UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

ANTONIO ALFREDO PIRES OLIVEIRA

São Luís
2005

ANTONIO ALFREDO PIRES OLIVEIRA

SAMARA
SOCIEDADE DE AGENTES PARA A MONITORAÇÃO DE ATAQUES E RESPOSTAS AUTOMATIZADAS

Dissertação de Mestrado submetida à Coordenação do Programa de Pós-Graduação em Engenharia de Eletricidade da UFMA como parte dos requisitos para obtenção do título de Mestre em Engenharia de Eletricidade.

Orientadores: Prof. Dr. Edson Nascimento

Prof. Dr. Zair Abdelouahab

São Luís

2005

Oliveira, Antonio Alfredo Pires

SAMARA – Sociedade de Agentes para Monitoração de Ataques e Respostas Automatizadas / Antonio Alfredo Pires Oliveira. – São Luís, 2005.

113f.

Dissertação (Mestrado em Engenharia de Eletricidade) – Universidade Federal do Maranhão, 2005.

1. Segurança de redes 2. Armadilhas de redes 3. *Honeypots* 4. *Honeynet* 5. Agentes inteligentes I. Título.

CDU 004.056.53

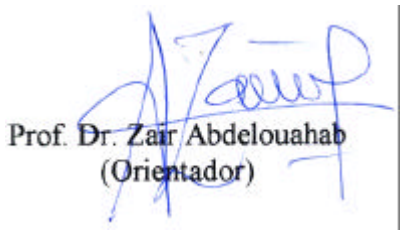
ANTONIO ALFREDO PIRES OLIVEIRA

SAMARA

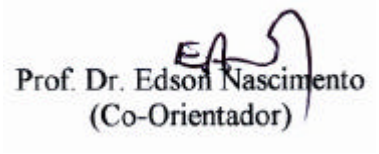
SOCIEDADE DE AGENTES PARA A MONITORAÇÃO DE ATAQUES E RESPOSTAS AUTOMATIZADAS

Dissertação de Mestrado submetida à Coordenação do Programa de Pós-Graduação em Engenharia de Eletricidade da UFMA como parte dos requisitos para obtenção do título de Mestre em Engenharia de Eletricidade.

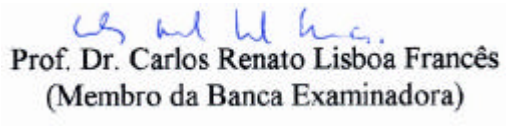
Aprovada em 17 de junho de 2005.



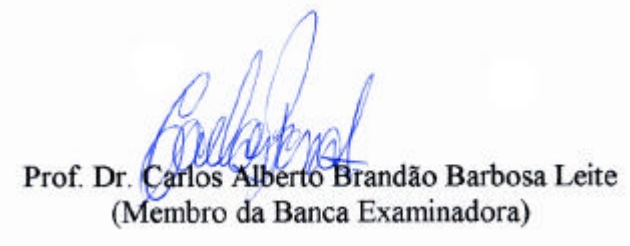
Prof. Dr. Zair Abdelouahab
(Orientador)



Prof. Dr. Edson Nascimento
(Co-Orientador)



Prof. Dr. Carlos Renato Lisboa Francês
(Membro da Banca Examinadora)



Prof. Dr. Carlos Alberto Brandão Barbosa Leite
(Membro da Banca Examinadora)

A Deus, ao meu pai, às minhas mães e à minha esposa, fontes de inspiração para a realização de todos os meus sonhos.

AGRADECIMENTOS

A Deus, ser maior e supremo, que me amparou nos momentos mais difíceis desta jornada.

Aos meus Orientadores, Prof. Dr. Edson Nascimento e Prof. Dr. Zair Abdelouahab, pela imensa paciência, compreensão e dedicação dispensadas à realização deste trabalho.

À minha esposa Alessandra, pela paciência, carinho e incentivo incessantes.

À Profa. Dra. Maria da Guia, por não esquecer de cobrar, a todo momento, a conclusão deste trabalho.

A Johnneth Fonseca, pelo valoroso esforço na implementação dos protótipos da arquitetura.

A Alice Almeida, por me dispensar de tarefas importantes para que eu pudesse me dedicar à conclusão deste trabalho.

A todas as outras pessoas que contribuíram diretamente e indiretamente e que, apesar de não terem sido citadas, não foram e nem serão esquecidas.

"É mais fácil proteger uma agência bancária do mundo físico do que a sua equivalente na Internet".

RESUMO

As técnicas tradicionais de segurança aplicadas em redes de computadores tentam bloquear ataques (utilizando *firewalls*) ou detectá-los assim que eles ocorrem (utilizando Sistemas de Detecção de Intrusos). Ambas são de reconhecido valor, porém, têm seus limites. Nesse sentido, há que se inovar em relação às técnicas e táticas de defesas, bem como em ferramentas e tecnologias que complementem os mecanismos tradicionais aplicados em segurança de redes e computadores. Uma dessas soluções tem sido o uso de *honeypots* (armadilhas de redes) na coleta de informações, motivos, táticas e ferramentas utilizadas em atividades maliciosas em redes e sistemas distribuídos. Este trabalho introduz a arquitetura de respostas automatizadas a incidentes de segurança, denominada SAMARA, que é baseada em *honeypots* e agentes inteligentes, concebida para atender os requisitos funcionais dos agentes *decoy server* e *honeynet* propostos para o Projeto NIDIA – *Network Intrusion Detection System based on Intelligent Agents* [18], mas que pode se ajustar a outras abordagens de detecção e prevenção e reação a incidentes de segurança em redes e sistemas distribuídos.

Palavras-chave: Segurança de Redes, Armadilhas de Redes, *Honeypots*, *Honeynet*, Agentes Inteligentes

ABSTRACT

The traditional security techniques applied in computer networks try to block attacks (using firewalls) or to detect them as soon as they happen (using Intrusion Detection Systems). Both are of recognized value, however, they have limitations. In that sense, there is to innovate as for techniques and defense tactics, as well as the tools and technologies that complement the traditional mechanisms applied in network and computer security. One of these solutions have been using *honeypots* (networks traps) to collect information, motives, tactics and tools used in malicious network activities and distributed systems. This research work introduce an architecture for automated incident response, called SAMARA, based on *honeypots* and intelligent agents, created to support the functional requisites of decoy server and *honeynet* agents proposed for NIDIA Project – *Network Intrusion Detection System based on Intelligent Agents* [18], but that can be adjust to others detection, prevention and reaction approaches of security incidents in network and distributed systems.

Keywords: Network Security, Network Traps, *Honeypots*, *Honeynet*, Intelligent Agents

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ARP – Address Resolution Protocol
ATR – Agente de *Traceroute* Remoto
BOF – Back Officer Friendly
CCO – Console Central de Operações
CERT – Computer Emergency Response Team
CGM – Content Generation Module
CPU – Central Processing Unit
DMZ – DeMilitarized Zone
DNS – Domain Name Server
DTK – Deception ToolKit
EUA – Estados Unidos da América
FDDI – Fiber Distributed Data Interface
FTP – File Transfer Protocol
HD – Hard Disk
HTTP – Hypertext Transfer Protocol
ICMP – *Internet Control Message Protocol*
IDS – Intrusion Detection System
INPE – Instituto Nacional de Pesquisas Espaciais
IP – Internet Protocol
LAN – Local Area Network
NAI – Network Associates Inc.
NIDIA – Network Intrusion Detection System based on Intelligent Agents
POP3 – Post Office Protocol
PPP – Point-to-Point Protocol
PPPoE – Point-to-Point Protocol over Ethernet
RABD – Base de Dados de Ações
RAM – Random Access Memory
RAS – Remote Access Server
RPC – Remote Procedure Call
SAMARA – Sociedade de Agentes para a Monitoração de Ataques e Respostas Automatizadas
SCA – Agente Controlador de Ações

SLIP – *Serial Line Internet Protocol*

SMS – Short Message Service

SMTP – Simple Mail Transport Protocol

SO – Sistema Operacional

SSH – Secure Shell

STBD – Base de Dados de Estratégias

TCP – Transmission Control Protocol

UDP – *User Datagram Protocol*

VPN – Virtual Private Network

WAN – Wide Area Network

SUMÁRIO

1 INTRODUÇÃO

Ao longo dos anos, os sistemas conectados em rede têm crescido consideravelmente em tamanho, complexidade e susceptibilidade a ataques e invasões. Ao mesmo tempo, cresce enormemente o conhecimento, as ferramentas e técnicas disponíveis aos atacantes. Infelizmente, por outro lado, as técnicas e ferramentas de defesa não têm evoluído tão rapidamente devido à natureza reativa na qual elas são utilizadas [28].

Segundo [47], tradicionalmente, a segurança das informações tem sido defensiva. Mecanismos de segurança como *firewalls*, IDS's e criptografia são utilizados para proteger os recursos de uma organização. A estratégia é defender a organização de alguém, da melhor maneira possível, detectar todas as falhas da defesa e, em seguida, reagir a essas falhas. O problema dessa abordagem é que o administrador de rede (ou de segurança) sempre espera pela ação atacante, utilizando suas ferramentas de defesa tradicionais.

O atacante ou invasor, por um lado, antes de uma investida, levanta diversas informações sobre o sistema-alvo, destaca suas fraquezas, isto é, suas vulnerabilidades e, então, traça um plano para conseguir seu objetivo, depois de mensurar os aspectos positivos e negativos de uma possível investida [33]. Os administradores, por outro lado, não utilizam ou não dispõem de recursos de contra-espionagem ou proatividade e, geralmente, os mecanismos tradicionais são preparados para reagirem às falhas conhecidas [39].

Nesse sentido, soluções inovativas são requeridas para lidarem com as classes de ameaças atuais e futuras [7]. Uma dessas soluções tem sido o uso de armadilhas de rede na coleta de informações sobre os atacantes, seus motivos, táticas e ferramentas utilizadas em atividades maliciosas em redes e sistemas distribuídos. Essas informações constituem uma importantíssima arma para que os administradores de segurança possam aferir melhor o ambiente e estabelecer um nível de segurança que satisfaça os requisitos da política de segurança das organizações [11].

1.1 Cenário e definição do problema

As técnicas tradicionais de segurança aplicadas em redes de computadores tentam bloquear ataques (utilizando *firewalls*) ou detectá-los assim que eles ocorrem (utilizando IDS's). Ambas são de reconhecido valor, porém, têm seus limites. Devido ao enorme tempo disponível e o acesso à informação a qualquer hora e lugar, um atacante pode aprender a contornar um *firewall*. Uma vez vencida, essa barreira não servirá para coisa alguma. Da mesma forma, um IDS somente fornecerá informação uma vez que um ataque tenha acontecido. Além disso, um IDS não pode determinar se um novo ataque foi bem sucedido ou se ele seria bem sucedido contra outros sistemas [28].

Outro fato preocupante é que, quase sempre, quando uma organização tem sua rede ou sistema atacado ou invadido, dificilmente são divulgadas informações sobre o fato, principalmente nos casos que envolvem diretamente instituições financeiras ou organizações governamentais [33].

Segundo [28], [50] e [51], uma contramedida bem sucedida tem sido atrasar o atacante enquanto os administradores obtêm informações suficientes sobre seu inimigo para evitar que o ataque cause danos. O uso bem sucedido de mecanismos de fraude e decepção leva a esse objetivo. Enganando o atacante, o defensor alimenta-o com informações e alvos falsos, e força-o a perder tempo em ataques infrutíferos, fazendo-o pensar que aquele alvo não merece um futuro ataque. Além disso, uma boa decepção dará ao defensor informações significativas sobre o pensamento do atacante, seus motivos, táticas e ferramentas empregadas na operação. Estas informações podem então ser usadas para melhorar as medidas de segurança existentes, auxiliar na reconfiguração de recursos de missão crítica, *firewalls* e IDS's e, até mesmo, levar o atacante a sofrer uma possível punição legal.

Porém, para os administradores de redes, é desagradável ver suas redes serem atacadas e ficar de braços cruzados, como simples observadores. É necessário que medidas de segurança sejam tomadas durante uma tentativa de invasão ou imediatamente após o comprometimento de uma unidade de decepção ou um *host* qualquer dentro da rede.

Uma solução para esse dilema seria, por exemplo, o uso integrado das técnicas tradicionais de segurança de rede (*firewalls*, IDS e roteadores) com a aplicação de *decoy servers*[i] ou *honeypots* e agentes inteligentes, associados de forma que esses dispositivos possam interagir uns com os outros, criando uma estrutura de segurança mais eficiente e capaz de impedir que o invasor atinja recursos de missão crítica dentro ambiente, minimizando, assim, os possíveis danos freqüentemente observados em ataques dessa natureza [37].

A Figura 1.1 exemplifica a idéia de utilizar ferramentas de segurança tradicionais como *firewalls* e IDS's aliadas a armadilhas de redes e *decoy servers* como estratégia para detectar ataques e invasões, iludir intrusos e responder proativamente.

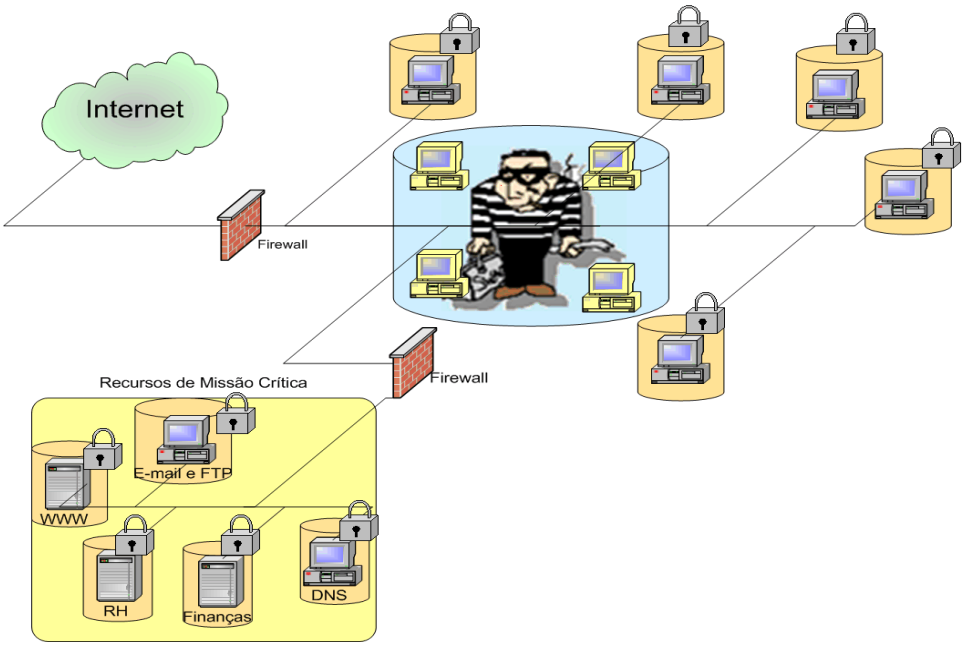


Figura 1.1 – Utilização de armadilhas de rede para responder a ataques

[i] *Decoy servers* (servidor fictício), *honeypots* e *deception systems* são referências a armadilhas de redes.

Conforme mostra a Figura 1.1, um sistema de detecção de intrusos com respostas automáticas, utilizando armadilhas de rede e agentes inteligentes, pode ajudar a detectar atividades maliciosas e responder proativamente, enrijecendo a segurança da rede, de forma a reduzir os riscos de danos ao ambiente [32] e [46].

1.2 Objetivos geral e específicos

Esta dissertação propõe, de forma geral, uma arquitetura que agrega uma estratégia de detecção de incidentes de segurança complementar à atividade de IDS's e *firewalls*, utilizando tecnologias baseadas em armadilhas de rede e um conjunto de agentes gerenciados por uma unidade central, além de possibilitar respostas automáticas aos incidentes reportados. Essa arquitetura possibilitará uma camada-extra para a detecção de incidentes de segurança em redes e sistemas distribuídos, fornecendo aos administradores recursos de proatividade que permitam minimizar potenciais danos causados por atividades maliciosas. De modo mais específico, espera-se:

- Apresentar um modelo estratégico de respostas automáticas a incidentes de segurança (SAMARA), utilizando *honeypots* e agentes inteligentes;
- Demonstrar o uso da tecnologia de *honeypots* e *honeynets* como complemento aos mecanismos tradicionais de detecção de ataques e intrusões;
- Reduzir o número de falsos positivos e negativos presentes nos alertas de IDS e *firewalls*;
- Contribuir para aumentar o nível de segurança de redes e sistemas distribuídos;
- Apresentar a arquitetura SAMARA como ambiente de decepção, baseado no uso de armadilhas de redes e agentes inteligentes conforme as proposições de

respostas a incidentes do sistema NIDIA [18], porém, flexível o bastante, de forma a possibilitar a interação com outros mecanismos de segurança;

1.3 Organização do trabalho

Esta dissertação está organizada em sete capítulos.

No Capítulo 1 encontra-se o cenário que motivou a realização deste trabalho.

O Capítulo 2 apresenta uma visão geral em torno dos conceitos, tecnologias e ferramentas baseadas em *honeypots*, destacando-se suas características, vantagens e desvantagens, sua categorização e classificação. Enfoca-se, também, a questão do posicionamento e tipos de distribuição desses mecanismos. Sumariza-se com um quadro comparativo que cruza as opções disponíveis no mercado, alternativas caseiras e acadêmicas.

No Capítulo 3 é apresentada a arquitetura baseada em agentes para a monitoração e respostas a ataques (SAMARA), as responsabilidades dos agentes envolvidos. É enfatizado o princípio de operação do sistema, destacando-se, também, suas principais funcionalidades e vantagens. Por fim, sumariza-se com um comparativo entre as soluções baseadas em *honeypots* disponíveis e essa arquitetura.

O Capitulo 4 mostra a aplicabilidade e integração da arquitetura SAMARA ao Projeto NIDIA.

O Capítulo 5 mostra alguns detalhes das implementações parciais da arquitetura, destacando-se os protótipos dos agentes que compõe o sistema e da console de gerenciamento.

O Capítulo 6 mostra alguns resultados parciais obtidos nos testes em laboratório.

Por fim, no Capítulo 7 são apresentadas as considerações finais, ressaltando-se as contribuições deste trabalho e as sinalizações para a continuidade deste trabalho.

2 HONEYPOTS: ARMADILHAS DE REDES DE COMPUTADORES

Apresenta-se, neste capítulo, uma visão geral sobre *honeypots* como ferramentas ou artifícios que complementam as tecnologias tradicionais utilizadas em segurança de redes de computadores, enfatizando-se sua história, vantagens, limitações, características, importância, classificação e categorização. Por fim, são mostrados os principais representantes existentes no mercado e em ambientes de testes e de pesquisas, e um estudo comparativo entre os mesmos.

2.1 Considerações iniciais

O uso de decepção (ou fraudes) tem sido um subterfúgio utilizado por líderes militares para ganhar batalhas ante seus oponentes. Segundo [28], durante a Segunda Guerra Mundial, os alemães foram levados a acreditar que a invasão real dos aliados ocorreria em Pas-de-Calais ao invés de pela Normandia. Em outras ocasiões, segundo esse mesmo autor, os aliados utilizaram bases com aviões e veículos fictícios para atrair aviões e comboios inimigos. Artifícios similares também foram utilizados pelos EUA na chamada "Operação Tempestade no Deserto" no Iraque.

Outras associações à utilização de fraudes são:

- O Cavalo de Tróia, suposto presente dos gregos para os troianos, como prova de aliança de paz entre aqueles povos, mas que foi utilizado para os gregos transpassarem as muralhas da cidade e destruírem Tróia [22];
- O "Boi de Piranha", artifício utilizado por condutores de boiadas, principalmente em regiões alagadas do Brasil (Pantanal e Amazônia) para reduzir o número de ataques de piranhas aos rebanhos que são constantemente

> trasladados para pastagens livres de inundação. Neste caso, uma cabeça do gado é escolhida para atrair o cardume dos vorazes predadores (principalmente piranhas, daí o nome "boi de piranha") e servir de distração enquanto o restante da boiada é tangido em segurança [20].

Indivíduos que praticam crimes por computador, na maioria das vezes, também utilizam esses subterfúgios, isto é, instalam programas chamados cavalo de tróia (*trojan horse*) nos sistemas invadidos. Outras vezes, utilizam máquinas alheias ("boi de piranha") para cometer suas atividades maliciosas, impossibilitando, na maioria das vezes, a sua localização [10].

2.2 Conceitos básicos

O conceito de *honeypots* (potes de mel) já existe há alguns anos. De modo geral, *honeypots* são sistemas ou recursos criados para serem atacados ou comprometidos por um atacante ou invasor.

Armadilhas de rede foram discutidas, pela primeira vez, em alguns documentos escritos por diversos ícones da segurança dos computadores, destacando-se [4] e [44]. Ambos os casos usavam tecnologia do tipo "prisão" para capturar a sessão de um invasor e monitorar, com detalhes, o que ele queria fazer. O termo *honeypot* veio mais tarde, mas a intenção é a mesma: configurar um ou mais sistemas que parecem atraentes para os invasores de redes (daí, a associação ao termo "pote de mel"), mas que também podem monitorar com um grau mais alto de precisão o que está acontecendo. Ao monitorar a atividade por meio de um *honeypot*, é possível identificar o problema e estar relativamente certo de que o invasor entrou e o que ele está fazendo ou fez no sistema comprometido.

Formalmente, Lance Spitzner [47] define o termo *honeypot* como segue:

> *"Um honeypot é um recurso cujo valor está em ser atacado ou comprometido. Isto é, espera-se que um honeypot seja provado (descoberto), atacado e potencialmente explorado."*

Ainda segundo [47] e [48], os *honeypots* não consertam, nem solucionam, quaisquer problemas. *Honeypots* fornecem informações adicionais de significativo valor, o que IDS's e *firewalls*, tradicionalmente, não conseguem mostrar e, quando o fazem, apresentam incertezas, o que dificulta o trabalho do pessoal de segurança de computadores.

Um conceito ligeiramente diferente daquele sugerido por [47], porém, mais esclarecedor, é proposto por Reto Baumann [2]:

> *"Um honeypot é um recurso que finge ser um alvo real. Espera-se que um honeypot seja atacado ou comprometido. Os objetivos principais são a distração de um atacante e a obtenção de informações sobre um ataque e sobre o atacante".*

De forma geral, o termo *honeypot* está associado a armadilhas de redes, fraude, isca e decepção, principalmente.

Há dois outros termos que são bastante referidos quando fala-se em segurança de redes e, lógico, *honeypots*: ataque e invasão.

Ataque é normalmente qualquer tentativa maldosa ou danosa de descobrir vulnerabilidades em um sistema e explorá-las. Quando não consegue explorá-las e ganhar acesso, o atacante pode reagir radicalmente, como tentar indisponibilizar serviços (no caso dos ataques de DoS[ii]). O atacante pode também divulgar as vulnerabilidades do sistema atacado. Invasão é quando um atacante consegue "entrar em um sistema", ou seja, quando é possível realizar atividades locais e remotas, pois o sistema foi de fato comprometido. Isto pode acontecer através de um acesso legítimo (por exemplo, o *login* de uma conta administrativa do recurso), através de um "cavalo de tróia" (*trojan horse*) ou através de "portas dos fundos" (*backdoors*). No primeiro caso, tudo ocorre como se fosse um acesso normal. No outro caso, o invasor ou instala uma aplicação que lhe permite entrar e sair do

[ii] Denial of Service ou Negação de Serviço. Ataque no qual o ataque tenta indisponibilizar os serviços de um sistema remoto através da sobrecarga de requisições para os serviços de um determinado sistema-alvo. Se o sistema não tiver recursos suficientes para tratar todas as requisições, ocorre uma espécie de pane e o sistema fica indisponível.

sistema sem deixar rastros, ou substitui uma aplicação legítima por outra com o mesmo nome, mas com funções extras que normalmente lhes permite acessar o sistema de forma segura e sem que o administrador desconfie. Ambos os casos são dificilmente auditados, até mesmo impossíveis de serem detectados a tempo [7].

2.3 Importância dos *honeypots*

Segundo Spitzner [47], a segurança das informações tradicionalmente tem sido defensiva, ou seja, são empregados mecanismos seguindo-se a estratégia de defender a organização de alguém, da melhor maneira possível, detectar todas as falhas e corrigi-las. Assim, *firewalls*, IDS's e outros recursos são utilizados de forma defensiva. Porém, o inimigo está no ataque. Há uma horda de desocupados e outros indivíduos ávidos para encontrar uma vulnerabilidade ou um sistema desprotegido o suficiente para derrubar um site ou obter alguma informação importante [26].

Dessa forma, os *honeypots* oferecem aos administradores uma chance de contra-atacar e agir proativamente. Um *honeypot* funciona como um alarme de rede. Coletando informações sobre o inimigo, as equipes de segurança de TI podem parar um ataque ou contornar uma falha antes dela ser explorada [48].

2.3.1 Vantagens

2.3.1.1 *Importância dos dados em um honeypot*

Ao contrário de *firewalls* e IDS's que coletam grandes quantidades de informações que na maioria das vezes são humanamente impossíveis de serem tratadas a

tempo por um administrador de rede (normalmente este profissional acumula diversas funções), um *honeypot*, pelo próprio princípio de não ser um recurso de produção, registra poucos dados. Estes, porém, são de elevado valor e significação. Os dados provenientes de um *honeypot* podem ser tratados mais rapidamente, além de conterem um grau de certeza bem mais elevado, ao contrário dos inúmeros falsos positivos e falsos negativos gerados por IDS's [39].

Apesar da precisão dos dados coletados pelos *honeypots*, a quantidade de bytes coletados é desprezível quando comparados com os registros de IDS's e *firewalls* (quando estes registram os *logs*). Segundo o Projeto *Honeynet*[iii] [47], os *honeypots* utilizados pelo grupo de pesquisa coletam, em média, menos de 1 MB (megabyte) por dia. Os *logs* de IDS's e *firewalls*, no mesmo período, passam de gigabytes [2] e [47].

Outra vantagem a respeito dos dados coletados por um *honeypot*, em relação a outros mecanismos utilizados em segurança de rede (*firewalls* e IDS's), é que é mais fácil modelar estatísticas e tendências, detectar ataques e, até mesmo, pesquisar atacantes.

Lance Spitzner compara em [39] a análise dos dados de um *honeypot* à análise microscópica, onde uma pequena amostra de material pode revelar informações importantes. Um exemplo disso pode ser visto através da Tabela 2.1, a seguir:

Tabela 2.1 – Varreduras de portas detectada por uma rede de *honeypots*

Date	Time	Src-IP	Dst-IP	Protocol	Src-Port	Dst-Port
02/02/15	23:50:15	213.68.213.135	10.1.1.101	UDP	5298	18030
02/02/15	23:50:15	213.68.213.134	10.1.1.109	UDP	18986	10903
02/02/15	23:50:15	213.68.213.133	10.1.1.108	UDP	16932	16219
02/02/15	23:50:16	213.68.213.140	10.1.1.107	UDP	1348	5274
02/02/15	23:50:16	213.68.213.130	10.1.1.105	UDP	19841	15316
02/02/15	23:50:17	213.68.213.144	10.1.1.104	UDP	17773	3327

A Tabela 2.1 mostra que vários *honeypots* foram sondados em conexões UDP, por endereços de origens diferentes. A princípio, para um IDS isto não significaria um ataque ou

[iii] Uma rede de *honeypots*.

um evento relevante. Porém, como os *honeypots* não têm valor de produção, ou seja, não participam do conjunto de servidores de produção, qualquer conexão cujo destino seja um desses *honeypots* é tratada como uma sondagem ou mesmo um ataque, principalmente se a fonte for um endereço externo, válido na Internet.

Neste exemplo específico, o atacante está tentando determinar quais sistemas são alcançáveis através da Internet. Ele faz isso, enviando pacotes UDP com destino a portas altas. Como a maioria dos sistemas não agrega serviços nessas portas, os alvos simplesmente respondem enviando mensagens de erro de porta ICMP inalcançáveis. Porém, essas respostas dizem, por outro lado, que o sistema existe, está ativo e é alcançável. Um IDS convencionalmente não trataria adequadamente esses eventos isolados nem como falsos alarmes.

Isto demonstra que um *honeypot* pode ser utilizado para detectar ataques e prevenir invasões, complementando o trabalho de IDS's.

2.3.1.2 *Exaustão de Recursos (Escabilidade)*

A maioria dos mecanismos de segurança de redes, notadamente *firewalls* e IDS's, quando experimentam a exaustão de seus recursos, falham ou ignoram suas diretivas e funções.

Um *firewall*, por exemplo, quando tem tabela de conexões cheia, pode falhar ou ignorar a monitoração de conexões, o que leva a bloquear todas as conexões, ao invés de apenas bloquear os acessos não-autorizados [39].

Já no caso de um IDS, é comum, quando há muito tráfego de rede para monitorar, principalmente hoje com conexões que extrapolam centenas de megabytes por segundo, o descarte de pacotes.

Isto não ocorre com *honeypots*. Devido à captura e monitoração gerarem uma atividade mínima, a exaustão de recursos em *honeypots* é praticamente impossível de ocorrer. Antes que isto ocorra, já estará comprovado que o *honeypot* está sob domínio do atacante.

Em relação à exigência de requerimentos de *hardware* (RAM, CPU, HD), os *honeypots* levam novamente vantagem quando comparados aos IDS's e *firewalls*. Enquanto para esses recursos são necessárias tecnologias de última geração: grandes quantidades de RAM, poder de processamento e espaço em disco, *honeypots* podem ser implantados, utilizando-se recursos mais simples e mais baratos. Vale lembrar, mais uma vez, que os *honeypots* não substituem IDS nem *firewalls*, apenas complementam a função destes [39].

2.3.1.3 *Simplicidade*

Enquanto *firewalls* e IDS's requerem grande robustez e, no caso de IDS's, algoritmos complexos, *honeypots* são fáceis de implementar, não requerem bancos de dados, assinaturas de ataques, dentre outras necessidades. Como diz Spitzner em [39], basta pegar uma máquina, instalar um sistema operacional pela forma padrão, conectá-lo à rede e/ ou Internet, sentar e aguardar.

2.3.1.4 *Retorno de Investimento (ROI)*

Firewalls e IDS's são soluções caras. Além disso, para que uma organização usufrua de seus préstimos, é necessário que os mesmos estejam sempre atualizados, tanto no que se refere ao *hardware* hospedeiro, quanto ao *software* e assinaturas de ataques. Além disso, é difícil convencer os altos escalões de uma organização de que esses recursos precisam

ser atualizados ou substituídos, principalmente quando a organização nunca sofreu um ataque ou invasão [41].

Em contraste, *honeypots* podem rapidamente demonstrar sua importância e, daí, a necessidade de investimento que, conforme já abordado, não é alto quando comparado a IDS's e *firewalls*. Além disso, a implantação de *honeypots* pode justificar também investimentos em outros recursos de segurança, como autenticação forte e criptografia, dentre outros.

2.3.2 Desvantagens

2.3.2.1 *Campo de visão estreito*

Se, por um lado, a tecnologia de *honeypots* consegue obter informações extremamente relevantes com pequenas amostras de dados, esta característica faz com que se tenha uma visão muito limitada do que está realmente acontecendo no ambiente monitorado. A menos que um *honeypot* seja sondado ou atacado diretamente, nada pode ser inferido a respeito da ocorrência de qualquer sinistro [39].

2.3.2.2 *Comportamento previsível*

Um atacante pode identificar a verdadeira identidade de um *honeypot*, baseando-se no tipo de respostas provenientes deste. Por exemplo, num *honeypot* que emula um servidor de web, se as mensagens-padrão de erros não forem exatamente aquelas que seriam esperadas de um servidor de web real, o atacante pode deduzir que tal alvo se trata de um *honeypot*, podendo até alimentá-lo com informações ruins que levarão a equipe de segurança a tirar conclusões erradas [50].

2.3.2.3 *Risco para a organização*

Se um *honeypot* for atacado e comprometido, o atacante poderá utilizá-lo para atacar outros alvos, infiltrar-se na rede e até investir contras recursos de outras organizações. Segundo Spitzner [49], este risco é variável e depende da forma como se obtém um *honeypot* e também da maneira como o mesmo é implantado no ambiente.

2.3.3 O papel dos *honeypots* em segurança de redes

Segundo [47], há duas categorias de *honeypots*: de produção e de pesquisa.

Um *honeypot* de produção é assim chamado devido à sua aplicação num ambiente de produção de fato, visando a descoberta e a redução de riscos de ataques e invasões em uma organização. Um *honeypot* de pesquisa destina-se à obtenção da maior quantidade possível de informações sobre a atividade de atacantes, seus métodos e táticas, estratégias, ferramentas e até mesmo o pensamento destes, visando o compartilhamento desse aprendizado assim como a construção de melhores defesas contra as ameaças de segurança [39].

2.3.3.1 *Honeypots de produção*

Na visão de Lance Spitzner [39], *honeypots* de produção são comparados a oficiais da lei (policiais). Sua função é tomar conta dos indivíduos maus, prevenindo, detectando e respondendo à atividade não-autorizada desses elementos.

No que tange à prevenção, *honeypots* podem ser utilizados para fazer os atacantes e intrusos perderem tempo e confundi-los com informações fraudulentas. Porém, esta estratégia somente tem chances de funcionar para organizações que queiram proteger recursos

e informações de elevada relevância, por exemplo, um instituto de pesquisa militar de um país, que trabalha com informações que não podem cair em mãos erradas, principalmente terroristas. Esses *honeypots* não podem emular serviços e as informações residentes precisam parecer as mais verdadeiras possíveis, inclusive as organizações devem disponibilizá-las para *downloads* [37].

Como sugere Lance Spitzner [47], o *honeypot* deve ser "o mais doce possível", ou seja, tanto o sistema operacional quanto os serviços e informações têm que parecer para o atacante ou intruso, os mais realistas possíveis. Porém, *honeypots* menos sofisticados podem ser utilizados para prevenirem ataques baseados em ferramentas automatizadas, reduzindo o risco de maiores danos ao ambiente de rede, fornecendo informações que possam levar a administração da rede a revisar a configuração de sistemas operacionais e serviços, forçar o uso de senhas e autenticação mais fortes, além de usar *firewalls*, IDS's, criptografia e outros mecanismos e políticas de segurança [39].

Quanto à detecção, há um vasto potencial para a utilização de *honeypots*, o que é reforçado por Spitzner em [39]. Segundo o autor, os maiores desafios para a comunidade de segurança são os falsos positivos, falsos negativos e a agregação de dados sobre incidentes de segurança.

Falso positivo ocorre quando um sistema (principalmente IDS) acusa a ocorrência de um evento como sendo uma atividade maliciosa. Porém, na realidade, tudo não passa de um falso alarme. No caso de um falso negativo, o incidente ocorre de fato, porém, o sistema não consegue detectá-lo.

Já a agregação de dados constitui um problema porque é extremamente difícil consolidar várias bases de informações sobre atividades maliciosas devido à heterogeneidade de dispositivos, tecnologias e padrões utilizados em segurança de redes. Pela monumental

quantidade de dados gerados por esses sistemas, inferir com precisão se uma atividade é ou não maliciosa e oferece risco ao ambiente é quase sempre impossível [47].

Enquanto os mecanismos convencionais de segurança, notadamente IDS's, falham em resolver esses problemas, um *honeypot*, devido à sua concepção, consegue efetivamente vencer esses desafios. Por não se envolver no tráfego de produção do ambiente, há pouca atividade que leve um *honeypot* a gerar falsos positivos. A menos que haja um erro de configuração ou alguém tente acessar um endereço errado, um *honeypot*, na maioria dos casos, gera alertas confiáveis, o que pode reduzir o grau de incertezas sobre uma atividade.

Além disso, como os alertas dos *honeypots* se baseiam em atividades de sistema e tráfego de rede e não em assinaturas ou padrões como ocorre com IDS's, o número de falsos negativos em *honeypots* tendem a ser mínimos. Os *honeypots* também não requerem as constantes atualizações de bases de dados de assinaturas. Eles simplesmente trabalham com a atividade de rede que lhes for direcionada.

Outro problema da detecção em que os mecanismos tradicionais como *firewalls* e IDS's, deixam a desejar, é quanto à agregação de dados. Devido à grande massa de dados capturada pelas tecnologias mais modernas de segurança de redes, que chega facilmente na ordem de gigabytes [39] por dia em redes movidas a tecnologias baseadas em gigabites por segundo, esses dispositivos apresentam grandes dificuldades para coletar e tratar esses dados e responder a tempo de evitar maiores danos. Entretanto, *honeypots*, devido à concepção, produzem quantidades de dados bem menores. Segundo o Projeto *Honeynet* [47], um *honeypot* de pesquisa coleta, em média, menos de 1MB de dados por dia. Além disso, esses dados são de elevada relevância por conterem somente atividades não-autorizadas, ou seja, tráfego potencialmente malicioso por natureza.

A título de ilustração, suponha-se uma DMZ (Zona Desmilitarizada) como mostrado na Figura 2.1.

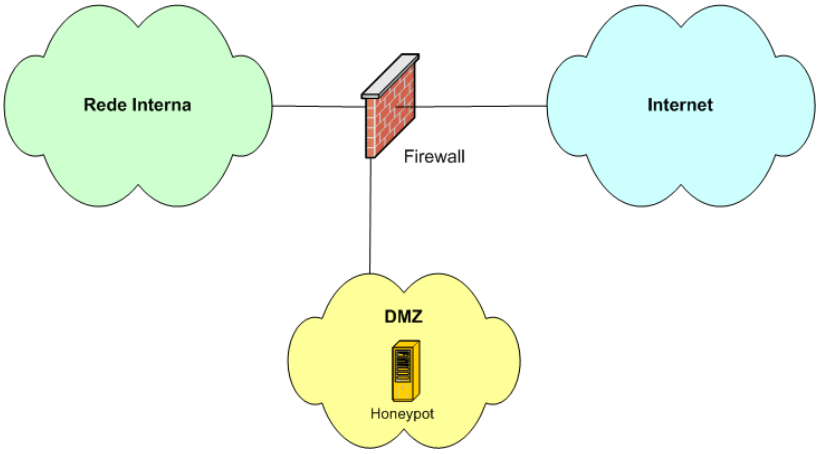


Figura 2.1 – Honeypot implantado na DMZ

Nessa região (ou zona) encontram-se os serviços disponíveis para o mundo externo de uma rede, isto é, a Internet. É nela onde ficam, geralmente, os serviços de web, correio eletrônico, ftp e outros serviços públicos. Esses serviços são altamente vulneráveis, pois qualquer um na Internet pode iniciar uma conexão para os mesmos, de modo que, mesmo com todo o aparato de segurança lógica, são alvos constantes de ataques e invasões.

A detecção, neste caso, é crítica. O problema é que, devido à grande quantidade de atividade de produção, a detecção de padrões de ataques nessa região é muito difícil. Por conta disso, na maioria das vezes, os administradores ignoram os alertas, independentemente de serem ataques de fato (verdadeiros positivos) ou falsos positivos. Também, por conta da grande quantidade de tráfego, a agregação de dados se torna um desafio, o que leva a problemas na detecção de falsos negativos.

Um *honeypot* poderia ser implantado nesta região (DMZ) para ajudar a detectar ataques. Seguindo o conceito de *honeypots*, por não terem valor de produção, além de não terem referências nos servidores de nomes, nem estarem ligados a outros sistemas de produção, sua função seria tipicamente para detectar ataques. Por conta da pouca atividade experimentada por esses dispositivos e, pela alta relevância desses dados, o número de falsos positivos iria diminuir consideravelmente. Se houvesse alguma conexão a partir da Internet

para ele, isso indicaria que aquela fonte estaria sondando, procurando por vulnerabilidades ou atacando os sistemas contidos na DMZ. Se as conexões tivessem origem nos servidores da própria DMZ, significaria que os serviços ali presentes já haviam sido comprometidos e agora estariam sendo utilizados para sondarem ou atacarem outros sistemas vulneráveis.

Assim, por detectarem e registrarem somente a atividade que lhes é direcionada e, por natureza, não-autorizada, os *honeypots* também ajudam a reduzir a quantidade de dados coletados, tornando a agregação de dados mais fácil. Apesar de não poderem detectar sondagens e tentativas de ataques nos sistemas de produção, *honeypots* podem ser ferramentas bastante úteis na detecção e prevenção de ataques, na medida em que também reduzem falsos negativos que constituem uma ameaça invisível para o ambiente de produção.

Porém, conforme já citado, *honeypots* não constituem uma alternativa a IDS's e *firewalls* ou a qualquer outro mecanismo ou estratégias de segurança de redes. Eles notadamente ajudam a detectar atividades não-autorizadas, o que é de significativa importância, pois o tráfego não-autorizado com destino a eles pode ser impedido de chegar até os recursos de missão crítica do ambiente.

Quando um ataque bem-sucedido a um sistema de produção é detectado, normalmente a organização aplica as correções de sistemas operacionais numa tentativa de restabelecer o serviço aos usuários e inibir novos ataques. Quase que invariavelmente, o pensamento comum é retornar à normalidade. Nada é feito para investigar o incidente, saber como ele ocorreu ou se ficou alguma evidência do ataque ou atacante no *host* afetado. Isso ocorre porque, na maioria das vezes, as organizações não podem ficar sem o serviço ativo por algum tempo para uma análise mais detalhada do incidente. Assim, a habilidade de responder a incidentes fica restrita a intervenções que, muitas vezes, prejudicam a captura de evidências.

Quanto à capacidade de responder a um incidente, os *honeypots*, por não terem atividade de produção ligada a eles, podem suportar paradas programadas para a análise de

evidências depois de ataques ou comprometimentos. Assim, detalhes de como um atacante conseguiu acesso a um sistema, o que ele fez após obter o acesso desejado, se deixou portas de fundo preparadas para a sua volta, se modificou, destruiu ou capturou informações críticas, podem ser obtidos pela análise criteriosa de *honeypots* comprometidos. Os resultados alcançados podem ser de significativa importância para a redução de riscos de ataques e invasões nos sistemas de produção.

Spitzner em [39] ilustra como exemplo, uma rede DMZ onde múltiplos servidores de web (recursos de produção) que fazem balanço de carga e ajudam a aumentar a performance do serviço. Nesse ambiente, um *honeypot* é colocado com o propósito de detectar ataques. Este, apesar de executar as mesmas aplicações daqueles servidores de web, por não ter entradas nos servidores de nomes, não deverá receber qualquer requisição para fornecer páginas web, o que implica em não ter tráfego de produção.

Se um atacante disparar sondagens seqüenciais para cada sistema na DMZ (normalmente, os endereços IP são parte de uma faixa bem definida), buscando vulnerabilidades ou simplesmente procurando por sistemas ativos, é praticamente certo que o *honeypot* será sondado. Isto é válido também quando sistemas rodando serviços semelhantes são comprometidos. Fatalmente, o atacante terá usado as mesmas ferramentas e métodos em todos os sistemas, incluindo o *honeypot*. Apesar dos sistemas de produção não poderem ser desativados para a coleta e inspeção de evidências, os administradores podem utilizar o *honeypot* na análise do incidente e, então, aplicar as lições aprendidas aos outros sistemas da DMZ.

2.3.3.2 *Honeypots de pesquisa*

É de consenso geral que conhecimento é poder, ou seja, quem tem informação pode transformar o ambiente em que vive. Segundo [47], quanto melhor se conhece o

inimigo, mais chance se tem de derrotá-lo. Esta máxima é muito utilizada pela inteligência de guerra. Aliás, o termo militar utilizado para definir informação é inteligência.

A obtenção de informações sobre a comunidade de hackers e outros entes mal-intencionados que podem comprometer a integridade, a disponibilidade e a confiabilidade de um sistema ou rede é um dos maiores desafios que a comunidade de segurança enfrenta. Coletando informações sobre as atividades maliciosas, as organizações podem parar um ataque ou corrigir uma falha na defesa antes dela ser utilizada pelo inimigo.

O administrador de segurança, além de, na maioria das vezes, não dispor de ferramentas e pessoal suficiente para auxiliá-lo, tem uma tarefa árdua a cumprir: proteger um sistema ou rede contra uma ameaça, um inimigo invisível, não conhecendo as suas táticas ou os motivos que levam o atacante a investir contra determinado sistema-alvo, ficando praticamente de braços atados, ou seja, não consegue realizar seu trabalho satisfatoriamente. Geralmente, após um ataque, o máximo que se sabe são algumas conseqüências óbvias, ou seja, o que foi danificado ou que serviço não está disponível. Questões sobre como a comunidade *blackhat*[iv] identifica e sonda os sistemas vulneráveis, o que acontece depois que um sistema é comprometido, se há outros sistemas vulneráveis no ambiente normalmente não podem ser respondidas [50].

Nesse sentido, os *honeypots* têm, talvez, sua principal aplicação. Os *honeypots* de pesquisa têm a finalidade de obter informações sobre um atacante, suas ferramentas, táticas e métodos. Assim, ao invés de apenas encontrar os estragos após o comprometimento de um ou mais recursos, a comunidade de segurança pode observar todo o processo, desde a sondagem até a posse do sistema-alvo pelo atacante. Com *honeypots* de pesquisa é possível observar ação, passo a passo, tecla a tecla.

[iv] Termo utilizado pela comunidade de segurança para se referir, de modo geral, a *hackers*, *crackers*, *script kiddies*, dentre outros.

Segundo [2], o valor de tais informações é altíssimo, e oferece uma variedade de usos potenciais, desde a captura de *worms* e ferramentas, passando por técnicas desconhecidas, podendo ajudar a entender melhor o motivo que levou o atacante a investir naquele sistema ou rede. Podem ainda servir como mecanismos de alerta, predizendo futuros ataques ou mostrando as fraquezas de sistemas de produção similares aos dos *honeypots* de pesquisa.

Apesar de não reduzirem diretamente o risco de ataques para uma organização, os *honeypots* de pesquisa podem fornecer subsídios significativos para melhorar a prevenção, a detecção e a resposta a incidentes de segurança.

Seu uso não é indicado para organizações que queiram aumentar o nível de segurança de seu ambiente de rede. Neste caso, é aconselhável, segundo [39], a utilização de *honeypots* de produção, pela facilidade de implementação e manutenção.

Honeypots de pesquisa são indicados como o nome já diz, para organizações que se preocupam com pesquisa e que estão interessadas em aprender mais sobre a comunidade *hacker*, tipicamente, governos, universidades e grandes companhias que têm infra-estrutura de suporte a tais *honeypots*. O Projeto *Honeynet* [39] e [47] é um exemplo de uma organização que utiliza *honeypots* de pesquisa para obter informações sobre a comunidade *blackhat*. Tais informações são, posteriormente, difundidas em fóruns, conferências, e outros eventos sobre segurança de redes. Desse grupo participam diversos profissionais, estudantes e cientistas. Existem até psicólogos, que traçam o perfil de atacantes. No Brasil, o INPE possui um grupo chamado *Honeynet*.BR, que é inspirado nesse projeto e que vem contribuindo em muito para o Projeto *Honeynet* [13].

Lance Spitzner em [47] alerta que *honeypots* não podem ser soluções baseadas em produtos puramente comerciais, isto é, eles não são produtos que funcionam assim que você os tira da caixa. O autor diz ainda que *honeypots* e *honeynets* precisam ser cultivados [49].

2.4 Classificação dos *honeypots*

Um *honeypot* pode ser classificado de acordo com o seu nível de envolvimento, ou seja, o nível de interação que um atacante ou intruso tem com o sistema operacional ou serviço disponível em um *honeypot*. Há três tipos de níveis de interação: baixo, médio e alto [2]. Cada nível tem suas vantagens e desvantagens, conforme descrito a seguir:

2.4.1 *Honeypots* de baixo nível de interação

Esse tipo de *honeypot* somente fornece certos serviços, que são, geralmente, emulações de sistemas operacionais ou aplicações servidoras. São *scripts* ou aplicativos que não oferecem ao atacante todas as funcionalidades esperadas de um sistema operacional de fato ou da aplicação servidora na qual o mesmo é baseado. Esses *honeypots* apresentam-se como uma conexão unidirecional. Assim, o atacante não pode interagir da mesma forma como se estivesse em contato com um sistema real, o que constitui uma desvantagem, pois, o agressor ao perceber tal fato, identifica facilmente que se trata de um *honeypot*. Como exemplo de *honeypots* de baixo nível podem ser citados: o DTK (*Deception Toolkit*) [5] e o *Specter* [42].

2.4.2 *Honeypots* de médio nível de interação

Honeypots desse nível de interação são geralmente mais complexos que os *scripts* de baixo nível de interação, porém, ainda não fornecem um acesso real ao sistema operacional do alvo. Apesar disso, os serviços providos apresentam funcionalidades mais apuradas que aquelas encontradas em *honeypots* de baixo nível de interação, chegando a aumentar o risco de exposição do sistema, pois o atacante tem uma ilusão melhor de um sistema operacional ou serviço real.

Devido a isso, exigem cuidados maiores com relação à checagem de segurança. Com isso, deve haver a constante preocupação de que as versões desses *honeypots* devam ser mais seguras que as correspondentes versões reais. O *Honeyd* [25] é um bom exemplo desse tipo de investida.

2.4.3 *Honeypots* de alto nível de interação

Honeypots de alto nível de interação fornecem ao atacante a possibilidade real de interagir com um sistema operacional de verdade, o que aumenta consideravelmente o risco de exposição do alvo e, conseqüentemente, o sistema fica mais suscetível a ataques. Por outro lado, as possibilidades de obtenção de mais informações são maiores [2].

Assim, tudo o que for possível realizar em um sistema ou aplicação real, é permitido em um *honeypot* desse tipo. Isso exige cuidados maiores com o fluxo de dados, principalmente, de saída, para que um sistema comprometido não possa ser utilizado por *blackhats* ou, caso seja utilizado, minimize o alcance e os danos a outros recursos dentro do ambiente ou na Internet [47].

O objetivo desses *honeypots* é obter o máximo possível de informações sobre o incidente. Segundo [2], isso compensa o alto risco da exposição desses *honeypots*. O *Mantrap* [27] é um exemplo desse tipo de *honeypot*.

Para finalizar a Tabela 2.2, extraída de [2], resume as características dos *honeypots* em relação aos diferentes níveis de interação:

Tabela 2.2 – Características dos *honeypots* versus níveis de interação

Característica	**Nível de Interação dos Honeypots**		
	BAIXO	**MÉDIO**	**ALTO**
Interação com o Sistema Operacional do *Host*	Não	Não	Sim
Risco de Comprometimento do *Host*	Baixo	Médio	Alto
Coleta de Informações	Conexões	Requisições	Tudo
Chamariz para o Comprometimento do *Host*	Não	Não	Sim
Experiência para implantação	Baixa	Baixa	Alta
Experiência para desenvolvimento	Baixa	Alta	Alta
Tempo gasto para manutenção	Baixo	Baixo	Muito Alto

2.5 Posicionamento de *honeypots*

Normalmente, não há um ponto certo para se posicionar um *honeypot* num ambiente de rede. A rigor, um *honeypot* pode ser colocado em qualquer local onde um servidor pode ser instalado. Porém, dependendo da abordagem, ou seja, da estratégia de implantação dos *honeypots*, algumas regiões são mais propícias que outras [2]. Por exemplo, o posicionamento de um *honeypot* na região da Intranet é interessante para a detecção de atividades suspeitas dentro da rede privada.

Assim, um *honeypot* pode ser colocado nas seguintes regiões:

- Em frente ao *firewall*, ou seja, diretamente na Internet;
- Atrás do *firewall*, ou seja, na Intranet;
- Na DMZ;
- Em outra interface do *firewall.*

A Figura 2.2 mostra graficamente essas regiões:

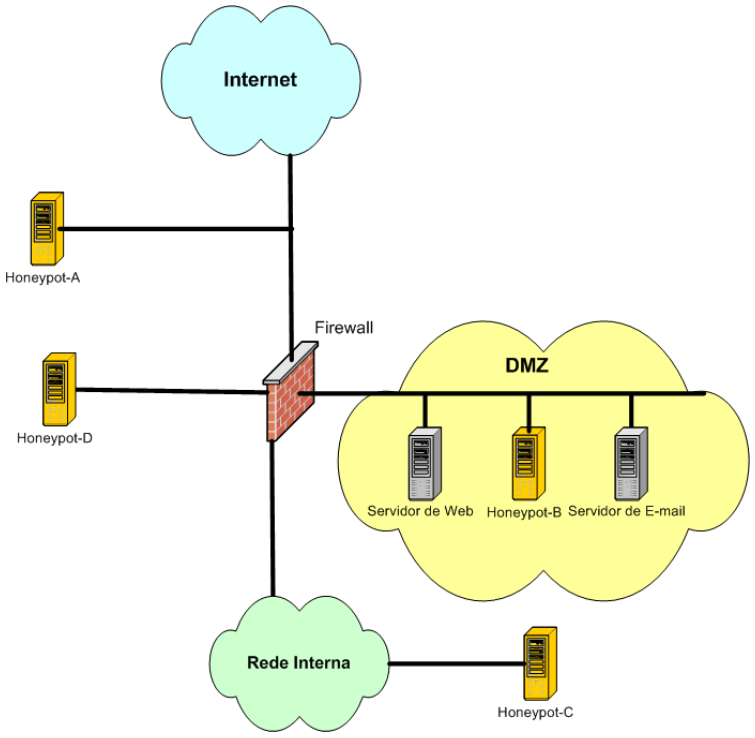


Figura 2.2 – Regiões para o posicionamento de *honeypots*

2.5.1 Na Internet

Nesta configuração podem-se detectar varreduras de portas e outros tipos de ataques aos recursos da rede que têm endereços e serviços válidos na Internet.

Colocando-se *honeypots* em frente ao *firewall* (*Honeypot*-A na Figura 2.2), é o mesmo que se ter um recurso com endereço IP válido exposto diretamente na Internet. Como existe um *firewall* entre esse *honeypot* e a rede interna, o mesmo não aumenta o risco para o ambiente. Porém, devem-se tomar medidas para que o *firewall* impeça o acesso desse *honeypot* ao ambiente interno [2].

2.5.2 Na Intranet

Um *honeypot* atrás do *firewall*, ou seja, dentro da rede local (*Honeypot*-C na Figura 2.2), potencialmente introduz novos riscos para o ambiente, principalmente se não houver um esquema de segurança interna. Isto pode ser conseguindo utilizando-se *firewalls* adicionais contra possíveis investidas, a partir dos *honeypots* internos [2]. O maior risco ocorre quando um *honeypot* interno é comprometido por um atacante externo. Por esse motivo, o *firewall* de entrada da Internet deve conter regras mais restritivas quanto à exposição de recursos da rede interna.

A vantagem de se colocar um *honeypot* dentro da rede local é que é possível detectar atacantes internos. Segundo pesquisas mais de 23% dos ataques e invasões provêm da rede interna [21] e [30].

2.5.3 Na DMZ

Honeypots colocados nessa região (*Honeypot*-B, Figura 2.2) podem ser bastante úteis para se detectar ataques nos recursos e serviços disponíveis publicamente. Entretanto,

deve-se endurecer a segurança dos outros recursos da DMZ e da rede interna para que, uma vez comprometido, um *honeypot* não seja utilizado para alcançar esses recursos [2].

Honeypots também podem ser posicionados em uma outra interface do *firewall*, ficando isolado das outras regiões, podendo formar, inclusive uma *honeynet*. Na Figura 2.2, o *honeypot*-D exemplifica essa possibilidade.

2.6 Estratégias para Implantação de *honeypots*

Há basicamente três tipos de estratégias:

- *Minefield;*
- *Shield*;
- *Honeynet*s.

2.6.1 *Minefield* (Campo Minado)

Nesta estratégia, os *honeypots* são colocados entre os servidores reais como espelhos dos serviços destes. Quando um invasor varre a rede é praticamente garantido que os *honeypots* e suas vulnerabilidades também sejam detectadas [27]. A Figura 2.3 ilustra a implantação de *honeypots* segundo uma estratégia de campo minado.

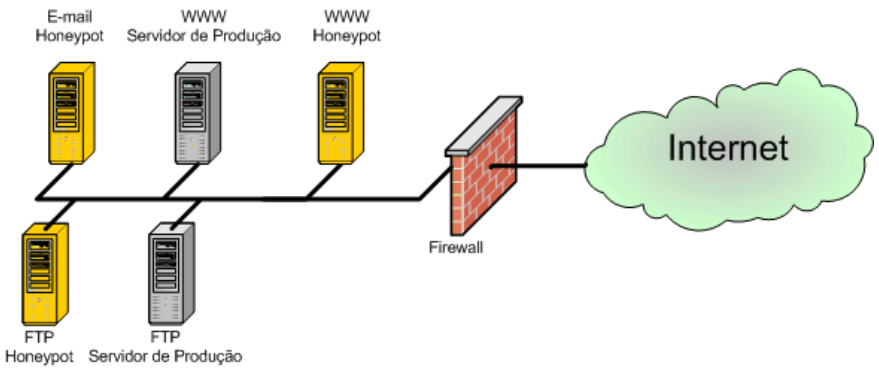


Figura 2.3 – *Honeypots* e Servidores de Produção na Configuração *Minefield*

2.6.1.1 *Vantagens*

- O alerta de invasão é livre de incertezas, pois, parte-se do pressuposto de que tráfego de entrada ou de saída de um *honeypot* é originado pelo intruso;
- Faz o atacante perder tempo, ou seja, o atacante tentará explorar as vulnerabilidades encontradas naquele *honeypot* e quanto mais tempo demorar melhor, pois ele não estará tentando invadir um sistema de produção;
- Facilita a correção de problemas nos servidores reais, pois, por exemplo, tendo-se um sistema de produção e um *honeypot* configurados identicamente, quando o *honeypot* informar que ocorreu um ataque bem sucedido, o mesmo poderia ter ocorrido no servidor de produção. Com base nessas informações, fica mais fácil corrigir o problema [47].

2.6.1.2 *Desvantagens*

- Precisa de um sistema de contenção forte e coleta de informações garantida [2];
- Não evita nem reduz as chances de que o servidor real sofra um ataque [27].

2.6.2 *Shield* (Escudo)

Nesta estratégia, cada *honeypot* fica emparelhado com um servidor real. Assim, o tráfego regular de e para o servidor real não é afetado e o tráfego suspeito é desviado para o *honeypot* [27]. Um exemplo de implantação de *honeypots* e servidores de produção, na estratégia *shield*, encontra-se ilustrado na Figura 2.4.

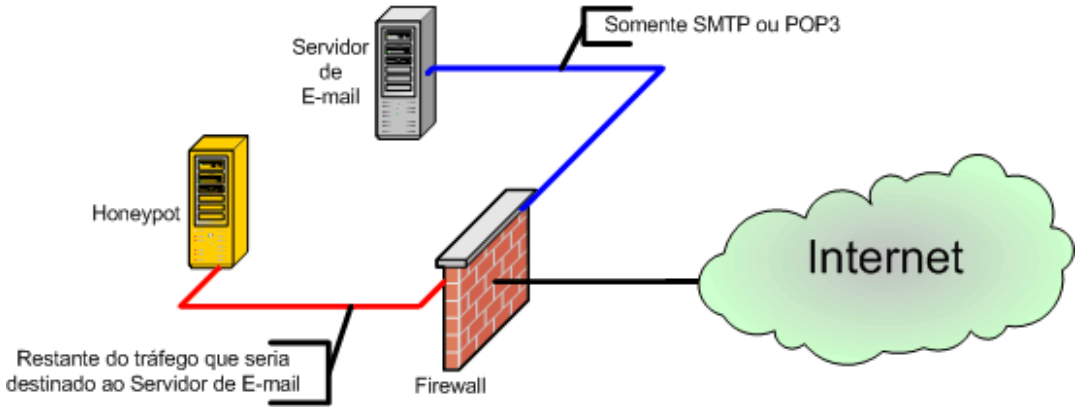


Figura 2.4 – Exemplo de implantação de *honeypots* e servidores de produção, na configuração *shield*

2.6.2.1 *Vantagens*

Assim como na estratégia *Minefield*:

- O alerta de invasão é livre de incertezas, pois parte-se do pressuposto de que tráfego de entrada ou de saída de um *honeypot* é originado pelo intruso;
- Faz o atacante perder tempo, ou seja, o atacante tentará explorar as vulnerabilidades encontradas naquele *honeypot* e quanto mais tempo demorar melhor, pois ele não estará tentando invadir um sistema de produção.

Além disso:

- Reduz a possibilidade de que um servidor real seja atacado, pois há uma filtragem do tráfego que deveria chegar ao servidor de produção, minimizando, assim, as vulnerabilidades que seriam detectadas e exploradas naquele servidor [27].

2.6.2.2 *Desvantagens*

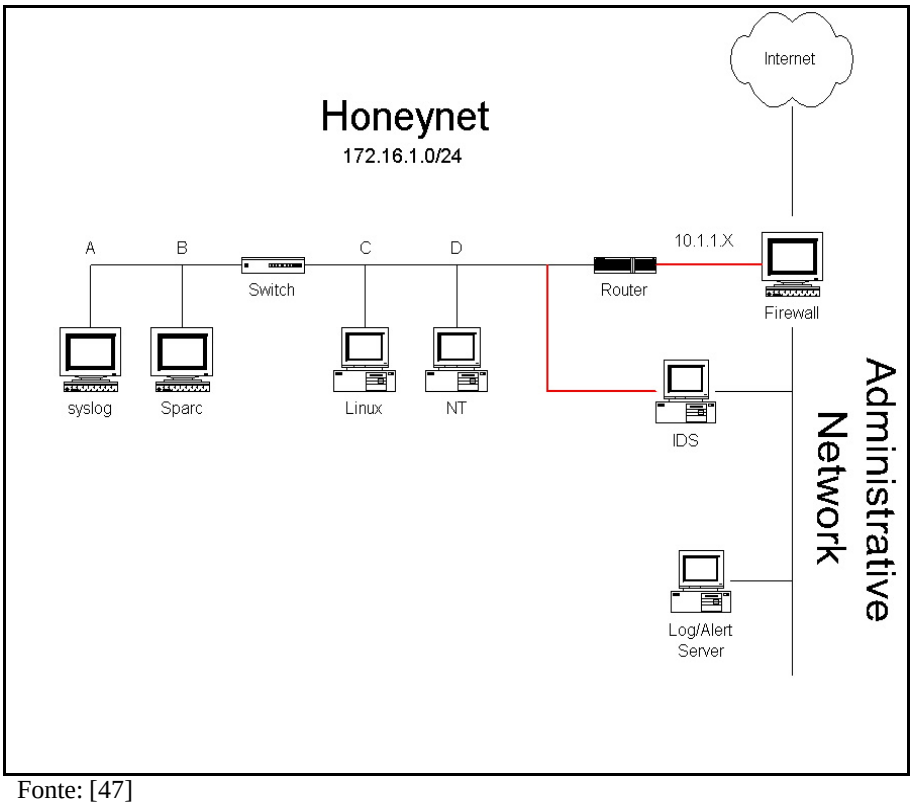
- Também necessita de um sistema de contenção forte e coleta de informações garantida [2];

- Não evita que o servidor real sofra um ataque com base no serviço que este oferece. Por exemplo, em um *Web Server*, se o serviço *http* tiver alguma vulnerabilidade conhecida e não corrigida, o atacante poderá ter êxito na sua investida.

2.6.3 *Honeynet*

Esta estratégia é uma extensão do conceito de *honeypot* onde múltiplas iscas formam uma rede criada para ser atacada e comprometida. Ou seja, nenhum servidor tem valor de produção. Seu objetivo é capturar e controlar o fluxo de dados. Cada pacote que entra ou sai é considerado suspeito por natureza [49].

Sua configuração pode conter escudos e campos minados. As possibilidades são infinitas. Na Figura 2.5 é mostrado um exemplo de *Honeynet* [2].



Fonte: [47]

Figura 2.5 – Exemplo de uma *Honeynet*

2.6.3.1 *Vantagens*

- O alerta de invasão é totalmente livre de incertezas;
- Faz o atacante perder tempo;
- Consegue-se aprender sem colocar em risco o ambiente de produção.

2.6.3.2 *Desvantagem*

- Também necessita de um sistema de contenção forte e coleta de informações garantida [2].

2.7 Aspectos sobre a legalidade dos *honeypots*

Segundo [31], quando o assunto é legalidade dos *honeypots*, é necessário considerar três aspectos: armadilha, privacidade e responsabilidade.

2.7.1 Armadilha

Coagir ou induzir alguém a fazer algo que normalmente não faria, ou seja, instigar a prática de um delito, pode acarretar processo judicial. A discussão é intensa sobre esse aspecto, mas na maioria dos casos não poderia ser caracterizado crime pelas seguintes razões [38]:

- *honeypots* não induzem ninguém, até porque muitas vezes são emulações dos sistemas de produção da empresa;
- os ataques são por iniciativa do invasor;

2.7.2 Privacidade

O sistema que o atacante está usando não pertence a ele. Portanto, toda monitoração realizada no sistema não pode caracterizar quebra de privacidade [31].

2.7.3 Responsabilidade

Se o *honeypot* for comprometido e utilizado para prejudicar outras redes, pode acarretar processo civil [38].

2.8 Principais *honeypots* disponíveis

2.8.1 DTK

Em 1997, sete anos depois da publicação de [44], Fred Cohen tornava disponível a primeira solução pública baseada no conceito de *honeypot*: a versão 0.1 do *Deception ToolKit* (DTK). O objetivo do DTK não era somente obter informações sobre o atacante, mas enganá-lo e psicologicamente confundi-lo, dando aos administradores um arsenal de vantagens sobre seus oponentes [6].

O DTK é baseado numa série de *scripts* (linguagem Perl e C e ambiente de execução Unix) que dão a impressão de que o sistema apresenta diversas vulnerabilidades conhecidas, tais como um arquivo de senhas falso, característico de uma velha vulnerabilidade de *Sendmail*. É uma alternativa bastante simples e gratuita, porém, é facilmente reconhecida pelo atacante, principalmente por não oferecer ao atacante um ambiente de interação convincente [5], o que o classifica como um *honeypot* de baixo nível de interação. A Figura 2.6 mostra a interface gráfica do DTK.

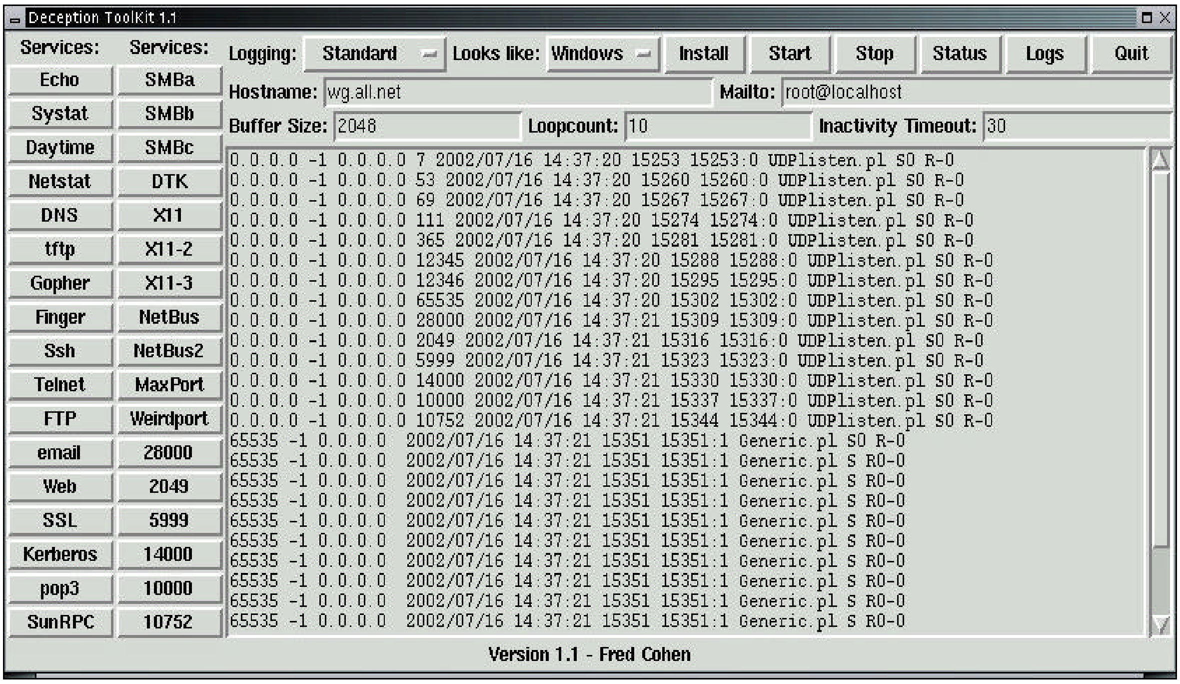


Figura 2.6 – Interface do DTK

2.8.2 *CyberCop Sting*

O *CyberCop Sting* foi o primeiro produto comercial baseado no conceito de *honeypot*. Desenvolvido originalmente por Alfred Huger, da Secure Networks Inc., o produto foi, posteriormente, comprado pela Network Associates Inc. (NAI), porém, não conseguiu alcançar parcelas significativas do mercado e, então, foi descontinuado [39].

Ao contrário do DTK, ele rodava em Windows NT e, além disso, podia emular diferentes sistemas simultaneamente, especificamente, roteadores Cisco, servidores Solaris e sistemas NT, podendo emular uma rede inteira. Embora tenha sido tecnologicamente mais avançado que o DTK, o *CyberCop Sting* ainda apresentava como desvantagem a limitação de que os sistemas emulados não ofereciam uma interface real que possibilitasse acesso real ao sistema operacional. Assim, esse *decoy server* era considerado um *honeypot* de médio nível de interação [39].

2.8.3 *Specter*

O *Specter* é uma ferramenta de decepção disponível comercialmente pela empresa NeoWorx[v]. Roda sobre o Windows NT/2000, não requerendo um *hardware* de grande poder de processamento. Sua placa de rede é configurada no modo promíscuo, capturando todos os pacotes que passam no barramento, os quais são confrontados com uma base de assinaturas de padrões de ataques. O *Specter* pode investigar a origem de um atacante através de registros WHOIS, comandos *finger* e varreduras de portas, além de *traceroute* [42].

O *Specter* pode simular nove sistemas operacionais diferentes (*Windows* NT, *Windows* 95/98, MacOS, *Linux*, SunOS/Solaris, *Digital* UNIX, neXTStep, Irix e *Unisys* UNIX) e permite cinco tipos de configurações diferentes: máquina pessimamente configurada, máquina apresentando defeito de *hardware* e *software*, comportamento estranho (imprevisível) e modo agressivo (coleta informações sobre o atacante e depois se anuncia). Além disso, o *Specter* também simula cinco serviços de rede (SMTP, FTP, *Telnet*, *finger*, Net-Bus), sete vulnerabilidades conhecidas, tais como DNS, Sun-RPC, POP3, IMAP4 e *Back Orifice*[vi], possibilitando ainda ao administrador monitorar uma porta específica [2].

Sua unidade de controle baseia-se numa pequena janela de status onde são providos detalhes de alertas em tempo real. O administrador também pode ser alertado via e-mail. Os *logs* também são armazenados em disco [42].

Quando o *Specter* detecta uma conexão suspeita, ele pode ser configurado para executar um comando *finger* e/ou uma varredura de portas para obter informações mais detalhadas do *host* atacante [2].

---

[v] http://www.neoworx.com/products/specter/

[vi] Poderoso trojan (Cavalo de Tróia) que permitia a atacantes controlarem remotamente máquinas Windows 9x, capturando, inclusive, pressionamento de teclas. Uma vez instalado, permitia que o atacante tivesse mais controle sobre a máquina que seu usuário legítimo.

O *Specter* é classificado como um *honeypot* de baixo nível de envolvimento, pois somente simula serviços, sistemas operacionais e vulnerabilidades, não oferecendo ao atacante interação real com o sistema [2].

2.8.4 BOF

O *BackOfficer Friendly* (BOF) é um produto gratuito, roda sobre o Windows e Unix e opera basicamente como o *Specter*, porém, de modo mais simples. Apresenta um pequeno conjunto de serviços que simplesmente ficam "escutando" portas. É, portanto, um *honeypot* de baixo nível de envolvimento. Seu projeto inicial era para servir como alarme de contra atacantes, alertando quando alguém estivesse fazendo varreduras nos sistemas [23].

O BOF é uma aplicação servidora de *spoofing*[vii], que notifica os administradores quando alguém tenta controlar remotamente um sistema utilizando *Back Orifice* [23].

Segundo [47], o BOF não pode ser tratado como um *honeypot* sério, mas pode ser muito útil na detecção de ataques a uma estação de trabalho. Porém, segundo [39], devido à sua simplicidade e seu custo (gratuito), o BOF é uma excelente ferramenta para demonstrar os conceitos de *honeypots.*

2.8.5 sNET

O sNET (*Secure Network*) é um sistema distribuído que utiliza unidades de decepção entre os seus principais nós, monitoradas e controladas por uma unidade de gerenciamento, objetivando inspecionar, detectar e proteger a rede [34].

[vii] Uma técnica utilizada para ganhar acesso não-autorizado a computadores onde o atacante envia mensagens a um computador utilizando um endereço IP que indica que a mensagem está vindo de um *host* confiável.

Cada unidade tem um papel bem definido e, para tanto, são configuradas apropriadamente e colocadas em posições estratégicas, de modo a formarem um sistema coeso.

Nem todos os elementos que compõem o sNET são unidades de decepção. Existem unidades que servem para a recepção de eventos, alertas para o reconhecimento e obtenção de pistas (análise forense), visualização de um ataque em tempo real, dentre outras unidades que constituem os nós ou ilhas de segurança dentro do sistema e da rede a ser protegida [15].

2.8.6 *Mantrap*

O *Mantrap* é um sistema disponível comercialmente desenvolvido pela *Recourse Technologies*[viii] que visa monitorar e trilhar atividades de intrusão em tempo real [27].

É baseado no conceito de "prisão", onde uma máquina pode suportar até 04 (quatro) *honeypots* virtuais diferentes, completamente isolados (logicamente) uns dos outros, inclusive do *host* real, que hospeda e gerencia a aplicação *Mantrap*. O interessante desse sistema é que ele não emula um sistema operacional, mas, segundo [48], executa uma imagem de um sistema operacional dentro de outro. Entretanto, há uma limitação quanto aos sistemas operacionais que o fabricante pode fornecer e quanto ao *hardware* necessário para rodá-lo: somente sistemas operacionais Solaris [39].

Cada "prisão" pode ser configurada independentemente das outras. Um ponto interessante é que o *Mantrap* possui um mecanismo de customização automática (*Mantrap's Content Generation Module* - CGM) o qual cria conteúdos únicos para cada *honeypot*, dificultando a identificação de uma máquina virtual pelo invasor [27].

[viii] Adquirida posteriormente pela Symantec.

Um problema apontado pela equipe do Projeto *Honeynet* é que os usuários devem solucionar o problema de como conter o intruso depois que o sistema estiver comprometido, pois o *Mantrap* não tem a capacidade de limitar a atividade do invasor [39].

2.8.7 Alternativas caseiras ou acadêmicas

Embora algumas armadilhas de rede estejam disponíveis para implementação imediata, conforme sugere [2], muitos interessados em segurança de rede estão construindo suas próprias implementações, baseados em informações detalhadas que podem ser obtidas em sites na Internet. Muitas vezes a tecnologia é baseada num misto de ferramentas disponíveis gratuitamente ou comercialmente tais como VMWare, VNC, Linux e Snort, dentre outras opções [14] e [41].

Ainda segundo [2], esses *honeypots* chegam a oferecer um alto nível de envolvimento. Infelizmente, muitos não oferecem filtros de tráfego de saída, possibilitando o uso dessas unidades, uma vez comprometidas, como ponto de partida para outros ataques.

A vantagem dessas alternativas caseiras, segundo [39], é que eles possibilitam a implementação mais rápida de novas idéias, o que não acontece tão facilmente com os produtos comercialmente disponíveis no mercado.

2.8.7.1 *Honeyd*

Inicialmente desenvolvido por Niels Provos[ix], o *Honeyd* ganhou adeptos no mundo inteiro, principalmente na área acadêmica, como uma alternativa de baixo custo (*opensource*) e relativamente fácil de ser implementado. O *Honeyd* é um *framework*

[ix] Niels Provos é atualmente engenheiro de *software* no Google, PhD em *Computer Science and Engineering* pela Universidade de Michigan [24].

disponível livremente para a implementação de *honeypots* virtuais. Com ele, é possível configurar *honeypots* com diferentes personalidades e serviços em uma única máquina [25].

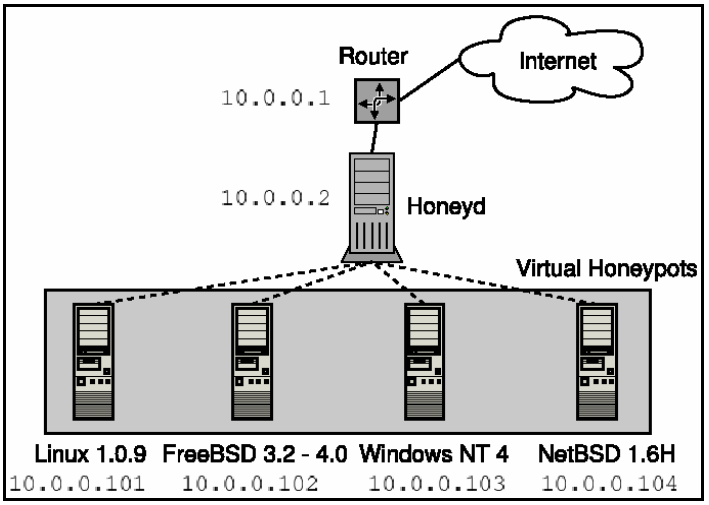
O *Honeyd* emula as diferentes pilhas IP dos principais sistemas operacionais e associa certos *scripts* a uma porta desejada para emular um serviço específico. O *Honeyd* é capaz de enganar ferramentas de *fingerprinting*[x] de rede fazendo "pensarem" que se trata de um sistema operacional real [24].

Suas principais características são:

- Simula sistemas, executando em espaços de endereçamento não alocados;
- Simula diversos *hosts* virtuais ao mesmo tempo;
- Permite a configuração de vários serviços;
- Simula um SO no nível de pilha do TCP/IP, enganando o *nmap* e o *xprobe;*
- Suporta redirecionamento de um serviço para outras máquinas em vez de o simular;
- Suporta somente os protocolos TCP, UDP e ICMP;
- Recebe o tráfego de rede, utilizando o *proxy* ARP (*arpd*) e através de roteamento específico para os endereços IP virtuais.

A Figura 2.7 exemplifica o *Honeyd* recebendo o tráfego de seus *honeypots* virtuais através de um roteador ou *Proxy* ARP. Para cada *honeypot*, o *Honeyd* pode simular o comportamento da pilha de rede de um sistema operacional diferente [29].

[x] Sistemas operacionais, aplicações servidoras e dispositivos de rede, em geral, apresentam certas características que funcionam como uma impressão digital Tais padrões podem ser capturados através da rede e, assim, pode-se identificar o SO ou qual a aplicação que está rodando remotamente. A essa técnica dá-se o nome de f*ingerprinting* [19].



Fonte: [25]

Figura 2.7 – Exemplo de aplicação do *Honeyd*

2.8.8 Projeto *Honeynet*

Honeynet não é um sistema único, mas sim uma rede de sistemas e aplicativos múltiplos (daí o nome *honeynet*), projetada para ser comprometida e observada. *Honeynet* é um tipo de *honeypot* de alto nível de interação, utilizada principalmente para pesquisa [47].

É criado um ambiente que reflete uma rede de produção real, onde os riscos e vulnerabilidades descobertos serão os mesmos que existem em ambientes de produção. Nada é emulado e nada é feito para tornar os sistemas inseguros [49]. A Figura 2.8 ilustra uma *honeynet*.

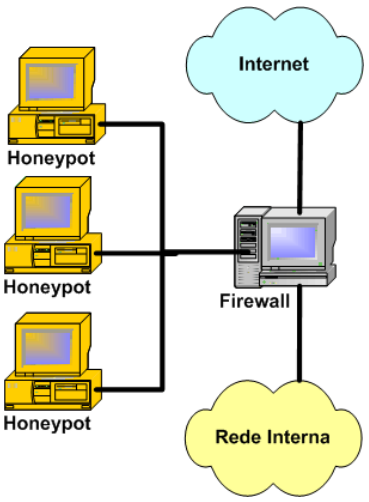


Figura 2.8 – Uma *honeynet*

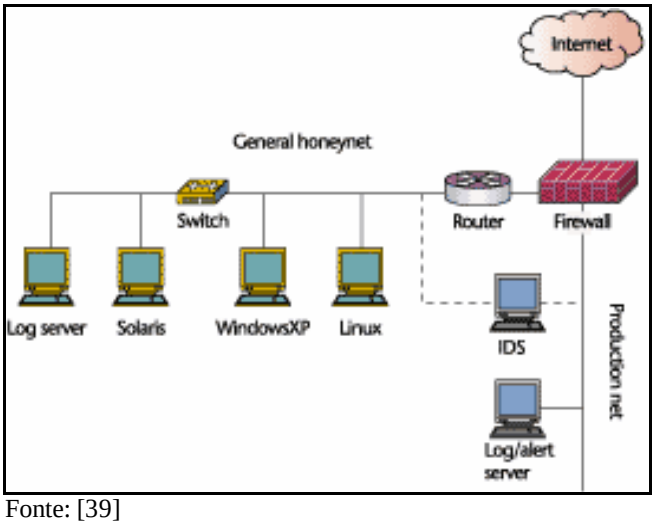
Em uma *honeynet* todo o tráfego é suspeito por natureza, caracterizando intrusão. Qualquer conexão efetuada deve ser monitorada, facilitando, assim a detecção de riscos, pois não há uma sobrecarga de dados ocasionada pelo tráfego normal da rede.

Em geral, *honeynets* não oferecem mecanismos de resposta automática pelo fato de que tudo o que é invadido nelas foi projetado para ser realmente comprometido. Porém, deve haver um mecanismo de contenção de pacotes e conexões que partem da *honeynet*, utilizado para evitar que a *honeynet* seja utilizada como ponto de partida para ataques a outras redes [39].

Seu monitoramento e gerenciamento são feitos através do envio de *logs* dos diversos dispositivos e unidades de decepção existentes e pela análise forense das unidades que tenham sido comprometidas [47].

O Projeto *Honeynet* é caracterizado por utilizar *honeynet*s baseadas em máquinas e sistemas operacionais reais, ou seja, nada é virtual ou emulado. Todos os elementos constituintes são unidades de decepção (com exceção das unidades de *log* e administração que servem para monitorar as atividades intrusivas), não tendo sistemas de produção de fato envolvidos. Tem o caráter puramente científico [47].

A Figura 2.9 mostra o esquema da primeira geração da *honeynet* desenvolvida pelo Projeto *Honeynet*.

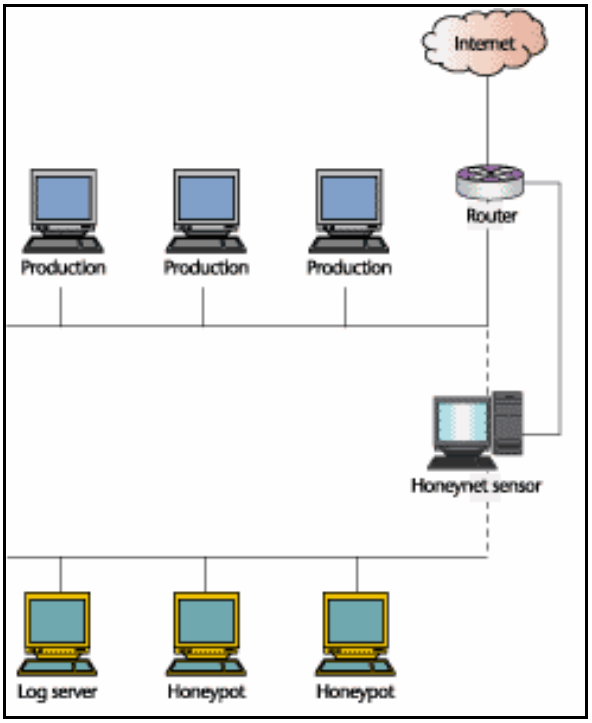


Fonte: [39]

Figura 2.9 – *Honeynet* Gen I

Pela Figura 2.9 percebe-se que a rede foi dividida em três partes: *Honeynet*, Internet e Rede Administrativa e que todos os pacotes trafegam através do *firewall* e roteador.

A finalidade do projeto Gen I (fase I) foi identificar e divulgar as ameaças mais comuns às instalações, além de implementam controle e captura de dados simples, mas efetivos. Com a expansão do projeto, surgiu o modelo Gen II ou fase II do Projeto *Honeynet* [31]. O modelo Gen II é mostrado na Figura 2.10.



Fonte: [39]

Figura 2.10 – *Honeynet* Gen II

A fase II visou a melhoria e simplificação da tecnologia empregada em *honeynets*, apresentando métodos mais avançados para controlar as atividades dos invasores e tornando mais difícil a descoberta do monitoramento, além da redução no risco de ataques a outras redes. Foi dada mais flexibilidade aos atacantes, proporcionado maior aprendizagem [40]. Por outro lado, o processo de controle, captura e coleção dos dados são feitos em um único dispositivo, facilitando a administração da *honeynet*. O *gateway*, dispositivo que separa as

redes, é denominado *Honeywall*; possui uma camada com duas pontes, permitindo integração e verificação de ameaças também nas redes internas [37].

O objetivo da Fase III foi o desenvolvimento de um CD-ROM de boot, contendo a tecnologia adotada na Fase II, numa tentativa de padronizará as *honeynets* e permitir o desenvolvimento de sistemas distribuídos. A idéia final é fazer com que os registros possam ser enviados para um sistema central e depois disponibilizados publicamente.

Atualmente o Projeto *Honeynet* encontra-se na Fase IV. A expectativa é desenvolver um sistema de coleta de dados centralizado para capturar os resultados das *honeynets* distribuídas mundialmente [35] e [36].

2.9 Quadro comparativo entre os principais *honeypots* disponíveis

Na Tabela 2.3, mostrada a seguir, tem-se um quadro-resumo com as principais características e fatores apontados nos produtos e soluções vistas nesta abordagem.

Tabela 2.3 – Quadro comparativo entre as principais soluções baseadas em *honeypots*

	Utilidade	**Nível de Interação/Emulação**	**Disposição Física**	**Mecanismos de Proteção do Hospedeiro**	**Mecanismos de Resposta Automática**
Deception Toolkit (DTK)	Pesquisa	Baixo/Sim	Unidade Isolada	▪ Não Existe	▪ *Log*
CyberCop Sting	Produção	Médio/Sim	Unidade Isolada	▪ Não Existe	▪ *Log*
Specter	Produção	Baixo/Sim	Unidade Isolada	▪ Não Existe	▪ *Log* ▪ *E-Mail* ▪ *Rastreamento*
BackOfficer Friendly (BOF)	Pesquisa	Baixo/Sim	Unidade Isolada	▪ Não Existe	▪ *Log*
ManTrap	Produção	Alto/Não	Unidade Isolada	▪ Não Existe	▪ *Log* ▪ *E-Mail* ▪ *SNMP traps* ▪ Execução de *Scripts* e binários
Alternativas Caseiras	Pesquisa/ Produção	Alto/Sim	Unidades Distribuídas Virtuais	▪ Não existe	▪ *Log* ▪ *E-Mail* ▪ Controle de Fluxo de Saída
sNET	Produção	Alto/Não	Unidades Distribuídas Mistas	▪ Mudança do Estado de Portas ou Serviços ▪ Redirecionamento de Portas e Serviços ▪ Desligamento do *host*	▪ *Log* ▪ *E-Mail* ▪ *Pager* ▪ Celular
Projeto *Honeynet*	Pesquisa	Alto/Não	Unidades Distribuídas Reais	▪ Não Existe ▪ Controle de pacotes do Fluxo de Saída	▪ *Log* ▪ *E-Mail* ▪ Controle de Fluxo de Saída
Honeyd	Pesquisa	Médio/Sim	Unidades Distribuídas Virtuais	▪ Redirecionamento de Portas e Serviços ▪ Controle de pacotes do Fluxo de Saída	▪ *Log* ▪ *Execução de Scripts e binários* ▪ *Controle de Fluxo de Saída*

Na Tabela 2.3 observa-se que a maioria das soluções prontas e disponíveis no mercado, em ambientes de pesquisa e alternativas experimentais ou caseiras, possui apenas recursos de respostas de notificação. Outro fato é que, embora consigam enganar atacantes e invasores em seus diversos níveis de interação, na sua maioria, não oferecem, dentre outras características e habilidades para [32]:

- Remover em tempo real um o intruso de um sistema alvo;
- Modificar dinamicamente as tabelas de regras de *firewalls* e roteadores;
- Bloquear portas, encerrar conexões usadas pelos atacantes, rejeitar tráfego vindo do endereço do atacante ou desviá-lo.

Enfim, há uma escassez de soluções que tenham habilidades de prover respostas automáticas e que não fiquem restritas apenas a respostas notificativas [32]. Ou seja, há necessidade de recursos que possibilitem, por exemplo, uma solução de reconfiguração automática do ambiente atacado de modo que o invasor possa ser contido sem causar danos aos recursos, ao mesmo tempo em que sua atividade possa ser observada e registrada e sem que o atacante e nem os usuários legítimos daqueles serviços percebam a fraude e as medidas adotadas.

2.10 Considerações finais

Neste Capítulo, percebeu-se o quanto a tecnologia baseada em *honeypots* pode ser útil na coleta de informações sobre ataques, atacantes, seus motivos, técnicas e as ferramentas que utilizam em suas investidas maliciosas. Porém, não basta apenas coletar informações sobre os atacantes e suas investidas. A comunidade de segurança necessita de soluções não apenas notificativas e que interajam com os mecanismos tradicionalmente utilizados em segurança de redes com o objetivo de tornar as redes das organizações mais seguras e preparadas para as ameaças virtuais [33].

Assim, o Capítulo a seguir apresentará a proposta de uma arquitetura baseada em agentes inteligentes e, associada a estratégias de implantação de armadilhas de rede, visando justamente aliar a eficiência das armadilhas de rede em reduzir falsos positivos à eficiência de *firewalls*, IDS's e roteadores, fazendo-os agirem proativamente. Dessa forma, será possível reconfigurar a rede para um nível maior de segurança, compondo um sistema que possa automatizar respostas a incidentes de segurança em redes e sistemas distribuídos, além de detectar, alertar e coletar informações sobre a atividade de atacantes, seus motivos e ferramentas e, se possível, suas origens, fornecendo subsídios para a investigação de crimes de computador.

3 SAMARA – SOCIEDADE DE AGENTES PARA A MONITORAÇÃO DE ATAQUES E RESPOSTAS AUTOMÁTICAS

3.1 Considerações iniciais

Embora não seja padrão ou obrigatório, normalmente, uma topologia de rede corporativa se apresenta como mostrado na Figura 3.1, a seguir:

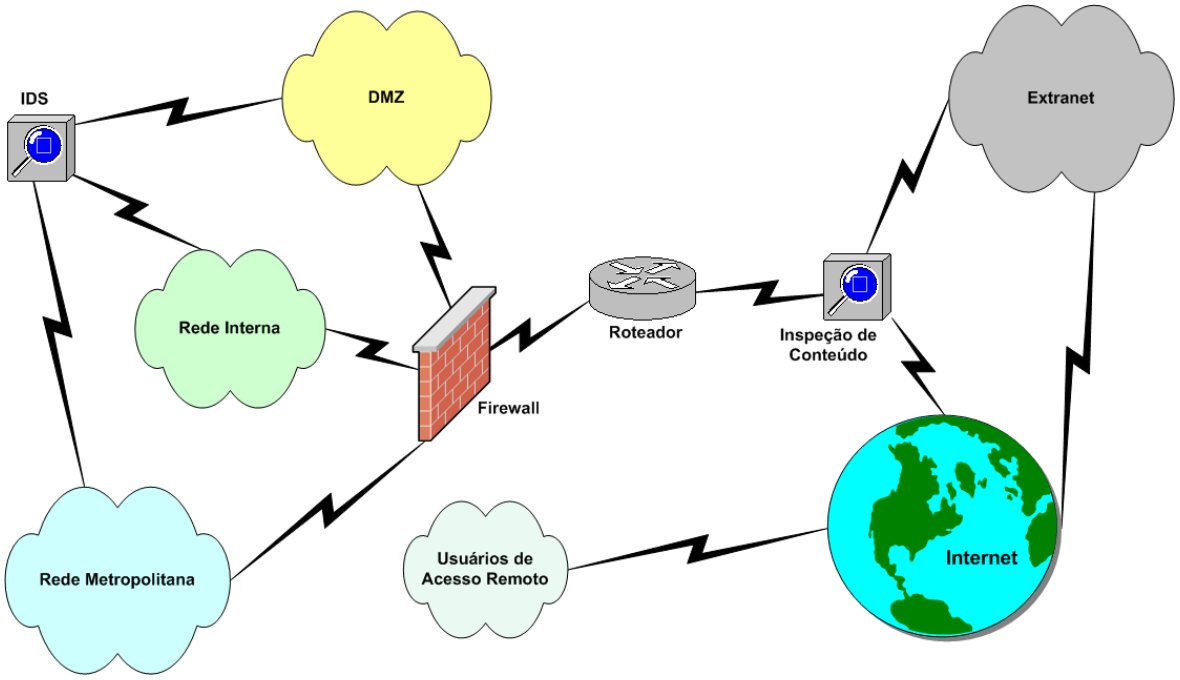


Figura 3.1 – Visão hipotética de uma rede corporativa e suas diversas conexões

A área denominada LAN ou simplesmente rede local é onde ficam normalmente as estações de trabalho e serviços internos de um ambiente de rede. A área denominada DMZ é a região onde ficam os serviços públicos, ou seja, disponíveis na Internet. Outra área que pode estar presente é a rede WAN. Nessa região estão normalmente os *links* para filiais e/ou localidades remotas. Existe também uma área chamada Extranet, que interliga a corporação com fornecedores e/ou parceiros. Embora ainda exista esta possibilidade de interconexão de redes, a mesma vem sendo substituída por conexões indiretas através da Internet, utilizando

VPN, isto é, redes virtuais privadas. Essa mesma tendência é utilizada pelos serviços de acesso remoto discado da instituição. Interligando essas regiões, existem os dispositivos de interconexão de redes, normalmente roteadores, pontes e servidores de acesso remoto. Para controlar toda essa estrutura, existem dispositivos como *firewalls*, sistemas de detecção de intrusos, inspetores de conteúdo e antivírus. Há ainda uma infindável lista de sistemas operacionais, aplicativos e utilitários, além de inúmeros protocolos, padrões e formatos para a interoperabilidade de redes e sistemas distribuídos. Enfim, pode-se perceber que a estrutura de um ambiente de rede é bastante complexa [3].

Em meio a esse ambiente caótico, os fabricantes de *softwares* disponibilizam seus produtos muito prematuramente, sem que as equipes de desenvolvimento tenham tempo suficiente para verificar eventuais problemas. Outro fator crítico é que sistemas operacionais e aplicações em geral são instalados e disponibilizados aos usuários, quase que geralmente, nas suas configurações padrões. Como conseqüência surgem os incontáveis *"bugs"* e vulnerabilidades, que tornam a segurança plena do ambiente impossível de ser alcançada.

Assim, o contexto de segurança da informação aqui focado pela presença de controles de acesso em roteadores, *firewalls*, IDS's, antivírus e outros recursos de segurança, não garante que o ambiente de rede fique a salvo de atacantes, intrusos e códigos maliciosos. Segundo [7], isto se deve a fatores como:

- O estrito relacionamento entre as classes de ataques de rede e vulnerabilidades derivadas de erros de implementação ou abuso de alguma ação legítima;
- O surgimento constante de *softwares* com elevada complexidade;
- O pouco comprometimento de programadores em desenvolver aplicações seguras;
- O crescente número de indivíduos interessados em investir contra redes e sistemas de computadores;

- A grande diversidade e disponibilidade de ferramentas e informações sobre vulnerabilidades e técnicas de ataques e invasões que podem ser utilizadas por indivíduos sem grandes conhecimentos técnicos;
- Usuários desinformados e/ou relapsos com suas senhas e procedimentos;
- Pessoal de suporte e capacitação insuficiente para manter estações, servidores e aplicações atualizadas com as mais recentes correções de segurança.

Dessa forma, por mais que os sistemas operacionais, aplicações e IDS's estejam atualizados e os *firewalls* estejam restringindo o acesso às diversas regiões da rede, a cada dia surgem novas vulnerabilidades que tornam praticamente impossível a tarefa de proteger plenamente os recursos de missão crítica de uma organização.

Conforme [1] e [7], um grande desafio enfrentado pelos administradores de segurança de rede é manter seus IDS's atualizados, isto é, capazes de identificar as variações de técnicas de ataques que surgem. Por mais atualizado que um IDS esteja, há ainda o problema da enorme quantidade de dados que deve ser analisada por esses sistemas, principalmente com o advento das tecnologias de redes que possibilitam tráfegos a *gigabits* por segundo. Se o *hardware* do IDS não for robusto e de alta performance, muitos produtos não conseguem analisar todos os pacotes que trafegam na rede, descartando uma boa parte. Além disso, a maioria dos IDS's gera uma quantidade muito grande de falsos positivos e falsos negativos. Como conseqüência, muita atividade maliciosa passa despercebida pela análise dos IDS's ou, devido à quantidade de eventos, administradores de rede não têm condições de tratá-los adequadamente [39]. As equipes de suporte de rede, geralmente, não dispõem de profissionais em número suficiente, capacitados e com tempo disponível para se dedicarem à tarefa de sentar em frente ao IDS, analisar todos os incidentes reportados e inferir se um evento constitui realmente um problema ou se trata de uma operação normal da rede. Assim, na maioria das vezes, muitos verdadeiros positivos são desprezados.

Como é muito difícil inferir se quem está por trás de um computador é uma fonte maliciosa ou constitui um usuário legítimo, um intruso ou atacante visualiza e tem acesso a um ambiente de rede da mesma forma que um usuário normal. Ou seja, num ambiente de rede os mecanismos tradicionais de segurança e conectividade tratam usuários normais e usuários maliciosos da mesma forma. Salvo algumas exceções, as permissões e restrições de acesso são as mesmas para todos os usuários, quer sejam maliciosos ou não. Quando o administrador de segurança bloqueia ou libera uma porta ou serviço, principalmente se disponível na Internet, ele o faz tanto para um atacante como para um usuário legítimo. Assim, há uma única visão de um ambiente de rede por qualquer usuário, conforme visto na Figura 3.2, a seguir. Cada setor da rede, aqui representado por hexágonos, significa uma daquelas áreas vistas na Figura 3.1, no início deste Capítulo.

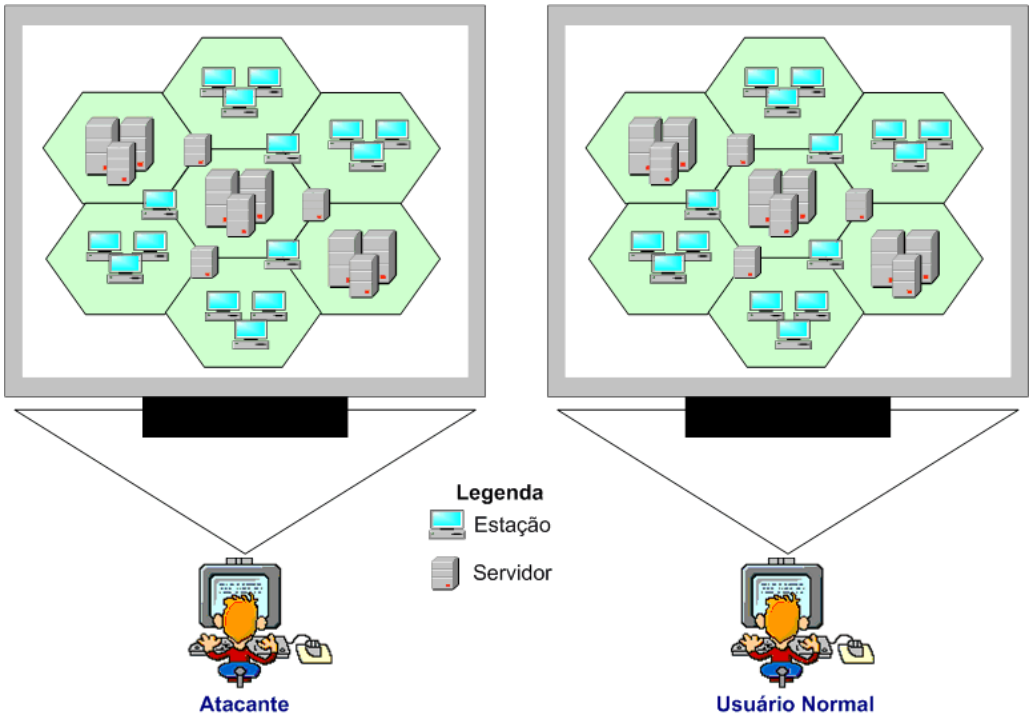


Figura 3.2 – Visão hipotética da rede para qualquer usuário de um ambiente de rede

Nesse contexto, a implantação de *honeypots* pode complementar a ação dos mecanismos tradicionais de segurança como *firewalls* e IDS's, ajudando a equipe de

segurança de redes a responder mais rápido aos incidentes de segurança, agregando proatividade e, conseqüentemente, tornando o ambiente mais seguro.

Assim, como contribuição desta dissertação, propõe-se a arquitetura SAMARA (Sociedade de Agentes para a Monitoração de Ataques e Respostas Automatizadas), baseada no uso de agentes inteligentes, instalados em *honeypots* e servidores de produção, de modo a introduzir no ambiente de rede novos recursos de detecção de atacantes e intrusos e brechas de segurança, tornando o sistema de segurança proativo e capaz de reconfigurar-se dinamicamente, aumentando o nível de segurança do ambiente monitorado.

3.2 Visão geral da arquitetura

A arquitetura SAMARA é composta por uma sociedade de agentes dispostos estrategicamente num ambiente controlado de modo a agir proativamente, detectando, alertando, reconfigurando a rede para um grau maior de segurança e coletando informações sobre a atividade do atacante, dentre outras funcionalidades.

A proposta consiste em instrumentar *honeypots* através da instalação de agentes de alerta e colocá-los nas diversas regiões do ambiente de rede, formando uma rede de monitoramento com pontos de prova estratégicos. Esses pontos de prova (*honeypots* e agentes de alerta) serão capazes de detectar e alertar sobre atividades maliciosas não triviais (por exemplo, a varredura mostrada através da Tabela 2.1), ou seja, aquelas atividades de rede que os sistemas de *firewall* e IDS não conseguem detectar. Na ocorrência de incidentes, será possível que uma unidade de gerenciamento central dispare respostas automáticas através de outros agentes, de forma a conter e rastrear a fonte maliciosa, impedindo que o atacante alcance os recursos de missão crítica da organização até que o suporte de segurança da rede possa entrar em ação.

A intenção, porém, não consiste em isolar totalmente o atacante ou invasor, mas restringir seu alcance, deixando certas áreas livres, a *honeynet*, para sua investida, fazendo com que ele perca tempo tentando explorar as vulnerabilidades encontradas nos *honeypots.*

A Figura 3.3 fornece uma visão geral da arquitetura, para uma melhor compreensão.

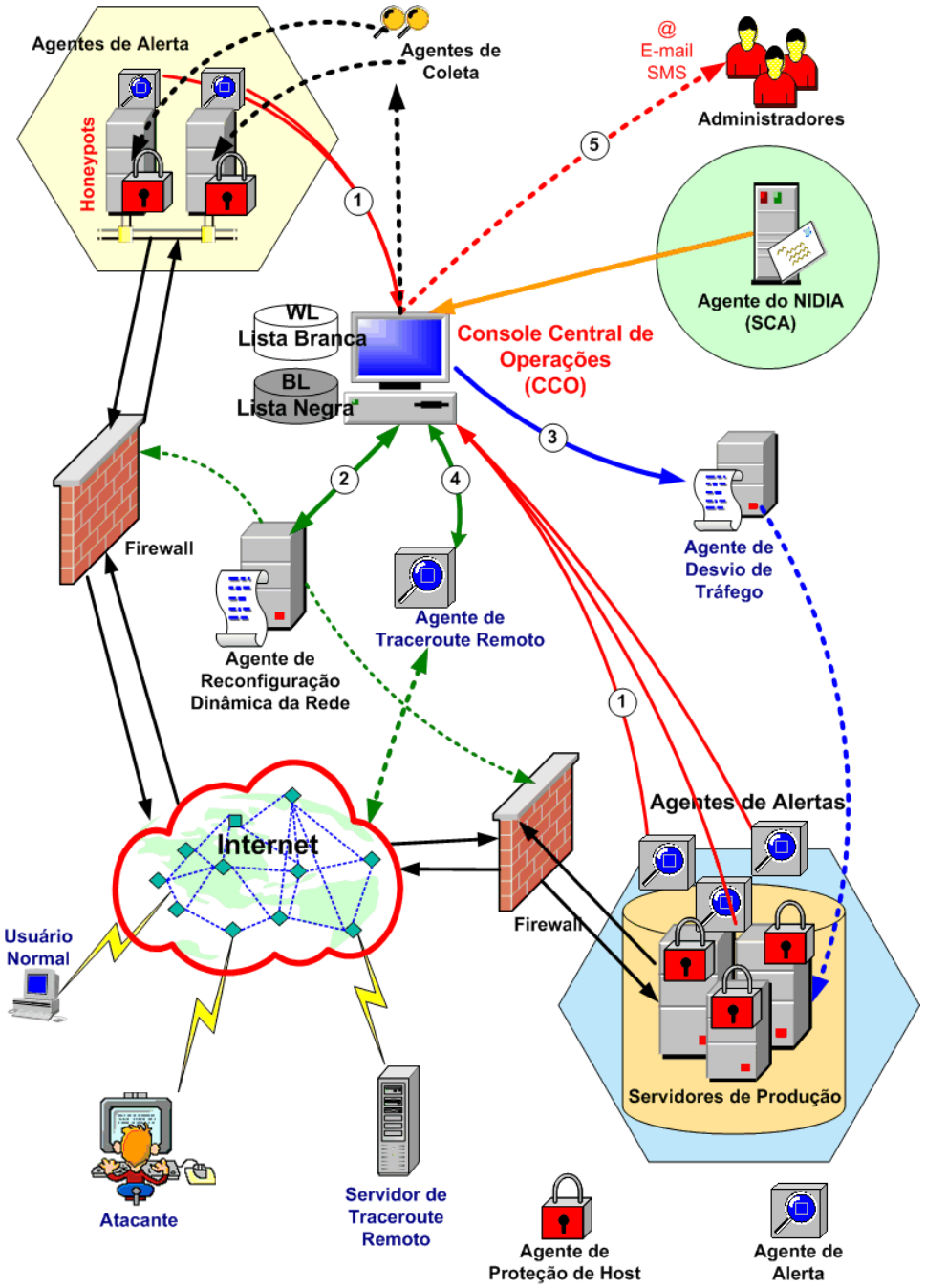


Figura 3.3 – Visão geral da arquitetura

3.2.1 *Honeypots* utilizados pela arquitetura SAMARA

A arquitetura SAMARA é voltada para ambientes de produção, porém, é flexível o bastante para ser utilizada em ambientes de pesquisa.

Segundo [47], é recomendável que os *honeypots* utilizados sejam baseados em soluções de sistemas operacionais e aplicações servidoras reais, iguais àquelas utilizadas em ambientes de produção. As razões para essa abordagem são várias, dentre as quais destacam-se:

- Se um *honeypot* baseado em sistemas operacionais e aplicações reais, não emuladas, for alcançado e até mesmo comprometido, é praticamente certo que um recurso de produção que utilize esse mesmo sistema operacional e aplicações servidoras também seja atingido [47];
- Sistemas operacionais e aplicativos reais podem ser obtidos facilmente na Internet, muitos inclusive até gratuitamente. Além disso, conforme [47], *honeypots* baseados em soluções reais não precisam de configurações complexas, bastando fazer a instalação padrão, tal como ocorre no dia-a-dia. Obviamente, esses *honeypots* precisam ser configurados convincentemente, de modo a parecerem com aplicações de ambientes de produção de modo que o atacante não desconfie;
- Administradores de rede já convivem diariamente com sistemas operacionais e suas aplicações servidoras, o que torna mais fácil a instalação e manutenção de *honeypots* baseados nessas soluções.
- *Honeypots* emulados são fáceis de serem detectados e não oferecem interatividade real para os atacantes, não permitindo, por exemplo, que o atacante instale seus *rootkits* [39].
- *Honeypots* sofisticados, como o Mantrap, são muito caros e exigem requisitos de *hardware* e *software* muitos caros, além de muita manutenção [27].

Porém, não pode ser descartada a opção de ser ter *honeypots* em ambientes virtuais como o VMWare [41] e [25], o que possibilita a expansão do número de *honeypots*, aumentando as possibilidades destes serem alcançados pelos atacantes [39].

Pode ser inferido que é temerário disponibilizar recursos numa rede para serem atacados e comprometidos, principalmente por se tratarem de soluções que são utilizadas também nos recursos de produção. Porém, conforme [39], os recursos de produção das redes da maioria das organizações correm os mesmos os riscos, pois suas vulnerabilidades são as mesmas. Aliás, uma outra vantagem de se instalar *honeypots* baseados em sistemas e aplicativos reais, não emulados, é que os problemas e vulnerabilidades dos sistemas de produção podem ser detectados através desses *honeypots* e resolvidos antes que atacantes causem danos ao ambiente de produção.

3.3 Principais componentes da arquitetura

O Capítulo anterior apresentou as características e particularidades da tecnologia baseada em *honeypots*. Assim, para uma melhor visão da arquitetura SAMARA, serão enfocadas mais as particularidades da arquitetura e seus agentes do que sobre *honeypots*. A Figura 3.4 mostra as principais categorias dos agentes que compõem a arquitetura.

Figura 3.4 – Categorias dos agentes da arquitetura SAMARA

A seguir, são descritos os principais componentes da arquitetura e suas respectivas responsabilidades:

- **Console Central de Operações (CCO)**: Console de monitoração e administração do ambiente. Possibilita uma visão geral dos eventos em tempo real e permite ao administrador realizar operações de rotina ou de emergência, tendo como foco os agentes do sistema, os *honeypots* e o ambiente de rede;
- **Agentes de Alerta**: Responsáveis por detectar e alertar sobre a presença de supostos intrusos em um dos *hosts* (quer sejam *honeypots*, quer sejam recursos de produção). Seus alertas são encaminhados à CCO.
- **Agente de Reconfiguração Dinâmica da Rede**: Responsável por reconfigurar *firewalls* e roteadores, de modo a limitar a acesso do intruso aos *hosts* de produção e também minimizar o uso das unidades de decepção, após um comprometimento, como trampolim para outros ataques.
- **Agentes de Desvio**: Responsáveis por desviar o tráfego malicioso para um *honeypot* com serviços similares aos do servidor de produção alvo de um ataque.
- **Agente de *Traceroute* Remoto**: Responsável por tentar descobrir o percurso até o atacante. O objetivo de fornecer informações sobre a origem verdadeira do tráfego, mesmo que o endereço fonte seja falso, de modo que o administrador possa reprimir ou limitar a ação do atacante.
- **Agentes de Coleta**: Responsáveis por coletar informações sobre a atividade do intruso a partir de um *honeypot* e enviar à CCO.
- **Agente de Proteção de *Host***: Responsável por reconfigurar *hosts* para um estado de segurança intensificado. Esse agente pode proteger um *host* de ser aniquilado pelo invasor. Esse agente pode ser instruído a mudar o estado de disponibilidade de portas e serviços, bloqueando assim o acesso, ou desligar o

host, medida extrema utilizada quando um sistema de arquivos precisa ser preservado para casos de perícia forense.

- **Administrador do Sistema**: Agente humano que interage com o ambiente através da CCO para realizar tarefas como: configurar *hosts* e dispositivos de rede; monitorar a atividade intrusiva; analisar os dados coletados pelos agentes; ativar e desativar unidade de decepção e agentes, dentre outras atividades.
- **Bases de Dados:** Responsáveis por prover um repositório para armazenar as informações provenientes dos diversos agentes, de modo que a CCO possa correlacioná-las e prover subsídios para que o administrador do sistema tenha estatísticas em tempo hábil que favoreçam medidas que evitem ou, num pior caso, minimizem os danos de um eventual ataque.
- ***Honeypots***: armadilhas de rede dispostas estrategicamente no ambiente. Têm por finalidade atrair os atacantes e alimentá-los com serviços e informações fraudulentas, fazendo-os perderem tempo em alvos sem valor de produção.
- **Recursos de Produção**: Recursos que devem ser protegidos.
- **Dispositivos de rede e segurança**: Elementos de conectividade e segurança da rede que deverão ser reconfigurados automaticamente a partir de diretivas da CCO. Podem ser *firewalls*, IDS's e roteadores.

3.4 Princípio de Operação

Os agentes de alerta, instalados nos *honeypots* e recursos de produção, ficam "escutando" na interface de rede desses *hosts*. A qualquer tentativa de conexão (entrada ou saída), esses agentes capturam os seguintes dados:

- Identificador único da conexão;

- A data de ocorrência do incidente no formato dd/mm/aaaa;
- A hora de ocorrência do incidente no formato hh:mm:ss;
- A porta de destino sondada;
- O nome do serviço/protocolo sondado;
- O endereço IP de origem;
- O endereço IP sondado.

Esses dados são repassados para a Console Central de Operações (CCO) que, por sua vez, os registra na base de dados de incidentes reportados.

Conforme dito anteriormente, idealmente, os *honeypots*, por não serem recursos de produção, nem serem recursos conhecidos pelos usuários e pelo público, não apresentam atividade de rede. Assim, qualquer atividade de rede envolvendo um *honeypot* deve ser considerada suspeita por padrão. Logo, tudo o que envolver tráfego de rede entre os *honeypots* do ambiente é capturado pelos agentes de alerta e enviado à CCO.

De posse desses dados, a CCO entra em ação disparando diversos procedimentos de resposta ao incidente reportado, conforme fluxograma mostrado na Figura 3.5:

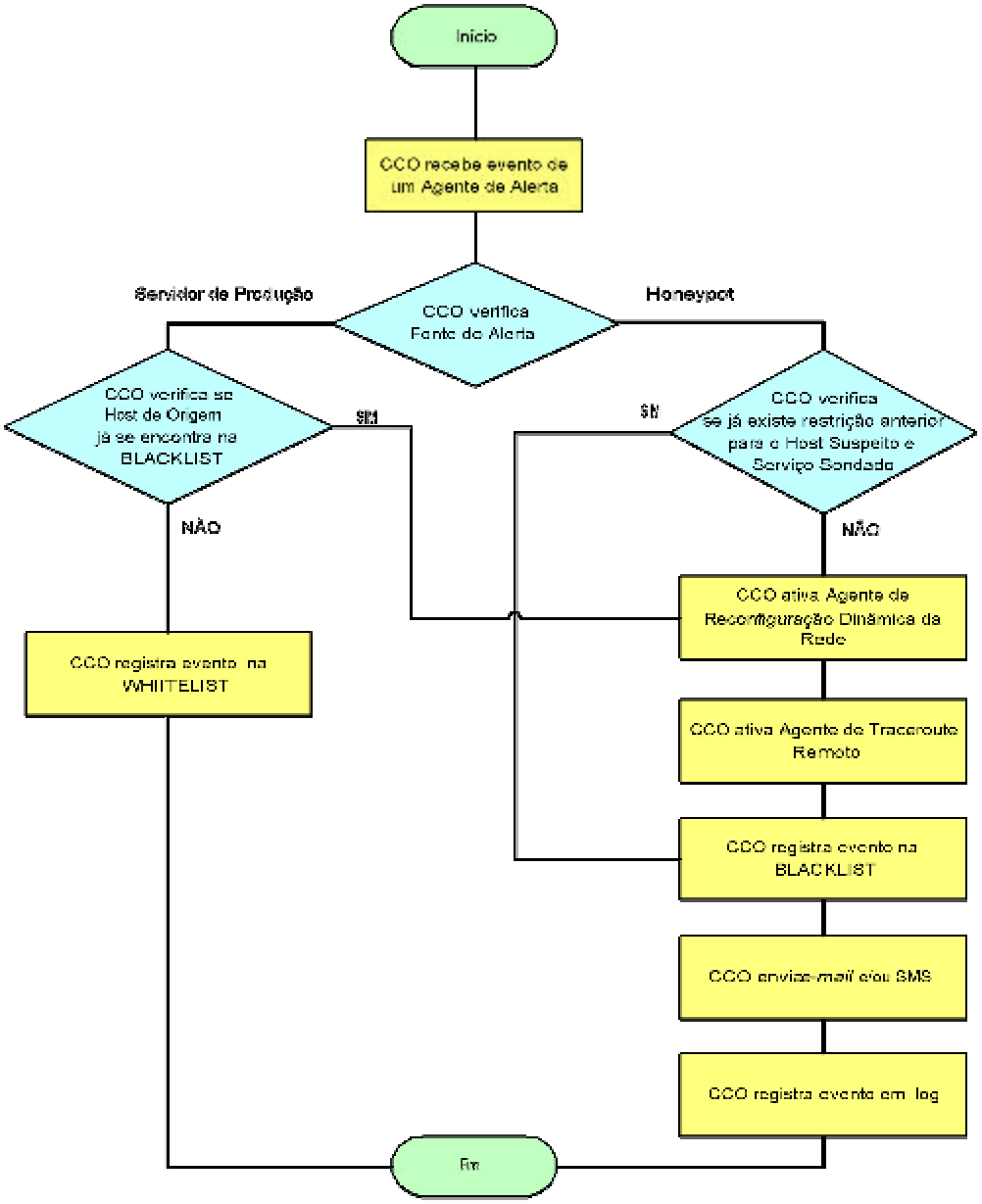


Figura 3.5 – Tratamento de um evento recebido pela CCO

Ao receber um evento, a CCO verifica a fonte daquele alerta. Se a fonte do evento for um *honeypot*, o evento será considerado um incidente de segurança de fato. Se a fonte for um recurso de produção, o evento, a princípio, não representa um incidente de segurança. Mesmo assim, a CCO verifica se o *host* de origem (aquele que tentou a conexão para o recurso de produção) já consta da lista negra (*blacklist*). Se não constar, o

evento será registrado na lista branca (*whitelist*), concluindo o tratamento do evento. Caso contrário, o evento será considerado um incidente de segurança de fato.

Caso o evento seja considerado um incidente de segurança de fato, a CCO repassa os dados do evento para o Agente de Reconfiguração Dinâmica da Rede para que este proceda a reconfiguração dinâmica da rede de modo a limitar o acesso da suposta fonte maliciosa. Isso é feito através da criação de uma combinação de regras de acesso (ACL), de tradução de endereço (NAT) e redirecionamento (FORWARD) de endereço e portas nas tabelas dos *firewalls* e roteadores do ambiente que possibilitem o desvio do tráfego supostamente malicioso para a zona dos *honeypots.*

A CCO também ativa o Agente de *Traceroute* Remoto, que é responsável por descobrir um possível percurso até a suposta fonte maliciosa. Para evitar que o suposto intruso descubra quem o está sondando, esse agente recorre a um Servidor de *Traceroute* Remoto, disponível na Internet.

A seguir, a CCO registra na *blacklist* o evento sobre o *host* suspeito.

A CCO também envia um *e-mail* e, se possível uma SMS para o administrador de segurança da rede. Por fim, o evento é registrado em arquivo de log, concluindo-se o tratamento de um incidente de segurança.

3.4.1 Bases de dados

A arquitetura propõe as seguintes bases de dados para o registro de todos os fatos que envolvam os alertas de incidentes:

- lista negra,
- lista branca e
- *log* de incidentes reportados.

A lista negra serve para registrar resumidamente os eventos sobre as fontes potencialmente hostis que o sistema capturar através dos agentes de alerta que estarão distribuídos no ambiente através dos *honeypots* ou através de notificações de agentes externos como é o caso do NIDIA [18]. O administrador de segurança, através da CCO, poderá editar a lista negra, verificar a consistência dos dados e, eventualmente, inserir ou excluir algum endereço IP dessa lista. A lista negra contém as seguintes informações:

- Fonte Hostil
- *Honeypot*/Recurso Sondado
- Porta/Serviço/Protocolo Sondado
- Data e hora do incidente

A Tabela 3.1, mostrada a seguir, exemplifica uma lista negra e seus principais campos:

Tabela 3.1 – Exemplo de entradas na *blacklist*

FONTE HOSTIL	HONEYPOT/ RECURSO SONDADO	PORTA/SERVIÇO/ PROTOCOLO SONDADOS	TIMESTAMP
200.165.132.154	10.0.0.2	709 RPC TCP	2005/03/06 21:12:26-3:00 GMT
64.45.38.211	172.16.14.45	23 TELNET TCP	2005/03/06 13:51:29-3:00 GMT
200.165.132.154	10.0.0.2	1033 DNS UDP	2005/03/06 21:12:26-3:00 GMT
64.45.38.211	172.16.14.45	23 TELNET TCP	2005/03/06 13:51:29-3:00 GMT
64.45.38.211	172.16.14.45	23 TELNET TCP	2005/03/06 13:51:29-3:00 GMT
200.165.132.154	10.0.0.2	1033 DNS UDP	2005/03/06 21:12:26-3:00 GMT
64.45.38.211	172.16.14.45	23 TELNET TCP	2005/03/06 13:51:29-3:00 GMT
200.165.132.154	10.0.0.2	1033 DNS UDP	2005/03/06 21:12:26-3:00 GMT

Todos os eventos mostrados na Tabela 3.1 são evidências de que alguém tentou acessar os *honeypots* do ambiente. O primeiro evento da lista indica que um *host* cujo endereço IP é `200.165.132.154` tentou uma consulta de RPC no *honeypot* `10.0.0.2` através da porta 709, utilizando o protocolo TCP.

Uma possível aplicação dessa lista, além de servir para análises de comprometimento de *honeypots*, seria gerar um gráfico para uma melhor visualização dos incidentes de segurança em tempo real.

Na evidência de algum endereço IP do ambiente, seja ele público ou privado, encontrar-se inserido nessa lista, significa que, possivelmente, o *host* em questão encontra-se comprometido por alguma ação intrusiva. Isto se deve ao fato de que, por definição, não há possibilidade de algum *host* descobrir por si só um outro, no caso um *honeypot*, sem saber o endereço ou nome deste. Ocorrendo isto, o administrador será alertado tanto pela console como pela negação de acesso daquele *host* legítimo aos recursos de produção.

A princípio, todos têm acesso aos serviços de um recurso de produção, sem restrições de origem, a menos que haja alguma restrição anterior nas regras dos *firewalls* e roteadores. Entretanto, pode ocorrer que, eventualmente, um atacante atinja primeiro um servidor de produção e não um *honeypot*. Nesse caso, se o recurso estiver vulnerável, ele conseguirá o que quiser, pois não haverá alerta e, dependendo do IDS, o evento será considerado uma atividade normal na rede.

Por esse motivo, os agentes de alerta também são instalados nos recursos de produção. Evidentemente, esses alertas sempre significarão falsos positivos, pois os agentes de alerta são desprovidos de inteligência para detectar se determinada fonte constitui ou não ameaça ao recurso ou ambiente monitorado. Eles simplesmente reportam qualquer atividade de rede.

Porém, esses alertas servirão para compor uma lista branca, isto é, uma lista onde serão registrados todos os eventos envolvendo as fontes que acessaram os recursos de produção do ambiente. Esses dados poderão ser úteis para eventuais consultas e análises no caso de comprometimento comprovado de algum *host* do ambiente. A lista branca pode também sofrer inclusões e exclusões manuais pelo administrador de segurança, utilizando a CCO. Esta lista contém as seguintes informações:

- *Host* de Origem;
- *Host* de Destino;

- Porta/Serviço/Protocolo Solicitados;
- Data e hora do evento.

A Tabela 3.2 mostrada a seguir exemplifica uma lista branca e seus principais campos:

Tabela 3.2 – Exemplo de entradas na *whitelist*

HOST DE ORIGEM	*HOST* DE DESTINO	PORTA/SERVIÇO/ PROTOCOLO SOLICITADO	TIMESTAMP
200.165.132.154	10.0.0.2	1033 DNS UDP	2005/03/06 21:12:26-3:00 GMT
64.45.38.211	172.16.14.45	23 TELNET TCP	2005/03/06 13:51:29-3:00 GMT
200.165.132.154	10.0.0.2	1033 DNS UDP	2005/03/06 21:12:26-3:00 GMT
64.45.38.211	172.16.14.45	23 TELNET TCP	2005/03/06 13:51:29-3:00 GMT
200.165.132.154	10.0.0.2	1033 DNS UDP	2005/03/06 21:12:26-3:00 GMT
64.45.38.211	172.16.14.45	23 TELNET TCP	2005/03/06 13:51:29-3:00 GMT
200.165.132.154	10.0.0.2	1033 DNS UDP	2005/03/06 21:12:26-3:00 GMT
64.45.38.211	172.16.14.45	23 TELNET TCP	2005/03/06 13:51:29-3:00 GMT
200.165.132.154	10.0.0.2	1033 DNS UDP	2005/03/06 21:12:26-3:00 GMT
64.45.38.211	172.16.14.45	23 TELNET TCP	2005/03/06 13:51:29-3:00 GMT
200.165.132.154	10.0.0.2	1033 DNS UDP	2005/03/06 21:12:26-3:00 GMT
64.45.38.211	172.16.14.45	23 TELNET TCP	2005/03/06 13:51:29-3:00 GMT
200.165.132.154	10.0.0.2	1033 DNS UDP	2005/03/06 21:12:26-3:00 GMT

3.4.2 Reconfiguração Dinâmica da Rede

As diretivas de acesso são aplicadas aos roteadores e *firewalls* através do Agente de Reconfiguração Dinâmica da Rede. Esse agente interage com esses dispositivos através de conexões seguras, principalmente SSH e diferentes das conexões com a rede de produção. Para tanto, a CCO possui duas interfaces de rede: uma para receber os alertas e outra para reconfigurar o ambiente.

Quando um suspeito investe contra um *honeypot*, as diretivas disparadas aos mecanismos de segurança normalmente são reforços de segurança ao tráfego dos servidores de produção. Ou seja, a partir do momento em que o suspeito faz a investida contra um *honeypot*, o ambiente tentará evitar que o atacante também alcance os recursos de missão crítica.

O Agente de Reconfiguração Dinâmica da Rede aplica regras de controle de acesso aos *firewalls* e roteadores do ambiente, fazendo com que recursos de produção fiquem fora do alcance do intruso ou atacante, redirecionando um possível tráfego da fonte potencialmente maliciosa para um ou mais *honeypots* com serviços similares aos daqueles servidores de produção.

Dessa forma, pode-se enrijecer o nível de segurança do ambiente e o atacante não poderá alcançar nos recursos de produção os serviços sondados nos *honeypots*. Porém, sua atividade não cessará, ficando apenas restrita aos *honeypots* do ambiente. A partir de então, o endereço IP do *host* suspeito fica registrado numa lista negra. Uma vez na lista negra, o endereço IP suspeito não conseguirá alcançar os recursos de produção. A qualquer tentativa, seu tráfego será desviado para os *honeypots*, de acordo com o serviço solicitado. Dessa forma, o pessoal de segurança poderá observar melhor a atividade maliciosa e ajustar a defesa do ambiente.

Vale ressaltar que a reconfiguração dinâmica da rede não deve ser a única reação do ambiente. Seu papel é prestar uma espécie de primeiro socorro, isto é, uma primeira reação até que a equipe de segurança de rede tome ciência do problema e inicie os procedimentos de respostas baseados na política de segurança da instituição. Assim, a expectativa é que o sistema seja capaz de realizar as seguintes reações:

1. Ao detectar um suspeito em potencial, seja através dos *honeypots*, seja através da análise da rede neural do NIDIA, o sistema seja capaz de responder primariamente, limitando o alcance desse suspeito somente aos *honeypots* do ambiente até que a equipe de segurança possa se estruturar para responder efetivamente ao incidente de segurança;
2. Alertar a equipe de segurança de rede da instituição sobre o incidente em andamento;

3. Fornecer mais informações sobre o incidente de segurança para uma análise mais aprofundada e corrigir as distorções nas políticas de segurança que culminaram na ocorrência do incidente reportado.

Após a reconfiguração dinâmica dos *firewalls* e roteadores do ambiente de rede, o *host* suspeito ficará confinado aos *honeypots* do ambiente. A Figura 3.6 dá uma visão hipotética da rede pelo atacante depois da reconfiguração dinâmica disparada pela CCO.

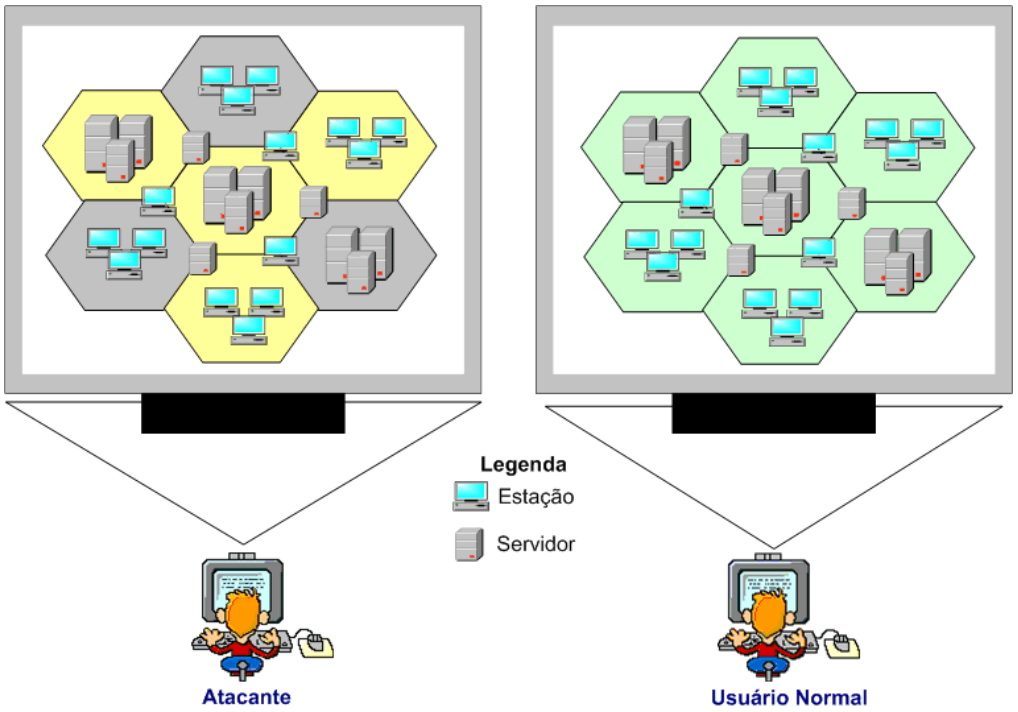


Figura 3.6 – Visão hipotética da rede após reconfiguração dinâmica do ambiente

Na Figura 3.6, as áreas representadas em amarelo compõem o escopo de visão do atacante. Ou seja, nessas áreas encontram-se os *honeypots*, únicos recursos que a fonte maliciosa consegue acessar, após os agentes de alerta informarem a CCO de uma possível atividade maliciosa. As áreas em cinza representam os recursos de produção, agora com segurança enrijecida e, conseqüentemente, fora do alcance do atacante. O usuário normal continua visualizando todo o escopo da rede, como se nada estivesse acontecendo.

3.4.3 Otimização do desempenho de *firewalls* e roteadores

Pode acontecer que, com o passar do tempo, o conjunto de regras de acesso dos *firewalls* e roteadores fique muito extenso e isso prejudique o desempenho e gerenciamento desses dispositivos. Para evitar isso, a CCO permite que o administrador de segurança edite, a qualquer tempo, o conjunto de regras desses mecanismos, agrupe *hosts* em regras similares, faça a limpeza de regras redundantes, otimizando, assim, o desempenho dos dispositivos.

3.4.4 Controle do fluxo de dados

Para evitar que a CCO e o administrador recebam uma avalanche de alertas a cada pacote que entre ou saia de um *honeypot*, é previsto um módulo controlador de fluxo de dados que permite controlar o nível de alerta enviado pela CCO. Assim, esse módulo permite definir um limite máximo de alertas que serão enviados.

Esse controle é feito pelo número de conexões requisitadas por uma mesma fonte a um determinado *honeypot* e pela verificação da lista negra. Isso evita também que a CCO dispare diretivas de acessos redundantes, impedindo que uma fonte potencialmente hostil gere muitas entradas nas tabelas de regras de *firewalls* e roteadores. Para tanto, a CCO verifica a lista negra antes de gerar cada diretiva de acesso. Se a fonte potencialmente hostil já estiver na lista negra, o evento será registrado, mas não serão gerados alertas por *e-mail* ou SMS. A diretiva somente será gerada caso o serviço que a fonte suspeita sondou seja diferente daquele registrado anteriormente. Porém, se o alvo for outro *honeypot* ou recurso de produção, o evento será tratado da maneira normal, isto é,

serão gerados os alertas usuais e as diretivas de acesso nos *firewalls* e roteadores do ambiente.

3.4.5 Proteção de *Hosts* e *Honeypots*

Para evitar que um *honeypot* já comprometido por uma fonte hostil seja utilizado como trampolim para novas investidas, a CCO também dispara diretivas que limitam o fluxo de dados a partir dos *hosts* comprometidos.

Assim, mesmo que o *host* esteja comprometido, o invasor enfrentará problemas para utilizá-lo, visto que nos mecanismos de controle de acesso (*firewalls* e roteadores), há restrições de quantidades de pacotes que deixarão o *host* comprometido, bem como a taxa de transmissão dos mesmos.

A arquitetura também prevê o Agente de Proteção de *Host*, que ficará residente no *host* a ser protegido, podendo este ser um *honeypot* ou um recurso de produção. Esse agente pode ser instruído para mudar o estado de disponibilidade de portas e serviços, bloqueando assim o acesso, chegando até a desligar o *host*, medida extrema utilizada quando um sistema de arquivos precisa ser preservado para casos de perícia forense. A expectativa é que esse agente evite que um *host* seja aniquilado pelo invasor.

3.4.6 Descoberta do percurso até o suspeito

Outra funcionalidade da CCO é a ativação do Agente de *Traceroute* Remoto (ATR). Esse agente passa como parâmetro, para um ou mais servidores de *traceroute* remotos, o endereço IP de uma fonte suspeita. A Figura 3.7, a seguir, esboça a estratégia de rastreamento do suspeito.

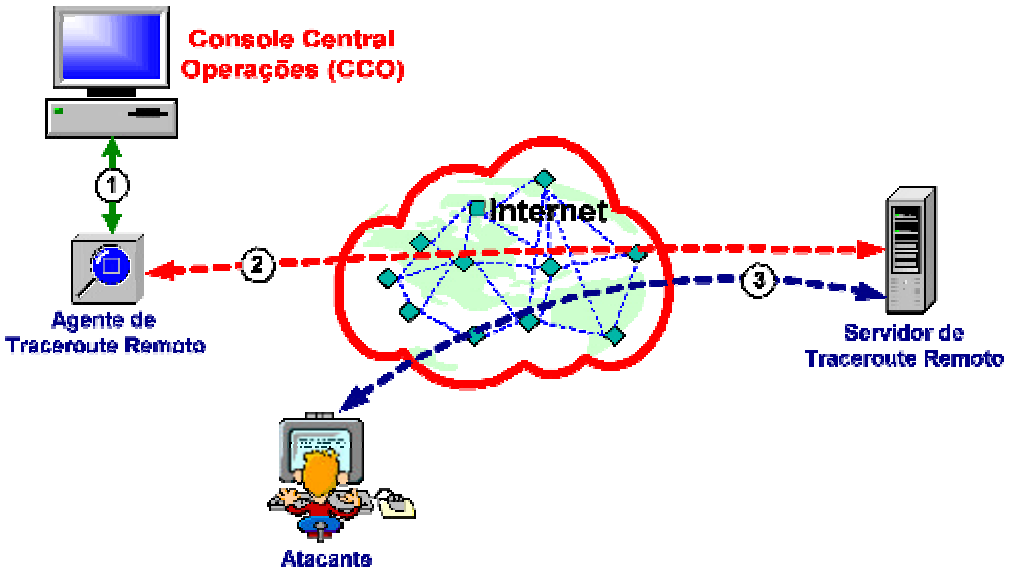


Figura 3.7 – Estratégia de rastreamento do suspeito através do Agente de *Traceroute* Remoto

A estratégia de utilizar servidores de *traceroute* remotos para assegurar que o suspeito não descubra a origem da CCO que é quem efetivamente o está sondando o endereço da fonte suspeita. O resultado é armazenado em arquivo texto e a CCO concatena então o percurso desde a conexão do ATR até o servidor de *traceroute* (já conhecido) com o percurso desse servidor até o endereço IP do suspeito, gerando o trajeto definitivo. A partir de então, o administrador de segurança poderá consultar esse arquivo, analisar os dados e, quando possível, contatar o responsável por aquele endereço IP, solicitando as providências cabíveis. Evidentemente, o atacante pode ser precavido e utilizar artifícios para que seu endereço IP verdadeiro não venha a ser descoberto. Porém, por mais que isso venha a ocorrer, o percurso descoberto pelo agente de *traceroute* pode ajudar a descobrir outras máquinas que foram comprometidas pelo atacante, inclusive em redes externas. Esses dados podem ser encaminhados a administradores de redes de provedores de acesso, universidades e entidades como os CERTs [3]. A Figura 3.8 mostra o resultado de um *traceroute* obtido através do agente.

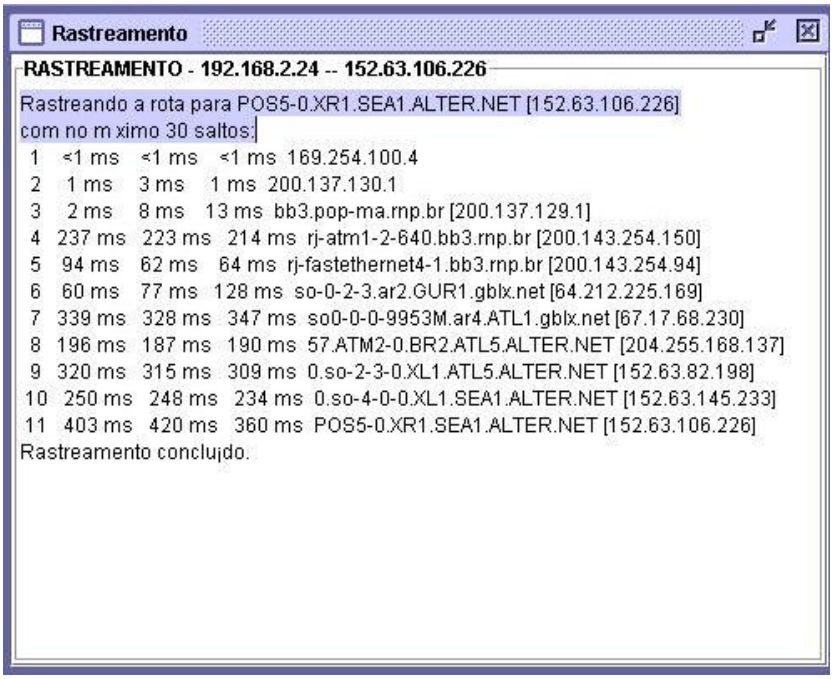


Figura 3.8 – Percurso levantado através do Agente de *Traceroute* Remoto

Analisando-se, por exemplo, o resultado do rastreamento indicado na Figura 3.8, pode-se inferir que a fonte suspeita é provavelmente de Seattle, nos EUA. Eventualmente, o percurso levantado pelo Agente de *Traceroute* Remoto pode diferir do caminho que o suspeito utilizou para chegar ao *honeypot* ou recurso de produção do ambiente. Além disso, o atacante poderá utilizar artifícios para encobrir seu rastro, inclusive utilizar um endereço IP de uma outra fonte, provavelmente comprometida por ele anteriormente. Mesmo assim, o rastreamento levantado pelo agente ainda é válido para notificar os responsáveis pelas redes levantadas no percurso de que alguém está utilizando aquela rota para realizar, possivelmente, uma atividade maliciosa [43].

3.4.7 Notificações ao administrador

Além da área de trabalho da CCO, que permite visualizar todos os incidentes reportados, em tempo real, a console se comunica com o administrador através de mensagens

de *e-mail* e/ou de SMS para o caso do administrador estar ausente do ambiente, por exemplo, à noite ou finais de semana. Normalmente, podem ser enviadas duas formas de alerta. Um módulo encaminhador de mensagens *de e-mail* e de SMS é ativado tão logo que a CCO recebe os alertas dos agentes instalados nos *honeypots.*

3.4.8 Coleta de dados

A arquitetura SAMARA também prevê a coleta de dados para fins de análise de unidades atacadas ou comprometidas. Para tanto, o projeto global incorpora um Agente de Coleta. Esse agente será ativado imediatamente após o agente de alerta de um *honeypot* informar à CCO que alguma fonte suspeita alcançou o *honeypot*. A partir de então, o agente de coleta passará a capturar o tráfego que a fonte maliciosa estabelecer com o *honeypot*, inclusive pressionamento de teclas e tráfego criptografado.

3.5 Vantagens da arquitetura

As principais vantagens da arquitetura SAMARA são:

- A utilização conjunta de *honeypots* e agentes para detecção de incidentes de segurança como forma de reduzir o número de falsos positivos gerados por mecanismos de detecção de intrusos convencionais;
- Utilização de agentes inteligentes para a reconfiguração dinâmica da rede, visando reduzir os potenciais danos experimentados quando da ocorrência de incidentes de segurança, sem prejudicar os usuários normais, com eventuais desativações de serviços e sem que a fonte maliciosa suspeite;

- A utilização de *honeypots* baseados em sistemas operacionais e aplicações reais, não emulados, podendo ser constituídos por ambientes virtuais em alguns casos. Essa característica permite que os requisitos de *hardware* e *software* não constituam obstáculos para a sua implantação, podendo utilizar recursos de desempenho inferior aos dos recursos de produção de fato;
- Os componentes podem estar distribuídos. Isto permite que o ambiente monitorado ultrapasse os limites de uma rede, podendo abranger quaisquer possibilidades de interconexão de redes. Ou seja, a arquitetura permite, por exemplo, que os agentes de alerta estejam em uma organização e a console central de operações em outra localidade. Assim, pode-se ter uma estrutura de segurança de rede global que monitore incidentes de segurança em várias organizações, compartilhe esses dados entre todos os nós monitorados e aplique as respostas automaticamente a partir de um ponto qualquer no globo.

Em resumo, esta arquitetura utiliza, associadamente aos *honeypots*, agentes estrategicamente distribuídos no ambiente de rede, inclusive em recursos de produção que, aliados a uma topologia de rede, permitem detectar incidentes de segurança, reduzir falsos positivos, restringir acessos indesejados através da reconfiguração dinâmica da rede, a fim de prover um nível de segurança que permita à rede continuar oferecendo os serviços normais enquanto a ameaça permanece no ambiente, porém, restrita a uma *Honeynet*, sem que os usuários normais e a própria fonte maliciosa percebam e sem que provoque danos ao ambiente.

3.6 Aplicabilidade da Arquitetura SAMARA ao Projeto NIDIA

Conforme foi visto no início deste Capítulo, a arquitetura SAMARA foi concebida para complementar a operação de *firewalls* e IDS, criando uma estrutura de detecção de incidentes e que provesse a reconfiguração dinâmica da rede, utilizando *honeypots* e agentes inteligentes, de modo a possibilitar que o tratamento de incidentes de segurança de rede seja mais efetivo e conseqüentemente aumentar o nível de segurança das organizações. Além disso, um outro objetivo deste trabalho é apresentar uma arquitetura que reflita as proposições de respostas a incidentes para o NIDIA, conforme [18] e [32]. Assim, esta seção descreve como é aplicada a estrutura da arquitetura SAMARA no modelo de IDS do Projeto NIDIA, apresentado por [18].

3.6.1 O Projeto NIDIA

O Projeto NIDIA, introduzido por [18], foi idealizado com o propósito de fornecer uma contribuição para a melhoria das técnicas de detecção de intrusos em redes de computadores com a utilização de agentes inteligentes e redes neurais em mecanismos de reconhecimento de ataques [7], associando, também, capacidades reativas [32]. A Figura 3.8 mostra a arquitetura geral do NIDIA.

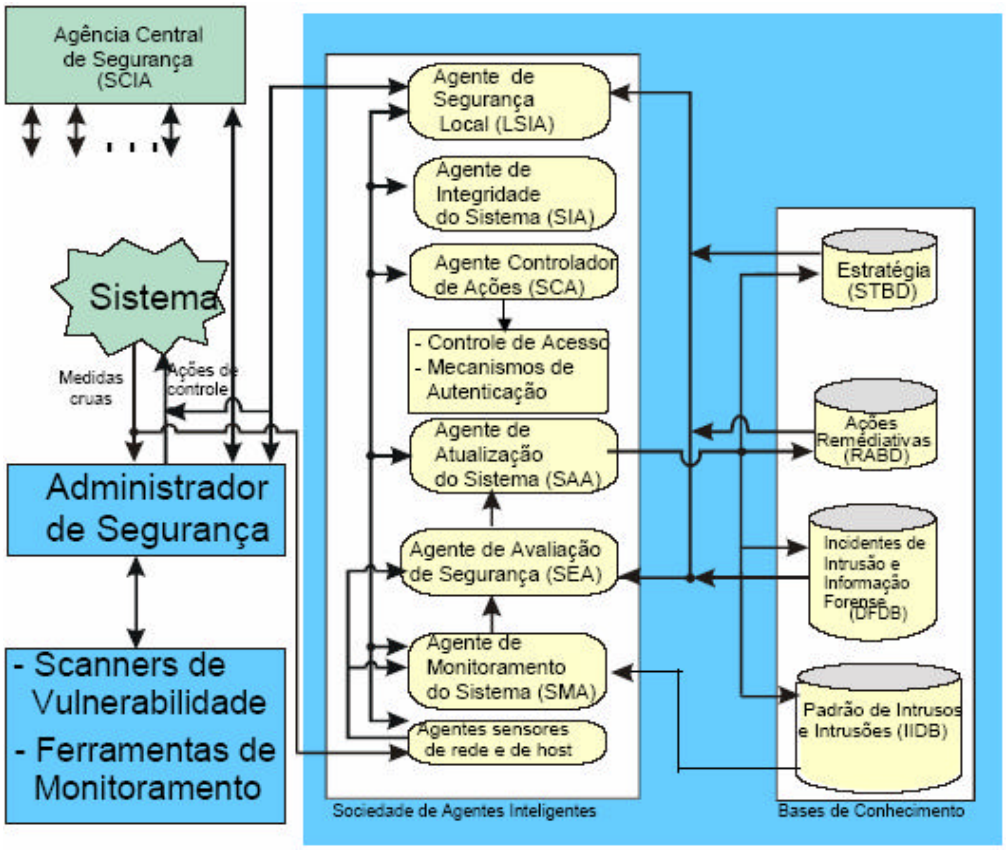


Figura 3.9 – Arquitetura Geral do NIDIA

O sistema NIDIA, esquematicamente visto através da Figura 3.9, baseia-se em uma sociedade multiagente [52] e é composto, basicamente, pelos seguintes elementos:

- **Agentes Sensores**, que capturam eventos em *hosts* (agentes sensores de *host*) e no tráfego da rede (agentes sensores de rede);
- **Agente de Monitoramento de Sistema**, que é responsável por receber os eventos dos agentes sensores, organizar e formatá-los para que possam ser identificados os padrões de ataques e comportamentos anormais na rede e recursos monitorados;
- **Agente de Avaliação de Segurança**, que verifica se o evento corresponde, de fato, a uma tentativa de ataque ou invasão ou se constitui apenas em um falso positivo;

- **Agente de Atualização do Sistema**, que é responsável por manter atualizadas as bases de dados de incidentes, padrões, estratégias e ações;
- **Agente Controlador de Ações**, responsável pelo controle das ações que o NIDIA deve tomar na ocorrência de incidentes de segurança;
- **Agente de Integridade de Sistema**, responsável por garantir a integridade do IDS, buscando eventos inesperados ou fora do padrão de atividade normal dos agentes do sistema;
- **Agente de Segurança Local**, que gerencia a sociedade de agentes e serve de interface entre o IDS e o administrador de segurança, mostrando os status e a configuração dos agentes, a atualização das bases de dados, o registro dos eventos reportados e as ações tomadas pelo sistema;
- **Agente Administrador de Segurança**, agente humano que interage com o IDS através do agente de segurança total para realizar diversas tarefas, tais como: configurar a política de segurança do ambiente, ativar e desativar agentes; gerenciar o status e a configuração do sistema, dentre outras;
- **Bases de Dados**, responsáveis pelos repositórios que armazenam as informações relevantes ao processo. A base STBD é responsável por registrar as estratégias adotadas. A base RABD registra as ações que devem ser tomadas de acordo com a severidade do evento.

Como implementações, [18] contribuiu, além da arquitetura geral do sistema, com os agentes sensores de rede e de *hosts*. Em [7] foram realizadas a modelagem e a implementação de reconhecimento de padrões de ataques utilizando redes neurais e em [32] foram especificadas as funcionalidades do agente controlador de ações, sendo implementado o protótipo desse agente.

3.6.2 Integração da arquitetura SAMARA ao NIDIA

Como parte do Projeto NIDIA, esta dissertação vem contribuir especificamente com a estrutura do Agente *Decoy Server*, idealizada inicialmente por [18] e abordada por [32]. Essa estrutura pode ser vista na Figura 3.10.

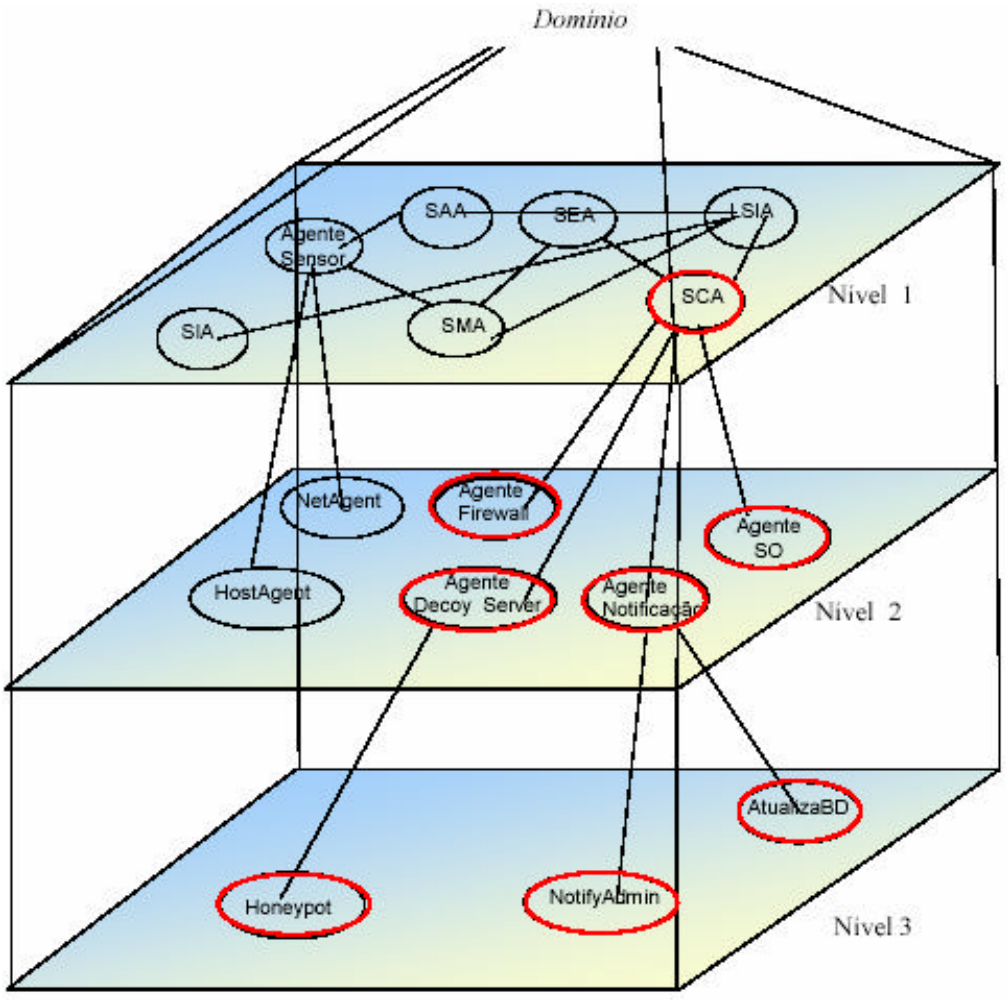


Figura 3.10 – Arquitetura NIDIA em profundidade

Segundo [18], o Agente Controlador de Ações (SCA) ao ser acionado, poderá ativar outros agentes para aplicarem as medidas necessárias contra as tentativas de ataques. Nesse contexto, o Agente *Decoy Server*, responsável por criar um ambiente virtual, poderá ser

ativado, permitindo que o sistema possa obter maiores informações a respeito do intruso, ativando o Agente *Honeypot*. O Agente de *Firewall* e o Agente de Sistema Operacional poderão ser acionados para reconfigurar o ambiente (alterar filtros de pacotes, mudar permissões de usuários, grupos e objetos) e impedir que novas tentativas de ataques sejam efetuadas.

Segundo [32], a arquitetura baseada no Agente *Decoy Server* e no Agente *Honeypot* é formada de uma coleção de agentes com as seguintes responsabilidades:

- Restringir o acesso em *firewalls* e roteadores;
- Desviar o tráfego malicioso para algum *honeypot* com serviços similares aos de um servidor de produção sob ataque;
- Coletar o tráfego malicioso e encaminhar, de modo seguro, ao servidor de *logs* remoto;
- Fazer o rastreamento do invasor e descobrir sua origem.

Com isso, esperava-se um sistema capaz de executar as seguintes respostas:

- Ao descobrir a origem do ataque obstruir, no roteador, todo o tráfego do atacante em direção a recursos de produção, como medida provisória até que a equipe de segurança pudesse entrar em ação;
- Desativar um *host* quando o seu sistema estiver tomado por um atacante;
- Quando um ataque for lançado através de um serviço ou uma porta conhecida para atingir um determinado sistema, parar efetivamente o ataque através do bloqueio dos recursos atingidos (porta e serviço), sem afetar quaisquer outros serviços do sistema;
- Investigar a conexão de rede para tentar identificar um atacante e desviá-lo para um *honeypot* e descobrir as suas intenções.

Durante este trabalho, percebeu-se que uma estrutura de *decoy server* que possibilitasse detectar atividades maliciosas, reagir automaticamente e modificar a estrutura de segurança do ambiente, além de criar um ambiente virtual para conter um suposto atacante (Figura 1.1) com apenas os dois agentes (Agente *Decoy Server* e Agente *Honeypot*) do Projeto NIDIA, acarretaria numa sobrecarga de responsabilidades para esses agentes. Para resolver esse problema, novos agentes foram especificados.

O Agente *Decoy Server* é equivalente na arquitetura SAMARA à CCO. O Agente de *Firewall* é equivalente ao Agente de Reconfiguração Dinâmica da Rede. Na estrutura do NIDIA, o agente SCA é quem comanda o Agente de *Firewall*. Porém, para que a estrutura de *decoy server* funcionasse como mostrado na Figura 1.1, constatou-se que esse agente deveria ser controlado pela CCO e não pelo agente SCA do NIDIA [32], assim como o Agente de Sistema Operacional, equivalente ao Agente de Proteção de *Host*.

Sobre o Agente *Honeypot* do NIDIA, percebeu-se, também que, ao invés de criar-se um *honeypot* para NIDIA, seria melhor acoplar agentes de alerta a *honeypots* baseados em sistemas operacionais e aplicações reais.

O Agente de *Traceroute* Remoto e o Agente de Desvio de Tráfego foram idealizados para atender as especificações de [32].

Dessa maneira, o ponto de integração da arquitetura SAMARA com o NIDIA é o agente SCA. Na ocorrência de um incidente de segurança, esse agente poderá disparar um alerta à CCO, informando os parâmetros de endereços, portas, serviços de origem e destino, além dos protocolos envolvidos. A CCO, por sua vez, dispara os procedimentos vistos no Capítulo anterior.

Assim, baseando-se nas características e funcionalidades especificadas por [18] e [32], a arquitetura SAMARA propõe-se a atender aos propósitos do Projeto NIDIA.

3.7 Comparativo entre as soluções disponíveis e a arquitetura SAMARA

No final do Capítulo anterior foi mostrado um quadro comparativo entre as soluções disponíveis baseadas em *honeypots*. Agora que a arquitetura foi apresentada, cabe compará-la com as outras soluções. A Tabela 3.3 mostra as funcionalidades e recursos da arquitetura aqui apresentada em relação às soluções disponíveis atualmente.

Tabela 3.3 – Quadro comparativo entre as principais soluções baseadas em *honeypots* e a arquitetura SAMARA

	Utilidade	**Nível de Interação/Emulação**	**Disposição Física**	**Mecanismos de Proteção do Hospedeiro**	**Mecanismos de Resposta Automática**
Deception Toolkit (DTK)	Pesquisa	Baixo/Sim	Unidade Isolada	▪ Não Existe	▪ *Log*
CyberCop Sting	Produção	Médio/Sim	Unidade Isolada	▪ Não Existe	▪ *Log*
Specter	Produção	Baixo/Sim	Unidade Isolada	▪ Não Existe	▪ *Log* ▪ *E-Mail* ▪ Rastreamento
BackOfficer Friendly (BOF)	Pesquisa	Baixo/Sim	Unidade Isolada	▪ Não Existe	▪ *Log*
ManTrap	Produção	Alto/Não	Unidade Isolada	▪ Não Existe	▪ *Log* ▪ *E-Mail* ▪ *SNMP traps* ▪ Execução de *Scripts* e binários
Alternativas Caseiras	Pesquisa/ Produção	Alto/Sim	Unidades Distribuídas Virtuais	▪ Não existe	▪ *Log* ▪ *E-Mail* ▪ Controle de Fluxo de Saída
sNET	Produção	Alto/Não	Unidades Distribuídas Mistas	▪ Mudança do Estado de Portas ou Serviços ▪ Redirecionamento de Portas e Serviços ▪ Desligamento do *host*	▪ *Log* ▪ *E-Mail* ▪ *Pager* ▪ Celular
Projeto *Honeynet*	Pesquisa	Alto/Não	Unidades Distribuídas Reais	▪ Não Existe ▪ Controle de pacotes do Fluxo de Saída	▪ *Log* ▪ *E-Mail* ▪ Controle de Fluxo de Saída
Honeyd	Pesquisa	Médio/Sim	Unidades Distribuídas Virtuais	▪ Redirecionamento de Portas e Serviços ▪ Controle de pacotes do Fluxo de Saída	▪ *Log* ▪ *Execução de Scripts e binários* ▪ *Controle de Fluxo de Saída*
SAMARA	Produção/ Pesquisa	Alto/Não	Unidades Distribuídas Reais	▪ Controle de pacotes do Fluxo de Saída ▪ Limitação de taxa de transmissão do Fluxo de Saída	▪ Criação de Bloqueios de Portas e Serviços ▪ Desvio de Tráfego através das tabelas de *firewalls* e roteadores ▪ Rastreamento do suspeito; ▪ *Log* ▪ *E-Mail* ▪ SMS

Como pode ser observado, os sistemas e aplicações que simulam ou utilizam *honeypots* são vários. Há soluções simples como o DTK, que apenas simula vulnerabilidades conhecidas de certas aplicações e gera *logs* sobre qualquer tentativa de exploração daquelas vulnerabilidades, sem oferecer, porém, a possibilidade do atacante interagir com os recursos do hospedeiro. Outras são mais sofisticadas como o *Honeyd* e o *Mantrap* que simulam ambientes virtuais e apresentam funcionalidades complexas muito próximas a sistemas operacionais e aplicações reais. Têm-se, ainda, soluções como o Projeto *Honeynet*, que propõe um ambiente de rede constituído apenas por recursos sem valor de produção, totalmente voltado para a pesquisa de atividades maliciosas.

Por outro lado, soluções voltadas para ambientes de produção que não se restrinjam a apenas notificar o administrador da rede são raras e, quando existem, utilizam dispositivos proprietários e/ou, na maioria das vezes, são muito complexos, além de exigir requisitos de *hardware* e *software* de alta performance, o que desestimula sua adoção.

Assim, como alternativa a esse contexto, foi visto que a arquitetura SAMARA combina o uso de *honeypots* e agentes inteligentes, provendo uma infra-estrutura voltada para a detecção, prevenção e reação a incidentes de segurança como complemento aos esforços de sistemas de detecção de intrusões e outros mecanismos de segurança de redes. Além disso, a arquitetura possibilita o uso de honeypots de baixo custo, que podem ser obtidos e implantados de maneira mais simples pelo pessoal de suporte, já que os honeypots são baseados em sistemas operacionais e aplicações que, por sua vez, já são utilizadas nos recursos de produção do ambiente. A arquitetura SAMARA possibilita o isolamento de atividades maliciosas e suspeitas, permitindo o acompanhamento seguro destas e impedindo ou minimizando os riscos de que recursos de produção e missão crítica venham a sofrer danos e que os serviços de redes e sistemas distribuídos sejam prejudicados. Com isso, espera-se reduzir os riscos de ataques e invasões a recursos de produção.

3.8 Considerações finais

Neste Capítulo foi apresentada a arquitetura SAMARA, constituída por uma sociedade de agentes inteligentes associados a *honeypots* baseados em sistemas operacionais e aplicações encontradas nos ambientes produção. Foram enfatizados os principais componentes da arquitetura, responsabilidades, o princípio de operação e a interação entre os agentes. Foram vistas também as principais vantagens da arquitetura SAMARA, destacando-se a integração de agentes inteligentes e *honeypots* e a reconfiguração dinâmica do ambiente monitorado. Por fim, foi revisto o comparativo entre as principais soluções disponíveis, que utilizam o conceito de *honeypots*, incluindo-se as características da arquitetura SAMARA.

Para demonstrar a viabilidade da arquitetura proposta, o Capítulo a seguir é dedicado às implementações parciais dos protótipos dos principais componentes da arquitetura SAMARA.

4 IMPLEMENTAÇÕES PARCIAIS

Neste capítulo são apresentadas as implementações dos protótipos dos principais componentes da arquitetura SAMARA, destacando-se o Agente de Alerta, a CCO, o Agente de Reconfiguração Dinâmica da Rede e o Agente de Traceroute Remoto.

4.1 Considerações iniciais

Escolheu-se a linguagem Java [45] para as implementações, por apresentar as características de portabilidade em várias plataformas de sistemas operacionais. Foram utilizadas também as bibliotecas *Jpcap* [9] e *libcap* [17] para as tarefas de captura de pacotes e a biblioteca *Jssh* [12] para comunicação segura entre os componentes distribuídos, além de possibilitar as intervenções em *firewalls*, roteadores e sistemas operacionais.

Conforme [18], o uso da biblioteca *libcap* apresenta diversas vantagens, podendo-se destacar a independência de tecnologia de rede utilizada e a portabilidade em várias plataformas de sistemas operacionais. A biblioteca *Jpcap* é um conjunto de classes Java que permitem a programação da biblioteca *libcap* em um nível mais alto de abstração, facilitando o desenvolvimento dos aplicativos. A biblioteca *Jssh* é também um conjunto de classes Java que permitem o estabelecimento de sessões SSH entre aplicações remotas.

4.2 Implementação do protótipo do agente de alerta

O Agente de Alerta é um elemento essencial na arquitetura. Sem ele, os outros componentes não têm o que fazer, pois os eventos reportados pelos agentes de alerta são os

subsídios para os demais componentes. Além disso, é ele quem dá inteligência aos *honeypots* e para os recursos de missão crítica monitorados.

Sua principal função consiste em capturar qualquer atividade de rede que envolva a interface de rede (entrada e saída) de qualquer um dos *honeypots* e recursos de produção monitorados e encaminhá-la para a Console Central de Operações (CCO) que, por sua vez, dispara uma série de procedimentos, conforme já descrito no capitulo anterior. Sua interação com a rede é passiva, atuando como um minúsculo *sniffer* na interface de rede do *host* monitorado. Para tanto, o agente de alerta tem suas rotinas baseadas em funcionalidades das versáteis bibliotecas *Jpcap* e *libcap*.

Na Figura 4.1 tem-se um trecho do código que permite a seleção da interface de rede que se deseja monitorar o tráfego.

```
/*****************************************************************************/
/**
 * Recebe o filtro a ser usado e o tipo de Host como parâmetro
 * @param Endereço do servidor;
 * @param Porta a ser usada;
 * @param Filtro utilizado;
 * @param Tipo de Host
 */
public ColetaConexoes ( InetAddress endServer, int port, String f, int tipoHost )
{
    try {
        this.enderecoConexao = InetAddress.getLocalHost();
    }
    catch ( IOException ioe ) { }

    this.enderecoServidor = endServer;
    this.numPorta         = port;
    this.filtro           = f;
    this.typeHost         = tipoHost;

    try {
        jpcapCaptura = Jpcap.openDevice ( Jpcap.getDeviceList()[1], 1000, false, 20);
        jpcapCaptura.setFilter ( this.filtro, true );
    }

    catch ( IOException ioe ) { }
}
```

Figura 4.1 – Seleção da interface de rede

O Agente de Alerta precisa de três parâmetros para iniciar a monitoração do trafego. São eles:

- A interface de rede que será utilizada na monitoração do tráfego;

- A quantidade de *bytes* a serem capturados;
- O modo de interação com a rede.

O Agente de Alerta somente capturará o tráfego dirigido especificamente para a interface de rede selecionada, por isso, a *flag* define o modo de interação permanece setada em *false*. O outro modo possível seria o modo promíscuo (*flag* setada para *true*), porém, não se encaixa nos propósitos desse agente. Esse modo é utilizado no agente sensor de rede do NIDIA [18].

Em tempo de execução essa seleção é feita através de uma interface gráfica como mostrado na Figura 4.2, a seguir.

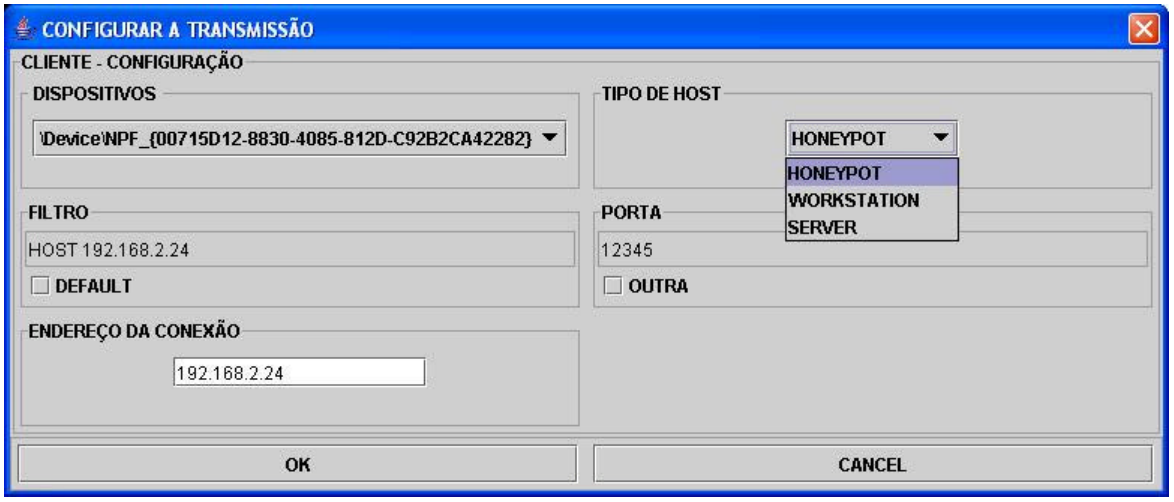


Figura 4.2 – Tela de seleção da interface de rede no agente de alerta

O agente de alerta não se interessa pelo *payload* (carga útil) dos pacotes que atravessam pela interface de rede. Por isso, os dados selecionados correspondem a campos dos cabeçalhos IP, TCP, UDP e ICMP de cada pacote. Essa função já existe no agente sensor de rede que foram implementados no NIDIA por [18]. Além disso, esta arquitetura também propõe um agente de coleta, que será responsável por capturar toda a carga útil dos pacotes que passarem pelas interfaces de rede dos *honeypots* com fins de uma análise mais detalhada

da atividade maliciosa nesses recursos. Assim, o Agente de Alerta captura basicamente as seguintes informações:

- Identificador único da conexão;
- Data e hora de ocorrência do fato no formato dd/mm/aaaa hh:mm:ss;
- O nome do serviço sondado;
- A porta sondada no *host*;
- O protocolo sondado;
- A porta de origem;
- O endereço IP de origem;
- O endereço IP de destino.

Devido às características inerentes da função do agente de alerta, não há necessidade de interface visual para este agente em modo de execução. Ele executa como serviço, ficando escondido enquanto opera. A CCO apresenta o seu *status* assim que o *honeypot* ou o recurso de produção monitorado é ligado, conforme visto no canto superior esquerdo da Figura 4.3, a seguir.

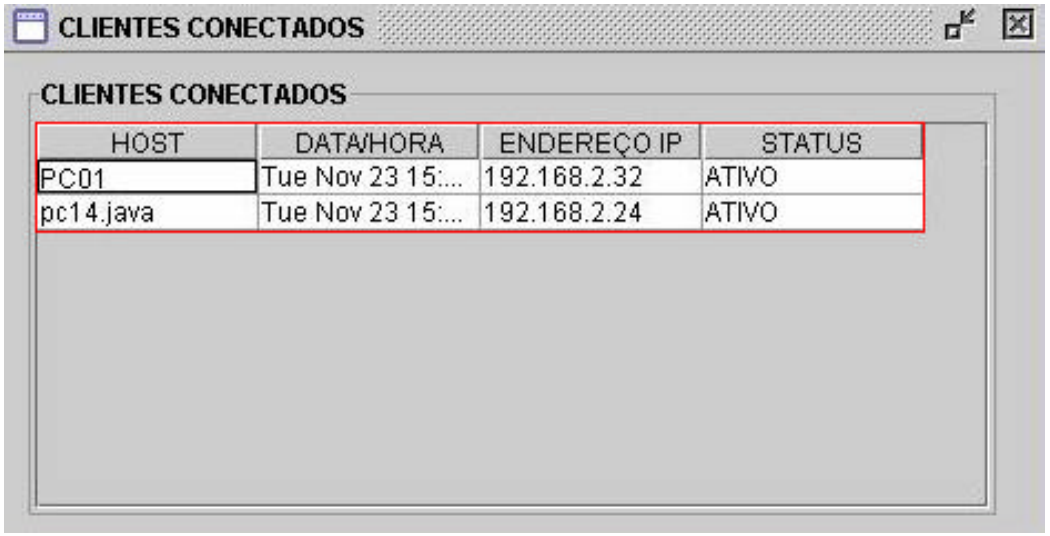


Figura 4.3 – *Status* do Agente de Alerta na CCO

4.3 Implementação do protótipo da Console Central de Operações (CCO)

Conforme visto Capítulo 3, a CCO é o componente da arquitetura SAMARA responsável por várias funcionalidades, destacando-se:

- Agregar e apresentar todos os eventos reportados pelas instâncias do agente de alerta distribuídas nos *honeypots* e recursos de missão crítica;
- Encaminhar ao agente de reconfiguração dinâmica da rede as informações necessárias para que este possa disparar e inserir diretivas em ACLs dos roteadores e *firewalls* do ambiente;
- Encaminhar ao Agente de *Traceroute* Remoto o endereço IP da fonte suspeita para que seja descoberto o possível trajeto até o suspeito;
- Registrar os eventos reportados pelos agentes de alerta em uma base única, de forma a permitir a consolidação das informações e inferir globalmente sobre incidentes reportados isoladamente;
- Notificar o administrador sobre o incidente de segurança reportado;
- Apresentar os status dos agentes de alerta;
- Apresentar e editar as listas negra e branca;
- Apresentar ACLs em roteadores e tabelas de *firewalls*;
- Apresentar a possível origem do atacante, com os dados recebidos do Agente de *Traceroute* Remoto.

A interface gráfica da CCO pode ser vista na Figura 4.4, mostrada a seguir.

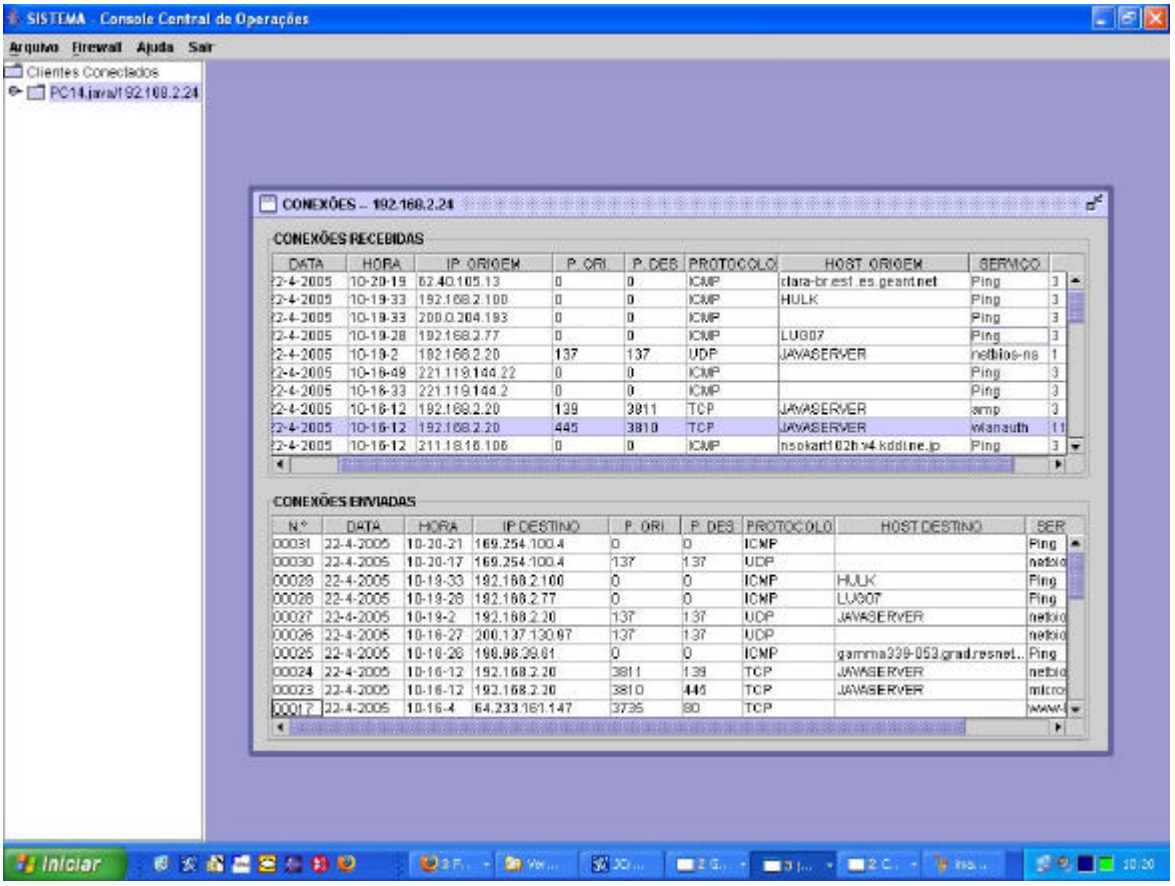


Figura 4.4 – Console Central de Operações (CCO)

Conforme pode ser visto através da Figura 4.4, a CCO mostra em sua janela central uma panorâmica das conexões de entrada e saída reportadas pelos agentes de alerta. Na parte superior dessa janela são mostradas as conexões de chegada. Na parte inferior, são mostradas as conexões de saída dos *hosts* monitorados. A atualização dessa janela é feita em tempo real, no momento em que cada evento acontece. Na janela superior a CCO mostra os status dos agentes distribuídos no ambiente monitorado.

4.4 Implementação do Agente de Reconfiguração Dinâmica da Rede

Conforme já foi visto no Capítulo 3, o Agente de Reconfiguração Dinâmica da Rede é responsável por aplicar diretivas de acesso em roteadores e *firewalls* que compõem o

parque efetivo de mecanismos de segurança da rede. Para tanto, a CCO repassa para esse agente a diretiva de acesso ao mecanismo de segurança ou acesso. São, basicamente, as seguintes informações:

- O endereço IP da fonte suspeita;
- O endereço IP do *firewall* ou *router*;
- O nome do usuário e a senha com poder de administrador no dispositivo;
- O tipo de diretiva:
  - Bloqueio de acesso a um endereço IP, porta e/ou serviço específico ou grupo de *hosts*, portas e/ou serviços;
  - Desvio de acesso para uma porta e/ou serviço em um endereço IP especificado.

A CCO mantém também uma tabela com tipos de *firewalls* e roteadores, pois, a sintaxe da diretiva depende do tipo (*hardware* ou *software*) e da plataforma de *software* (Windows, Linux ou outra qualquer) desses dispositivos.

O Agente de Reconfiguração Dinâmica da Rede é disparado através da CCO. Assim, não precisa de uma interface visual para executar.

A CCO tem uma opção onde o administrador seleciona um determinado *firewall* ou *router* e estabelece uma sessão segura com o mesmo para visualizar ou alterar a configuração, conforme suas necessidades, utilizando a própria interface do dispositivo. Na Figura 4.5 é mostrada a interface do firewall, acessado através da CCO.

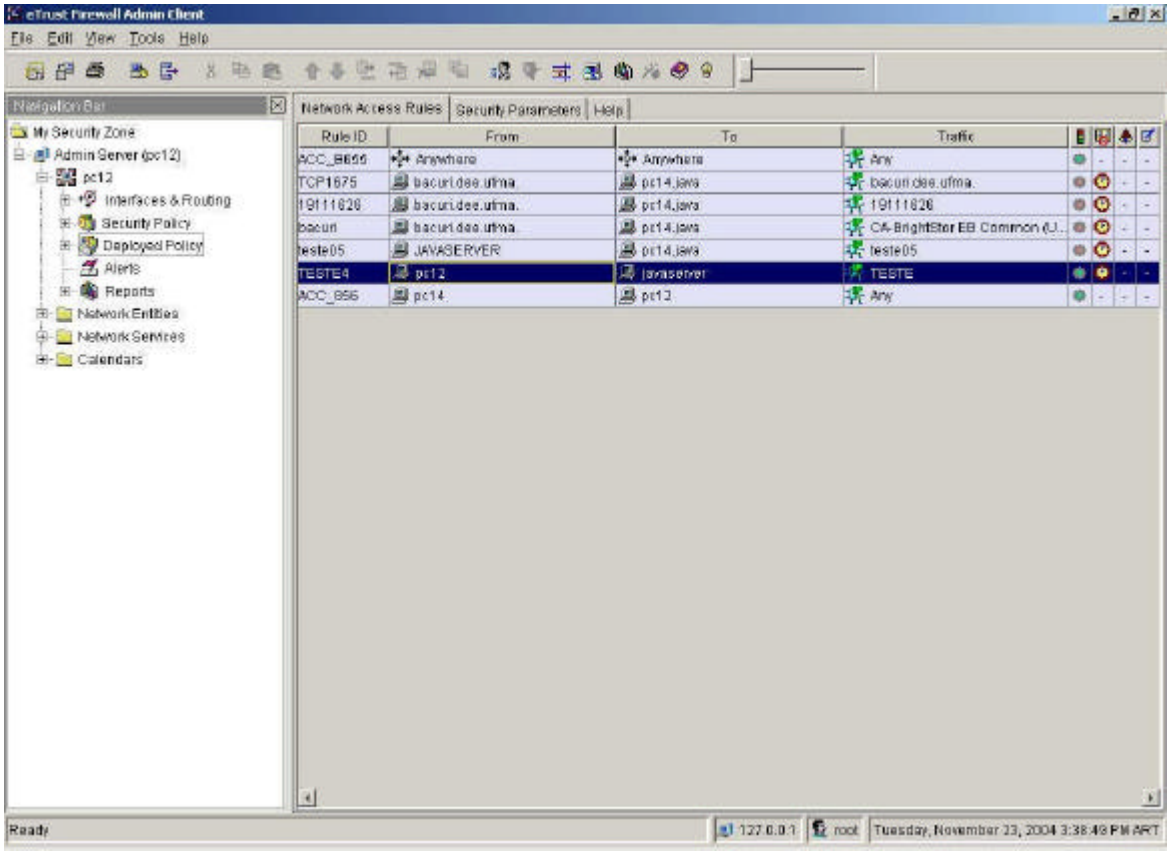


Figura 4.5 – Tela para visualização e configuração do *firewall*

4.5 Implementação do Agente de *Traceroute* Remoto

Conforme visto anteriormente, o Agente de *Traceroute* Remoto possibilita à CCO descobrir o caminho até a fonte que originou a sondagem, invasão ou ataque ao ambiente monitorado. Para tanto, a CCO repassa a esse agente o endereço IP e vários parâmetros sobre a fonte suspeita, o que corresponde a uma entrada na lista negra.

Também foi previsto que o administrador pudesse visualizar o percurso descoberto. Assim, a CCO tem uma opção que permite ao administrador selecionar um endereço da lista negra ou entrar diretamente com o endereço IP de um *host*, obtendo como resultado o percurso (quando possível) até aquele endereço IP. Na Figura 4.6 pode-se visualizar a janela correspondente na CCO.

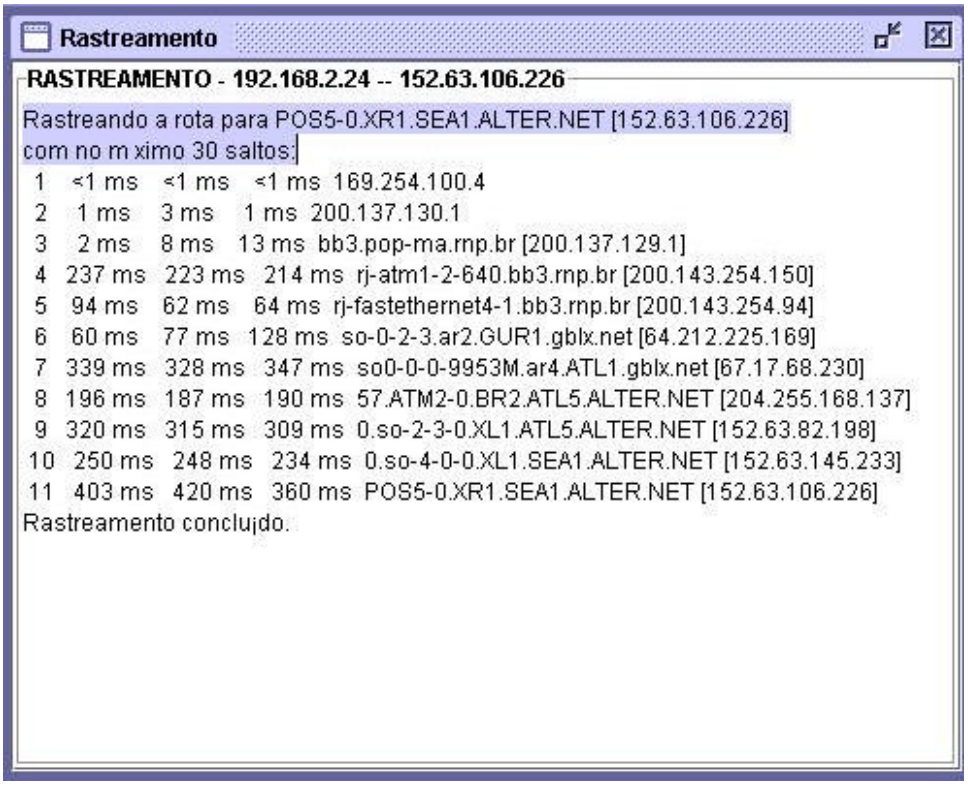


Figura 4.6 – Agente de *Traceroute* Remoto

4.6 Considerações finais

Neste capítulo foram apresentadas as implementações dos protótipos dos principais componentes da arquitetura SAMARA, destacando-se o Agente de Alerta, a CCO, o Agente de Reconfiguração Dinâmica da Rede e o Agente de Traceroute Remoto.

Com se pôde observar, os protótipos ainda não foram implementados integralmente, apenas as funcionalidades básicas. O objetivo das implementações até então realizadas foi demonstrar a viabilidade da arquitetura SAMARA com a utilização conjunta de *honeypots* e agentes inteligentes na detecção, prevenção e resposta automatizada a incidentes de segurança.

5 RESULTADOS PARCIAIS DE SIMULAÇÕES EM LABORATÓRIO

Para demonstrar, de fato, que a arquitetura SAMARA tem viabilidade prática, foram utilizados os protótipos dos principais agentes e da CCO implementados, conforme visto no Capítulo anterior. Neste Capítulo são apresentados os resultados obtidos nos testes de laboratório.

5.1 Considerações iniciais

Como ambiente de testes, foi utilizada a seguinte topologia, conforme mostra a Figura 5.1.

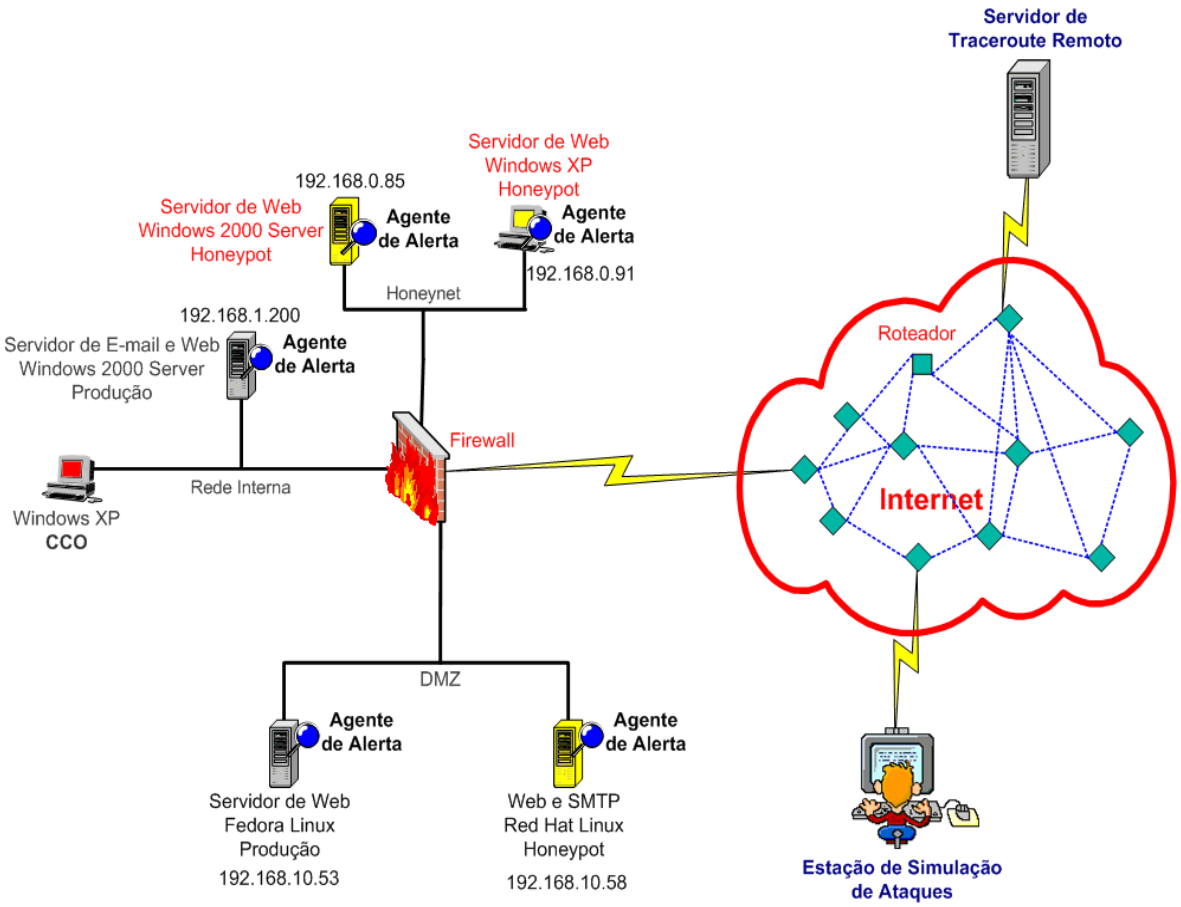


Figura 5.1 – Ambiente de teste da arquitetura

Como pode ser observado na Figura 5.1, foi instalado um *firewall* com 4 (quatro) interfaces de rede, objetivando a criação de um ambiente de quatro regiões:

- Uma Honeynet, composta por dois *honeypots* (um Servidor Windows 2000 rodando IIS 5.0 e uma Estação com Windows XP rodando IIS 5.0), conectados através de um *hub* de 8 portas;
- Uma Intranet, composta por dois *hosts* (um servidor Windows 2000, rodando IIS 5.0 e MS Exchange e a estação de gerenciamento do sistema (CCO)), conectados através de um *hub* de 8 portas;
- Uma DMZ, composta por um *honeypot* (Servidor de Web e E-mail, rodando Red Hat Linux e um servidor de Web de produção, rodando Fedora Linux), conectados através de um *hub* de 8 portas;
- Uma conexão com a Internet. Na realidade, uma conexão com a rede do Laboratório, que possibilitou o acesso à Internet. A estação da simulação de ataques ficou nessa região.

A CCO foi posicionada na Intranet. Foram adicionadas regras no *firewall* de modo a prevenir quaisquer acessos ao *host* da CCO, provenientes da conexão Internet.

5.2 Varredura de Portas

Conforme [18], a varredura de portas é o processo de se conectar a portas TCP e UDP do sistema-alvo para determinar quais serviços estão em execução ou em estado de escuta (*listening*). A identificação de portas nesse estado é crucial para determinar o tipo de SO e aplicativos que são utilizados na rede-alvo. Existem vários tipos de varredura de portas, a saber: a varredura de conexão TCP, a varredura TCP SYN, a varredura TCP FIN, a varredura TCP de árvore de natal, varredura TCP nula e a varredura UDP.

A partir da estação do atacante, identificada na Figura 6.1 como "*Attack Desktop*", e utilizando-se uma ferramenta de varredura de portas disponível na Internet, *SuperScan* [8], realizou-se a varredura na faixa de endereços IP do ambiente de testes (Figura 5.1) e foram obtidos os resultados mostrados na Figura 5.2, a seguir.

```
* + 192.168.0.85
            |___  80  World Wide Web HTTP
                        |___ HTTP/1.1 200 OK..Server: Microsoft-IIS/5.0..Content-Location: http://192.168.0.85/INDEX.HTM..Date: Sun, 01
May 2005 21:13:53


* + 192.168.0.91
            |___  80  World Wide Web HTTP
                        |___ HTTP/1.1 200 OK..Server: Microsoft-IIS/5.0..Content-Location: http://192.168.0.91/INDEX.HTM..Date: Sun, 01
May 2005 21:13:55

* + 192.168.1.200
            |___  25  Simple Mail Transfer
                        |___ 220 honeypot3.domain.com ESMTP MDaemon 7.2.0; Thu, 28 Apr 2005 18:04:29 -0300..
            |___  80  World Wide Web HTTP
                        |___ HTTP/1.1 200 OK..Connection: close..Date: Thu, 28 Apr 2005 21:04:29 GMT..Server: Microsoft-
IIS/6.0..Content-Length: 511..Conten
            |___  110  Post Office Protocol - Version 3
                        |___ +OK domain.com POP MDaemon 7.2.0 ready <MDAEMON-
F200504281804.AA0427890MD0012@domain.com>..

* + 192.168.10.53
            |___  80  World Wide Web HTTP
                        |___ HTTP/1.1 200 OK..Date: Sun, 01 May 2005 21:15:32 GMT..Server: Apache/1.3.33 (Unix) PHP/4.3.10..X-
Powered-By: PHP/4.3.10..Connec

* + 192.168.10.58
            |___  25  Simple Mail Transfer
                        |___ 220 honeypot2.domain.com ESMTP Postfix..
            |___  80  World Wide Web HTTP
                        |___ HTTP/1.1 200 OK..Date: Sun, 01 May 2005 22:11:52 GMT..Server: Apache/2.0.40 (Red Hat Linux)..Accept-
Ranges: bytes..X-Powered-By
```

Figura 5.2 – Resultado de uma varredura de portas no ambiente de testes

Analisando a Figura 5.2, conclui-se que a ferramenta de varredura de portas encontrou todas as máquinas do ambiente de testes. A única exceção foi o *host* da CCO. A razão para o não aparecimento da estação da CCO na varredura da estação de simulação de ataques é que, no *firewall*, havia uma regra bloqueando o acesso ao endereço IP da CCO. A ferramenta ainda identificou as portas e os serviços que estavam rodando nas máquinas do ambiente de testes, a saber:

- No *host* 192.168.0.85, foi identificado um SO Windows, rodando IIS 5.0 e tendo a porta 80 (HTTP) aberta;
- No *host* 192.168.0.91, foi identificado um SO Windows, rodando IIS 5.0 e tendo a porta 80 (HTTP) aberta;

- No *host* 192.168.1.200, foi identificado um SO Windows, rodando IIS 6.0 e tendo as portas 25 (SMTP), 80 (HTPP) e 110 (POP3) abertas;
- No *host* 192.168.10.53, foi identificado u SO Unix, rodando o servidor Apache 1.3.33 e tendo a porta 80 (HTTP) aberta;
- No *host* 192.168.10.58, foi um identificado um SO Red Hat Linux, rodando o servidor Apache 2.0.40 e tendo as portas 25 (SMTP) e 80 (HTTP) aberta.

A ferramenta *SuperScan* conseguiu identificar, em alguns casos, a aplicação servidora nos *hosts* encontrados.

5.2.1 Respostas aos incidentes reportados

Após a reconfiguração dinâmica do ambiente, utilizou-se novamente a ferramenta de varredura de portas. Na repetição do teste, a ferramenta *SuperScan* mostrou apenas os *honeypots* do ambiente, conforme pode ser observado através da Figura 5.3, a seguir.

```
* + 192.168.0.85
        |___   80  World Wide Web HTTP
                  |___ HTTP/1.1 200 OK..Server: Microsoft-IIS/5.0..Content-Location: http://192.168.0.85/INDEX.HTM..Date: Sun, 01
May 2005 21:13:53


* + 192.168.0.91
        |___   80  World Wide Web HTTP
                  |___ HTTP/1.1 200 OK..Server: Microsoft-IIS/5.0..Content-Location: http://192.168.0.91/INDEX.HTM..Date: Sun, 01
May 2005 21:13:55

* + 192.168.10.58
        |___   25  Simple Mail Transfer
                  |___ 220 honeypot2.domain.com ESMTP Postfix..
        |___   80  World Wide Web HTTP
                  |___ HTTP/1.1 200 OK..Date: Sun, 01 May 2005 22:11:52 GMT..Server: Apache/2.0.40 (Red Hat Linux)..Accept-
Ranges: bytes..X-Powered-By
```

Figura 5.3 – Resultado da varredura de portas após reconfiguração dinâmica da rede

Além disso, a CCO enviou um *e-mail* para o administrador, alertando sobre o incidente e, registrou na lista negra o endereço IP do *host* que sondou o ambiente. O Agente de *Traceroute* Remoto também foi acionado, registrando o percurso até a fonte que originou as varreduras. As Figuras 5.4 e 5.5 mostram os detalhes dessas respostas.

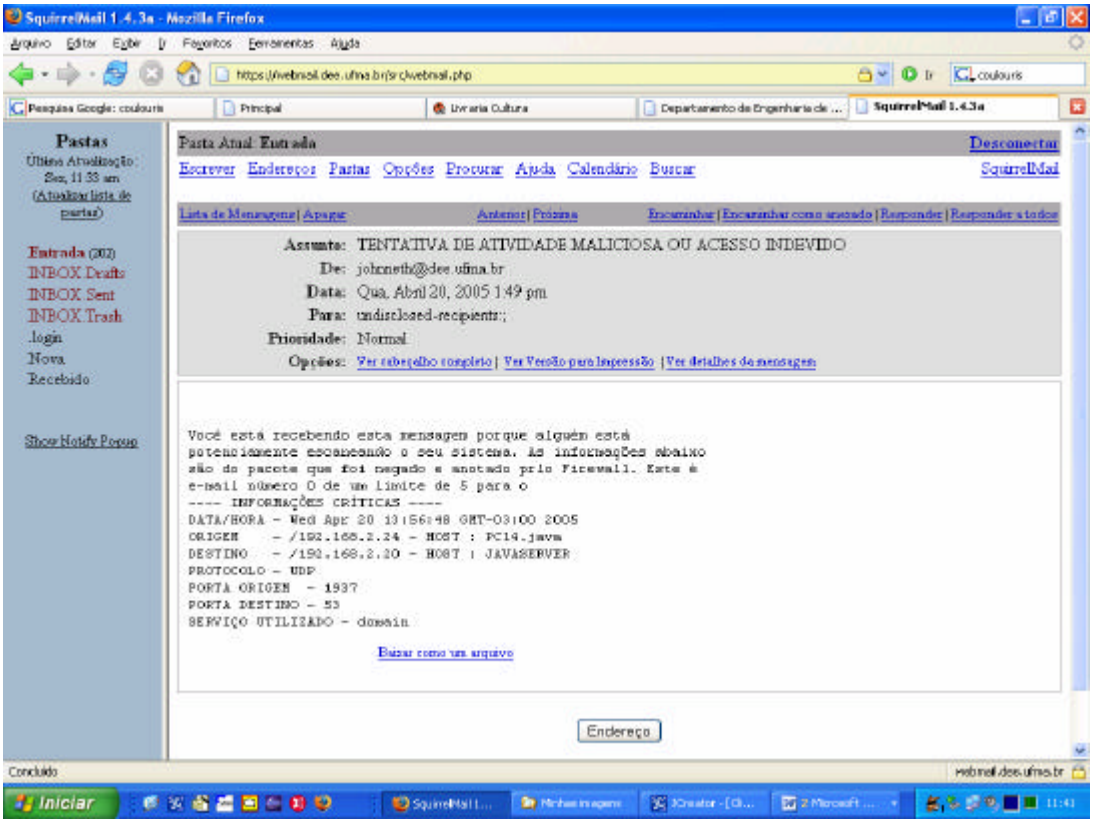


Figura 5.4 – *E-mail* notificando o administrador

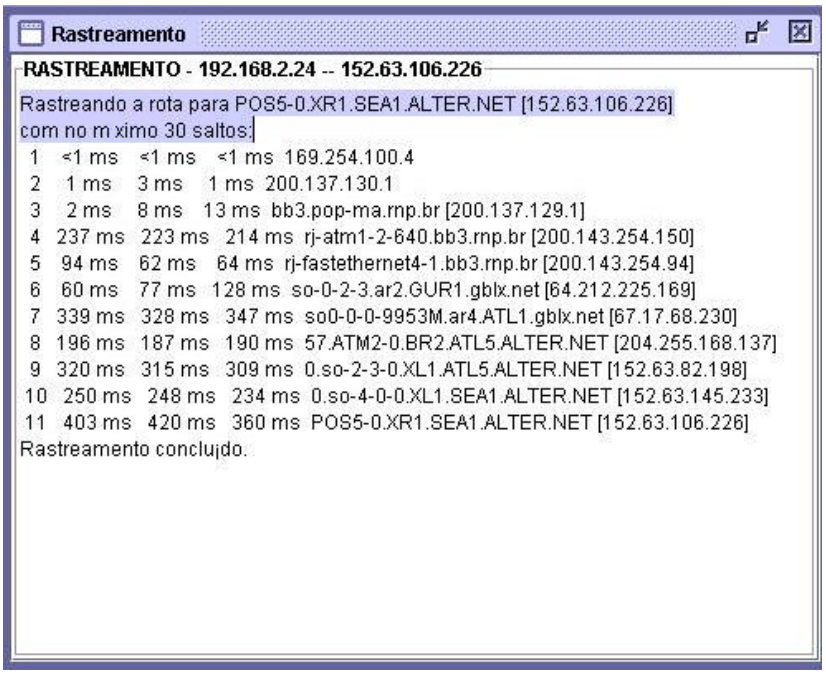


Figura 5.5 – Resultado do *traceroute* remoto

5.3 Considerações finais

Foram realizados vários testes similares no ambiente do laboratório. Os resultados dos testes demonstram a potencialidade de arquitetura quanto à sua utilização na detecção e em respostas a incidentes de segurança, mostrando que o uso de *honeypots* e agentes inteligentes pode ser uma alternativa a ser considerada em sistemas de detecção e resposta a incidentes de segurança.

6 CONCLUSÃO

6.1 Contribuições deste trabalho

Constatou-se a problemática em que vivem os administradores para proverem um nível de segurança adequado para as redes dos ambientes em que trabalham. Isto ocorre, principalmente, devido à falta de informações sobre as classes de ameaças que os recursos de suas redes sofrem no dia-a-dia. Foi visto também que essa falta de informações os deixa praticamente de braços atados, diante de ataques e invasões, vendo suas redes serem atacadas e invadidas e, na maioria das vezes, não podendo fazer nada para impedir ou reduzir os riscos e danos dessas atividades.

Evidenciou-se também o uso de *honeypots* como mecanismos de coleta de informações sobre ataques, atacantes, seus motivos, técnicas e as ferramentas que utilizam em suas investidas maliciosas, danosas e, em muitos casos, onerosas para as organizações atacadas ou invadidas. Há várias iniciativas, tanto em nível comercial quanto em nível acadêmico, voltadas para o uso de *honeypots* em redes e sistemas distribuídos.

Porém, não basta apenas coletar informações sobre os atacantes e suas investidas. Além de obter essas informações, faz-se necessário estar, pelo menos, um passo à frente da comunidade de *hackers* e suas variantes. As técnicas e ferramentas de defesa têm que evoluir face às necessidades atuais e futuras na área de segurança de redes.

Assim, como contribuição, foi proposta neste trabalho a arquitetura SAMARA como alternativa para a utilização conjunta de estratégias de implantação de *honeypots* e agentes inteligentes para complementar a ação dos mecanismos tradicionais de segurança como *firewalls*, IDS's e roteadores. Aliando-se as capacidades dos *honeypots* quanto à detecção, prevenção e agregação de dados sobre atividades maliciosas à versatilidade da

tecnologia de agentes, é possível agir sobre os mecanismos tradicionais de segurança para a detecção e monitoração mais efetivas de ataques e invasões. Com isso, pode-se complementar a ação de *firewalls*, roteadores e IDS's, criando-se uma estrutura que permita a reconfiguração automática e dinâmica da rede quando da ocorrência de incidentes segurança. Conseqüentemente, espera-se reduzir o número de falsos positivos, além de agregar proatividade, eficiência e eficácia à tarefa dos administradores de redes e, conseqüentemente, tornar as redes e sistemas distribuídos mais seguros.

Além disso, outra contribuição do trabalho foi apresentar uma arquitetura que refletisse as proposições de respostas automáticas a incidentes de segurança para o Projeto NIDIA, no que tange a questão do ambiente de decepção baseado no uso de armadilhas de redes e agentes inteligentes, porém, flexível o bastante de forma a possibilitar interagir com outros mecanismos de segurança.

Para demonstrar a aplicabilidade da arquitetura SAMARA, foi apresentada sua integração com a arquitetura multiagentes do Projeto NIDIA, através da a implementação de um protótipo funcional com os principais componentes do modelo apresentado, utilizando-se a plataforma de desenvolvimento Java JBuilder**.**

6.2 Considerações finais

Com o desenvolvimento desse trabalho foi possível mostrar a potencialidade que sistemas baseados em *honeypots* e agentes inteligentes têm, principalmente se utilizados como ferramentas complementares daquelas tradicionalmente utilizadas na defesa e manutenção da segurança de redes.

Apesar das vantagens de soluções baseadas em *honeypots* é importante ressaltar que a implantação desses sistemas em ambientes de produção deve ser feita com

planejamento e sabendo-se dos riscos e limitações dessa tecnologia. Além disso, as organizações precisam estar cientes e preparadas para os resultados e repercussões, inclusive em aspectos legais.

Dificuldades existiram, sobretudo em relação à disponibilidade de referências bibliográficas em quantidade e qualidade suficientes. Um provável motivo para esta constatação é que as organizações ainda não despertaram para o fato de que a divulgação de informações sobre os incidentes de segurança que acontecem na maioria das redes e sistemas distribuídos pode subsidiar pesquisas e o desenvolvimento de novas ferramentas nessa área e conseqüentemente, ajudá-las a minimizar os riscos porque passam seus recursos de missão crítica.

Contudo, a realização deste trabalho foi de significativa importância, pois possibilitou conhecer um pouco mais dessa área de pesquisa que vem crescendo de maneira intensa, sobretudo com a ajuda da Internet, que permite unir esforços em várias partes do mundo em busca de soluções que a comunidade de segurança possa ter em mãos para enfrentar os desafios atuais e futuros.

6.3 Sugestões para trabalhos futuros

Para a continuidade deste trabalho e sugestão para trabalhos futuros nessa área, pode-se citar:

- A implementação dos agentes de coleta, proteção local de *hosts* e desvio de tráfego nos próprios *hosts*;
- O aprimoramento das técnicas de reconfiguração dinâmica da rede e do rastreamento do atacante;

- A implementação de um módulo que permita monitorar graficamente através de símbolos os eventos reportados de modo que reduza o tempo que administrador leva para ficar a par da situação.
- O aprimoramento do agente de alerta de modo que seja possível agregar o índice de severidade já presente no NIDIA e, com isso, gerar alertas mais precisos.
- O aprimoramento da CCO visando o gerenciamento a partir de uma plataforma web.
- O aprimoramento da segurança do sistema, principalmente no que diz respeito à exposição da CCO em um ambiente de rede de produção e quanto ao tráfego de informações entre os agentes e a CCO.
- A expansão efetiva da arquitetura SAMARA para sistemas distribuídos de forma seja possível intercambiar e correlacionar incidentes e contramedidas de segurança globalmente, além de possibilitar o rastreamento mais efetivo das fontes de atividades maliciosas.
- A pesquisa sobre ataques e invasões em redes *wireless* ainda é escassa. Assim, a implementação de pesquisas com *honeypots*, aplicados nesse tipo de meio de transmissão, permitirá conhecer as vulnerabilidades para desenvolvimento de tecnologias e políticas de segurança mais consistentes.
- Estabelecimento de um formato padrão de alertas (XML) para divulgação de vulnerabilidades entre CERTs, visando facilitar o encaminhamento de ações e respostas.
- Verificação da performance e da escalabilidade da solução em um ambiente de produção, comparando os tempos de respostas com e sem o uso da arquitetura.

REFERÊNCIAS

[1] BARBATO, Luiz Gustavo C., MONTES, Antonio. **Técnicas de monitoração de atividades em honeypots de alta interatividade**. Anais do V Simpósio sobre Segurança em Informática (SSI'2003), (São José dos Campos, SP), Novembro/2003.

[2] BAUMANN, Reto, PLATTNER, Christian. **Honeypots**. Zurich, 2001. Dissertação (Mestrado em Ciência da Computação), Swiss Federal Institute of Technology, 2001.

[3] CERT - COMPUTER EMERGENCY RESPONSE TEAM. **Defending yourself:** the role of intrusion detection systems. Disponível em <http://www.cert.org/>. Acesso em Julho/2004.

[4] CHESNICK, Bill. **An evening with Berferd in which a cracker is lured, endured, and studied**. Proceedings of Winter USENIX Conference, San Francisco, CA, January 1992.

[5] COHEN, Fred. **Deception toolkit**. Disponível em <http://all.net/dtk/index.html>. Acesso em Janeiro/2004.

[6] DECEPTION TOOLKIT (DTK). **The deception toolkit home page**. Disponível em <http://www.all.net/dtk>. Acesso em Dezembro/2004.

[7] DIAS, Rômulo Alves. **Um modelo de atualização automática do mecanismo de detecção de ataques de rede para sistemas de detecção de intrusão**. Dissertação (Mestrado em Engenharia de Eletricidade) – UFMA, Maranhão, Novembro, 2003.

[8] FOUNDSTONE. **SuperScan** version 3.00. Disponível em <http://www.foundstone.com>. Acesso em Maio/2004.

[9] FUJII, Keita. **Jpcap**: Java package for libpcap. Disponível em <http://www.goto.info.waseda.ac.jp/~fujii/jpcap/>. Acesso em Junho/2004.

[10] GERLACH, Cristiano. **Técnicas adotadas pelos crackers para entrar em redes corporativas e redes privadas**. http://www.rnp.br/newsgen/9903/crackcorp.shtml. Acesso em Junho/2004.

[11] GLOBAL INTEGRITY CORPORATION. **Honeypot efectiveness study**. Disponível em <http://www. globalintegrity.com>. Acesso em Julho/2003.

[12] GOURIO, Cedric. **Package jssh**. Disponível em <http://www.pitman.co.za/projects/jssh/index.html>. Acesso em Janeiro/2005.

[13] *HONEYNET*.BR. **Desenvolvimento e implantação de um sistema para avaliação de atividades hostis na internet brasileira**. Disponível em <http://www.lac.inpe.br/security/*Honeynet*/papers/hnbr_ssi2002.pdf>. Acesso em Maio/2004.

[14] *HONEYPOTS*.NET. **Intrusion detection, honeypots and incident handling resources**. Disponível em <http://www.*honeypots*.net/>. Acesso em Junho/2004.

[15] ITWORLD.COM. **Start-up's 'decoy' server helps track down hackers**. Disponível em: <http://www.itworld.com/>. Acesso Julho/2004.

[16] ITWORLD.COM. **To trap a thief**. Disponível em: <http://www.itworld.com>. Acesso em Janeiro/2005.

[17] LAWRENCE BERKELEY LABORATORY. **WinCap**: **free packet capture architecture for Windows**. Disponível em< http://netgroup-serv.polito.it/winpcap/>. Acesso em Setembro/2004.

[18] LIMA, Christiane Ferreira Lemos. **Agentes inteligentes para detecção de intrusos em redes de computadores**. Dissertação (Mestrado em Engenharia de Eletricidade) – UFMA, Maranhão, Janeiro, 2002.

[19] MANDIA, K and PROSISE, C. **Incident response**: investigating computer crime, Osborne/McGraw-Hill, USA, 2002.

[20] MICROSOFT TECHNET. **Acesso indevido**. Disponível em <www.technetbrasil.com.br/forum/>. Acesso em Fevereiro/2004.

[21] MÓDULO SECURITY SOLUTIONS. **9ª Pesquisa Nacional sobre Segurança da Informação**. Disponível em: <http://www.modulo.com.br>. Acesso em Janeiro/2005.

[22] NBSO - Brazilian Computer Emergency Response Team. **Cartilha de segurança para internet**. Disponível em < http://www.nbso.nic.br >. Acesso em Maio/2004.

[23] NFR SECURITY. **Back Officer Friendly**. Disponível em <http://www.nfr.com/resource/backofficer.php>. Acesso em Dezembro/2004.

[24] PROVOS, Niels. **A virtual honeypot framework**. 13th USENIX Security Symposium, San Diego, CA, Ago. 2004.

[25] PROVOS, Niels. **Honeyd**: a virtual honeypot daemon. Disponível em <http://www.honeyd.org/index.php/>. Acesso em Maio/2004.

[26] RECOURSE TECHNOLOGIES. **Attacks and countermeasures**. Disponível em <http://www.recourse.com>. 2002. Acesso em Julho/2002.

[27] RECOURSE TECHNOLOGIES. **Mantrap**: a secure deception system. Disponível em <http://www.recourse.com>. Acesso em Novembro/2002.

[28] RECOURSE TECHNOLOGIES. **The evolution of deception technologies as a means for network defense**. Disponível em <http://www.recourse.com>. 2002. Acesso em Julho/2002.

[29] REINING, Christopher J. **Design of a honeypot**. Disponível em <http://blow.packetfu.org:1337/hnd.html>. Acesso em Novembro/2004.

[30] RNP. **Entidades mais vulneráveis a um ataque**. Disponível em <http://www.rnp.br>. Acesso em Julho/2004.

[31] ROJAS, Gislaine Aparecida. **Análise de intrusões através de honeypots e honeynets**. Curso de Tecnologia em Processamento de Dados. Faculdade de Tecnologia de Americana, Setembro, 2003.

[32] SANTOS, Glenda de Lourdes Ferreira. **Um agente inteligente controlador de ações do sistema**. Dissertação (Mestrado em Engenharia de Eletricidade) - UFMA, Maranhão. Novembro, 2003.

[33] SCAMBRAY, J., MCCLURE, S. and KURTZ, G. **Hacking exposed**: network security secrets and solutions, 2nd Edition, McGraw-Hill Co., USA, 2003.

[34] SCHLOSSBERG, Barry. **sNET**: design and implementation of a deception system. Information Security Bulletin, CHI Publishing Ltd., Out.2003, pág. 13-26.

[35] SPITZNER, Lance. **Dynamic honeypots**. Disponível em <http://www.securityfocus.com/infocus/1731>. Acesso em Julho/2004.

[36] SPITZNER, Lance. **Honeypot farms**. Disponível em <http://www.securityfocus.com/infocus/1720>. Acesso em Julho/2004.

[37] SPITZNER, Lance. **Honeypots** - the future. Disponível em <http://www.blackhat.com/presentations/bh-usa-03/bh-us-03-spitzner.pdf>. Acesso em Julho/2004.

[38] SPITZNER, Lance. **Honeypots**: are they illegal? Disponível em <http://www.securityfocus.com/infocus/1703>. Acesso em Julho/2004.

[39] SPITZNER, Lance. **Honeypots**: tracking hackers, 3rd Printing, Addison-Wesley, USA, 2002.

[40] SPITZNER, Lance. **Honeytokens**: the other honeypot. Disponível em <http://www.securityfocus.com/infocus/1713>. Acesso em Julho/2004.

[41] SPITZNER, Lance. **Open source honeypots**: learning with honeyd. Disponível em <http://www.securityfocus.com/infocus/1659>. Acesso em Julho/2004.

[42] SPITZNER, Lance. **Specter**: a commercial honeypot solution for Windows. Disponível em <http://www.securityfocus.com/infocus/1683>. Acesso em Julho/2004.

[43] STEDING-JESSEN**,** Klaus, HOEPERS, Cristine, MONTES, Antonio. **Mecanismos para contenção de tráfego malicioso de saída em honeynets**. Anais do V Simpósio sobre Segurança em Informática (SSI'2003), São José dos Campos, SP. Novembro/2003.

[44] STOLL, Clifford. **The cuckoo´s egg**. London: Bodley Head, 1990.

[45] SUN MICROSYSTEMS, **JAVA (TM) SDK documentation**. Disponível em <http://java.sun.com/products/jdk/1.2/docs/>. Acesso em Setembro/2004.

[46] TEO, Lawrence, Zheng1Yuliang. **Intrusion detection force**: an infrastructure for internet-scale intrusion detection. 2003 IEEE International Workshop on Information Assurance (IWIA 2003). Março/2003.

[47] THE *HONEYNET* PROJECT. **Conheça o seu inimigo**: O Projeto *Honeynet*, 1ª Ed., MAKRON Books, São Paulo, 2002.

[48] THE *HONEYNET* PROJECT. **Honeypots**: definitions and value of *honeypots*. Disponível em <http://project.*Honeynet*.org/papers/>. Acesso em Maio/2003.

[49] THE *HONEYNET* PROJECT. **Know your enemy**: *honeynet*s - what a *honeynet* is, its value, how it works, and risk/issues involved. Disponível em <http://project.*Honeynet*.org/papers/>. Acesso em Novembro/2002.

[50] THE *HONEYNET* PROJECT. **Know your enemy**: the tools and methodologies of the script kiddie. Disponível em <http://project.*Honeynet*.org/papers/>. Acesso em Novembro /2004.

[51] THE *HONEYNET* PROJECT. **Know your enemy**: tracking the blackhat's moves. Disponível em <http://project.*Honeynet*.org/papers/>. Acesso em Novembro /2004.

[52] WEISS, Gerhard; **Multiagent systems - a modern approach to distributed artificial intelligence**, The MIT Press - Cambridge, Massachusetts, London, England, 1999.