve ser capaz de transportar um datagrama de 68 octetos sem mais fragmentação. Isto porque um cabeçalho internet pode ter mais de 60 octetos, e o fragmento mínimo é de 8 octetos”

Nota-se contudo, que para propostas de segurança, não é suficiente meramente garantir que um fragmento contenha no mínimo 8 octetos de dados além do cabeçalho IP porque informações importantes do cabeçalho do nível de transporte (isto é, o campo CODE do cabeçalho TCP) pode estar situado além do 8º octeto de dados.

No exemplo da figura 2.5, pode-se observar que o primeiro fragmento contém somente oito octetos de dados (o tamanho mínimo de um fragmento). No caso de protocolo TCP, isto é suficiente para conter os números da porta origem e destino, mas o campo de flags será forçado a ficar em um segundo fragmento.

Os filtros que por ventura tentem impedir uma requisição de conexão (datagramas TCP tendo SYN=1 e ACK=0) serão incapazes de testar estes flags no primeiro octeto, e tipicamente o ignorarão nos fragmentos subseqüentes.

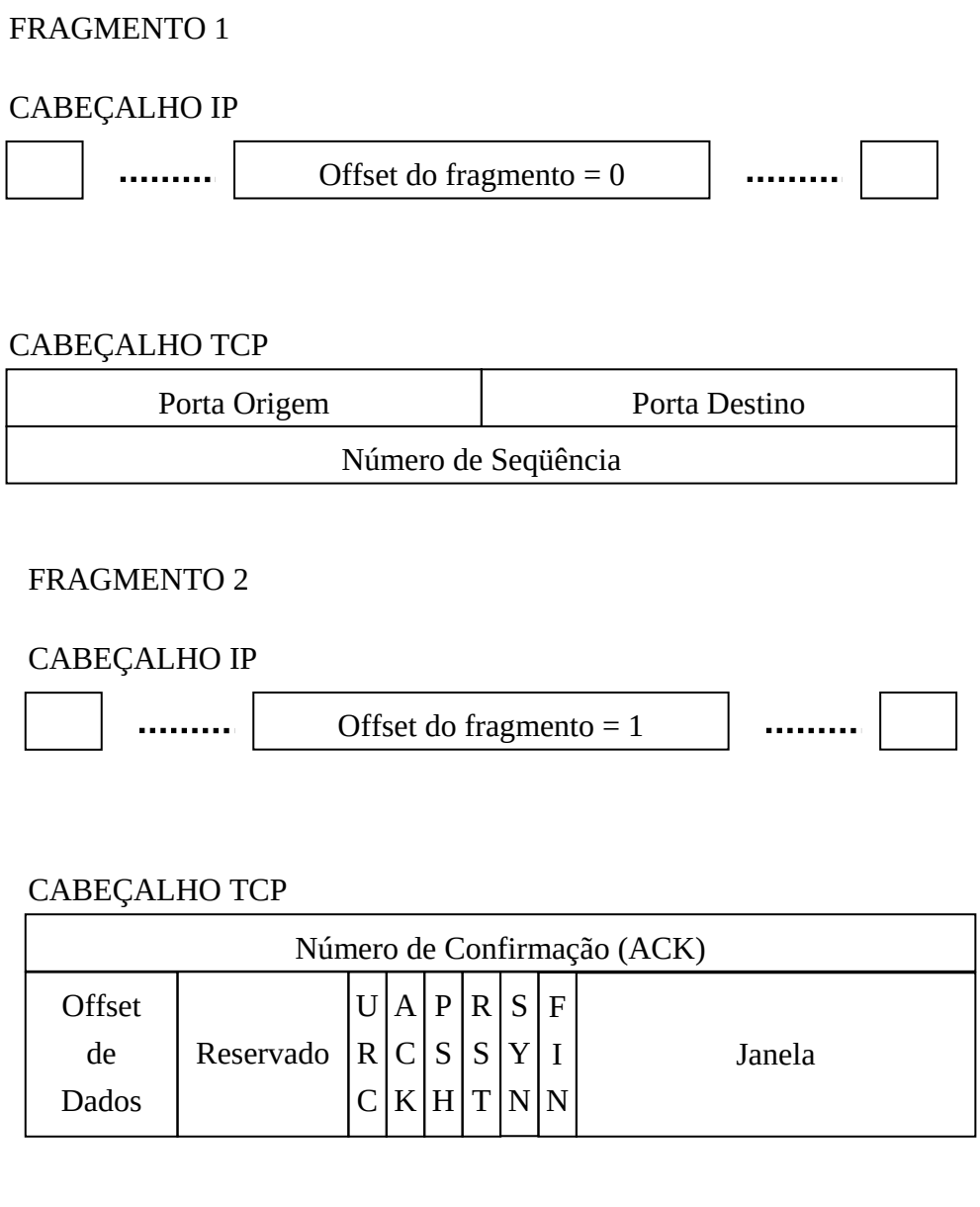

Figura 2.5: Fragmentação do cabeçalho TCP

Uma variação para um ataque utilizando técnicas de fragmentação, é conhecido como “ataque do fragmento sobreposto”. Ele é demostrado a seguir.

A RFC 791[POS81], atual especificação para o protocolo IP, descreve um algoritmo para remontagem de pacotes (figura 2.6). Observe que o seu processamento resulta nos novos fragmentos rescrevendo qualquer porção que sobreponha, mesmo parcialmente, um fragmento anteriormente recebido.

```
Notation:

        FO     -  Fragment Offset
        IHL    -  Internet Header Length
        MF     -  More Fragments flag
        TTL    -  Time To Live
        NFB    -  Number of Fragment Blocks
        TL     -  Total Length
        TDL    -  Total Data Length
        BUFID  -  Buffer Identifier
        RCVBT  -  Fragment Received Bit Table
        TLB    -  Timer Lower Bound

Procedure:

        (1)  BUFID <- source|destination|protocol|identification;
        (2)  IF FO = 0 AND MF = 0
        (3)     THEN IF buffer with BUFID is allocated
        (4)             THEN flush all reassembly for this BUFID;
        (5)          Submit datagram to next step; DONE.
        (6)     ELSE IF no buffer with BUFID is allocated
        (7)             THEN allocate reassembly resources
                             with BUFID;
                             TIMER <- TLB; TDL <- 0;
        (8)          put data from fragment into data buffer with
                     BUFID from octet FO*8 to
                                        octet (TL-(IHL*4))+FO*8;
        (9)          set RCVBT bits from FO
                                       to FO+((TL-(IHL*4)+7)/8);
        (10)         IF MF = 0 THEN TDL <- TL-(IHL*4)+(FO*8)
```

```
(11)          IF FO = 0 THEN put header in header buffer
(12)          IF TDL # 0
(13)           AND all RCVBT bits from 0
                                        to (TDL+7)/8 are set
(14)             THEN TL <- TDL+(IHL*4)
(15)                  Submit datagram to next step;
(16)                  free all reassembly resources
                      for this BUFID; DONE.
(17)          TIMER <- MAX(TIMER,TTL);
(18)          give up until next fragment or timer expires;
(19) timer expires: flush all reassembly with this BUFID;
              DONE.

In the case that two or more fragments contain the same data
either identically or through a partial overlap, this procedure
will use the more recently arrived copy in the data buffer and
datagram delivered.
```

Figura 2.6: Algoritmo de remontagem de pacotes proposto pela RFC971[POS81]

Devido a tal implementação de remontagem de pacotes, um agressor pode construir uma série de pacotes nos quais o fragmento de menor deslocamento (offset 0) no datagrama original pode conter dados inócuos (passando assim pelo filtros de pacotes), e no qual algum pacote subsequente tendo um offset diferente de zero possa sobrepor as informações do cabeçalho TCP (porta destino, por exemplo), causando a sua modificação. O segundo pacote deverá passar pela maioria das implementações de filtros hoje existentes por ter um offset de fragmento diferente de zero.

No exemplo a seguir (figura 2.7), os fragmentos são grandes o suficiente para satisfazer o requisito de tamanho mínimo exigido, como descrito anteriormente. O filtro está configurado para rejeitar pedidos de conexão TCP.

O primeiro fragmento contém os valores SYN=0 e ACK=1, que permitem a passagem ilesa através do filtro.

O segundo fragmento, com um offset de oito octetos, contém flags TCP que diferem daqueles no primeiro fragmento, ou seja, SYN=1 e ACK=0. Considerando que este é um fragmento com offset diferente de zero, ele não será verificado, e assim sendo, também passará pelo filtro.

O host destino, se preenche todos os requisitos de implementação estipulados nos algoritmos da RFC 791, reconstituirá o pacote como uma requisição de conexão, devido aos “dados” recebidos anteriormente.

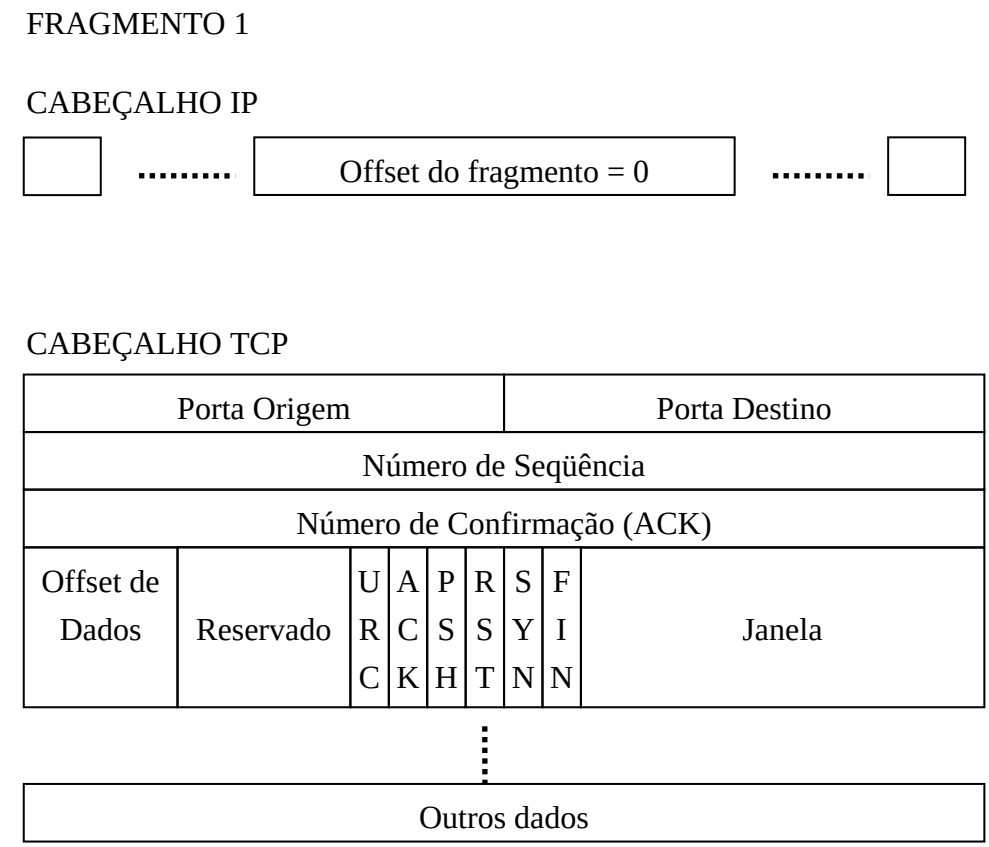

Figura 2.7: Sobreposição do fragmento do cabeçalho TCP

Caso o host destino possua um algoritmo de remontagem que previna novos dados de sobreporem dados recebidos anteriormente, pode-se ainda enviar o fragmento número 2 primeiro, seguido pelo número 1, e assim realizar o mesmo ataque com sucesso.

2.1.4 SYN Flooding

Esta é uma técnica freqüentemente utilizada em conjunto com outras como ip-spoofing (visto anteriormente). Seu principal objetivo é impedir que um determinado host consiga se comunicar com outros, impedindo que ele receba qualquer tráfego da rede e inundando-o com pacotes. Existem muitas maneiras diferentes de se fazer isto, e aqui será abordada uma delas, conhecida como TCP SYN flooding.

Como demonstrado em capítulo anterior, uma conexão TCP é iniciada com um cliente emitindo um pedido para o servidor através do flag SYN no cabeçalho do pacote TCP. Normalmente o servidor envia de volta um SYN/ACK para o cliente identificado pelo endereço origem no cabeçalho IP. O cliente envia então um ACK para o servidor (figura 2.3), e então se inicia a transmissão dos dados. Entretanto, há um limite máximo para o número de pedidos de conexão que possam ser atendidos simultaneamente, isto é, existe um limite máximo de pacotes com o flag SYN configurado requisitando uma conexão TCP que pode ser processado concorrentemente em um mesmo socket [STE90]. Este limite é conhecido como *backlog*, e ele é o tamanho da fila de entrada onde são enfileiradas as requisições de conexões que ainda não foram completadas. Este limite de fila se aplica para ambos os números de conexões incompletas (3-*way handshake* não completados), mais o número de conexões completadas que ainda não foram retiradas da fila por aplicações utilizando a função *accept()* de chamada ao sistema. Se o limite *backlog* é alcançado, o nível TCP silenciosamente descartará todos os pedidos (SYN) que chegarem, até que todas as requisições possam ser tratadas. Nisto reside o ataque.

O host atacante envia várias requisições SYN para a porta TCP que ele deseja bloquear. O host atacante também precisa garantir que o endereço IP da origem é falso, preferencialmente um host inexistente (o host atacado enviará uma resposta

para este endereço). O nível IP pode informar que o host não foi encontrado (*unreachable*), mas o nível TCP considerará estes erros como passageiros, e deixa para a camada de baixo (IP) solucioná-los (rerotear pacotes, etc), efetivamente ignorando-os. O endereço IP colocado no cabeçalho como originador necessariamente deve ser inexistente, evitando assim que qualquer host receba um pacote resposta (SYN/ACK) e envie uma resposta (RST), frustrando assim o ataque.

Os passos do ataque podem ser verificados na figura abaixo:

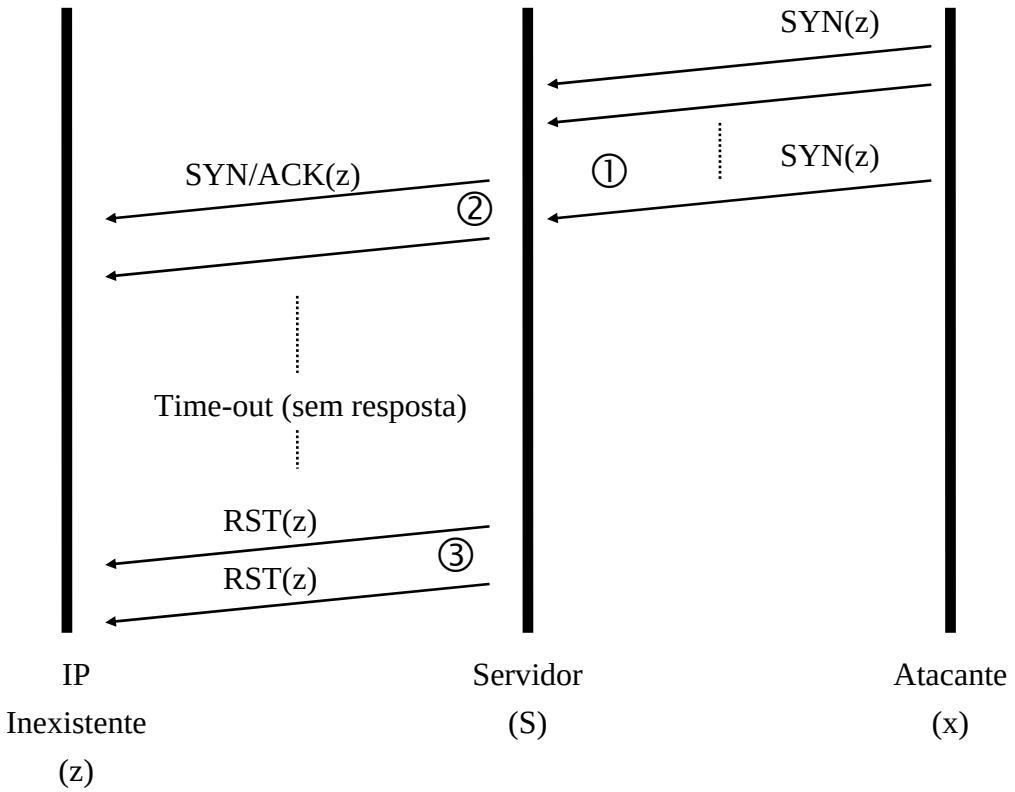

Figura 2.8: Simulando um ataque do tipo SYNflooding

Em (1) o host agressor envia uma enxurrada de requisições de conexão (SYN) para o alvo, até preencher a fila de conexões pendentes (*backlog*). Em (2) a máquina destino dos pacotes de SYN responde com SYN/ACK para o host que ela acredita ser a fonte dos SYN que estão chegando. Durante este momento todas as outras conexões para esta porta TCP serão ignoradas. Em (3) a máquina alvo envia pacotes RST após o seu time-out.

Diferentes implementações TCP possuem diferentes tamanhos de *backlog*. Sistemas do tipo BSD (FreeBSD) geralmente tem um *backlog* de 5, sistemas como Linux possuem *backlog* de 6.

2.2 Tecnologias existentes de Proteção

Como apresentado no capítulo anterior, pode-se notar que os sistemas operacionais existentes hoje, mesmo os do nível C2, segundo os critérios de avaliação do DoD [DOD85] ainda são vulneráveis. Isto exige que seja realizado um esforço complementar para garantir a sua segurança, não se esquecendo nunca que mesmo com um esquema bastante completo de segurança, o controle por parte dos operadores deve ser uma constante, analisando logs e verificando possíveis alarmes.

2.2.1 TCPWrapper

O TCPWrapper [VEN92] (também conhecido como tcpd) é um dos muitos pacotes de segurança criados por Wietse Venema. Ele é utilizado para realizar basicamente filtragem de pacotes em hosts unix [HUG95].

Dentre os serviços oferecidos pelo tcpd destacam-se dois: filtragem e logs. Cada um destes serviços pode ser utilizado independentemente, embora uma utilização em conjunto seja considerada mais eficiente. Para quem deseja um mínimo de segurança ao se conectar na Internet, filtros como TCPWrapper são indispensáveis.

Tcpd é uma ferramenta que fica situada entre o serviço inetd e o programa servidor para uma determinada porta, de forma totalmente transparente para o usuário final. O funcionamento do mecanismo é o seguinte: o serviço inetd invoca o programa tcpd que registra e filtra todas as conexões, tcpd então invoca a aplicação servidora, caso ela seja aprovada pelo inspeção do filtro.

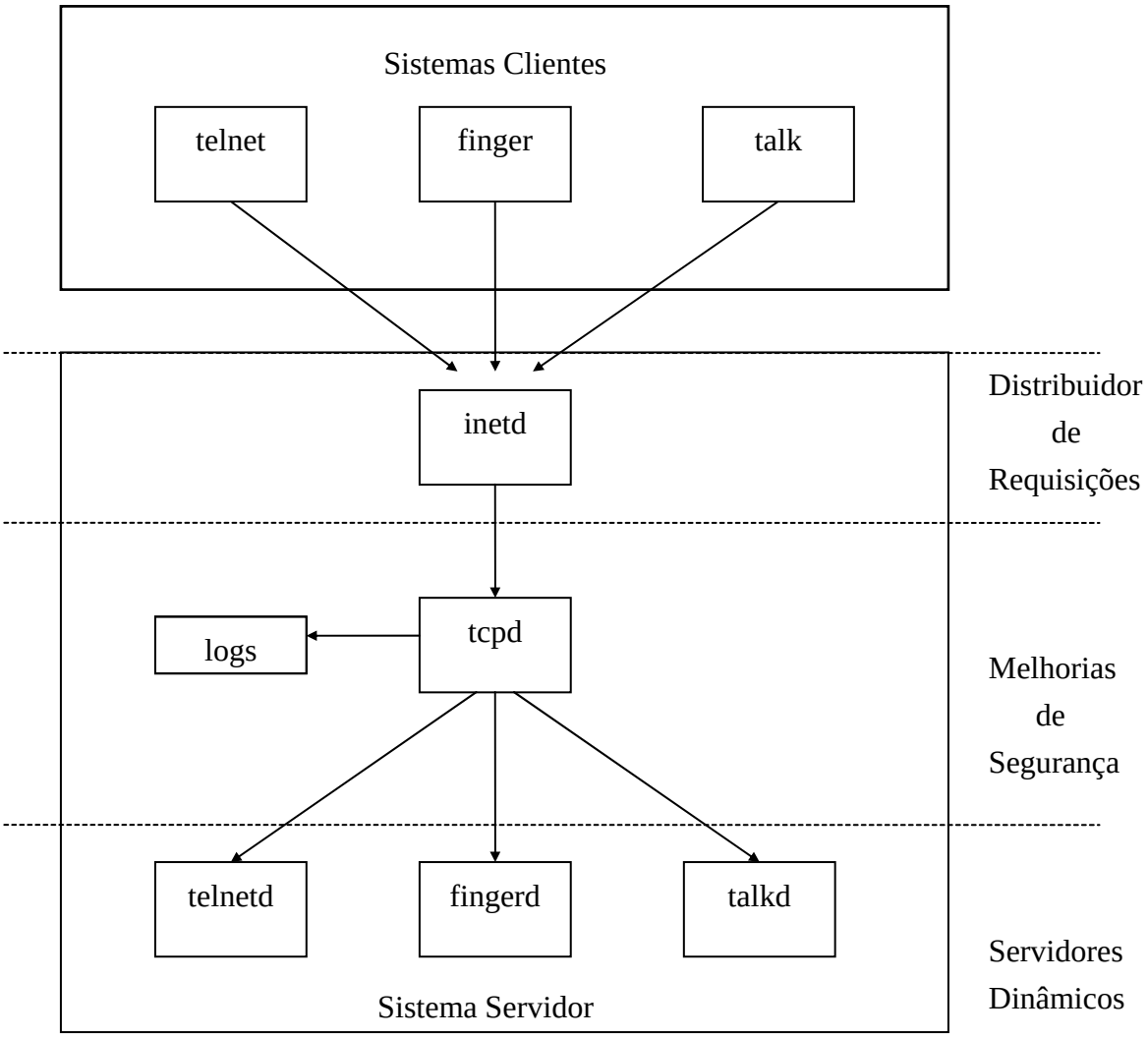

Figura 2.9: Inetd operando com tcpd

O TCPWrapper necessita que sejam especificadas regras de filtragem em uma ou duas bases de dados diferenciadas, onde são especificadas os tipos de pedidos de conexões ou pacotes que devem ser aceitos (/etc/hosts.allow) ou negados (/etc/hosts.deny). Este aplicativo foi implantado em segmentos considerados importantes da universidade, pelos logs gerados e, principalmente, por suas características de filtragem.

```
# Permitido para penta e maquinas do pop na rede da Ufrgs
ALL: 143.54.1.20 143.54.1.124 143.54.1.179 143.54.1.190

# Permitido a partir da maquina da puc-rio(refletor)
ALL: 139.82.17.17

# Maquina da liane (noc.tche.br)
ALL: 200.17.171.137

# Permitido para as maquinas do pop
ALL: 200.132.0.21 200.132.0.30 200.132.0.24 200.132.0.25

# Permissoes temporarias (trentin,trentin,liane)
ALL: vitoria.upf.tche.br pampa.inf.ufrgs.br npd.uel.br

# Permissoes temporarias para atualizacao via rdist (eleicoes)
ALL: 200.19.119.125 200.17.132.34 200.17.132.55 200.19.246.20
```

Figura 2.10: Regras de filtragem no *tcpd* (/etc/hosts.allow)

2.2.2 Firewalls

As Novas tendências em segurança de redes apontam o uso de firewalls como a melhor solução para os problemas de segurança. Firewalls são um conjunto de filtros e gateways que protegem uma rede considerada “confiável”, com um perímetro de segurança bem definido da rede externa, considerada perigosa e “não confiável”. Utilizar uma estratégia de firewall isola a rede interna, supostamente considerada controlável, dos perigos da rede externa, definitivamente fora de controle. Isto funciona da mesma forma que uma porta corta fogo que age como uma barreira impedindo que as chamas se alastrem em um incêndio. Diferente no entanto, da associação com o mundo real, uma firewall deve permitir que alguns tipos de tráfego passem através dela, provendo assim, acesso a um conjunto de serviços geralmente bastante limitado, mas presumivelmente seguros.

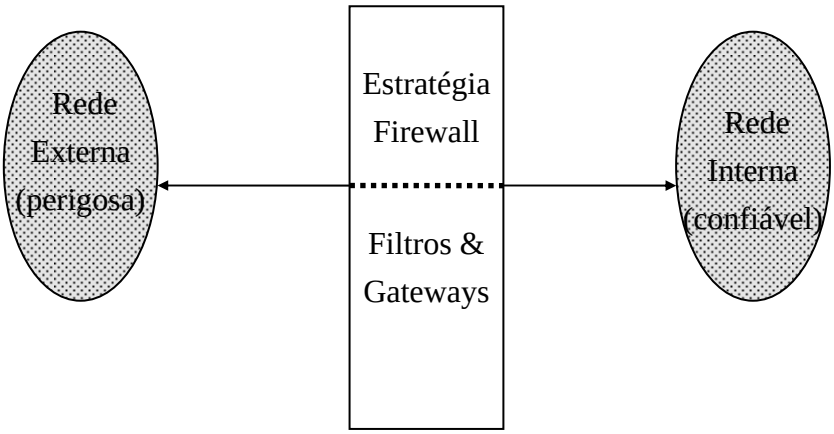

Figura 2.11: Estratégia de firewall genérica

Uma firewall deve ser mais segura que a maioria dos hosts. A única grande razão para isto é que ela não é uma máquina genérica; portanto não necessita de aplicações com características duvidosas de segurança, que existem apenas para conveniência dos usuários, como NIS (Network Information Service), rlogin e NFS (Network File System) [SUN91a] [SUN91b] [SUN91c].

Existem hoje inúmeras táticas de segurança envolvendo o uso de firewalls, oferecendo assim diferentes níveis de segurança que variam quanto a preço, equipamento, software, e material humano necessários. Dentre as estratégias específicas de firewall existentes, pode-se citar [BEL94]:

- filtros de pacotes;
- gateways de aplicação; e
- gateways de circuito.

Filtros de pacotes podem prover de uma maneira barata e útil a primeira linha de defesa em qualquer estratégia firewall. Comumente situados entre os limites de vizinhança da rede, os filtros de pacotes trabalham examinando pacotes advindos de outras redes, os descartando segundo regras de filtragem previamente definidas pelo administrador.

A filtragem dos pacotes é realizada examinando os cabeçalhos IP e TCP, e levando-se em conta dados como: endereço IP origem e IP destino da conexão, portas TCP origem e TCP destino, tipo de protocolo usado, flags do nível TCP e opções do protocolo IP, portanto, cabe ao usuário definir estes itens em suas regras de filtragem.

O exemplo mais comum de filtros de pacotes são aqueles implementados no próprio software dos roteadores. Como este equipamento já se utiliza de regras para permitir a interconexão de sistemas na Internet, novas regras de filtragem podem ser facilmente incorporadas, com o único custo de configuração e teste. Posteriormente sofrendo alguns ajustes. Roteadores que implementam funções de filtragem são conhecidos como roteadores escrutinadores [RAN93].

Abaixo está um exemplo de uma lista de acesso configurada em um roteador:

```
access-list 100 permit tcp any 200.132.0.0 0.0.0.255 established
access-list 100 permit tcp any host 200.132.0.1 eq www
access-list 100 permit tcp any host 200.132.0.1 eq ftp
access-list 100 permit tcp any host 200.132.0.1 eq ftp-data
access-list 100 permit tcp any host 200.132.0.1 eq smtp
access-list 100 permit tcp any host 200.132.0.1 eq pop3
access-list 100 permit udp any host 200.132.0.1 eq domain
access-list 100 permit udp any host 200.132.0.250 eq domain
access-list 100 permit icmp any any
access-list 100 permit tcp any host 200.132.0.253 eq ftp
access-list 100 permit tcp any host 200.132.0.253 eq ftp-data
access-list 105 permit tcp 200.132.0.0 0.0.0.255 any
access-list 106 permit tcp any 200.132.0.0 0.0.0.255 established
access-list 106 permit icmp any 200.132.0.0 0.0.0.255
access-list 106 permit tcp any 200.132.0.0 0.0.0.255 eq www
```

Figura 2.12: Filtro de pacotes em um roteador

Gateways de aplicação representam o extremo oposto em projetos de firewall. Ao invés de utilizar um mecanismo de propósito geral, como um roteador, para tratar os diferentes tipos de tráfego, é usada uma implementação especial para tratar com cada aplicação. Embora isto pareça um desperdício, é provavelmente mais seguro que as outras alternativas. Ninguém precisa se preocupar sobre interações entre diferentes conjuntos de regras de filtragem, nem sobre falhas em centenas de hosts que oferecem serviços nominalmente seguros para o mundo externo. Somente alguns poucos programas necessitam ser observados.

Gateways de aplicação tem ainda outra vantagem que em alguns ambientes é bastante crítica: é fácil de registrar e controlar todo o tráfego de entrada e saída. Eles são freqüentemente utilizados em conjunto com outros gateways, filtros de pacotes e gateways de circuito. A principal desvantagem de um gateway de aplicação é a necessidade de um programa especializado para interfacear com os serviços providos.

O terceiro tipo de gateway, preferencialmente utilizado em conexões com destino em uma máquina externa é o do nível de circuito. Gateways de circuito são repassadores de conexões TCP. O chamador se conecta em uma porta TCP em um gateway, que o conecta para algum destino na rede externa. Durante a chamada, o programa que reside no gateway repassa os bytes de um lado para o outro, agindo assim como um fio(circuito) que conecta as duas máquinas. Em geral, estes servidores de retransmissão não examinam os dados que passam por eles.

Um dos grandes problemas com a utilização de gateways de circuito é a necessidade de prover novos programas cliente, para desta forma possibilitar uma conexão em dois passos. Várias estratégias estão hoje disponíveis para realizar esta mudanças, sendo a mais comum delas o uso de bibliotecas como *socks* [KOB92]. Ela consiste de várias substituições a várias chamadas ao sistema: socket, connect, bind, etc.

2.2.8 SATAN

SATAN (Security Administrator Tool for Analyzing Networks) é uma ferramenta criada para ajudar administradores de sistemas a identificar problemas comuns relativos a segurança da rede e colaborar na sua eliminação. O pacote consiste de um pequeno kernel que se baseia em uma base de dados com regras e vários outros programas responsáveis por diferentes testes de validação de segurança. Cada um destes programas pode gerar ao final de sua execução até vários megabytes de informação que são analisadas e repassadas ao usuário sob forma de hipertexto, o que torna a ferramenta extremamente amigável e inovadora.

A sistemática de funcionamento da ferramenta está baseada em informações obtidas pela rede, e através das quais pode-se deduzir o fabricante (SUN, IBM, DEC, etc.), bem como a natureza do sistema que está rodando (servidores de arquivos, estações diskless, etc.) além de identificar os vários serviços disponibilizados por cada um dos seus alvos (WWW, Gopher, Ftp anônimo, rexd, etc.). A ferramenta identifica vários problemas, sem contudo tentar explorá-los. Todos os problemas por ele identificados já foram anteriormente notificados por órgãos de segurança como o CERT (Computer Emergency Response Team) e CIAC (Computer Incident Advisory Capability), ou então explorados por livros e artigos sobre práticas de segurança, como em [GAR94], [BRY94] e [FAR95]. O sistema SATAN foi desenvolvido por Dan Farmer e Wietse Venema, utilizando a linguagem C, scripts shell, PERL e o software Mosaic[VEN95].

SATAN é composto de um kernel central, que implementa a parte mais geral do sistema, e por vários outros pequenos programas responsáveis por detalhes de implementação de serviços de rede e testes de vulnerabilidades. O kernel é composto pelos seguintes subsistemas:

Cripto-ignição: responsável pela geração de uma chave pseudo-randômica para validar a comunicação entre o cliente HTML e o servidor HTTP (Este servidor é implementado pelo próprio SATAN, e possui apenas um subconjunto dos comandos definidos pelo protocolo).

Máquina de política: Este subsistema determina quais hosts serão pesquisados, e quais os níveis de teste são adequados para aquele hosts. Isto é realizado baseando-se no arquivo de configuração *satan.cf.* Aqui é definida a política para a análise de segurança a ser usada pelo SATAN.

Aquisição de alvos: dada uma lista de hosts, este subsistema gera uma lista de testes a serem utilizados para aqueles hosts

Aquisição de dados: A partir da lista de testes gerada pelo subsistema de aquisição de alvos, este subsistema executa o conjunto de operações correspondente, e gera novos fatos, que servirão como entrada para a máquina de inferência

Máquina de inferência: responsável por gerar uma nova lista de hosts, testes e fatos, baseada nos dados obtidos pelo sistema

Relatórios e análise: subsistema responsável pela tabulação dos dados na linguagem HTML, para ser repassados ao browser WWW. É composto basicamente por scripts PERL, que lêem os arquivos de resultado dos testes e os formatam em textos HTML.

SATAN opera iniciando conexões para diferentes portas em computadores remotos, e então determina que tipo de informação cada porta retorna. Ele testa um grande número de vulnerabilidades, incluindo NFS (Network File System) e NIS (Network Information Service).

2.2.9 MD5

O MD5 (Message Digest Algorithm) [RFC1321] é uma proposta de um padrão para autenticação de dados. O MD5 processa o texto de entrada em blocos de 512 bits, divididos em 16 superblocos de 32 bits. A saída do algoritmo é um conjunto de quatro blocos de 32 bits, concatenados de forma a gerar um único valor de 128 bits.

O MD5 tem a peculiaridade de que cada bit da entrada é utilizado para gerar o código resumido. A repetição da execução de várias funções básicas produz

resultados que são bem mixados, ou seja, é bastante improvável que duas mensagens escolhidas aleatoriamente, mesmo sendo as duas bastante regulares e semelhantes, possuirão uma mesma representação resumida [STA95].

2.2.10 DES

É um dos algoritmos de criptografia mais largamente utilizado nos sistemas de hoje. O DES (Data Encription Standard) foi inicialmente desenvolvido pela IBM na década de 70 [SCH95]. O algoritmo baseia-se na utilização funções de permutação, substituição e recombinação executadas em blocos de 64 bits de dados e blocos de 56 bits de chave, ou seja, 8 caracteres de 7 bits. O algoritmo é estruturado de tal forma, que a troca de qualquer bit da entrada tem o efeito de afetar a maioria dos bits da saída.

O algoritmo DES pode ser usado de quatro modos:

– Eletronic Code Book (ECB)

– Cipher Block Chainning (CBC)

– Cipher Feedback (CFB)

– Output Feedback (OFB)

Cada modo tem particular vantagens em algumas circunstâncias, tais como transmitir dados sobre um canal com ruídos, ou quando é necessário decriptografar somente uma porção do arquivo. Os quatro modos são descritos abaixo:

Modo ECB - Cada bloco de entrada é encriptado usando a chave, e a saída é escrita como um bloco. Este método é uma simples criptografia de uma mensagem, um bloco de cada vez. Este bloco pode não indicar onde as porções da mensagem foram inseridas, ou removidas. Ele funciona bem em canais com ruído de transmissão, pois a alteração de alguns poucos bits somente afetará um único bloco de 64 bits.

Modo CBC - O texto da mensagem passa por uma operação XOR com o valor encriptado no bloco anterior. O resultado é encriptado usando uma chave. Devido a propagação de cada bit da mensagem ocorrerem até a saída, o último bloco pode ser usado como uma assinatura de que a encriptação não foi alterada. Igualmente, após sucessões de repetidas trocas, a saída será mascarada.

Modo CFB - No modo de cifragem por realimentação, a saída alimenta novamente o mecanismo. Depois de cada bloco ser encriptado, parte é deslocada em um registrador de deslocamentos. Os conteúdos deste registrador são então encriptados com a chave do usuário, usando o modo ECB, e esta saída submetida a uma operação XOR com a seqüência de dados, produzindo assim o resultado da criptografia. Este método é auto-sincronizante, e possibilita a um usuário decifrar apenas uma porção de uma grande base de dados apenas iniciando em uma distância fixa anterior ao início dos dados desejados.

Modo OFB - No modo de realimentação de saída, a saída também é realimentada na saída. Um registro é inicializado com um valor. Este registro é o encriptado com o modo ECB usando a chave do usuário. O resultado disto é então usado como chave para encriptar os blocos de dados (usando uma operação XOR), e então colocado novamente no registro para ser usado no próximo bloco.

O DES usa a mesma chave para encriptar e decriptografar os dados. Por este motivo, é essencial usar técnicas que mantenham o sigilo desta chave. Um mal gerenciamento ou uma má escolha desta chave pode levar a uma redução na efetividade, ou total ineficácia deste algoritmo.

3 O SISTEMA CUCO

Conforme o que foi apresentado no capítulo anterior, pode-se entender o porquê de muitos profissionais da área de segurança considerarem impossível a existência de um sistema 100% seguro. O que foi apresentado certifica que em se tratando de segurança não há como provar se uma estratégia usada é ou não efetiva, nem se os sistemas que foram desenvolvidos por este ou aquele fabricante são ou não seguros[CHE94].

Visando solucionar paleativamente o problema de um sistema aberto, diretamente conectado a uma rede de computadores, surgiram pesquisas que visam a detecção automatizada de intrusões[9]. Estes mecanismos, totalmente automáticos tentam precisar tentativas de violação na segurança em uma determinada máquina, e assim revelar novas falhas como a exploração de novos bugs, estouros de pilha (vide anexo A-1) e outros. Existem alguns desenvolvimentos nesta área que se utilizam de técnicas de inteligência artificial, e que hoje abrangem basicamente dois grupos[SPA93]:

- detecção de anomalias;
- detecção de violação.

O sistema CUCO utiliza a primeira abordagem. A detecção por anomalia é baseada na premissa que atividade intrusa freqüentemente se manifesta como uma anormalidade. Esta abordagem se baseia em constantes medidas realizadas pelo sistema, de forma a detectar através de grandes variâncias nestas métricas a indicação de uma intrusão. Um exemplo disso é um grande número de conexões sendo rejeitadas pelo sistema por *time-out*, uma possível evidência de um ataque por negação de serviço.

[9] Conceitualmente, intrusão é compreendido como a detecção de atividade ilegal e aquisição de privilégios que não pode ser detectado através do fluxo normal de informações e os modelos de controle de acesso

Para detectar uma anomalia, o sistema mantém uma base de conhecimento sobre diferentes sistemas operacionais e usuários, e a utiliza para parametrizar com o estado atual de uma determinada máquina. Isto será melhor explicado nos capítulos a seguir.

A detecção de intrusão é uma abordagem nova de segurança. Ela provê uma sensação de segurança em uma rede de computadores e dados, enquanto permite a eles operarem de uma forma "aberta". A meta desta técnica é identificar preferivelmente em tempo real, uso não autorizado, mau uso, e abuso de sistemas de computação por ambos, usuários internos e/ou invasores externos [MUK94].

A segunda abordagem, detecção de violação ou abuso, baseia-se em técnicas específicas de representação de conhecimento sobre "comportamentos inaceitáveis" e tentativas de detectar estas ocorrências. Um exemplo disso seria a detecção do abuso do fingerd e sendmail, usado no ataque do "Internet Worm" [SPA88].

3.1 A Estrutura do Sistema CUCO

A estrutura de funcionamento da aplicação usa uma filosofia de monitoração, onde um único gerente monitora via SNMPv2 as métricas de dezenas de hosts (figura 3.1.1).

Cada host possui um agente SNMPv2, que se comunica com o gerente, repassando a ele possíveis alertas de violações, ou respondendo a consultas sobre o estado de seus objetos. A consulta normal é uma forma de verificação de atividade do agente, pois a indicação contrária pode significar que um ataque está em andamento. Isto é, toda a segurança acrescentada ao sistema depende basicamente da comunicação confiável entre os hosts monitorados e o programa gerente rodando na máquina diretora.

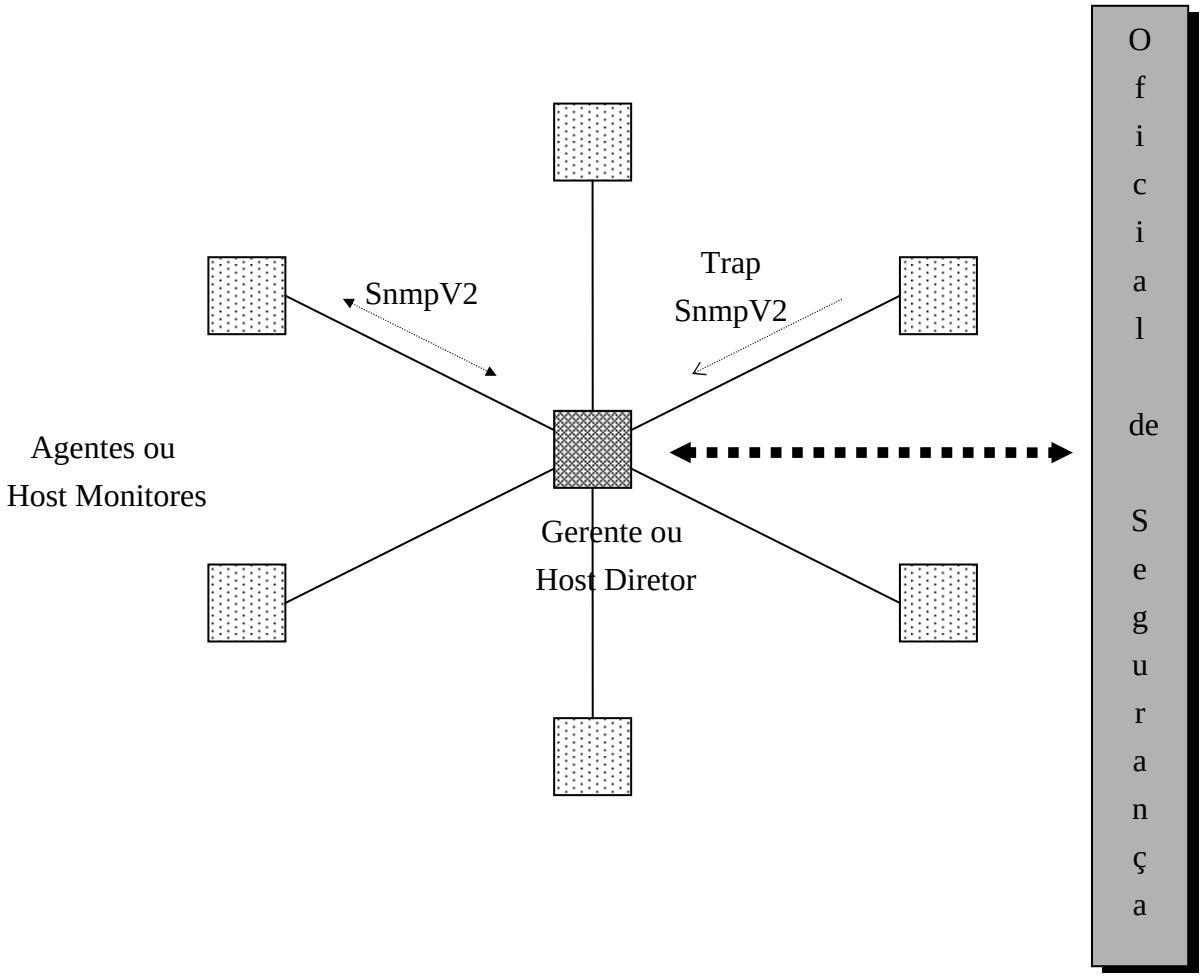

Figura 3.1.1: A estrutura do Sistema CUCO

A estrutura da aplicação é formada por dois módulos básicos, que implementam além do protocolo de gerência algumas outras funções:

Módulo Monitor: Composto por vários programas de aplicação, responsáveis por identificar uma possível invasão.

Módulo Diretor ou Gerente: Composto por uma interface amigável que permite realizar consultas SNMP e é ser capaz de notificar uma anomalia qualquer para o administrador responsável pela segurança (oficial de segurança) do *site*, para que este tome as devidas providências.

A troca de mensagens entre os módulos gerente e agente é realizada utilizando-se o protocolo SNMPv2. Esta escolha é devido às suas características incorporadas a partir do protocolo S-SNMP, características estas que garantem o sigilo e autenticidade das mensagens trocadas entre os parceiros. O protocolo de comunicação entre gerente e agente será demostrado em um capítulo a parte.

3.2 O Módulo Monitor

O módulo monitor é o coração do sistema CUCO. Ele é o responsável direto pela segurança do host onde ele está instalado. Cabe a ele detectar qualquer tipo de ataque ou modificações em arquivos de configuração; instalação de *trap doors*[10], tanto no sistema operacional quanto de aplicativos; garantir a integridade de dados sensíveis do usuário; manter uma base de informações (MIB) sobre o estado dos objetos e acionar os traps quando necessário. O módulo monitor possui cinco atribuições igualmente importantes, e estão assim distribuídas:

1.A implementação dos diversos módulos capazes de reconhecer a ocorrência de uma invasão: verificadores de integridade do sistema operacional e das configurações do sistema, monitor das conexões do sistema, controlador dos estados das interfaces de rede do hosts.

2.Controle automatizado dos logs do sistema; obtém um maior controle sobre o estado do sistema envolvendo para isto programas que automaticamente vasculham os logs existentes na máquina a procura de indícios.

3.A implementação do protocolo de gerência, o agente SNMPv2.

[10] Back doors ou trap doors, conceitualmente, são segmentos de programa inseridos no código de aplicações ou sistemas operacionais que permitem acesso dos programadores a programas sem necessitarem passar pelos métodos normais de autenticação de acesso. Freqüentemente, hackers instalam versões modificadas de login, telnetd, ftpd, rshd e outros que através de uma seqüência especial de entrada de dados dispara um shell para o usuário [GAR91].

4.A implementação de um módulo capaz de prover informações a partir da máquina atacada, bem como manter a base de informações (MIB) para ser consultado pelo host diretor (aplicação gerente).

5.Por último, um módulo responsável por recolher maiores informações sobre atividades suspeitas que estejam sendo realizadas por um usuário do sistema. Este módulo deve ser composto de ferramentas que implementam modificações nos códigos do programa shell e de um monitor a nível de terminal de usuário (tty), além de vários scripts que permitam extrair maiores informações do sistema. Este módulo foi parcialmente implementado, pois já existe na bibliografia aplicações que implementam monitores de tty [BELa] e shells modificados, necessitando apenas adaptações para funcionar diretamente com a interface do sistema CUCO.

A figura abaixo mostra o inter-relacionamento de todos os submódulos existentes, e que operam em conjunto para permitir que as funções supra citadas sejam realizadas.

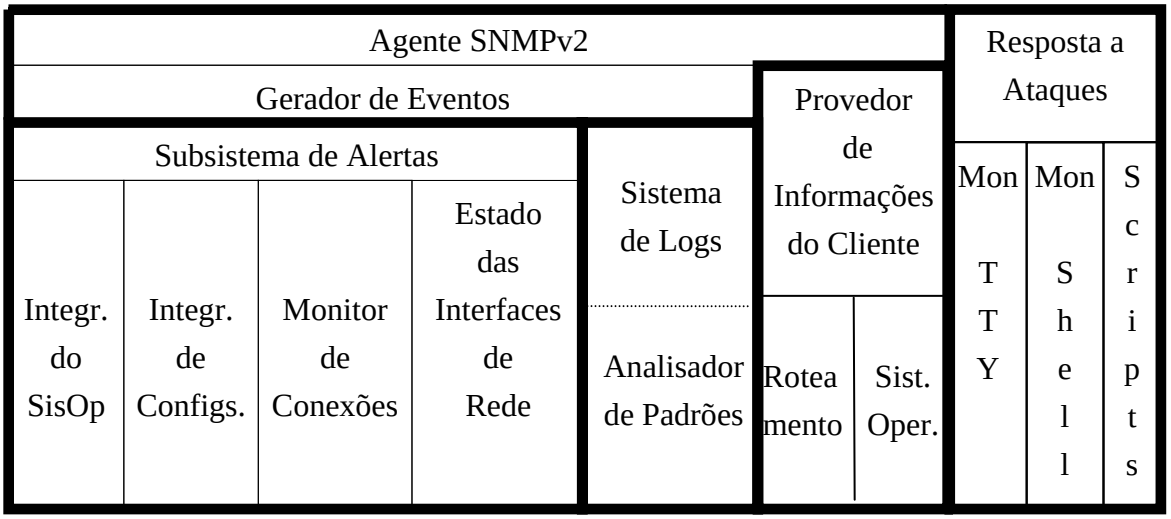

Figura 3.2.1: A estrutura do módulo monitor

A seguir serão explorados todos os componentes do módulo monitor, a exceção do submódulo de comunicação que implementa o agente SNMP. Este será, explanado em conjunto com o gerente SNMP do módulo gerente, em um capítulo a parte.

3.2.1 O Subsistema de Alertas

Este subsistema é composto pelos procedimentos de integridade de configurações e do sistema operacional, monitor de conexões e verificador de interfaces, as funções que formam o real sistema de alertas da aplicação CUCO.

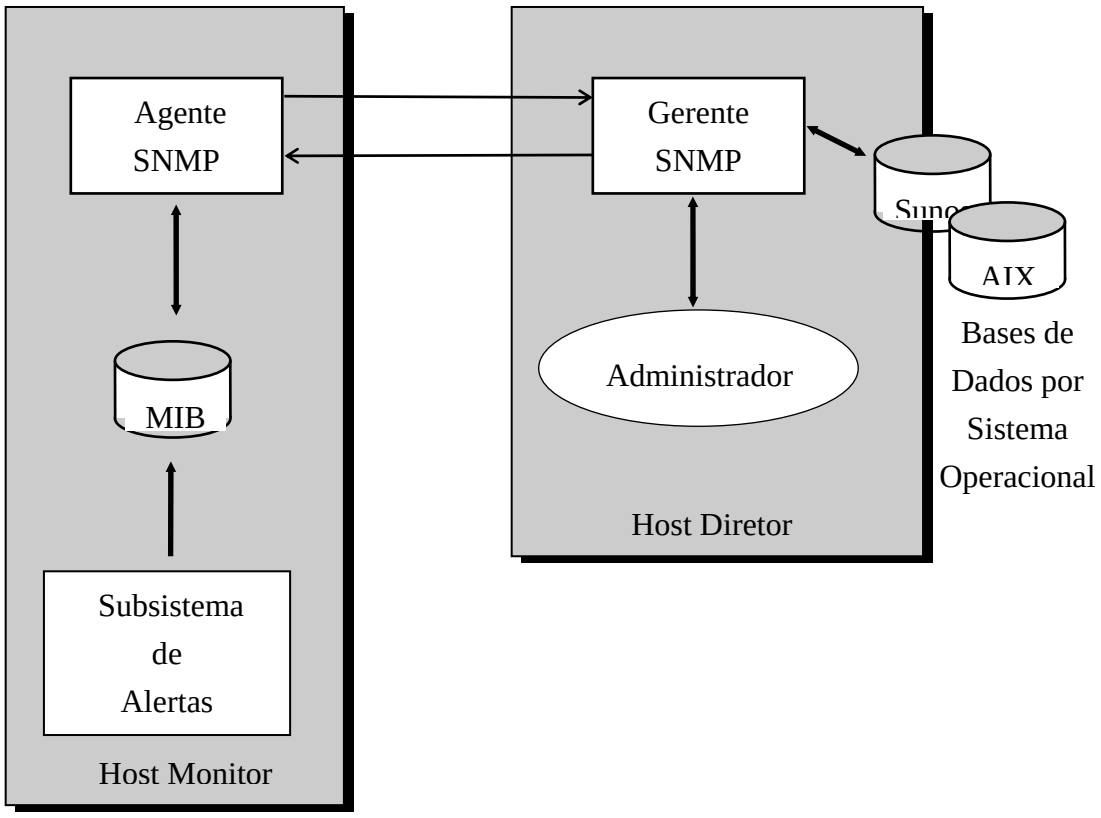

Figura 3.2.2: Os módulos de Integridade

Os alertas gerados por este subsistema baseiam-se em uma verificação periódicas dos arquivos de configuração e binários do sistema operacional, além de como se encontra o estado do sistema em determinados instantes, tal como será visto nas seções subsequentes.

3.2.1.1 Os Procedimentos de Avaliação de Integridade

A avaliação de Integridade, tanto do Sistema operacional quanto das configurações do sistema são realizadas utilizando o algoritmo MD5. Esta avaliação é realizada em dois níveis, um considerado obrigatório (binários do sistema operacional), e outro opcional (diversos arquivos de configuração e/ou dados). Os resultados obtidos com a execução do algoritmo alimentam diretamente os diversos objetos da MIB SNMP. Periodicamente o gerente consulta os dados de todos os agentes sob sua jurisdição, utilizando-os com uma base de dados existente no host diretor. Uma listagem da MIB experimental criada pode ser obtida no anexo A-4..

No caso da verificação de integridade do Sistema Operacional, o gerente compara o valor dos objetos da MIB fornecidos pelo agente, com uma base de dados local. Esta base de dados possui os MD5 de algumas aplicações consideradas vitais pelo sistema Unix. Se a verificação for referente a um arquivo de configuração do sistema, será consultada uma base de dados organizada por host monitorado, onde lá se encontra um índice com o MD5 de todos os arquivos de configuração considerados importantes, além de uma cópia destes arquivos, para que se verifique qual alteração foi realizada.

A manutenção das cópias dos arquivos de configuração não é um procedimento automático do sistema, cabe ao administrador mantê-las atualizadas. Este processo pode ser razoavelmente automatizado, mas é realizado ao próprio custo e risco do administrador. Apesar de parecer inviável a primeira vista, este procedimento não é tão penoso, e é de grande valia. Nos dois locais onde o sistema foi testado, mudanças de configurações são incomuns. No caso de uma invasão, estes dados são de grande utilidade para a verificação de alterações.

Nem sempre as alterações realizadas em arquivos de configuração do sistema são facilmente perceptíveis; como por exemplo a remoção de um único caracter, que pode deixar o site totalmente desprotegido.

Geralmente o caracter ‘#’ é usado como comentário nos arquivos de configuração dos sistemas Unix, como os de configuração do TCPWrapper, ou no próprio inetd. Este tipo de sutileza foi observado em uma das invasões descobertas.

```
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/ftpd -ld
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/telnetd
#shell   stream  tcp     nowait  root    /usr/sbin/tcpd /usr/sbin/rshd
#login   stream  tcp     nowait  root    /usr/sbin/rlogind      rlogind
#exec    stream  tcp     nowait  root    /usr/sbin/rexecd       rexecd
##
## Finger, systat and netstat give out user information which may be
## valuable to potential "system crackers."  Many sites choose to disable
## some or all of these services to improve security.
##
#finger stream  tcp     nowait  nobody  /usr/sbin/fingerd    fingerd
#systat       stream tcp    nowait nobody /usr/bin/ps           ps -ef
#netstat stream     tcp    nowait nobody /usr/bin/netstat      netstat -f
inet
#
# rexd uses very minimal authentication and many sites choose to disable
# this service to improve security.
#
#rexd   sunrpc_tcp   tcp    wait   root   /usr/sbin/rpc.rexd rexd 100017 1
```

Figura 3.2.3: Exemplo do Arquivo inetd.conf

A implementação atual do protótipo possui hoje onze arquivos binários, que foram escolhidos para estarem sob constante monitoração, vide tabela 3.2. A tabela 3.3 exibe os arquivos de configuração, que fazem parte do conjunto mínimo de monitoração.

SunOs 4.1.3	AIX 4.1.4
/bin/login	/usr/bin/login
/usr/kvm/ps	/usr/bin/ps
/usr/etc/in.telnetd	/usr/sbin/telnetd
/usr/etc/in/ftpd	/usr/sbin/ftpd
/usr/etc/rshd	/usr/bin/rshd
/usr/etc/rlogind	/usr/bin/rlogind
/usr/etc/inetd	/usr/sbin/inetd
/usr/bin/passwd	/usr/bin/passwd
/usr/bin/su	/usr/bin/su
/usr/etc/ifconfig	/etc/ifconfig

Tabela 3.1: Binários monitorados pelo sistema

SunOs 4.1.3	AIX 4.1.4
/etc/passwd	/etc/passwd
/etc/rc.local	/etc/rc.tcpip
/etc/hosts.equiv	/etc/hosts.equiv
/etc/hosts.allow	/etc/hosts.allow
/etc/hosts.deny	/etc/hosts/deny
/etc/inetd.conf	/etc/inetd.conf
/etc/syslog.conf	/etc/syslog.conf
/etc/sendmail.cf	/etc/sendmail.cf
/etc/services	/etc/services
/etc/aliases	/etc/aliases
$HOME/.forward	$HOME/.forward
$HOME/.rhosts	$HOME/.rhosts

Tabela 3.2: Arquivos de configuração monitorados pelo sistema

A tabela 3.3 exibe uma lista de eventos passíveis de geração de alarmes para o gerente, com uma possível conseqüência que explica o porquê destes objetos terem sido inicialmente escolhidos no protótipo

Objeto a ser gerenciado	Interpretação
/etc/hosts.equiv $HOME/.rhosts	Alterações permitem acesso ao sistema sem autenticação
/etc/ifconfig	Possibilidade de ocultar *eavesdropper* na rede
/vmunix ou correlato	Alteração no kernel do sistema para instalação de trap-doors
/etc/passwd	Usuário com privilégios especiais, ou facilitação para acesso indevido
rc.local, inetd.conf, services ou correlatos	Alteração no tipo de serviço de rede disponibilizado
/etc/syslog.conf	Tentativa de ocultar a passagem pelo sistema
Arquivos de diretórios /bin /usr/bin /sbin /usr/etc	Possibilidade de instalação de trap-doors

Tabela 3.3: Alguns eventos capazes de gerar alarmes

Acima estão algumas das verificações realizadas pelo sistema. Novos objetos podem ser incluídos, bastando que seja reservado inicialmente um objeto para ele na MIB.

3.2.1.2 O Monitor de Conexões do Sistema

O Monitor de conexões do sistema é também um dos componentes do subsistema de alertas. Através dele são identificadas, em uma primeira proposição, ameaças como SYN flooding. A informação sobre o estado das conexões TCP já é um objeto provido pela MIB SNMP hoje existente.

O objeto da MIB analisado é TcpConnEntry, que fornece uma visão geral das conexões TCP e o estado e que cada uma se encontra. Caso haja mais de cinco pedidos de conexões enfileiradas esperando para serem atendidas (estado synReceived(4)) , o sistema CUCO considera como um ataque contra o sistema.

```
        tcpConnTable OBJECT-TYPE
            SYNTAX  SEQUENCE OF TcpConnEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                    "A table containing TCP connection-specific
                    information."
            ::= { tcp 13 }

        tcpConnEntry OBJECT-TYPE
            SYNTAX  TcpConnEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                    "Information about a particular current TCP
                    connection.  An object of this type is
                    transient, in that it ceases to exist when
                    (or soon after) the connection makes the
                    transition to the CLOSED state."
            INDEX   { tcpConnLocalAddress,
                      tcpConnLocalPort,
                      tcpConnRemAddress,
                      tcpConnRemPort }
            ::= { tcpConnTable 1 }
        TcpConnEntry ::=
            SEQUENCE {
                tcpConnState
                    INTEGER,
                tcpConnLocalAddress
                    IpAddress,
                tcpConnLocalPort
                    INTEGER (0..65535),
                tcpConnRemAddress
```

```
                IpAddress,
            tcpConnRemPort
                INTEGER (0..65535)
        }
    tcpConnState OBJECT-TYPE
        SYNTAX  INTEGER {
                    closed(1),
                    listen(2),
                    synSent(3),
                    synReceived(4),
                    established(5),
                    finWait1(6),
                    finWait2(7),
                    closeWait(8),
                    lastAck(9),
                    closing(10),
                    timeWait(11),
                    deleteTCB(12)
                }
        ACCESS  read-write
        STATUS  mandatory
```

Figura 3.2.4: Trecho da MIB SNMP usado para controlar as conexões TCP

Através do comando netstat[GAR92] pode-se obter uma lista de todas as conexões TCP/IP ativas entre a sua máquina e a Internet, da mesma forma que a provida pela MIB.

Informações sobre o estado das conexões são de vital importância para que se confirmem suspeitas a respeito de invasões que estejam ocorrendo, ou mesmo que o host em questão esteja sendo utilizado como trampolim para realizar ataques contra outros sites. Dentre as informações obtidas por este tipo de consultas, estão inclusas: o número da porta em cada lado da conexão e o endereço IP das duas partes envolvidas, além do número de bytes disponíveis nos vetores de envio e recepção de mensagens. O sistema CUCO analisa estas informações através de um script em

PERL e repassa, no caso de alguma suspeita, as informações obtidas ao administrador de segurança da rede.

```
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address Foreign Address        (state)
tcp        0     44  circulo.nntp  jukebox.plug-in..1051  ESTABLISHD
tcp        0      0  circulo.nntp  lakesis.fapesp.b.2524  ESTABLISHD
tcp        0     80  circulo.telne porta3.tche.br.1024    ESTABLISHD
tcp        0      0  *.*           *.*                    CLOSED
tcp        0      0  *.smtp        *.*                    LISTEN
tcp        0      0  *.writesrv    *.*                    LISTEN
tcp        0      0  *.smux        *.*                    LISTEN
tcp        0      0  *.nntp        *.*                    LISTEN
tcp        0      0  *.www         *.*                    LISTEN
tcp        0      0  *.domain      *.*                    LISTEN
tcp        0      0  *.dtspc       *.*                    LISTEN
tcp        0      0  *.1026        *.*                    LISTEN
tcp        0      0  *.time        *.*                    LISTEN
tcp        0      0  *.daytime     *.*                    LISTEN
tcp        0      0  *.chargen     *.*                    LISTEN
tcp        0      0  *.discard     *.*                    LISTEN
tcp        0      0  *.echo        *.*                    LISTEN
tcp        0      0  *.telnet      *.*                    LISTEN
tcp        0      0  *.ftp         *.*                    LISTEN
tcp        0      0  *.sunrpc      *.*                    LISTEN
tcp        0      0  *.printer     *.*                    LISTEN
```

Figura 3.2.5: Uma máquina em funcionamento normal

```
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address Foreign Address        (state)
tcp        0     44  circulo.nntp  jukebox.plug-in..1051  ESTABLISHD
tcp        0      0  circulo.nntp  lakesis.fapesp.b.2524  ESTABLISHD
tcp        0     80  circulo.telne porta3.tche.br.1024    ESTABLISHD
tcp        0      0  *.*           *.*                    CLOSED
tcp        0      0  *.smtp        *.*                    LISTEN
tcp        0      0  *.writesrv    *.*                    LISTEN
tcp        0      0  *.smux        *.*                    LISTEN
tcp        0      0  *.nntp        *.*                    LISTEN
tcp        0      0  circulo.www   circulo.pop-rs.r.2342  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2343  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2344  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2345  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2346  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2347  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2348  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2349  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2350  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2351  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2352  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2353  SYN RECV
tcp        0      0  circulo.www   circulo.pop-rs.r.2354  SYN RECV
```

Figura 3.2.6: Conexões de uma máquina sob ataque ao serviço WWW

Mesmo através deste controle o sistema ainda pode ser ludibriado quanto à origem do ataque. Isto é explicado pelas falhas existentes na implementação dos serviços de rede, e que podem ser explorados visando mascarar temporariamente uma conexão, de forma a falsificar a sua origem (técnicas de *spoofing*), Não há nada que possa ser feito para se prevenir disto, a não ser se forem examinados os níveis inferiores da pilha TCP/IP; e, mesmo nesse caso, só há como se proteger se o ataque partir de uma máquina conectada diretamente na mesma rede, sem que exista nenhum roteamento entre o atacante e o defensor.

A análise deste comando permite a detecção de tentativas de ataque utilizando técnicas de SYN flooding. Dependendo da implementação que está sendo usada pelo agressor, o programa CUCO pode localizar a sua origem e gerar um relatório sobre a máquina que está sendo usada como fonte do ataque Se possível, o protótipo tenta obter informações a respeito da pessoa que está realizando esta agressão, além de outros dados como o responsável pelo site ou pessoa de contato (maiores informações podem ser obtidas no capítulo 3.3.2 “Localizando a Fonte do Ataque”) .

3.2.1.3 O Controlador de Estado das Interfaces de Rede

O controle sobre o estado das interfaces de rede é tido como de fundamental importância para o sistema CUCO. Esta verificação, por si só, evita que maiores danos sejam causados por hackers que consigam burlar algum sistema situado em pontos críticos da rede, ou seja, em um dos principais backbones. O objetivo primordial deste subsistema é evitar que se consiga realizar uma monitoração do tráfego da rede (eaversdropper). Um invasor pode tirar proveito disso para obter nomes de usuários e suas senhas em outras máquinas, podendo estas estarem situadas em pontos mais bem protegidos da rede, ou mesmo em outras redes ou sites.

O sistema CUCO realiza este controle através da manutenção de um objeto experimental na base de dados SNMP. O objeto criado é um complemento ao objeto ifAdminStatus, já existente na MIB. Este novo objeto é mais detalhado, possuindo uma conjunto de valores diferentes do existente hoje.

O objeto ifAdminStatus disponível é utilizado para descrever o estado em que se encontra uma determinada interface da rede. Seu range de valores varia entre up(1), down(2) e testing(3), o que é considerado insuficiente para se obter o nível de informação desejado.

Foi criado para tanto um novo objeto: ifStatus. Ele permite uma melhor especificação do estado de uma determinada interface. Neste novo conjunto de valores válidos para o objeto se inclui: broadcast(4), debug(5) e principalmente,

promiscuous(6). Através dele o host diretor pode avaliar se existe algo suspeito ocorrendo em determinada estação. Esta opção precisa ser desabilitada em hosts que utilizam agentes RMON ou rodam programas que executam monitorações na rede, como TCPDUMP. Caso contrário, estas máquinas sempre estarão sendo consideradas "sob ataque" pelo sistema CUCO.

A avaliação sobre o estado das interfaces é realizada por uma função denominada CheckStatus( ). Esta função usa os mesmos recursos que o comando ifconfig usa para obter uma lista das interfaces existentes em cada host, e o seu estado. É através desta função que o objeto ifStatus da MIB SNMP é mantido.

```
le0: flags=63<UP,BROADCAST,NOTRAILERS,RUNNING>
        inet 143.54.1.20 netmask ffffff00 broadcast 143.54.1.0
lo0: flags=49<UP,LOOPBACK,RUNNING>
        inet 127.0.0.1 netmask ff000000
```

Figura 3.2.7: Informação sobre o estado das interfaces obtido por **ifStatus()**

3.2.2 O Sistema de Logs e o Analisador de Padrões

```
>My name is Michael Hampton, and my site was hacked two weeks ago.
>Though I watch my system logs fairly carefully, I never noticed
>the breakin, until the administrator of another local site
>informed me of the details of the breakin at his site.
>Only then did I notice that my /bin/login had been replaced,
>and only then did I whip up a "raw" utmp viewer, and notice that
>someone had done a not-so-great job of editing my wtmp file.
```

Figura 3.2.8: Trecho extraído da lista ids@uow.edu.au em 20/11/96

O trecho da mensagem acima é um dos motivos que justificam totalmente a criação deste módulo do sistema CUCO. Ele relata um típico ataque onde o administrador, apesar de cuidadoso, foi ludibriado. Este é um dos exemplos de

uso das diversas ferramentas[11] existentes para esconder traços da passagem de um invasor pelo sistema.

Apesar de serem considerados como um sistema de auditoria, o sistemas de auditoria do sistema Unix ainda são bastante deficiente. Mesmo em sistemas mais novos e tido como mais completos no que tange a auditoria de usuários, eles ainda deixam muito a desejar. Um bom *audit trail* deveria no mínimo responder as seguintes questões:

- O que foi executado no sistema
- Quando foi executado
- De onde foi originada a requisição
- Qual foi o resultado da operação (sucesso ou falha)

Mesmo sistemas como AIX e Solaris não provêem log de toda esta informação.

Uma das possíveis soluções para evitar este tipo de situação seria o uso de *wrappers*, ou seja, substituir diversos programas como login, telnetd, ftpd, rlogind e outros, de forma a gerar logs mais completos e em duplicidade. Esta estratégia é conhecida como "*security through obscurity*" [GAR91]. No entanto, o uso desta técnica foi descartada, pois para se alcançar um nível de segurança satisfatório é necessário que os logs de todas as máquinas sejam realizados em um host considerado confiável, o mesmo conceito de *bastion host*, usado nas estratégias de *firewalls*. A maior dificuldade para isto é a substituição de vários programas (*wrappers*) em uma plataforma totalmente heterogênea (SunOs, Solaris, Linux, AIX, FreeBSD), que envolve um grande esforço de programação, somado a um tráfego inaceitável circulando pelos diversos backbones da rede.

[11] Ferramentas como wedit (editor de arquivos wtmp), uedit (editor de arquivos utmp) e outras como zap e flash, podem ser encontradas facilmente em sites hackers, ou mesmo construídas sem muito trabalho por um bom programador Unix.

A solução encontrada para resolver este problema foi o uso de um processamento imediato dos logs gerados pelos diferentes sistemas. Este processamento notifica o administrador através de um alerta do sistema, duplicando assim a informação no momento seguinte a sua geração, e não simultaneamente, como proposto por programas do tipo wrappers.

A grande vantagem desta variação é a conservação de todas as aplicações que geram os logs do sistema em sua forma original, conforme distribuído pelo fabricante.

O trecho de código da figura 3.2.9 demonstra uma das configurações utilizadas durante os testes de configuração com o sistema de logs. Neste caso é usado uma estratégia de log remoto. Este tema será retomado nos capítulos subsequentes.

Um outro problema é o sistema de contabilização mantido pelo programa /bin/login, ou seja, os arquivos /var/adm/wtmp /etc/utmp e /var/log/lastlog. Estes logs são até mais visados que os próprios logs gerados pela aplicação usando a facilidade syslog. Estes arquivos não são redirecionáveis como os que usam a aplicação syslog.

Os arquivos utmp e wtmp costumam manter um controle de quem entrou e saiu do sistema. Em Unix padrão Berkeley estes arquivo contém:

- O nome do terminal
- O nome do usuário
- O nome ou identificação do host que originou a conexão, caso o login tenha sido realizado pela rede
- O tempo que o usuário está logado

```
#     <destination> is:
#     /filename - log to this file
#     username[,username2...] - write to user(s)
#     @hostname - send to syslogd on this machine
#     * - send to all logged in users
#
# Configurado por Leandro Bertholdo em out/96
# Caso necessite alteracao me notifique com antecedencia

auth.crit;auth.emerg;auth.err;auth.alert;auth.notice;auth.warn;auth.i
nfo;auth.debug                      @circulo

daemon.crit;daemon.emerg;daemon.err;daemon.alert;daemon.notice;daemon
.warn;daemon.info;daemon.debug      @circulo

kern.crit;kern.emerg;kern.err;kern.alert;kern.notice;kern.warn;kern.i
nfo;kern.debug                      @circulo

mail.crit;mail.emerg;mail.err;mail.alert;mail.notice;mail.warn;mail.i
nfo;mail.debug                      @circulo
user.crit;user.emerg;user.err;user.alert;user.notice;user.warn;user.i
nfo;user.debug                      @circulo
#log do CUCO
local5.info                         /var/log/cucolog
#log do tacacs
local6.info                         /var/log/userlog
#log do roteador e servidor de comunicacao
local7.debug                        /var/log/nmslog
```

Figura 3.2.9: Configuração do sistema de logs utilizando redirecionamento para um *LogHost***,** no caso a estação “circulo”

Em Unix do tipo System V, os arquivos são um pouco diferentes, e cada entrada contém:

- O nome do usuário
- O número da linha do terminal
- O nome do device
- O número identificador do processo
- Um código para a entrada (estado da conexão)
- O código de exit da conexão
- A hora em que a entrada foi criada

Programas usados para reportar quem está logado no sistema (w, who, users, finger), realizam consultas ao /etc/utmp. O programa **last** que reporta os horários em que cada usuário esteve logado usa o arquivo wtmp.

```
/home/penta/berthold/tmp> last | head
berthold  ftp      porta2.tche.br   Thu Nov 28 03:08 - 03:09  (00:01)
berthold  ftp      porta2.tche.br   Thu Nov 28 02:31 - 02:33  (00:01)
berthold  ftp      circulo          Thu Nov 28 02:31 - 02:31  (00:00)
berthold  ttyp4    porta2.tche.br   Thu Nov 28 01:44   still logged
in
berthold  ttyp5    porta2.tche.br   Thu Nov 28 01:26   still logged
in
lauren    ttyp5    noc.tche.br      Thu Nov 28 01:15 - 01:24  (00:08)
lisianeh  ftp      tigrao.ufrgs.br  Thu Nov 28 01:08 - 01:09  (00:01)
ftp       ftp      200.238.55.35    Thu Nov 28 01:04 - 01:04  (00:00)
ftp       ftp      200.238.55.35    Thu Nov 28 01:03 - 01:04  (00:00)
berthold  ttyp4    porta2.tche.br   Thu Nov 28 01:01 - 01:42  (00:41)
```

Figura 3.2.10: Saída do comando last

```
/home/penta/berthold/tmp> w
3:07am  up 6 days,  6:54,  3 users,  load average: 0.19, 0.20, 0.01
User     tty       login@  idle   JCPU   PCPU  what
guga     ttyp1    Fri 9am 6days                -
lisianeh ttyp2    11:23pm     1     21      1  -csh
berthold ttyp4     1:44am    27      3         -csh
berthold ttyp5     1:26am     1     36      1  w
guga     ttyp7     2:11pm 10:18                -
```

Figura 3.2.11: Saída do comando w

Programas como wedit, uedit e zap, costumam ser executados em background, e removem automaticamente as entradas dos arquivos utmp, wtmp e lastlog que se assemelham ao padrão (username) determinado. Eles geralmente usam alguma coisa como as funções lseek() e bzero(), que geram estes registros vazios.

```
(...)
      printf("user[%8.8s] line[%8.8s] host[%-16.16s]\n",
            utmp.ut_name,utmp.ut_line,utmp.ut_host);

          if ( tty ? (!strcmp(utmp.ut_line, argv[1]))
                   : (!strncmp(utmp.ut_name,argv[1],8)) )
          {
             printf("\nVai zerar este\n");
             lseek( fd, -sizeof(struct utmp), SEEK_CUR );
             bzero( &utmp, sizeof(struct utmp) );
             write( fd, &utmp, sizeof(struct utmp) );
          }
          size = read( fd, &utmp, sizeof(struct utmp) );
(...)
```

Figura 3.2.12: Trecho de um programa que implementa um editor para zerar os registros do arquivo /etc/utmp

Sob condições normais, o sistema CUCO consegue detectar a presença destes registros vazios, podendo assim alertar a utilização desta artimanha. Abaixo estão registradas as saídas dos programas **wcheck** e **ucheck**, criadas exatamente para garantir a integridade destes arquivos de auditoria. As entradas com o comentário "<NULL?>", foram resultado da exclusão do usuário "teste", utilizando uma implementação do programa **wedit.c**. Consultas realizadas a estes arquivos por aplicações tradicionais (last, por exemplo) não são capazes de notar tais diferenças

```
user[lisianeh] line[   ttyp0] host[ tigrao.ufrgs.br]
user[        ] line[   ttyp0] host[                ]
user[lisianeh] line[   ttyp0] host[ tigrao.ufrgs.br]
user[lisianeh] line[ ftp9302] host[ tigrao.ufrgs.br]
user[        ] line[   ttyp0] host[                ]
user[        ] line[ ftp9302] host[                ]
user[lisianeh] line[   ttyp0] host[ tigrao.ufrgs.br]
user[     ftp] line[ ftp9522] host[mithrandir.ibase]
user[     ftp] line[ ftp9565] host[    200.17.81.71]
user[     ftp] line[ ftp9568] host[mithrandir.ibase]
user[        ] line[        ] host[                ] <NULL?>
user[        ] line[        ] host[                ] <NULL?>
user[ aluno96] line[   ttyp2] host[minuano.inf.ufrg]
user[        ] line[   ttyp2] host[                ]
user[        ] line[   ttyp0] host[                ]
user[lisianeh] line[   ttyp0] host[ tigrao.ufrgs.br]
user[     ftp] line[ftp10007] host[  200.255.253.23]
user[        ] line[ftp10007] host[                ]
user[lisianeh] line[   ttyp2] host[ tigrao.ufrgs.br]
user[        ] line[   ttyp0] host[                ]
user[lisianeh] line[ftp10128] host[ tigrao.ufrgs.br]
user[        ] line[ftp10128] host[                ]
user[berthold] line[   ttyp0] host[  porta2.tche.br]
user[     ftp] line[ftp10472] host[   200.238.55.35]
user[berthold] line[   ttyp4] host[  porta2.tche.br]
user[        ] line[ftp10472] host[                ]
user[     ftp] line[ftp10491] host[   200.238.55.35]
```

```
user[     ftp] line[ftp10497] host[   200.238.55.35]
user[        ] line[ftp10491] host[                ]
user[        ] line[        ] host[                ] <NULL?>
user[lisianeh] line[ftp10522] host[ tigrao.ufrgs.br]
user[        ] line[ftp10522] host[                ]
user[  lauren] line[   ttyp5] host[     noc.tche.br]
user[berthold] line[   ttyp5] host[  porta2.tche.br]
user[        ] line[   ttyp4] host[                ]
user[berthold] line[   ttyp4] host[  porta2.tche.br]
user[berthold] line[ftp10809] host[         circulo]
user[        ] line[ftp10809] host[                ]
user[berthold] line[ftp10810] host[  porta2.tche.br]
user[        ] line[ftp10810] host[                ]
```

Figura 3.2.13: Saída do programa wcheck para SunOs 4.1.3

Abaixo podemos verificar como o usuário “teste” foi retirado do sistema, não aparecendo assim em aplicações comuns como **w** ou **who**. Este tipo de modificação também só pode ser reconhecido por uma implementação como a do **ucheck**. As entradas dos processos do usuário teste foram removidas.

```
user[        ] line[ console] host[                ]
user[        ] line[        ] host[                ]
user[        ] line[        ] host[                ]
user[        ] line[   ttyp0] host[                ]
user[    guga] line[   ttyp1] host[                ]
user[lisianeh] line[   ttyp2] host[tigrao.ufrgs.br ]
user[        ] line[   ttyp3] host[                ]
user[berthold] line[   ttyp4] host[porta2.tche.br  ]
user[berthold] line[   ttyp5] host[porta2.tche.br  ]
user[        ] line[   ttyp6] host[                ]
user[    guga] line[   ttyp7] host[                ]
user[        ] line[   ttyp8] host[                ]
user[        ] line[   ttyp9] host[                ]
```

Figura 3.2.14: Saída parcial do programa ucheck para SunOs 4.1.3

3.2.3 O Submódulo de Resposta a Ataques

Este submódulo é usado opcionalmente pelo sistema CUCO. Para que ele seja utilizado, o módulo agente precisa, necessariamente, estar executando com id 0 (root), e assim, estar sujeito a vulnerabilidades e explorações típicas de aplicações que possuem este poder, como o sendmail.

Apesar do código dos componentes do sistema terem sido construídos observando sempre a necessidade de uma codificação a prova de falhas: utilizando funções não buferizadas; paths consistentes nas chamadas ao sistema; tratamento cuidadoso no uso de vetores; consistência das entradas de dados; etc., não há como garantir que o sistema esteja isento de falhas, e que de nenhuma forma alguém conseguiria se beneficiar dele para obter controle das máquinas onde estão executando os programas agentes.

Dentre as vantagens obtidas com o seu uso estão uma maior flexibilidade no controle das máquinas, e um menor tempo de resposta no caso do sistema estar sofrendo um ataque. As aplicações disponibilizadas ao administrador da rede são as seguintes:

- retirar um usuário, com todos os seus processos, do sistema;
- impedir o acesso de um determinado host ao sistema;
- ativar a monitoração de conexões TCP e datagramas IP;

e, futuramente:

- monitorar o shell do usuário, de forma a descobrir os comando que ele está digitando;
- monitorar o terminal (tty) do usuário, para além de descobrir quais comandos ele está digitando. Tomando assim, conhecimento das resposta que ele obtém em cada execução;

O primeiro dos itens, refere-se a um pequeno programa shell (quickoffuser.sh) residente no host monitorado, que pesquisa por um determinado usuário do sistema, matando todos os seus processo. Este usuário é informado ao programa agente, configurando o objeto **cuco.ação.acuser** da MIB experimental com o identificador do usuário, conhecido via cópia do arquivo de passwords mantido no host diretor. Caso o administrador tenha optado por não manter este arquivo de configuração, a operação não será realizada.

O segundo dos itens refere-se a sistemas que utilizam pacotes de software como o TCP Wrapper. Neste momento o usuário é questionado sobre o nome do host (IP) que terá todos os seus serviços bloqueados. O processo agente é informado via objeto **cuco.ação.acHost** da MIB experimental com o endereço IP do host a ter seu acesso bloqueado. Neste instante uma aplicação no host monitor (quickoffhost.sh) realiza a alteração nos arquivos de configuração necessários, tanto no host monitor quanto no diretor, calculando novamente os seus MD5 e atualizando-os na base de dados de configurações daquele host.

Por último , existe também a possibilidade de se ativar um log TCP/IP a partir da interface do host remoto, ou seja, obter-se todos os datagramas destinados a um determinado a um determinado host monitor. A vantagem desta opção é a de poder monitorar mais cuidadosamente determinados datagramas, tentando assim identificar novos possíveis ataques. Este log é acionado a partir da troca de valores do objeto **cuco.ação.acAtivadump**, de 0 para 1. A partir deste momento são realizadas monitorações na rede via programa TcpDump, e enviadas ao host diretor utilizando originalmente a facilidade local5[12] do sistema de log. O nome do host destino é obtido via objeto **cuco.sistema.hostDiretor** da MIB experimental. Ele contém o endereço IP da máquina que “teoricamente” está agindo como gerente.

[12] Sistemas UNIX prevêem facilidades de log para aplicações criadas por usuários que variam entre local1 e local7, vide arquivo de configurações da figura 3.2.8. Esta entrada pode ser alterada no arquivo de configurações do sistema.

3.2.4 O Analisador de Logs

Os logs de atividades do sistema são em geral informações preciosas para qualquer administrador que saiba e deseje fazer bom uso delas. O módulo de análise de dados envolve um conjunto de programas responsáveis pela geração e manuseio das informações providas pelo sistema operacional e aplicações, tais como: httpd, telnetd, ftpd, logind, rlogind, rshd, tcpd e xtacacsd e outros que o administrador deseje ter sua saída analisada.

Este subsistema baseia-se na procura de mensagens padrões do sistema, como os da figura abaixo. Também são tratadas muitas outras mensagens que venham a revelar indícios de tentativas de subverter um determinado host, desde que não sejam consideradas pelo administrador como rotineiras. É interessante salientar que o sistema notificará todas as mensagens não consideradas como inofensivas pelo administrador. Este módulo pode ser utilizado executando localmente na máquina, independentemente do módulo de gerenciamento.

```
Aug 12 06:03:14 telnetd[12345]: refused connect from 138.8.56.7
Aug 12 06:04:32 telnetd[12345]: refused connect from 138.8.56.7
Aug 12 06:08:48 telnetd[12345]: refused connect from 138.8.56.7
```

Figura 3.2.15: Logs gerados pelo TCPWrapper

Este submódulo foi projetado para automaticamente executar e verificar os arquivos gerados pelo sistema de logs, localizando violações ou atividades não usuais dirigidas contra o sistema. O analisador de logs utiliza para isso uma base de dados de regras de filtragem para os logs gerados pelo sistema operacional e aplicações.

O código da aplicação é subdividida em dois programas, que agem de forma idêntica: um que possibilita uma análise geral dos logs do sistema, gerando assim um relatório sobre todas as “infrações” que foram registradas nas diferentes aplicações; e outro responsável pela análise simultânea dos logs gerados em cada host monitorado. Além destes dois programas foi também incorporado ao sistema um analisador especialmente para logs gerados para o servidor tacacs (autenticador de

senhas para o servidor de comunicações). O analisador criado a partir dos logs gerados pelo servidor xtacacsd é de grande valia: ajuda a localizar os responsáveis por ataques advindos das linhas discadas; registrar possíveis tentativas de quebras de senhas utilizando algoritmos de força bruta; localiza tentativas de utilização o servidor de comunicações como trampolim para realizar ataques contra outros sites (telnet a partir do prompt do roteador).

Username	Data	Hora	Tty	Slip	Acumulado
CPVDF	21/Aug	00:17.51	-----	-----	0:01
CPVDF	15/Sep	17:25.15	-----	0:08	0:09
CPVDF	18/Sep	23:47.44	-----	0:01	0:11
CPVDF	18/Sep	23:49.59	-----	0:01	0:12
CPVDF	21/Sep	14:24.21	-----	0:10	0:22
CPVDF	24/Sep	00:27.23	-----	0:12	0:35
CPVDF	24/Sep	19:48.06	-----	0:02	0:39
CPVDF	24/Sep	19:56.53	-----	0:02	0:41
CPVDF	25/Sep	00:22.17	-----	-----	0:43
CPVDF	25/Sep	00:25.03	-----	-----	0:44
CPVDF	25/Sep	00:32.06	0:05	0:01	0:51

Figura 3.2.16: Saída gerada pelo analisador tacacs

```
REJECT: [Sep  3 19:01:47 penta tacacsd[138]: xlogin query from
200.19.246.24 TTY21 for cleber rejected]
REJECT: [Sep  3 19:01:52 penta tacacsd[138]: xlogin query from
200.19.246.24 TTY21 for cleber rejected]
REJECT: [Sep  3 19:02:12 penta tacacsd[138]: xlogin query from
200.19.246.24 TTY21 for cleber rejected]
REJECT: [Sep  3 20:50:22 penta tacacsd[138]: xlogin query from
200.19.246.17 TTY2 for proplan rejected]
```

Figura 3.2.17 Warnings gerados pelo analisador tacacs

A base de dados de regras é formada basicamente por três arquivos: um para identificar o que é considerado uma violação pelo sistema (log.violação), outro para identificar possíveis avisos importantes registrados pelo sistema, como reboots, daemons reestartados, etc. (log.warning), e ainda outro que especifica tudo o que deve ser ignorado pelo sistema (log.ignore).

No arquivo log.violação estão registrados padrões que certificam mais de 90% de chances de uma tentativa de hacking do sistema, como pode ser verificado no trecho do arquivo abaixo.

```
wiz
WIZ
debug
DEBUG
UUDECODE
VRFY decode
VRFY uudecode
VRFY lp
VRFY demo
VRFY guest
EXPN root
passwd
inetd.conf
```

Figura 3.2.18: Trecho do arquivo log.violação

Dentre algumas tentativas de violações que foram encontradas em vários meses de análise de logs, podemos citar: bugs antigos do sendmail como WIZARD, DEBUG, UUDECODE; sondagens realizadas no sistema a procura de contas de usuários como decode, uudecode, lp, guest e demo; tentativas de obter maiores informações sobre quem é responsável pelo sistema; referências a arquivos de configuração do sistema (geralmente obtidos via ftp, incluindo anônimo).

No arquivo de log.warning são especificadas quaisquer entradas de log que merecem uma atenção especial do administrador do sistema. Esta foi a idéia principal que gerou este submódulo: reduzir a quantidade de log a ser inspecionada pelo administrador.

Aqui são registradas tentativas de acesso realizadas ao sistema, pessoas que usaram a conta de root da máquina, se a usaram a partir da console ou via terminal, modificações realizadas nos aliases do sendmail, múltiplos erros ao tentar logar no sistema, reboots e shutdowns no sistema, etc., além de todos os outros já englobados pelas regras do log.violação.

Dependendo da configuração do sistema (utilizando ou não programas que gerem logs diferenciados), costuma-se registrar também a origem dos acessos . Criando assim uma lista indicativa das origens dos acessos e dos serviços prestados (telnet, ftp, ...). Este tipo de informação mostrou-se bastante útil em máquinas que possuiam uma política de acesso mais restrita.

```
deny
deny host
su:
su root
ROOT LOGIN
alias database
LOGIN FAILURE
LOGIN REFUSED
shutdown
wiz
WIZ
debug
DEBUG
failed
denied
vrfy
VRFY
```

```
expn
EXPN
reject
rshd
REFUSED
rexec
illegal
ILLEGAL
-ERR Password
!=
SITE EXEC
RETR group
RETR passwd
RETR pwd.db
CWD etc
```

Figura 3.2.19: O arquivo log.warning

No arquivo log.ignore estão inseridos todas as palavras que identificam entradas sem maior importância registradas pelo syslog. Conforme a configuração das facilities do arquivo syslog.conf utilizada, este arquivo pode vir a tornar-se extenso, registrando todas as informações possíveis. O objetivo deste procedimento é impedir que nenhuma mensagem até então desconhecida passe em branco. Esta ação é necessária para que se identifique possíveis violações realizadas contra aplicações como ftp, sendmail e serviço de nomes (DNS).

```
cron.*CMD
cron.*RELOAD
cron.*STARTUP
ftpd.*ANONYMOUS FTP LOGIN
ftpd.*FTP LOGIN FROM
ftpd.*retrieved
ftpd.*stored
mail.local
named.*Lame delegation
```

```
named.*Response from
named.*answer queries
named.*points to a CNAME
named.*reloading
named.*starting
sendmail.*User Unknown
sendmail.*User Unknown
sendmail.*alias database.*rebuilt
sendmail.*aliases.*longest
sendmail.*from=
sendmail.*lost input channel
sendmail.*message-id=
sendmail.*putoutmsg
sendmail.*return to sender
sendmail.*return to sender
sendmail.*stat=
sendmail.*timeout waiting
telnetd.*ttloop:  peer died
```

Figura 3.2.20: Trecho do arquivo log.ignore

Os arquivos de regras são formados por expressões regulares, similares às utilizadas por programas em PERL, portanto os asteriscos utilizados correspondem a repetição de qualquer padrão, e não realmente a um símbolo “*”.

Os programas que processam os arquivos de log trabalham de duas maneiras diferentes; um processa todo o arquivo de uma só vez, gerando diferentes arquivos de relatórios, outro distribuído entre os diferentes hosts monitores, processa os logs por demanda. Os dois usam a mesma estratégia de funcionamento, isto é, a pesquisa de strings entre as diversas linhas dos arquivos de log, e entregando seu parecer final ao administrador do sistema.

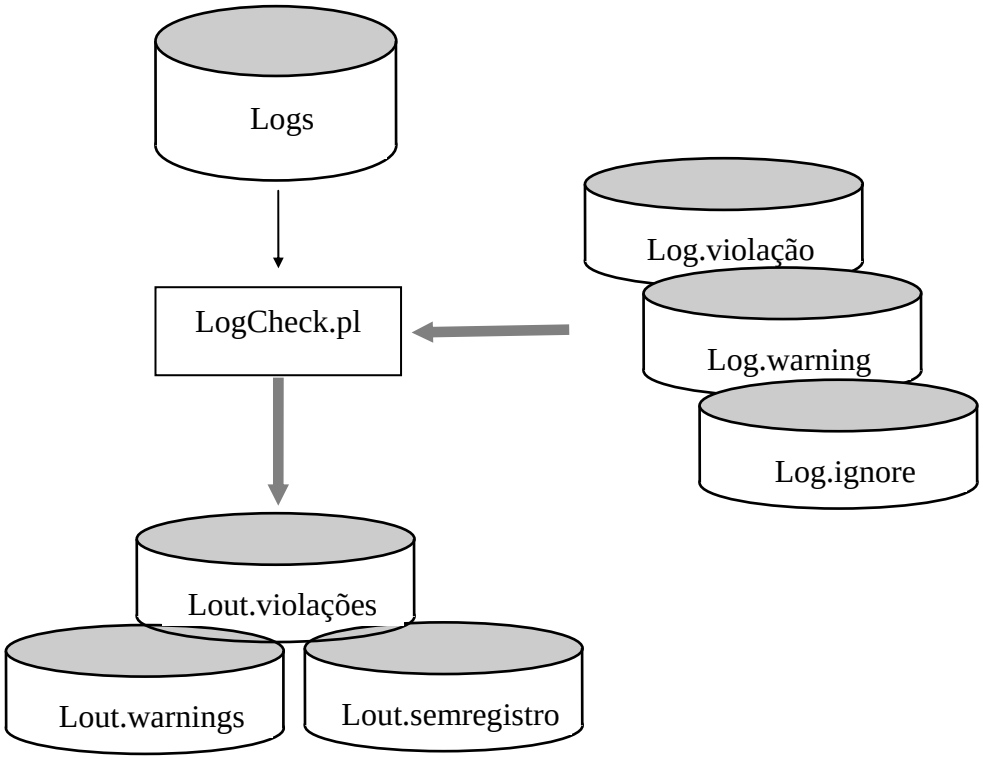

Figura 3.2.21: A estrutura do analisador de logs

O primeiro deles (logcheck.pl - figura acima), processa os logs do sistema de uma forma completa, ou seja, todos os dados que forem encontrados nos arquivos de logs configurados (diretório /var/log), são inspecionados. Como resultado o programa logcheck.pl gera três relatórios a respeito de suas conclusões:

- Lout.violações: Todas as tentativas de violação, segundo as regras configuradas em log.violações, ou seja, sabidamente consideradas como um ataque. A existência deste arquivo após o processamento dos logs indica que algum host monitor, ou o próprio host diretor tem ficado, ou está, sob alguma forma de ataque.
- Lout.warning: Todas as informações de log relevantes que o administrador deve inspecionar diariamente ou em períodos regulares curtos.
- Lout.semregistro: Relato de entradas desconhecidas ou inesperadas nos logs do sistema. No tempo de adaptação do sistema cuco à sua rede este tipo de mensagens são comuns, necessitando ser revisado

freqüentemente, para encaixar tais padrões em algum dos arquivos de regras existentes.

Em sua segunda forma (AnLog.pl), o analisador de logs é configurado no host monitor utilizando uma chamada ao sistema (tail -f $logfile) para monitorar em tempo real as ocorrências repassadas pelo daemon syslogd. Esta forma de obter os logs foi escolhida pela sua simplicidade e portabilidade, em relações a outros testes realizados durante a criação deste protótipo (inicialmente foram propostas mudanças em sistemas como syslog, e o próprio syslogd[13]). A desvantagem do procedimento adotado é a vulnerabilidade à qual o sistema fica exposto no caso de um ataque dirigido ao daemon syslogd.

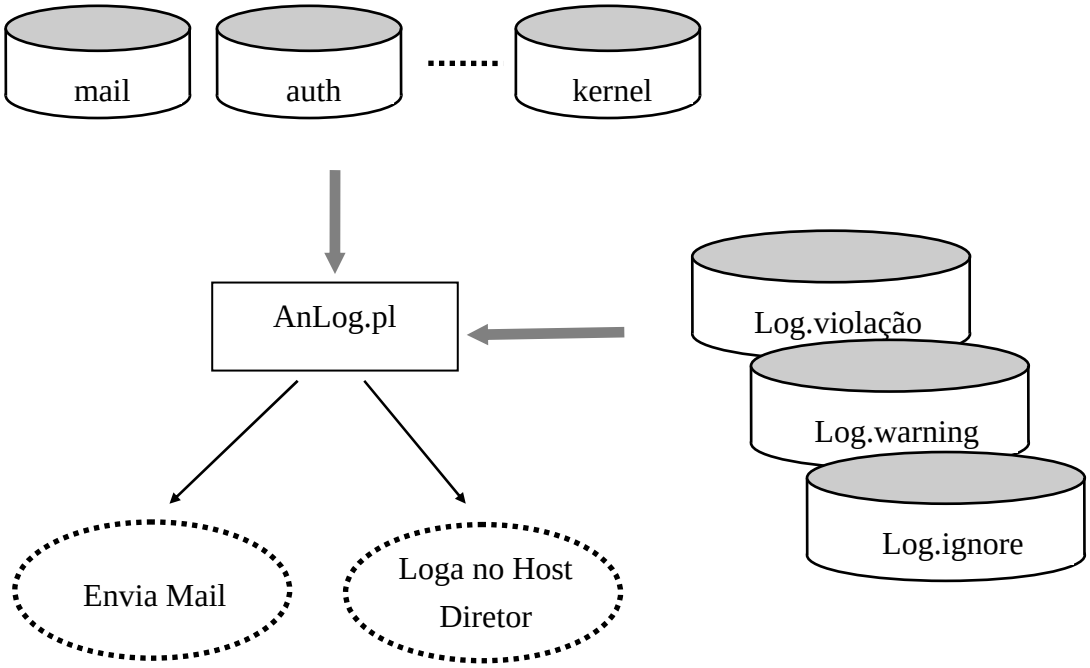

Figura 3.2.22: O analisador de logs no host monitor

[13] Inicialmente foram implementadas mudanças no sistema de logs (syslog e syslogd) em uma máquina com sistema operacional FreeBSD. O resultado obtido foi desconsiderado devido às dificuldades de portabilidade (ausência de documentação adequada) para estações com sistema operacional AIX4.1.4 que fazia parte do ambiente de testes.

O analisador de logs no host monitor possui uma forma diferente de registrar os problemas relatados ao host diretor. A forma mais comumente usada hoje em dia seria a de centralizar todos os logs gerados por todas as máquina, o que geraria um tráfego dispensável na rede, além de superlotar o sistema de arquivos do host diretor e exigir um esforço computacional multiplicado várias vezes. Em solução a isto, optou-se pela criação do programa AnLog.pl executando no host monitor e logando, via o próprio syslogd do host diretor, somente as entradas que julgue necessárias (regras do arquivo log.warning). Uma segunda opção do sistema é a de dar um tratamento especial para os casos que combinem com as regras descritas no log.violação: enviar um mail para o administrador responsável por aquele host monitor.

```
From root Fri Jan 10 14:40:26 1997
Received: from penta.ufrgs.br (penta.ufrgs.br [143.54.1.20]) by
circulo.pop-rs.rnp.br (8.7.5/8.7.3) with SMTP id OAA16072 for
<berthold@circulo.pop-rs.rnp.br>; Fri, 10 Jan 1997 14:40:25 -0300
From: cuco@penta.ufrgs.br
Received: from circulo.pop-rs.rnp.br (circulo.pop-rs.rnp.br
[200.132.0.21]) by penta.ufrgs.br (051895/8.6.11) with SMTP id
OAA14330 for berthold@circulo.pop-rs.rnp.br; Fri, 10 Jan 1997
14:40:55 -0300
Date: Fri, 10 Jan 1997 14:40:55 -0300
Message-Id: <199701101740.OAA14330@penta.ufrgs.br>
To: berthold
Subject: CUCO::ALERTA

*****************************
mail: Nov 22 16:28:04 penta.ufrgs.br sendmail[5911]: "wiz" command
from minuano.inf.ufrgs.br (143.54.7.18)
*****************************
```

Figura 3.2.23: Exemplo de mail para o administrador

3.3 O Módulo Diretor ou Gerenciador

O módulo gerente por sua vez é composto por três partes básicas:

1. A interface do sistema, toda baseada em hiperdocumentos, usando um navegador bem conhecido como o Netscape; um servidor para o protocolo HTTP [FIE97] que implementa algumas diretivas básicas do protocolo; um conjunto de programas geradores de código misto HTML [SAM96] [RAG97] e JAVA scripts [FLA96].
2. Um gerente SNMPv2, criado para negociar com os vários agentes remotos. A sua implementação utiliza o pacote CMU.
3. Conjunto de programas C [SCH89], scripts PERL [WAL96] e scripts shell capazes de: analisar e obter informações sobre uma possível fonte de ataque; verificar o funcionamento ou não dos hosts sobre monitoração; analisar dados gerados pelo sistema.

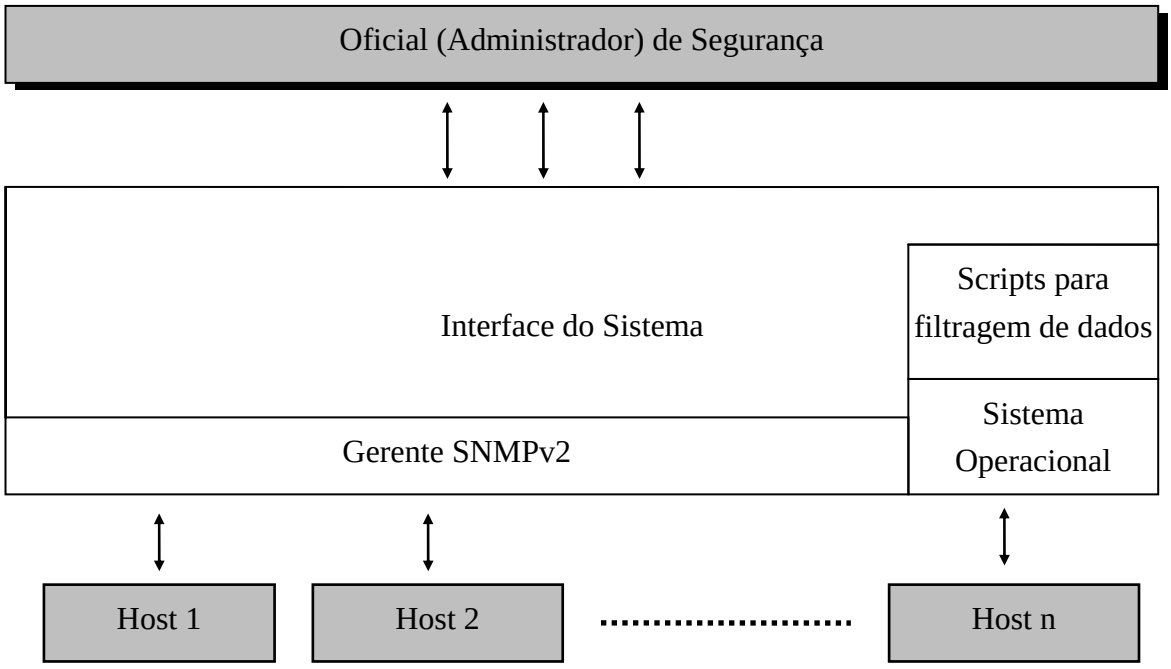

Figura 3.3.1: A estrutura do módulo diretor

3.3.1 A Interface do Sistema

Sendo parte do objetivo do sistema CUCO a sua utilização em uma variedades de máquinas situadas no campus da universidade, a portabilidade da aplicação para as diferentes plataformas existentes é um ponto fundamental. Durante a concepção do sistema, a portabilidade e conseqüente simplicidade foi sempre levada em consideração[14]. Em face disso, optou-se por construir a interface do sistema utilizando uma estratégia que está se consagrando dia a dia: uma interface utilizando navegadores (browsers) multimídia.

A interface possui scripts JAVA [FLA96] embebidos na linguagem HTML, permitindo desta forma acessar os dados gerados pelos diferentes programas de aplicação e pelo gerente SNMP.

O sistema é inicialmente disparado por programa chamado cuco. A partir deste programa, o processo inicial é dividido via chamada ao sistema (fork()) em dois: um responsável pela interface direta com o usuário, realizada através do browser Netscape ou Mosaic e outro responsável pelo controle do servidor HTTP. Os dois processos compartilham um mesmo número de porta de comunicação gerado anteriormente ao fork de processos. Desta forma a comunicação entre o cliente e o servidor HTTP é mantida (figura 3.3.2).

[14] Sempre foi levado em consideração princípios básicos de segurança na construção do código da aplicação, princípios como KISS (*Keep it Simple Stupid)*

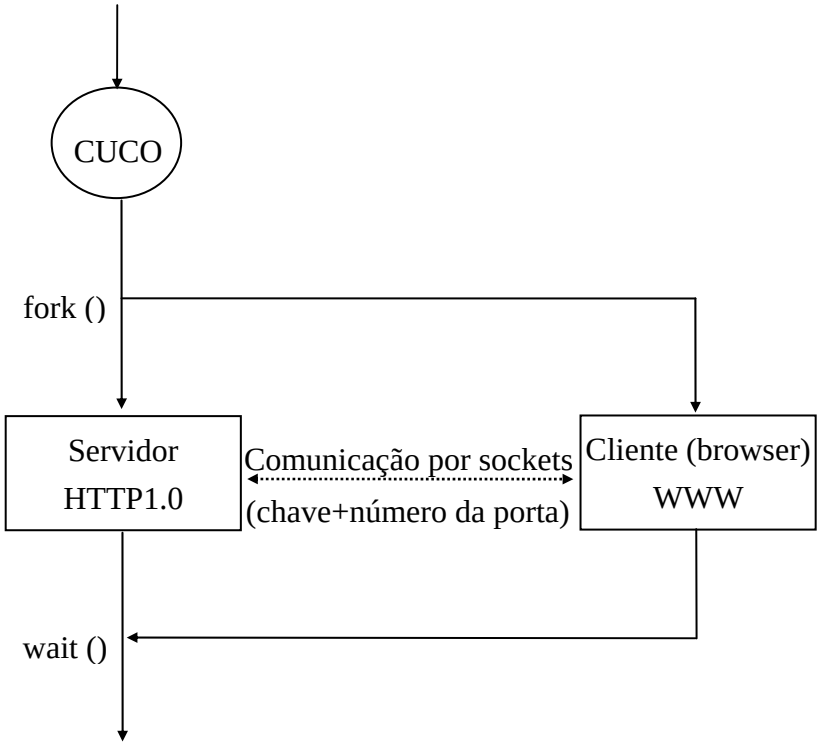

Figura 3.3.2: A comunicação servidor-browser.

Um nível de segurança básico é mantido através da geração de uma chave de 32 bytes exigida em cada passo da comunicação entre cliente e servidor. Esta senha é um conjunto suficientemente aleatório de caracteres, gerados a partir de um “ps axl&ps -el&netstat -na&netstat -s&ls -lLRt /dev*&w 2>/dev/null | $MD5” realizado por uma chamada ao sistema no momento que a interface é disparada. Tanto cliente como servidor tomam conhecimento desta password via compartilhamento de memória entre processos. Todo este procedimento é necessário para adequar as necessidades de exibição das informações geradas dinamicamente pelo sistema, à forma de comunicação pela qual a ferramenta de exibição (netscape) recebe as informações que ela deve exibir. No anexo A-2 podemos verificar um dos programas scripts responsáveis pela geração das informações exibidas pelo sistema CUCO, o resultado da execução do script inicial responsável pela homepage do sistema pode ser observado na figura 3.3.3.

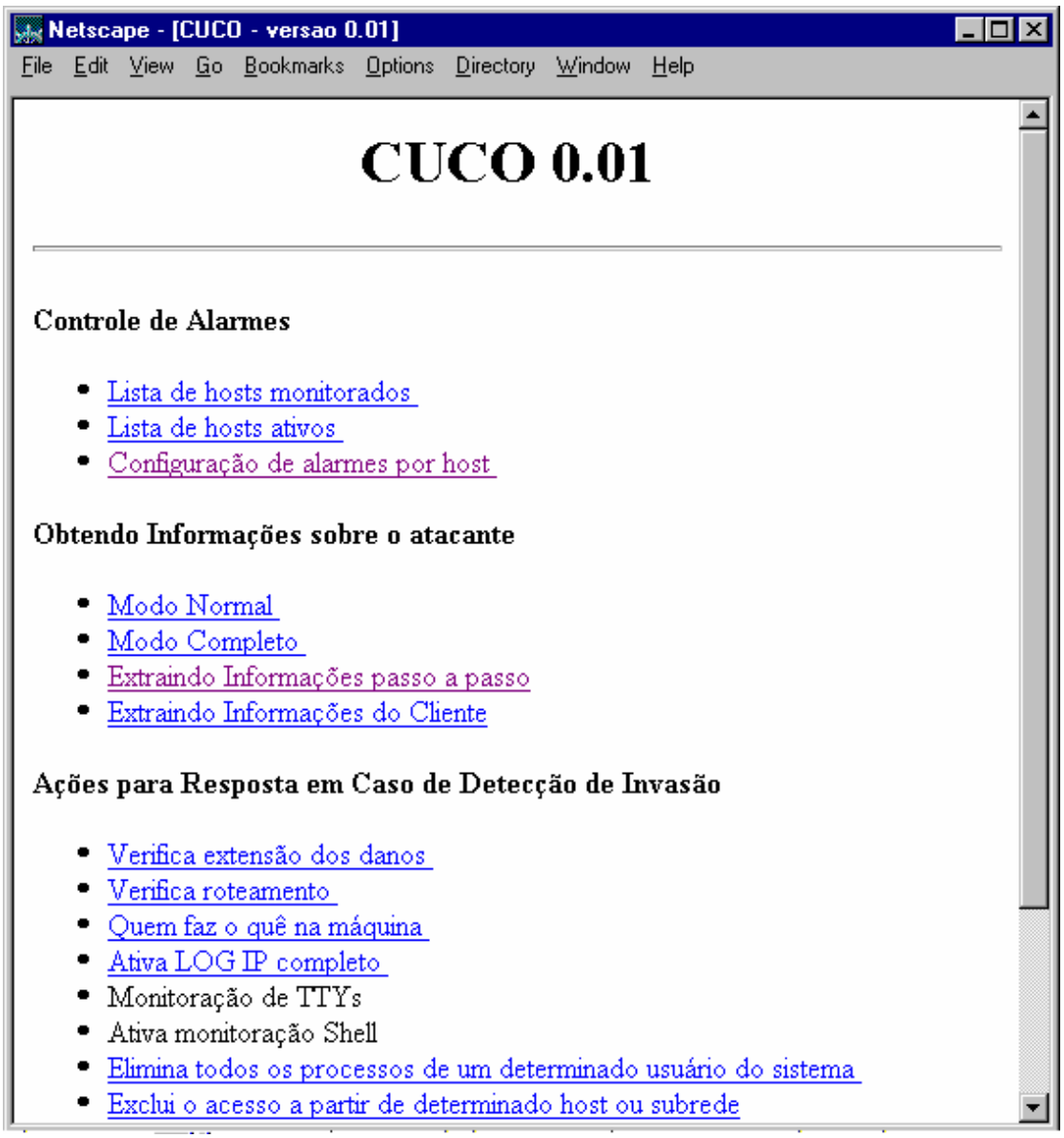

Figura 3.3.3: A interface do sistema CUCO

O programa que serve a interface com o usuário final mantém via diversos scripts e programas C. Ao serem executados, estes programas realizam a comunicação via HTML com o browser do usuário. Este interfaceamento é conhecido como CGI (Common Gateway Interface), e faz parte das diversas implementações de servidores WWW como o NCSA e CERN (figura 3.3.4). A idéia de um programa CGI é possibilitar ao cliente executar diversas operações, como atualizações em bancos de dados e obtenção de informações locais, além de várias outras tarefas que necessitem de execução na máquina servidora. Infelizmente, não são poucos os casos

onde a possibilidade de execução de comandos no servidor foi explorada visando obter informações que possibilitassem a invasão da máquina servidora[15]. Este, juntamente com a necessidade de modularidade e portabilidade de instalação da parte do sistema residente no host diretor, foram decisivos para a criação do servidor HTTP proprietário, ao invés de exigi-lo como requisito para instalação do CUCO. Sem dúvida, a necessidade de diminuir-se as chances de um ataque direto, explorando falhas de implementação nos módulos CGIs, foi quem contribuiu decisivamente pela opção de um servidor mais compacto e com maiores recursos de segurança.

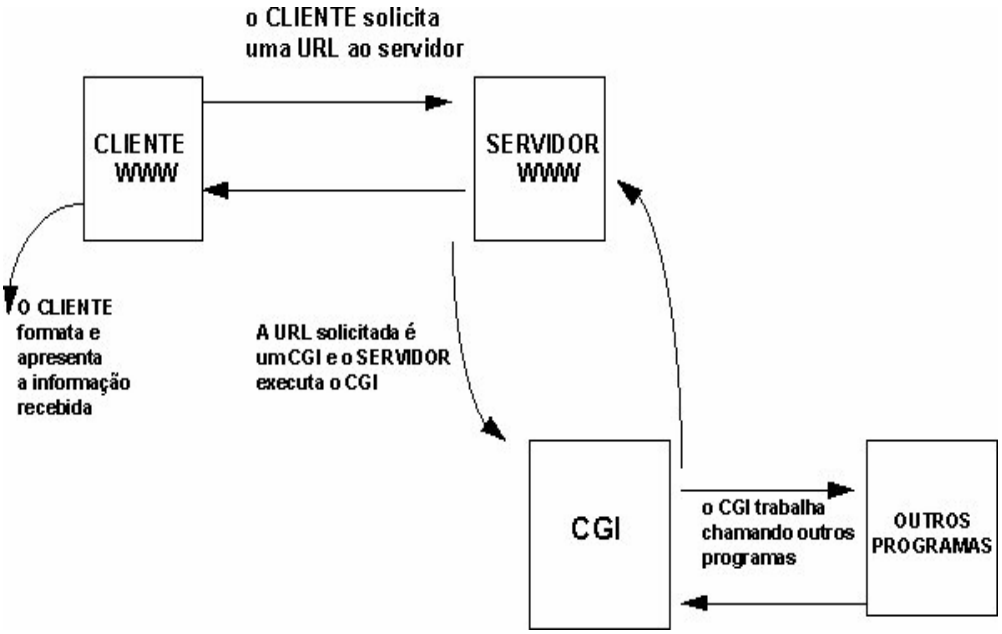

Figura 3.3.4: A descrição da interface cliente-servidor (CGI) padrão

A figura a seguir (figura 3.3.5) é o exemplo da utilização da interface CGI. Ela mostra o resultado da execução do programa testahosts.pl (anexo A-3), usando para verificar se os hosts que possuem o software monitor estão ativos ou não.

[15] O exemplo mais conhecido disto foi o bug do programa phf.pl, distribuído em conjunto com o servidor WWW da NCSA, que possibilitava a qualquer pessoa obter arquivos do sistema como o próprio /etc/passwd.

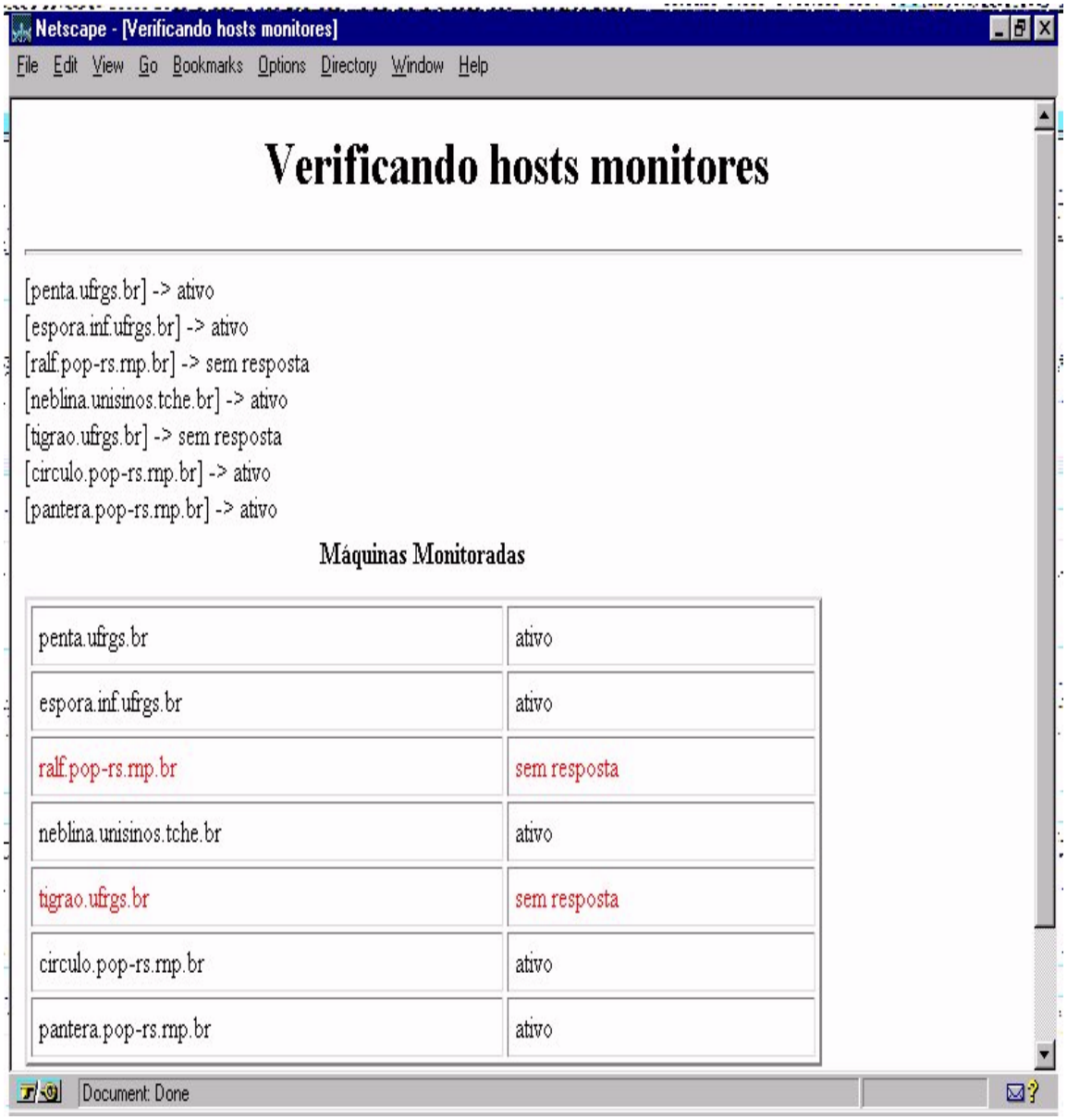

Figura 3.3.5: Consulta sobre a situação dos hosts

3.3.2 Localizando a Fonte do ataque

Um dos desafios para detecção de uma intrusão em um ambiente de rede é sem dúvida o de seguir as pistas deixadas pelo agressor, sejam elas baseadas em uma máquina ou em uma pessoa como originador do ataque, ou mesmo um

arquivo deixado como prova; lembrando sempre que esta origem pode se mover por diversos sites da Internet durante o transcorrer de um ataque (ataque sincronizado). Por exemplo, um invasor pode usar diferentes contas de usuários em diferentes máquinas no transcorrer de um único ataque, visando com isso dissuadir o administrador da rede, dissimulando o seu verdadeiro objetivo e ocultando a verdadeira fonte do ataque.

O sistema CUCO possui um módulo exclusivo para detecção do agressor. Este subsistema coleta informações de várias fontes independentes. Inclusive o próprio estado da rede e o seu comportamento no momento do ataque são levados em consideração. Todas estas informações são colocadas em um arquivo e posteriormente exibidas ao administrador encarregado da segurança da rede. A identificação de um possível agressor pode ser realizadas pelo responsável pela segurança da rede de duas formas diferentes, segundo o seu critério:

- Um modo **normal**;
- Um modo **agressivo**

Estes modos serão mostrados com maiores detalhes a seguir.

Cabe salientar que o modo normal é sempre usado pelo sistema, e cabe ao administrador, se achar necessário disparar, por suas conseqüências a segunda instância de investigação (modo agressivo).

3.3.2.1 Modo Normal

Neste modo de operação são realizadas buscas padrão para descoberta e investigação de um usuário em um determinado host. Entenda-se padrão como aquelas normalmente realizadas por um administrador de rede em sua tentativa de obter conhecimento sobre um determinado usuário ou máquina da Internet.

Quando se esta sofrendo um ataque, a grande maioria dos ataques sofridos resulta na obtenção de um único IP ou uma lista deles. Uma vez identificado

o ataque (via subsistema de monitoração), a primeira informação importante para o administrador do sistema é a validação de todos os usuários do sistema, sendo necessário para isto o conhecimento da origem de cada um. Como sistema não é capaz de identificar o uso de quaisquer brechas de segurança como sendmail, nem de apontar o usuário responsável por isto com um grau de confiança aceitável, a única opção do sistema é pesquisar todas as origens de conexões que estão sendo realizadas no momento para a máquina, gerando uma lista de todos os "suspeitos" e a origem de sua conexão. Qualquer conexão existente no momento do registro do ataque é considerada suspeita.

O processamento deste subsistema é realizado através de múltiplos programas PERL disparados simultaneamente após a geração da lista de endereços IP. A primeira implementação do sistema leva em conta somente conexões do tipo TELNET, e é gerada por uma consulta ao comando Who do sistema operacional Unix, que gera uma saída no seguinte formato:

```
berthold    pts/0       11 Nov 18:20    (200.19.246.20)
fest        pts/2       11 Nov 18:30    (penta.ufrgs.br)
```

Figura 3.3.6: Saída do comando *Who*

Nas próximas versões do sistema, a lista de conexões será fornecida pelo comando netstat, possibilitando desta forma a investigação da origem de todas as conexões, com as diferentes portas TCP que estão sendo consultadas. Levando em conta o custo da análise de todas as conexões (WWW, Ftp, Telnet, Gopher, etc.), chegou-se a conclusão que isto só seria válido se o sistema pudesse reconhecer em tempo real a ocorrência de uma invasão, o que por si só envolve a construção de um sistema especialista, que não é o objetivo do sistema CUCO.

Como dito anteriormente, o modo normal realiza uma identificação usando ferramentas para obtenção de informações consideradas tradicionais, ou seja, após obter uma lista de números IP ou nomes e domínios o sistema tentará identificar as pessoas que estão utilizando estes recursos, visando desta forma localizar contas de usuários que estejam sendo usadas para hackear outros sistemas. Por exemplo, alguém a partir de bobao.ufrgs.br conseguiu acesso a uma das máquinas protegidas pelo

sistema, entrou em desconfiado.ufrgs.br utilizando um conta válida, e rodou um programa que permite a ela acesso como administrador, neste momento é alterado um arquivo de configuração que permite acesso direto a esta máquina sem ter de passar por bobao.ufrgs.br. Dentro de instantes o sistema reconhece esta mudança, notifica o administrador e tenta automaticamente localizar quem está em bobao.ufrgs.br, possivelmente uma conta comprometida por um agressor externo. A partir deste momento os dois administradores tentam entrar em acordo para localizar a real origem do ataque.

O sistema realiza em paralelo as seguintes consultas, ao sistema operacional e ao sistema operacional de rede, através de diferentes módulos que processam as respostas obtidas e a entregam um relatório ao administrador. São eles:

- dns.pl: realiza consultas ao Domain Name Server local sobre um endereço IP e seu reverso. Também realiza uma consulta ao dns remoto a procura dos NS (Name Servers) responsáveis pelo domínio em questão. Ao final gera uma lista completa sobre as informações do IP pesquisado e dos NSs citados por ela. Neste ponto São obtidas informações como MX records e registros INFO se houver.
- finger.pl: realiza uma pesquisa ao daemon fingerd, visando obter uma lista de quem esta usando o sistema em determinado instante. Em um segundo passo tenta obter maiores informações sobre cada um dos usuários, ou seja, arquivos como .plan e .project.
- ftp.pl: Tenta obter maiores informações sobre o site em questão, caso as consultas ao DNS sejam insuficientes.
- rusers.pl: Outra forma de obter informações sobre usuários
- rwho.pl: Ainda outra forma de obter uma lista de usuários.
- sendmail.pl: Tenta obter via telnet para a porta 25 da máquina destino e através da diretiva “EXPN root”, o endereço de mail de alguns dos responsáveis pelo sistema.

```
Trying 200.132.0.21 ...
Connected to circulo.
Escape character is '^]'.
220 circulo.pop-rs.rnp.br ESMTP Sendmail 8.7.5/8.7.3; Mon, 11 Nov
1996 19:06:34 -0300
EXPN root
250 <berthold@circulo.pop-rs.rnp.br>
221 circulo.pop-rs.rnp.br closing connection
```

Figura 3.3.7: Obtendo informações através do sendmail

- snmp.pl: Tenta obter informações básicas sobre quem administra o sistema, e qual é o tipo do sistema operacional da máquina.
- telnet.pl: Caso não tenha tido nenhuma informação via DNS tenta obter maiores informações sobre o host analisando o banner de login do sistema.
- traceroute.pl: Verifica a rota entre o destino e a origem da conexão e vice-versa. Esta informação é utilizada com duplo objetivo: Identificar a existência ou não de rotas assimétricas entre os dois pontos. Caso a origem seja um PC conectado via linha discada, que sabidamente não responderá a nenhuma consulta realizada, o sistema vai disparar novamente todas as rotinas para obtenção de informações contra o host imediatamente anterior na rota dos pacotes IP. Observe a figura 3.3.8, sendo (4) a origem da conexão, (3) e (2) que possivelmente são roteadores ou servidores de comunicação, também serão inspecionados

```
1  rsc701.pop-rs.rnp.br (200.132.0.17)  2 ms  3 ms  2 ms
2  tchepoa (200.132.0.20)  2 ms  3 ms  3 ms
3  200.19.246.17 (200.19.246.17)  3 ms  4 ms  3 ms
4  200.19.246.20 (200.19.246.20)  775 ms  1741 ms  740 ms
```

Figura 3.3.8: Escolhendo um novo host provedor de informações

- whois.pl: Realiza uma consulta a base de dados WHOIS, a partir de nic.ddn.mil, rs.internic.net e whois.fapesp.br, pré processando suas respostas e realizando novas consultas caso sejam indicadas na resposta dos servidores. Aqui são utilizados os comando whois e rwhois.
- whodo.c: Gera uma listagem de todos os processos que estão sendo executados na máquina onde ocorreu/está ocorrendo a invasão, ordenados por usuários e terminal (tty) onde estão executando. Esta informação será extremamente útil no caso da necessidade de uma monitoração do terminal do usuário. Esta monitoração pode ser realizada localmente na máquina atacada utilizando o software ttywatcher. Esta capacidade de monitoração por tty está prevista para ser incorporada em futuras versões do sistema CUCO. Desta forma será possivel realizada uma monitoração remota, via a própria interface WWW do sistema. Estes programa faz parte do módulo monitor, que gera informação para a MIB experimental.

3.3.2.2 Modo Agressivo

Quando se usa este modo de operação, deve-se levar em conta a possibilidade do agressor estar usando um PC com aplicações do tipo winsock[16] e estar ligado em uma rede local (acesso típico a partir de universidades). Deve-se também contar com a possibilidade dele esconder-se atrás de uma firewall (um acesso

[16] Este tipo de aplicação implementa uma pilha de protocolos TCP/IP em máquinas Intel (PCs)

a partir de uma empresa), e mais comumente nos dias de hoje, de estar usando uma linha discada que na maioria das vezes não possui um DNS reverso (situação típica de acessos via provedores comerciais brasileiros). Todas estas situações dificultam o simples reconhecimento do domínio envolvido.

Este modo de operação, mais completo que o anterior, procura descobrir a qualquer custo a origem de uma determinada conexão, além de possibilitar um reconhecimento das máquinas situadas nos arredores do host agressor. Levando em conta as diferentes respostas que o sistema pode ter obtido a partir dos métodos anteriores, o sistema pode recomendar ao administrador que tente obter maiores informações sobre o domínio em questão através das diversas máquinas que se encontram entre ele e a máquina agressora. Isto possibilita realizar investigações sobre os endereços obtidos pelo programa **traceroute.pl**, bastando simplesmente uma seleção do mouse pelo administrador, para que um novo host seja investigado.

Entretanto, deve-se ter sempre em mente que a pesquisa é realizada nas diferentes máquinas que se situam no caminho entre a origem e o destino desta conexão. Portanto, esta “pesquisa”, pode ser considerada como politicamente incorreta pelos outros administradores envolvidos, já que são obtidas informações sobre o sistema operacional, pessoal responsável, aplicações executando, etc., em todas as máquinas envolvidas direta ou indiretamente. Este procedimento estará notadamente registrando no site consultado o host originador das consultas, no caso a máquina executando o gerente do sistema CUCO.

Este módulo pretende obter toda e qualquer informação, relevante ou não, que puder obter sobre a fonte do ataque. Ele deve ser evitado e somente usado como uma última alternativa para obter informações sobre um possível agressor. Ou seja, quando o administrador do domínio de onde partiu a agressão não se mostre cooperativo aos diversos mails relatando o ataque. Esta é uma situação comum na Internet dos dias de hoje, tanto no Brasil como no exterior.

Este nível de avaliação utiliza vários programas para obter maiores informações sobre o domínio fonte do ataque. Ele foi criado usando como base três funções do código do programa SATAN, TCP_SCAN, UDP_SCAN e FPING, sendo

que a última delas, o FPING, havia sido já é uma reutilização de código usada pelos próprios autores do programa SATAN.

O software CUCO gera uma listas de possíveis endereços a serem verificados ao redor dos domínios do hosts agressor, baseando-se nas informações obtidas pelos programas **dns.pl** e **snmp.pl**, onde devem ter sido obtidas informações sobre as máquinas que compõem o domínio, tal como servidores de nomes e de correio eletrônico e informações como a máscara de rede (via snmp). A partir destes dados o sistema CUCO gera uma lista de IP válidos, e a repassa para o programa **fping**, responsável por validar a atividade desta lista de IPs, verificando assim quais hosts respondem a ping. A nova lista gerado pelo **fping** é utilizada para se obter uma lista de hosts ativos. A diferença fundamental entre o ping e o **fping** é que o segundo recebe como parâmetro uma lista de IP, paralelizando a tarefa do primeiro, aumentando desta maneira o desempenho do sistema.

Após a obtenção e validação de uma lista de hosts, o subsistema de descoberta de informações do sistema tentará obter dados sobre quais aplicações estão sendo executadas na máquina alvo. São analisadas todas as portas de um determinado host a partir das ferramentas: TCP_SCAN e UDP_SCAN. Estas ferramentas procuram por todos os serviços disponibilizados por um host da rede. Através de um item de configuração ($intensidade_do_ataque), existe como especificar quais serviços (número de portas) serão vasculhados. Isto visa aumentar o desempenho do sistema, e, dependendo da configuração, gerar o mínimo de alarmes no sistema que está sendo inspecionado. Todo o processo de verificação das portas é realizado em paralelo.

Abaixo está um trecho do arquivo /etc/services, que mostra uma listagem das portas padronizadas no sistema unix.

```
bootps          67/udp                  # bootp server port
bootpc          68/udp                  # bootp client port
tftp            69/udp
gopher          70/tcp                  # Gopher
gopher          70/udp                  # Gopher
```

```
rje             77/tcp          netrjs
vettcp          78/tcp                  # vettcp
vettcp          78/udp                  # vettcp
finger          79/tcp
www             80/tcp                  # World Wide Web HTTP
www             80/udp                  # World Wide Web HTTP
hosts2-ns       81/tcp                  # HOSTS2 Name Server
hosts2-ns       81/udp                  # HOSTS2 Name Server
```

Figura 3.3.9: Um exemplo do arquivo /etc/services

O programa TCP_SCAN é usado para determinar a disponibilidade dos serviços TCP em um dado host. Ele pode também ser usado para obter banners de aplicações, podendo desta forma obter maiores informações sobre um determinado site, que se utilize por exemplo de enviar mensagens via aplicações como TCPWrapper.

```
/home/minuano/berthold> telnet 200.132.0.21
Trying 200.132.0.21 ...
Connected to 200.132.0.21.
Escape character is '^]'.
^G
        Esta maquina possui acesso restrito ao pessoal do
        Ponto de Presenca do Rio Grande do Sul!

Todas as suas atividades estao sendo monitoradas, e o seu acesso a
este  site, explicita a concordância com estes termos. Toda e
qualquer tentativa de abusar desta ou outras maquinas deste dominio
serao  consideradas  de  ma  fe, e  seu autor sujeito as penas
cabiveis da lei.
^G
Connection closed by foreign host.
/home/minuano/berthold>
```

Figura 3.3.10: Exemplo de Banner que pode ser obtido

Uma vez estabelecida a conexão com o HOST:PORTA destino, o programa tenta obter durante um intervalo de tempo preestabelecido o banner da aplicação que está instalada naquele número de porta. Após a recepção deste banner ou a obtenção de um time-out da porta, o TCP_SCAN envia um ‘QUIT\r\n’ e encerra a conexão. Todas as informações obtidas são armazenadas na base de dados do programa CUCO, para serem inspecionadas pelo administrador

A pesquisa através das portas UDP é um pouco mais trabalhosa, isto devido ao protocolo UDP não possuir nenhum tipo de controle de congestionamento ou retransmissão de pacotes. o programa precisa estabelecer um controle mínimo sobre estes itens, já que podem ser gerados aproximadamente 2800 pacotes UDP para as diferentes portas de cada host a ser analisado, e, cujas respostas são dependentes das velocidades de processamento dos hosts e do segmento de rede entre eles. Para este controle o programa se baseia em uma janela de transmissão que tem sua abertura diminuída quando forem constatadas as primeiras retransmissões realizadas.

O sistema CUCO gera, a princípio, uma analise das portas UDP entre 53 e 177, uma faixa que concentra os serviços mais considerados mais relevantes, como as portas para NETBIOS(137,138,139), CMIP(163,164), SNMP(161,162) e xdmcp(177). Após o envio de um pacote do tipo ‘0\r\n’ para HOST:PORTA destino, o programa aguarda a resposta na tentativa de obter mais informações sobre aquela porta. No caso do UDP_SCAN podem haver várias retentativas até que se esgote o tempo máximo de espera pela resposta. Este tempo está inicialmente configurado para 60 segundos, mas pode ser alterado via arquivos de configuração do sistema CUCO. Todas as informações obtidas a respeito das portas UDP são meramente informacionais, somente sendo armazenadas na base de dados para serem posteriormente exibidas pelo subsistema de apresentação de dados.

3.4 O Protocolo de Comunicação

O protocolo escolhido para implementar a comunicação e gerenciamento de informações foi SNMPv2 [STA93], por prover um nível aceitável

de confiabilidade e segurança. O protocolo SNMPv1[ROS91] não será utilizado, embora largamente difundido em um esquema de gerenciamento de rede, por possuir falhas inerentes de segurança. Falhas como a incapacidade para autenticar a fonte da mensagem bem como de prevenir a sua interceptação.

Mesmo incorporando algumas características do S-SNMP, SNMPv2 ainda possui algumas imperfeições intrínsecas ao protocolo sobre o qual trabalha, no caso UDP. Entretanto, considerando-se as necessidades de segurança na comunicação entre o agente e o gerente do sistema CUCO, e, apesar dos problemas que ainda possui, SNMPv2 se adapta bem ao problema e consegue certificar:

- Integridade dos dados: Elimina problemas como inserção, duplicação e ressequenciamento de mensagens através de um esquema de sincronismo e *time-stamp*.
- Autenticação da origem: Provê confirmação da fonte de uma mensagem usando um esquema de chave pública.
- Confidencialidade dos dados: Provê confidencialidade através de criptografia usando um algoritmo simétrico (DES).

Dentre os diversos modos de operação do SNMPv2, optou-se na implementação pelo uso do modo "privado e autenticado", que supre todas as características acima. As diretivas que definem o modo de comunicação entre os módulos podem ser vistas na figura abaixo.

```
authProtocol      snmpv2Md5Auth
privProtocol      desPriv
```

Figura 3.4.1: Diretivas utilizadas no arquivo party.conf

Na figura 3.4.2 podemos ver a diferença que resulta na criação dos formatos de mensagens SNMP, enquanto variam os parâmetros que definem sua privacidade e integridade. A figura descreve informalmente a estrutura de dados usada para o transporte das mensagens SNMPv2.

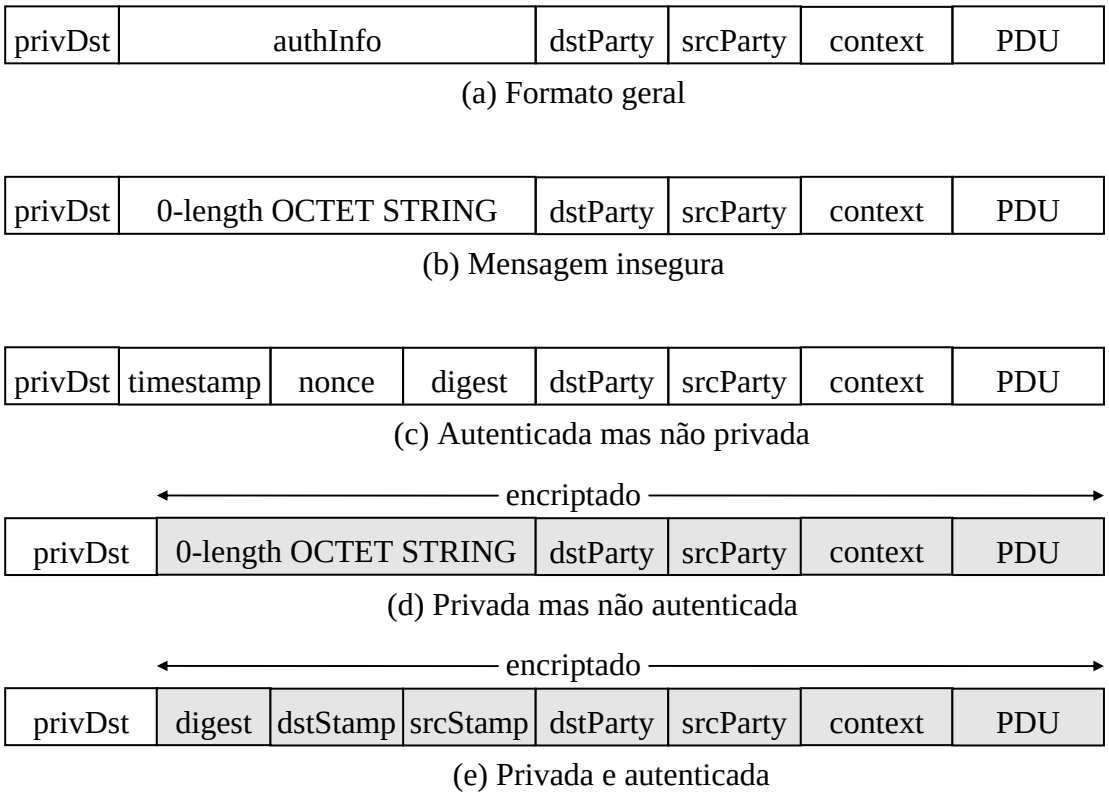

Figura 3.4.2: Formato das mensagens SNMPv2

Um segundo passo na criação do protótipo foi a definição de uma MIB experimental. Esta base de informações é mantida por vários programas que avaliam a segurança do host monitorado, conforme os vários procedimentos descritos em capítulos anteriores.

Os objetos aqui apresentados são uma proposta para criação de um grupo de objetos que servirão a um protótipo em desenvolvimento. Para tanto o grupo para o qual se insere o sistema CUCO é o “experimental”, cuja identificação é 1.3.6.1.3, ou seja, ISO.ORG.DOD.Internet.experimental (figura 3.4.3).

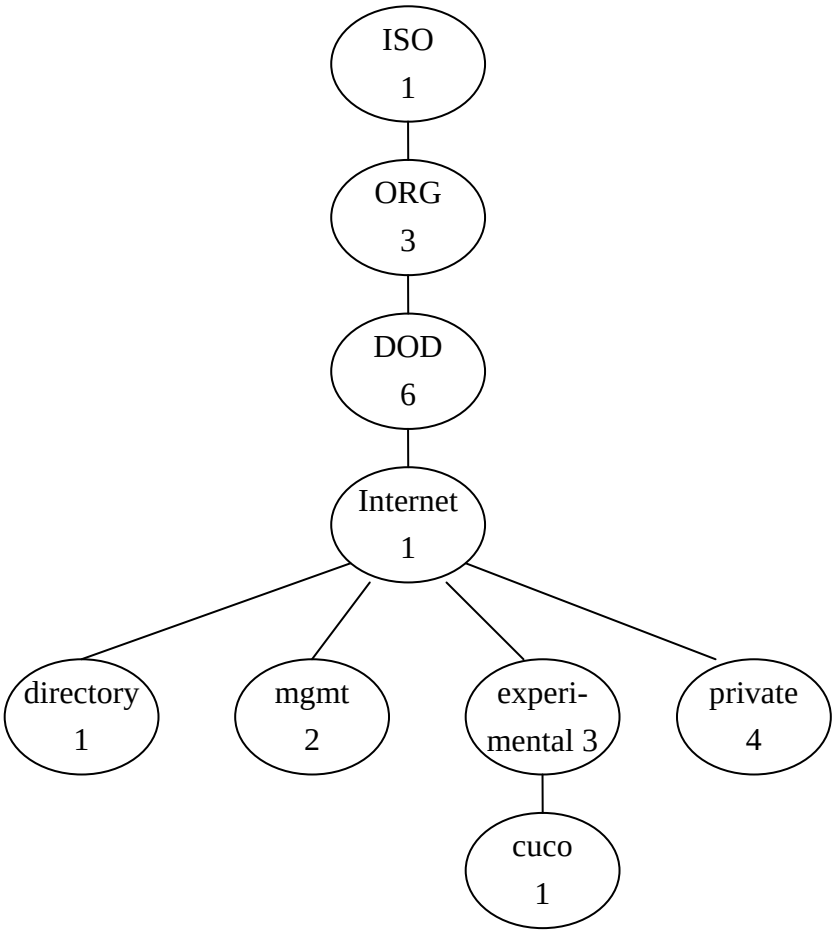

Figura 3.4.3: Hierarquia de objetos para identificar as variáveis da MIB

A seguir está um trecho da MIB SNMP que foi criada para atender às necessidades do sistema CUCO, juntamente com os subgrupos que o compõem. São eles:

- cfgmd5: Tabela que mantém informações sobre os dados de integridade (MD5) dos principais arquivos de configuração do sistema.
- binmd5: Tabela que mantém informações sobre dados de integridade (MD5) dos principais arquivos binários do sistema operacional.
- interfaces: Tabela com todas as interfaces do sistema, mantém principalmente o estado das interfaces.
- processos: Tabela utilizada para reter informações sobre os processos que estão sendo executados no host monitorado. Futuramente, técnicas de inteligência artificial serão usadas para definir uma baseline dos processos de cada host, visando com isso acusar comportamentos “estranhos” de usuários, ou do próprio administrador da máquina, horários de utilização, etc. Atualmente tem somente caráter informacional para facilitar a identificação e eliminação de problemas de segurança pelo próprio administrador.
- sistema: Informações de controle para o sistema CUCO, tal como o host diretor ao qual ele presta informações, além de outras informações pertinentes ao administrador de segurança.
- ação: Grupo de objetos utilizados para orientar o host monitorado sobre ações que deve tomar no caso de uma possível invasão, através da cláusula (SET).

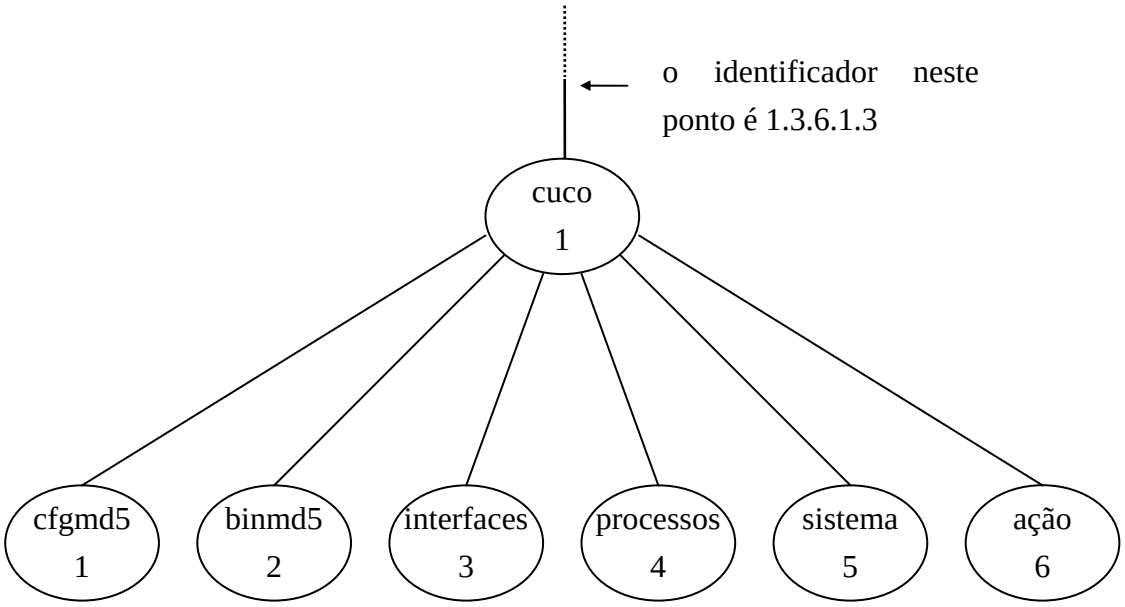

Figura 3.4.4: O grupo de objetos “cuco”

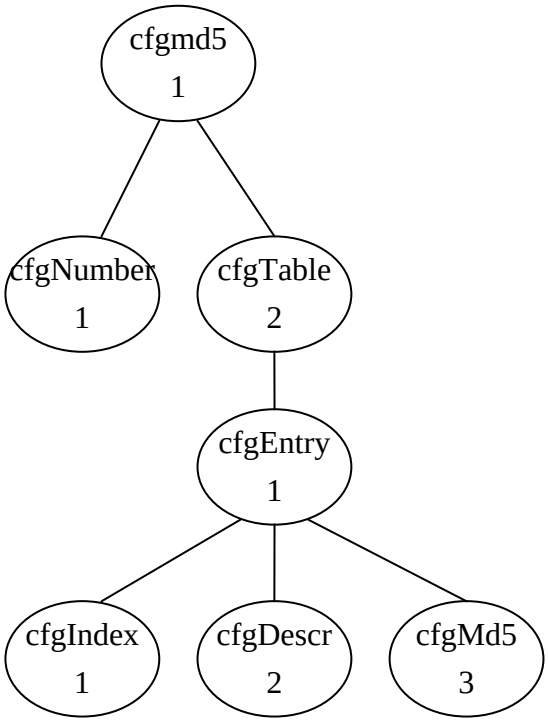

Figura 3.4.5: O grupo “cuco.cfgmd5”

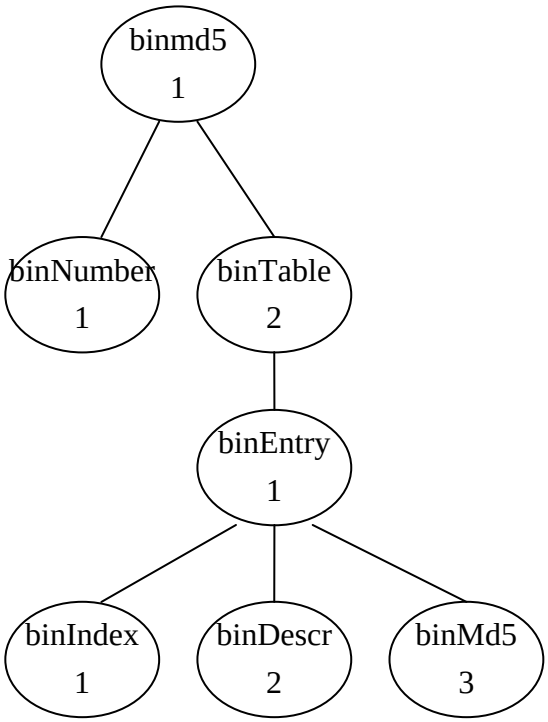

Figura 3.4.6: O grupo “cuco.binmd5”

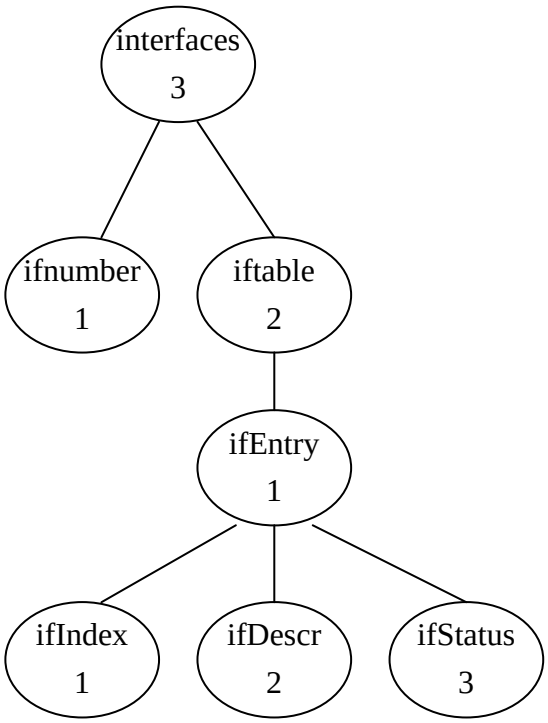

Figura 3.4.7: O grupo “cuco.interfaces”

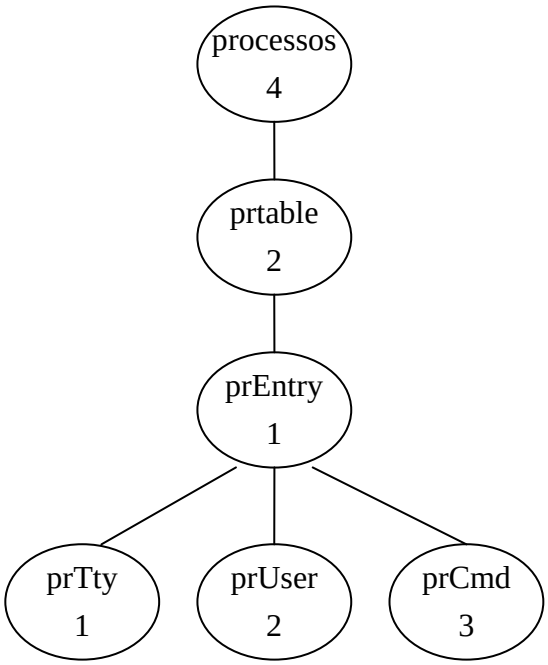

Figura 3.4.8: O grupo “cuco.processos”

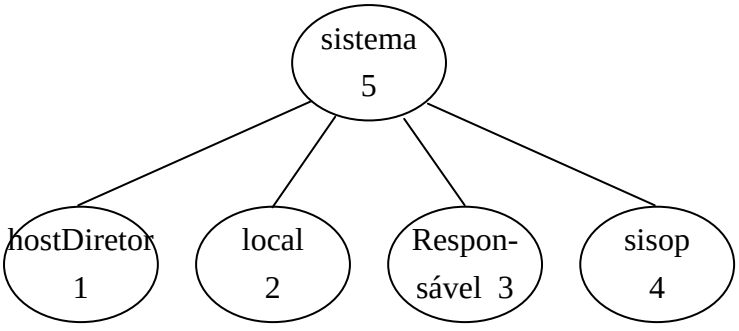

Figura 3.4.9: O grupo “cuco.processos”

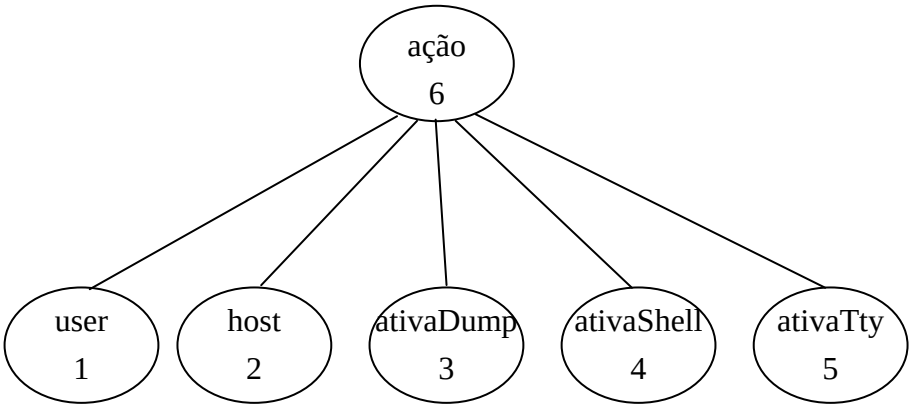

Figura 3.4.10: O grupo “cuco.ação”

A definição completa da MIB SNMPv2 usando a descrição ASN.1 pode ser encontrada no anexo A-4.

4. Análise do Sistema CUCO

Neste capítulo será realizada uma análise sobre a plataforma sobre a qual o protótipo do sistema foi desenvolvido, dificuldades de implementação e desempenho da estratégia que foi escolhida. Também serão descritas algumas falhas que o protótipo ainda possui, que podem transformá-lo em mais uma vulnerabilidade na segurança do que uma solução para os problemas hoje existentes.

O protótipo do sistema demonstrou-se uma excelente ferramenta para manter um nível aceitável de segurança no site. Ele mostrou-se eficaz no ambiente em que foi testado, ou seja, em instituições com características semelhantes a de uma universidade, onde por questões filosóficas, isolar a rede com firewalls geralmente é

inviável, assim como utilizar qualquer outra solução que se baseia em roteadores com regras de filtragem de pacotes. Nestes casos, o programa CUCO é uma maneira de aumentar o nível médio de segurança de toda a rede. Isto é de extrema importância onde existem relações de confiança [CHE94] entre as máquinas envolvidas, caso para o qual o sistema foi inicialmente projetado.

4.1 O Desenvolvimento e Implementação

O protótipo foi desenvolvido utilizando um ambiente com duas estações de trabalho, com diferentes sistemas operacionais diferentes. O ambiente heterogênio foi preterido para que se tivesse contato com as possíveis diferenças de implementações e as levasse em consideração. Esta opção levou o protótipo a se transformar em uma camada de aplicação do sistema operacional, de forma a ser transparente às diferentes implementações e comandos (como “ps”), principalmente do sistema operacional.

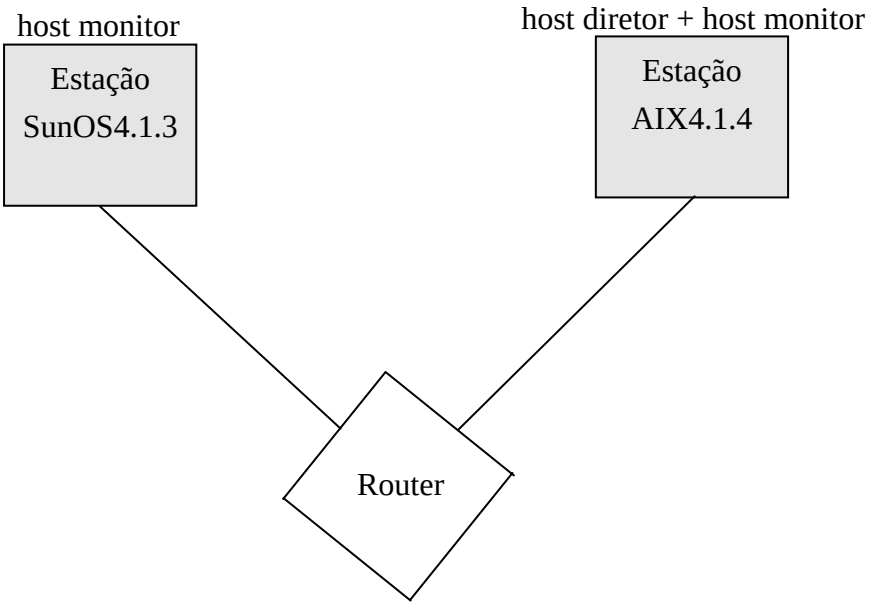

Figura 4.1: O ambiente de testes

A figura 4.1 esboça o ambiente no qual o sistema foi prototipado. As duas estações possuem um módulo host monitor que é controlado através do host diretor executando na estação AIX. Toda a interface do sistema foi inicialmente construída no sistema AIX, mas é totalmente portável para outros ambientes. Como descrito anteriormente a interface em funcionamento foi construído utilizando-se um browser netscape e PERL5.002.

No host diretor o pacote é composto pela árvore de diretórios, assim descrita:

- logs: mantém os logs do protótipo quando executando em modo debug, e arquivos de log gerados pelas diversas aplicações do sistema, como mensagens recebidas;
- bin: binários do sistema;
- config: os diversos arquivos de configuração do sistema: configuração para o tacacs, analisador de logs, paths do sistema, etc.;
- source: diversos programas de aplicação construídos em linguagem C, como ifcheck.c;

- html: abaixo deste diretório estão todos os fontes PERL que controlam a interface com o usuário: o servidor HTTP e todos os scripts geradores de documentos HTML, além de diversos scripts SHELL e PERL responsáveis por inspeções e processamento de segurança nos dados obtidos via SNMP e parte do analisador de logs;
- images: algumas imagens usadas durante a exibição dos documentos criados;
- util: aqui estão subdivididos alguns programas mais genéricos que fornecem dados à interface do usuário;
- work: diretório de trabalho onde são gerados grande parte dos documentos HTML;
- docs: diretório onde são colocados toda a documentação do sistema. Esta documentação se encontra também em formato HTML e pode ser acessada via a própria interface do sistema.
- dados: aqui são colocadas as bases de informações necessárias para o funcionamento do sistema CUCO. Md5 de diferentes sistemas operacionais e arquivos de configuração. Também estão aqui todas as cópias de arquivos relevantes das máquinas monitoradas.

A figura abaixo representa a disposição da árvore de diretórios:

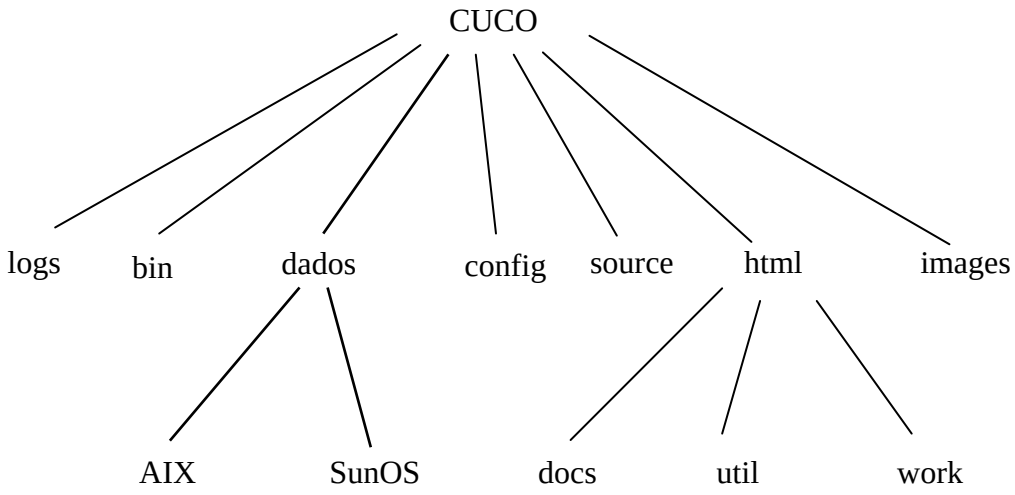

Figura 4.2: A árvore de diretórios do protótipo

Após as dificuldades na definição do sistema (MIB, interface, operacionabilidade) surgiram as dificuldades de implementação. Durante esta implementação, os principais problemas encontrados foram relativos à portabilidade do protótipo para os dois sistemas UNIX diferentes. O BSD (SunOs4.1.3) e o SystemV (AIX4.1.4). Durante o funcionamento do sistema, a dificuldade passou para o processamento das mensagens obtidas pelo sistema, as quais envolveram uma substancial pesquisa sobre mensagens de erro dos dois sistemas operacionais e dos diversos aplicativos existentes, de forma a adaptar o protótipo no ambiente de execução.

4.2 O desempenho da Estratégia Usada

Durante o tempo em que a ferramenta esteve em desenvolvimento, inúmeras tentativas de acesso indevido ou de tentativas de explorar bugs antigos em aplicações como sendmail, foram detectadas. Em simulações de ataques realizados contra o sistema, a aplicação teve no seu pior caso, apenas o registro de modificação de arquivos. Nesta simulação havia sido explorado um estouro de pilha. A aplicação

alertou o ataque quando houve a tentativa de instalar um sniffer e uma trapdoor na máquina. Isto evitou que o sistema fosse adulterado após a violação, mas não foi possível reconhecer através dos logs gerados pelo sistema, qual aplicação foi explorada.

Em outras simulações, onde o nosso arquivo de passwords foi transportado (ação comum realizada por hackers), o sistema registrou a saída deste arquivo via logs do sistema e relatou imediatamente. Isto tornou possível a localização da máquina agressora, bem como a identificação do usuário que a estava ocupando no momento, via contato com o administrador do site agressor.

4.3 Pontos Críticos do Sistema

O sistema CUCO, como qualquer aplicação que utilize a própria internet para trafegar dados sensíveis, possui suas próprias vulnerabilidades. Ou seja, vulnerabilidades que são inerentes aos protocolos e serviços sobre o qual ele trabalha. A seguir são citadas algumas destas possíveis fragilidades, ou pontos críticos do sistema.

Dentre os diversos tipos de análise das possíveis vulnerabilidades, convém citar os ataques que podem ser realizados ao protocolo de gerenciamento. Considerou-se como satisfatória a criptografia e autenticação do protocolo SNMPv2, mas deve-se estudar mais cautelosamente as possibilidades de bombardeio com requisições e pacotes, realizadas em um ataque conjunto nas duas partes do sistema (gerente / agente). Se utilizado em um ataque cooperativo com outras técnicas, o atacante pode ganhar tempo para continuar no seu intento.

Uma outra vulnerabilidade, também explorando negação de serviço pode ser disparada contra o serviço de logs da estação diretora, de forma a perder logs importantes do sistema. Isto se deve ao fato da utilização da facilidades do sistema de logs (syslogd), que opera utilizando também o protocolo UDP. Um ataque desta forma poderia provocar o descarte de mensagens do sistema operacional, ou mesmo algum nível de comunicação do sistema CUCO. Uma solução para os dois casos

citados é o uso de redundância de comunicação, utilizando por exemplo os traps SNMP. Outra melhoria a ser considerada é o uso do software TcpWrapper nas portas usadas pelo sistema CUCO.

O terceiro ponto considerado crítico pelo sistema é a sua própria interface. Por usar a interface HTTP/CGI e um browser WWW realizando sua comunicação via sockets, o sistema fica vulnerável. Podendo um agressor inclusive executar comandos na máquina diretora. A solução paliativa implementada para este problema é a geração de uma chave pseudo-aleatória de 32 bytes à la SATAN. Uma outra melhoria também seria a configuração de um filtro de pacotes como TcpWrapper e a sua instalação no pacote CUCO, para conseguir filtrar a porta dinamicamente criada pelo sistema.

A última falha que poderia ser explorada é o código da aplicação cliente. Esta vulnerabilidade existe devido a necessidade de execução de alguns programas com UID 0 (root). Estes programas são disponibilizados caso o módulo de resposta a ataques seja instalado, pois ele necessita alterar alguns arquivos de configuração do sistema. A solução para este tipo de vulnerabilidade é a não instalação deste módulo.

É de extrema importância para o sistema CUCO a segurança da máquina que estará executando o módulo diretor do sistema. Além da sua responsabilidade no controle da segurança de todo o site, ele pode estar configurado para manter cópias de arquivos sensíveis de outras máquinas. A simples leitura destes arquivos pode tornar-se uma grave falha de segurança. Este tipo de problema está sendo solucionado através da encriptação da base de dados onde estão os dados sobre os diversos sistemas operacionais e hosts monitorados.

4.4 Trabalhos Futuros

Dentre as possibilidades de expansões no uso da ferramenta, estão previstas algumas melhorias, tais quais um monitor para terminais de usuários (tty) e um monitor para aplicações como sh e csh. O custo desta melhoria é somente a de

adaptá-las a partir de ferramentas já existentes no protocolo de comunicação e de interface de usuário.

Uma outra evolução, como citado anteriormente, é o melhor uso das informações sobre os processos dos usuários. Disto compreende a formação de uma base de dados com aplicações e acessos mais freqüentemente realizados por cada usuário em cada máquina, horários de utilização e outros, e, uma aplicação inteligente para gerenciar estes dados. Através disto poderia vir a ser guardado um perfil do usuário. Quaisquer variações neste perfil poderiam ser notificadas ao administrador do site como sendo uma provável infração realizada contra o sistema. Um exemplo desta utilização são os acessos inoportunos realizados em datas não usuais: 03:00hs de 24 de dezembro, 23:00hs de 31 de dezembro, etc..

5. CONCLUSÃO

Durante o tempo de desenvolvimento desta dissertação houve a oportunidade de se acompanhar na prática os problemas relativos à questões de segurança junto a gerência de redes da Rede Tchê/RNP. Isto foi de fundamental importância, e foi o ponto de partida para a definição, desenvolvimento e aprimoração da solução prototipada. Neste período teve-se contato com diferentes tentativas de invasões, bem sucedidas ou não, que delinearam os principais objetivos propostos pelo sistema CUCO.

A experiência adquirida revelou que segurança é na prática totalmente dependente dos padrões e conhecimentos que os inúmeros administradores do site possuem sobre o assunto, sua capacidade investigativa e, principalmente, seu tempo disponível. Levando também em consideração que em ambientes heterogêneos em termos de usuários e máquinas, como são as universidades, o fator segurança se degrada exponencialmente com o tempo, torna-se impossível saber se uma determinada máquina foi ou não invadida. Ambientes como este são altamente propícios para uma ferramenta como o CUCO. Esta é a situação ideal para demonstrar a máxima eficiência da ferramenta, tendo sempre em consideração que o protótipo não exige um conhecimento uniforme sobre informações relativas à segurança entre os vários administradores. Além do mais, não exige que se perca um tempo precioso em verificações diárias a respeito da integridade, tentativas de acesso, e análise repetitivas de logs do sistema.

Neste cenário, uma ferramenta como o software CUCO, contribuí como um auxílio importante para que os administradores possam ficar mais tranqüilos quanto ao estado das suas máquinas. Estabelece-se assim um nivelamento padrão de segurança, tendo como garantia uma verificação periódica do sistema a respeito principalmente de tentativas de invasões, ou mesmo ataques já bem sucedidos por hackers. Ataques estes que no passado eram ignorados.

Tendo em vista o número crescente de incidentes relativo à segurança, e a falta de uma ferramenta que abordasse de forma completa um problema específico

como a segurança de máquinas em um ambiente sabidamente vulnerável, a estratégia de monitoração proposta é de grande valia. O sistema CUCO vem preencher uma lacuna deixada entre ferramentas como SATAN, para verificação pró-ativa de segurança e outras como firewalls e filtros, que visam solucionar estes mesmo problema através de restrições à usuários. Esta ferramenta é primeira a incorporar características como alarmes e gerenciamento remoto de segurança através de SNMPv2, além de possuir características de análise de dados e monitoração de usuários e processos. A ferramenta possui um grande potencial, podendo inclusive isolar e detectar novas formas de ataque aos protocolos e serviços hoje disponíveis via Internet.

O sistema CUCO inegavelmente aumentou a tranqüilidade com a qual problemas de segurança são tratados nas máquinas monitoradas. Ele prove uma garantia básica: o sistema operacional está intacto, tanto em seus arquivos quanto em sua configuração. As máquinas ainda estão vulneráveis a aplicações como sendmail, mas existe a garantia que mais cedo ou mais tarde as fontes dos ataques serão descobertas.

Conclui-se portanto, que o trabalho alcançou os objetivos inicialmente idealizados, e que o trabalho aqui apresentado serve definitivamente como uma fonte de informação para aqueles que desejam manter seus sites mais seguros.

BIBLIOGRAFIA

[BELa] BELLOVIN, Steven M.; **The "Session Tty" Manager.** AT&T Bell Laboratories, Murray Hill, NJ 07974.

[BEL89] BELLOVIN, Steven M.; **Security Problems in the TCP/IP Protocol Suite.** AT&T Bell Laboratories. Computer Communication Review, Vol. 19, No. 2, pp. 32-48, April 1989.

[BEL93] BELLOVIN, Steven M.. **Packets Found on an Internet.** Computer Communications Review, Vol. 23, No. 3, pg. 26-31. July 1993.

[BEL94] BELLOVIN, Steven M; CHESWICK, William R.. **Network Firewalls.** IEEE Communication Magazine, p.50-57. September 1994.

[BEL96] BELLCORE, **PINGWARE System Master Log Summary.** Http://www.belcore.com/demotoo/SECURITY/pingmaster.html, 1996

[BER95] BERTHOLDO, Leandro Márcio; TAROUCO, Liane M. R. **Relatório de Vulnerabilidades da Segurança das Redes de Computadores da UFRGS.** Relatório Interno (não publicado), dezembro, 1995.

[BER96] BERTHOLDO, Leandro Márcio; TAROUCO, Liane M. R. **Uma Análise do Software de Segurança SATAN - *Security Administrator Tool for Analysing Networks.*** Anais do Segundo Workshop sobre Administração e Integração de Sistemas (em conjunto com o SBRC'96) p. 149-160. Fortaleza-CE, 18 a 20 de maio de 1996.

[BRU96] BRUNO, Lee. **Internet Security: How Much is Enough?** Data Communications Volume 25 Número 5, pg. 60-72. April 1996

[BRY94] BRYANT, Richard Bringle. **Unix Security for the Organization: Implement a Secure Environment - and Keep It That Way.** SAMS Publishing, 1994.

[CER92] CERT/CC. **Generic Security Information.** Ftp://info.cert.org/ pub/tech_tips/

[CER93] CERT/CC. **Internet Security Scanner (ISS).** Computer Emergence Response Team Advisory 93:14, September, 1993.

[CER95a] CERT/CC. **Security Administrator Tool for Analyzing Networks (SATAN).** Computer Emergence Response Team Advisory 95:06, April, 1995.

[CER95b] CERT/CC. **Unix Computer Security Checklist.** Computer Emergence Response Team. Ftp://info.cert.org/pub/tech_tips/aucert-checklist1.1. December, 1995.

[CER96] CERT/CC. **Anonymous FTP configuration Guidelines.** April, 1996. Ftp://info.cert.org/pub/tecch_tips/anonymous_ftp_config.

[CHA95] CHAPMAN, Brent; ZWICKY, Elizabeth. **Building Internet Firewalls**. O'Reilly & Associates, 1995.

[CHE92] CHESWICK, William; **An Evening with Berferd, in Which a Cracker is Lured, Endured, and Studied.** Proceedings of the Winter USENIX Conference. San Francisco, January 1992.

[CHE94] CHESWICK, William; BELLOVIN, Steven. **Firewalls and Internet Security: Repelling the Wily Hacker**. Quarta edição. Addison Wesley Publishing Company, 1994.

[COH95] COHEN, Frederick B.; **Protection and Security on the Information Superhighway**. John Wiley & Sons, Inc., 1995.

[COM91a] COMER, Douglas E.; **Internetworking with TCP-IP. Vol I: Principles, Protocols, and Architecture - Second Edition**. Prentice Hall International, 1991.

[COM91b] COMER, Douglas E.; STEVENS, David L. **Internetworking with TCP-IP. Vol II: Design, Implementations, and Internals**. Prentice Hall International, 1991.

[COM95] COMER, Douglas E. **Internetworking with TCP/IP. Vol I: Principles, Protocols, and Architecture**. Third Edition. Prentice Hall, 1995.

[CUR90] CURRY, David A. **Improving the Security of Your Unix System.** Information and Telecommunications Sciences and Technology Division (ITSTD-721-FR-90-12) SRI International. Ftp://info.cert.org/pub/info/security-doc.tar.Z. 1990. December, 1990.

[DOD85] DoD Computer Security Center. **“Orange Book” Trusted Computer System Evaluation Critéria**. DoD 5200.28-STD, DoD, 1985. Ftp://info.cert.org/pub/info/orange-book.Z.

[FAR91] FARMER, Daniel; SPAFFORD, Eugene. **The COPS Security Checker System.** Purdue University Technical Report CSD-TR-993. Proccedings of the Usenix Conference, Anahein CA. September, 1991. Http://www.ja.net/newsfiles/janinfo/cert/Farmer_and_Spafford/cops.html

[FAR95] FARMER, Dan; VENEMA, Wietse**. Improving the Security of Your Site by Breaking Into it.** Intenet Security, march 1995, pg. 4-13.

[FIE97] FIELDING, J. Gettys; MOGUL J.; FRYSTYK, H.; BERNERS-LEE, T.. **Request For Comments 2068: Hypertext Transfer Protocol -- HTTP/1.1.** January, 1997.

[FLA96] FLANAGAN, David; **Java in a Nutshell. A Desktop Quick Reference for Java Programmers.** O’Reilly & Associates, Inc.. First Edition, February, 1996.

[GAR94] GARFINKEL, Simson; SPAFFORD Gene. **Practical Unix Security.** O’Reilly & Associates, Inc. 1991. Sétima edição 1994.

[HES] HESS, David K.; SAFFORD, David R.; POOCH, Udo W..**A Unix Network Protocol Security Study: Network Information Service.**Texas A&M University.

[HUG95] HUGHES, Larry. **Actually Useful Internet Security Techniques**. New Riders Publishing, 1995.

[ISS96] Internet Security Systems; **Identifying Network Security Vulnerabilities. A white paper for network security professionals**. Http://www.iss.net, 1996.

[KLE90] KLEIN, Daniel V.; "**Foilling the Cracker": A Survey of, and Improvements to, Password Security.** Proceedings of the USENIX Security Workshop, summer 1990.

[KOB92] KOBLAS, D.; KOBLAS, M.R.. **Socks.** USENIX - Unix Security III Symposium, Baltimore, p. 77-83. September, 1992.

[LIT96] LITTMAN, Jonathan; **The Fugitive Game: online with Kevin Mitnick. The inside story of the great cyberchase**. Little, Brown and Company, 1996.

[LIU94] LIU, Cricket; PEEK, Jeny; BUUS, Bryan, NYE, Adrian. **Managing Internet Information Services - World Wide Web, Gopher, FTP, and more.** O'Reilly & Associates, Inc.. First Edition, December 1994.

[LYO94] LYONS, Katherine; **Toll Fraud: The Billion-Dollar Heist**. Telecommunications, p.49-52, May 1994.

[MOR85] MORRIS, R. T.; **A Weakness in the 4.2BSD UNIX TCP/IP Software**. Computing Science Technical Report N. 117, AT&T Bell Laboratories, Murray Hill, New Jersey.

[MUF92] MUFFET, Alec, et al; **Almost Everything You Ever Wanted To Know About Security (but were afraid to ask).** Frequently Asked Questions in Usenet Newsgroups "comp.security.misc" and "alt.security". July, 1992..

[MUF95] MUFFETT, Alec; **WAN-hacking with AutoHack - Auditing security *behind* the firewall**. Sun Microsystems United Kingdom. The Fifth USENIX UNIX Security Symposium, Salt Lake City, Utah. June 5-7, 1995.

[MUK94] MUKHERJEE, Biswanath; HEBERLEIN, Todd L.; LEVITT, Karl N.; **Network Intrusion Detection.** IEEE Network, May/June 1994, p.26-41.

[POS81] POSTEL, J.. **Request For Comments 791: Internet Protocol - DARPA Internet Program Protocol Specification.** USC/Information Sciences Institute, September 1981.

[RAG97] RAGGETS, Dave.**W3C: HyperTxt Markup Language (HTML).** Http://www.w3.org/pub/WWW/MarkUp/. January, 1997.

[RAN92] RANUM, Marcus J.; **A Network Firewall.** In Proceedings of the World Conference on System Administration and Security, Washington, D.C..July 1992.

[RAN93] RANUM, Marcus J.; **Thinking About Firewalls.** In Proceedings of the Second International Conference on System and Network Security and Management (SANS-III). April 1993.

[RAN96] RANUM, Marcus J.; **Eletronic Commerce and Security**. V-One Corporation. Http://www.v-one.com/pubs/ecommerce/ecommerce.htm.

[REE95] REED, Daren; Fitzgerald, Tom; Traina, Paul. **Request For Comments 1858: Security Considerations - IP Fragment Filtering.** October, 1995.

[REY90] REYNOLDS, Joyce K.; POSTEL, Jon B.; **Request For Comments 1060: Assigned Numbers**, March 1990.

[ROO96] ROOTKIT - Release 1.0. **README.txt**, Ftp://penta.ufrgs.br/security

[ROS91] ROSE, Marshall. **The Simple Book - An Introduction to Management of TCP/IP-based Internets**. Prentice-Hall International Inc., 1991.

[SAM96] SAMPAIO, Cleuton. **Home Page sem traumas**. Brasport livros e multimídia Ltda. Rio de Janeiro, RJ. 1996.

[SCH89] SCHILDT, Herbert; **C Avançado - Guia do Usuário.** McGraw-Hill, Ltda, 2ª Edição, 1989.

[SCH94] SCHNEIER, Bruce; **Applied Cryptography. Protocols, Algorithms, and Source Code in C.** John Wiley & Sons, Inc. 1994.

[SMI93] SMITH, Danny; **Selected Aspects of Computer Security in Open Systems.** Australian Computer Emergence Response Team. University of Queensland, Australia. November, 1993.

[SPA88] SPAFFORD, Eugene H.; **The Internet Worm Program: An Analysis.** Departament of Computer Sciences, Purdue University. Purdue Technical Report CSD-TR-823. November, 1998. Ftp://ftp.cs.purdue.edu.

[SPA93] SPAFFORD, Eugene; KUMAR, Sandeep; **A Pattern Matching Model for Misuse Intrusion Detection.** The COAST Project, Departament of Computer Sciences, Purdue University. Ftp://cs.purdue.edu.

[STA95] STALLINGS, William. **Network and Internetwork Security - Principles and Practice.** Prentice Hall International Edition, 1995.

[SUN91a] SUN Microsystems; **SunOS Reference Manual.** Part Number 800-1751-10, May 1988.

[SUN91b] SUN Microsystems; **System and Network Administration.** Part Number 800-1733-10, May 1988

[SUN91c] SUN Microsystems; **Security Features Guide.** Part Number 800-1751-10, May 1988

[STA95a] STALLINGS, William. **Internet Security Handbook**. IDG Books Worldwide, 1995.

[STA95b] STALLINGS, William. **Network and Internetwork Security - Principles and Practice.** Prentice Hall International Editions.New Jersey, 1995.

[STE90] STEVENS, W. Richard. **Unix Network Programming.** Prentice Hall Inc., 1990.

[TAN89] TANENBAUM, Andrew S. **Redes de Computadores**. Rio de Janeiro, Ed. Campus, P. 80-84, p.210-221. 1994.

[VEN92] VENEMA, Wietse. **TCP WRAPPER: Network Monitoring, Access Control, and Booby Traps.** Mathematics and computing Science - Eindhoven University of Technology. In proceedings of the Third USENIX Security Symposium, Baltimore, Maryland. September 1992.

[VEN95] VENEMA, Wietse. S.A.T.A.N. **A Summary. Internet Security.** March 1995, pg. 3

[WAL95] WALL, Larry; **Perl Reference Manual - version 5.002beta1g.** December 1995, p. 441.

Anexo A-1 Implementando de um estouro de pilha

Esta mensagem foi extraída de uma lista sobre assuntos de segurança

```
Subject: Linux & BSD's umount exploit
Author:  Paulo Jorge Alves Oliveira <pjao@dux.isec.pt> at Internet
Date:    29/10/96 12:38


Hello,

  there is a bug in berkeley-derived umount, which allows attacker to
get
root access (see freebsd-security for details). Here is exploit for
Linux
(tested on 2.0.XX), for BSD (tested on FreeBSD 2.1) and a quick
soluction.

Best regards, Paulo

------------------- linux_umount_exploit.c ----------
#include <stdio.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>

#define PATH_MOUNT "/bin/umount"
#define BUFFER_SIZE 1024
#define DEFAULT_OFFSET 50

u_long get_esp()
```

```
{
  __asm__("movl %esp, %eax");

}

main(int argc, char **argv)
{
  u_char execshell[] =
   "\xeb\x24\x5e\x8d\x1e\x89\x5e\x0b\x33\xd2\x89\x56\x07\x89"
   "\x56\x0f\xb8\x1b\x56\x34\x12\x35\x10\x56\x34\x12\x8d\x4e"
   "\x0b\x8b\xd1\xcd\x80\x33\xc0\x40\xcd\x80\xe8\xd7\xff\xff"
   "\xff/bin/sh";

   char *buff = NULL;
   unsigned long *addr_ptr = NULL;
   char *ptr = NULL;

   int i;
   int ofs = DEFAULT_OFFSET;

   buff = malloc(4096);
   if(!buff)
   {
      printf("can't allocate memory\n");
      exit(0);
   }
   ptr = buff;

   /* fill start of buffer with nops */

   memset(ptr, 0x90, BUFFER_SIZE-strlen(execshell));
   ptr += BUFFER_SIZE-strlen(execshell);

   /* stick asm code into the buffer */

   for(i=0;i < strlen(execshell);i++)
```

```
      *(ptr++) = execshell[i];

   addr_ptr = (long *)ptr;
   for(i=0;i < (8/4);i++)
      *(addr_ptr++) = get_esp() + ofs;
   ptr = (char *)addr_ptr;
   *ptr = 0;

   (void)alarm((u_int)0);
   execl(PATH_MOUNT, "umount", buff, NULL);
}
```

Here is a little solution --
chmod -s /bin/umount
This way only root can run this command.

With best regards, Paulo

--

Anexo A-2 Programa CGI usado para gerar a homepage do sistema CUCO: cuco.pl

```
print <<EOF;
<HTML>
<HEAD>
<title>CUCO - versao 0.02</title>
<LINK REV="made" HREF="mailto:berthold\@pop-rs.rnp.br">
</HEAD>
<BODY>
<CENTER>
<H1> CUCO 0.02 </H1>
<H5> <font color=#FF0000>CU</font>idado! ou <font
color=#FF0000>C</font>hamo
 o <font color=#FF0000>O</font>perador!
<!-  Can yoU Call Operator? ->
  
(<font color=#FF0000>C</font>a<font color=#FF0000>U</font>tion! or
I'll
<font color=#FF0000>C</font>all <font color=#FF0000>O</font>perator!)
</H5>
</CENTER>
<HR>
<HL>
<p>
<strong> Controle de Alarmes </strong>
        <ul>
        <li> <a href=$HTML_SERVER/util/testahosts.pl> Lista de hosts
ativos </a>
        <li> <a href=$HTML_SERVER/hostsmon.pl> Lista de hosts
monitorados </a>
        </ul>
```

```
<strong> Obtendo Informa&ccedil;&otilde;es sobre o atacante </strong>
        <ul>
        <li> <a href=$HTML_SERVER/mon.pl> Modo Normal </a>
        <li> <a href=$HTML_SERVER/mon.pl> Modo Completo </a>
        </ul>

<strong> A&ccedil;&otilde;es para Resposta em Caso de
Detec&ccedil;&atilde;o de
      Invas&atilde;o</strong>
        <ul>
      <li> <a href=$HTML_SERVER/fullmd5.pl> Verifica extens&atilde;o
dos danos </a>
      <li> <a href=$HTML_SERVER/util/tac.pl> Verifica login via
Tacacs </a>
        <li> <a href=$HTML_SERVER/whodo.pl>   Quem faz o qu&ecirc; na
m&aacute;quina </a>
        <li> Ativa LOG IP completo
        <li> Monitora&ccedil;&atilde;o de TTYs
        <li> Ativa monitora&ccedil;&atilde;o Shell
      </ul>
<strong> No <a href=$HTML_SERVER/help.html> Centro de Recursos </a>
voc&ecirc; pode
encontrar ajuda e mais informa&ccedil;&otilde;es.</strong>

</p>
<HR>
<HL>
Esclarecimentos e Coment&aacute;rios:
<ADDRESS><A HREF="mailto:berthold\@pop-rs.rnp.br">berthold\@pop-
rs.rnp.br</A></ADDRESS>
</BODY>
</HTML>
EOF
```

Anexo A-3 Exemplo de um programa CGI usado no sistema CUCO: testahosts.pl

```
require "config/cucocfg.pl";
require "config/paths.pl";
require "html/util/shell.pl";

#
# pinga uma lista de hosts e verifica se estao ativos
#
sub check_alive {

   foreach $host (@_) {
      print "pingando host [$host]\n" if $debug;


        #
        # Usa o programa fping(suid)
        #
        &open_cmd(FPING, $long_timeout, "$FPING $host")
                || die "Nao consigo executar $FPING: $!\n";

        while(<FPING>) {
                if (/(\S+) is alive/) {
                  print "[$host] -> ativo\n" if $debug;
                  print CLIENT "[$host] -> ativo<BR>\n";
                  print CLIENT "<TR><TD>$host</TD><TD>";
                  print CLIENT "<FONT
COLOR=#000000>ativo</FONT></TD></TR>\n";
                } elsif ($debug && /(\S+) is unreachable/) {
                  print "[$host] -> sem resposta\n" if $debug;
                  print CLIENT "[$host] -> sem resposta<BR>\n";
```

```
                    print CLIENT "<TR><TD><FONT
COLOR=#FF0000>$host</FONT></TD><TD><BLINK>";
                    print CLIENT "<FONT COLOR=#FF0000>sem
resposta</FONT></BLINK></TD></TR>\n";
                }
        }
        close(FPING);
   }
}


print CLIENT <<EOF;
<HTML>
<HEAD>
<title>Verificando hosts monitores</title>
<LINK REV="made" HREF="mailto:berthold\@pop-rs.rnp.br">
</HEAD>
<BODY>
<CENTER>
<H1> Verificando hosts monitores </H1>
</CENTER>
<HR>
<HL>
<TABLE BORDER=2 CELLSPACING=3 CELLPADDING=5 WIDTH=80%>
<CAPTION><B>M&aacute;quinas Monitoradas</B></CAPTION>
EOF

# Verifica se os hosts estao ativos...
&check_alive( @Mhosts ) ;

print CLIENT <<EOF;
</TABLE>
<HR>
<HL>
Esclarecimentos e Coment&aacute;rios:
```

```
<ADDRESS><A HREF="mailto:berthold\@pop-rs.rnp.br">berthold\@pop-
rs.rnp.br</A></ADDRESS>
</BODY>
</HTML>
EOF
```

Anexo A-4 Definição ASN.1 da MIB cuco

```
RFC1213-MIB DEFINITIONS ::= BEGIN
--- groups
          cuco        OBJECT IDENTIFIER ::= { mib-2 3 1 }
          cfgmd5      OBJECT IDENTIFIER ::= { cuco 1 }
          binmd5      OBJECT IDENTIFIER ::= { cuco 2 }
          interfaces  OBJECT IDENTIFIER ::= { cuco 3 }
          processos   OBJECT IDENTIFIER ::= { cuco 4 }
          sistema     OBJECT IDENTIFIER ::= { cuco 5 }
          acao        OBJECT IDENTIFIER ::= { cuco 6 }

          cfgNumber OBJECT-TYPE
              SYNTAX  INTEGER
              ACCESS  read-only
              STATUS  mandatory
              DESCRIPTION
                      "Numero de arquivos de configuracao
                       monitorados"
              ::= { cfgmd5 1 }

          cfgTable OBJECT-TYPE
              SYNTAX  SEQUENCE OF CfgEntry
              ACCESS  not-accessible
              STATUS  mandatory
              DESCRIPTION
                      "Uma lista de MD5 de arquivos. O numero de
                       entradas e dado por  cfgNumber."
              ::= { cfgmd5 2 }

          cfgEntry OBJECT-TYPE
              SYNTAX  CfgEntry
              ACCESS  not-accessible
```

```
    STATUS  mandatory
    DESCRIPTION
            "Descricao dos objetos arquivos de
             configuracao"

    INDEX   { cfgIndex }
              ::= { cfgTable 1 }

CfgEntry ::=  SEQUENCE {
    cfgIndex  INTEGER,
    cfgDescr  DisplayString,
    cfgMd5    DisplayString,
    }

cfgIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Um valor único para cada arquivo, com
             base em um indice global que define todos
             os arquivos de configuração possiveis em
             um dado sistema operacional. Arquivos de
             usuario possuem indice 0(zero) ."
    ::= { cfgEntry 1 }

cfgDescr OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Descrição textual da interface."
    ::= { cfgEntry 2 }

cfgMd5 OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..32))
```

```
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "MD5 de 32 bytes do arquivo de configuracao."
    ::= { cfgEntry 3 }

binNumber OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Numero de arquivos de configuracao
             monitorados"
    ::= { binmd5 1 }

binTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF BinEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "Uma lista de MD5 de arquivos. O numero de
             entradas e dado por  binNumber."
    ::= { binmd5 2 }

binEntry OBJECT-TYPE
    SYNTAX  BinEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "Descricao dos objetos arquivos de
             configuracao"

    INDEX   { binIndex }
              ::= { binTable 1 }

BinEntry ::=  SEQUENCE {
```

```
    binIndex  INTEGER,
    binDescr  DisplayString,
    binMd5    DisplayString,
    }

binIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Um valor único para cada arquivo, com
             base em um indice global que define todos
             os arquivos de configuração possiveis em
             um dado sistema operacional. Arquivos de
             usuario possuem indice 0(zero) ."
    ::= { binEntry 1 }

binDescr OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Descrição textual da interface."
    ::= { binEntry 2 }

binMd5 OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..32))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "MD5 de 32 bytes do arquivo de configuracao."
    ::= { binEntry 3 }

ifNumber OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
```

```
    STATUS  mandatory
    DESCRIPTION
            "Número de interfaces de rede do sistema"
    ::= { interfaces 1 }

ifTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF IfEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "Uma lista de interfaces. O número de
             entradas é dado por ifNumber."
    ::= { interfaces 2 }

ifEntry OBJECT-TYPE
    SYNTAX  IfEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "Cada interface contem mais alguns objetos"

    INDEX   { ifIndex }
              ::= { ifTable 1 }

IfEntry ::=  SEQUENCE {
      ifIndex  INTEGER,
      ifDescr  DisplayString,
      ifStatus INTEGER,
      }

ifIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Um valor único para cada interface, com
```

```
                valor variando entre 1 e ifNumber."
        ::= { ifEntry 1 }

    ifDescr OBJECT-TYPE
        SYNTAX  DisplayString (SIZE (0..255))
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Descrição textual da interface."
        ::= { ifEntry 2 }

    ifStatus OBJECT-TYPE
        SYNTAX  INTEGER {
                    up(1),
                    down(2),
                    testing(3),
                    broadcast(4),
                    debug(5),
                    promiscuous(6)
                   }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "O estado da interface, segundo os valores
                 retornados
                 pela função ifstatus()"
        ::= { ifEntry 3 }

   prTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF PrEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "Uma lista de processos."
        ::= { processos 1 }
```

```
prEntry OBJECT-TYPE
    SYNTAX  PrEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "Define os objetos de cada processo"

    INDEX   { prIndex }
              ::= { prTable 1 }

PrEntry ::=  SEQUENCE {
    prTty    DisplayString,
    prDescr  DisplayString,
    prCmd    DisplayString
    }

prTty OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..5))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Identificacao do tty do processo, se o valor
             for nulo, entao nao ha tty associado."
    ::= { prEntry 1 }

prUser OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..7))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Identificacao do usuario responsavel pelo
             processo."
    ::= { prEntry 2 }

prCmd OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..49))
```

```
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Descricao do comando e dos parametros
             utilizados pelo usuario."
     :: { prEntry 3 }

hostDiretor OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..30))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Informa o host diretor ao qual deve se
             reportar e ter como único gerente do
             sistema CUCO."
    ::= { sistema 1 }

local OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..50))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Informa o local onde esta intalado esta
              maquina "
    ::= { sistema 2 }

responsavel OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Informa o responsavel pelo sistema,
             ou uma pessoa/telefone para contato."
    ::= { sistema 3 }

sisop OBJECT-TYPE
```

```
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Informacoes complementares sobre o sistema,
             que esta sendo executado nesta maquina"
    ::= { sistema 4 }

acUser OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..8))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "Informa um usuário a ser eliminado do
             sistema. Somente utilizado se o sistema
             CUCO estiver com módulo de resposta a
             ataques habilitado."
    ::= { acao 1 }

acHost OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..30))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "Informa um host que deve perder o acesso ao
             sistema. As conexões atuais são terminadas
             e novos pedidos são indeferidos. Somente
             utilizado se o sistema CUCO estiver com o
             módulo de resposta a ataques habilitado."
    ::= { acao 2 }

acAtivadump OBJECT-TYPE
    SYNTAX  INTEGER {
                turnoff(0),
                turnon(1)
                }
```

```
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
             "Ativa dump de pacotes. Somente utilizado
             se o sistema CUCO estiver com o módulo
             de resposta a ataques habilitado."
    ::= { acao 3 }

acAtivashell OBJECT-TYPE
    SYNTAX  INTEGER {
                turnoff(0),
                turnon(1)
                }
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "Ativa monitoracao shell de processo.Apesar
             de existir uma entrada definida ainda nao
             esta implementado - falta entrada para
             identificacao do processo shell. Por hora
             ainda nao eesta implementado"
    ::= { acao 4 }

acAtivatty OBJECT-TYPE
    SYNTAX  INTEGER {
                turnoff(0),
                turnon(1)
                }
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
             "Ativa monitoracao de tty de usuarios
              So vai funcionar se estiver com o módulo
             de resposta a ataques habilitado. - Por
             hora ainda nao esta implementado"
    ::= { acao 5 }
```

```
END
```