

UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

JEAN PABLO MARQUES MENDES

*AVALIAÇÃO DE UM MECANISMO AUTONÔMICO PARA
SEGURANÇA EM REDES BASEADO EM METODOLOGIA DE
DECEPÇÃO*

São Luís
2013

JEAN PABLO MARQUES MENDES

*AVALIAÇÃO DE UM MECANISMO AUTONÔMICO PARA
SEGURANÇA EM REDES BASEADO EM METODOLOGIA DE
DECEPÇÃO*

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Zair Abdelouahab, Ph. D.

Co-orientador: Ariel Soares Teles, M. Sc.

São Luís

2013

Mendes, Jean Pablo Marques

Avaliação de um mecanismo autônomo para segurança em redes baseado em metodologia de decepção / Jean Pablo Marques Mendes - 2013

93.p

Orientador: Zair Abdelouahab, Ph. D.

Monografia (Graduação) - Universidade Federal do Maranhão, Curso de Ciência da Computação, 2013

1.Segurança em redes. 2. Sistemas autônomos. 3. Internet. 4. Honeypots. I.Título.

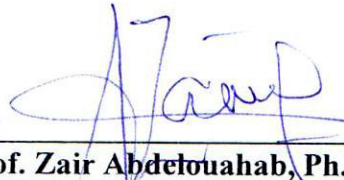
CDU 004.056:007.5

Jean Pablo Marques Mendes

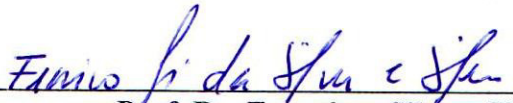
*AVALIAÇÃO DE UM MECANISMO AUTÔNOMICO PARA SEGURANÇA
DE REDES BASEADO EM METODOLOGIA DE DECEPÇÃO*

Monografia apresentada ao curso de Ciência da
Computação da Universidade Federal do Maranhão,
Como parte dos requisitos necessários para a obtenção
do grau de BACHAREL em Ciência da Computação.

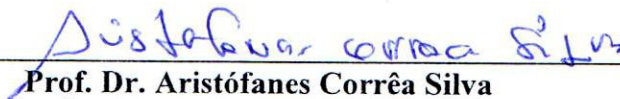
Aprovado em 27 / 12 / 2013



Prof. Zair Abdelouahab, Ph. D
Orientador



Prof. Dr. Francisco Silva e Silva
Examinador 1



Prof. Dr. Aristófanos Corrêa Silva
Examinador 2

Aos meus pais e irmãos.

Aos amigos, pelo apoio e companheirismo.

Resumo

Avaliação de um mecanismo autônomo para segurança em redes. Descrevem-se os conceitos básicos da área de segurança em redes de computadores. Apresentam-se todos os fundamentos da computação autônoma. Expondo-se sobre as funções de cada fase com informações pertinentes para o contexto de segurança de redes. Os resultados desta avaliação mostram que o AutonomicSec atingi as propriedades pertinentes a computação autônoma, o auto-gerenciamento, auto-cura, auto-proteção, auto-otimização, além disso, as métricas são capazes de olhar para Sistemas Autônomos e avaliar com rigor suas propriedades autônomas.

Palavras-chave: Internet, segurança, ataques, honeypots, computação autônoma.

Abstract

Evaluation of an autonomic mechanism for network security. We describe the basics of the safety area in computer networks. We present all the essentials of autonomic computing. Exposing themselves on the functions of each phase with information relevant to the context of network security. Evaluation results show that AutonomicSec reach the relevant properties to autonomic computing, self-managing, self-healing, self-protection, self-optimization, in addition, the metrics are able to look for Autonomic Systems and rigorously evaluate their autonomic properties.

Keywords: Internet security attacks, honeypots, autonomic computing.

Agradecimentos

A Deus, o que seria de mim sem a fé que eu tenho nele.

Aos meus pais, José Pedro e Maria Ivoneide, aos meus irmãos e toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

Ao Professor Zair Abdelouahab pela paciência na orientação e incentivo que tornaram possível a conclusão desta monografia.

Ao meu Co-orientador Ariel Soares que com sua ajuda me propiciou elaboração, criação e desenvolvimento deste trabalho.

Aos colegas de laboratório, em especial a Mario Henrique, Leonardo, Steve, Luiz, Willian, Cláudio, Dileane e Wesley, Renato, Higo pela ajuda e apoio que todos prestaram na produção deste trabalho.

A todos os professores do curso, que foram tão importantes na minha vida acadêmica e no desenvolvimento desta monografia.

Aos amigos e colegas, pelo incentivo e pelo apoio constantes.

*“É bonito e virtuoso, que tanto valor
merece, quem tem viva fé em Deus, nada
de mal lhe acontece”.*

Valdemor Uchoa Mendes

Lista de Figuras

3.1	Propriedades gerais para Computação Autônoma[12]	27
3.2	Ciclo de gerenciamento de sistemas[7]	30
3.3	MAPE-K: Ciclo de gerenciamento autônomo[11]	31
3.4	Definição de uma regra do tipo ECA	34
4.1	Arquitetura do <i>Framework</i> [31]	37
4.2	Comunicação entre componentes do <i>Framework</i> [31]	38
4.3	Arquitetura do Primeiro Ciclo Autônomo[31]	40
4.4	Topologia do Primeiro Ciclo Autônomo[31]	41
4.5	Lista Cinza[31]	42
4.6	Lista Negra[31]	43
4.7	Arquitetura do Segundo Ciclo Autônomo[31]	44
4.8	Topologia do Segundo Ciclo Autônomo[31]	46
5.1	Diagrama de componentes do <i>Framework</i> [31]	49
5.2	Diagrama de classe do <i>Framework</i> [31]	50
5.3	Critérios para seleção da Técnica de Avaliação[32]	54
5.4	Taxonomia das técnicas de avaliação de desempenho[32]	55
5.5	Métricas de Desempenho[32]	56
5.6	Estados alcançado pelo <i>AutonomicSec</i> [11]	57
5.7	Categorização de trabalho autônomo descrito[31]	60
5.8	Lógica do sensor no primeiro ciclo autônomo[31]	62
5.9	Lógica da estratégia autônoma do primeiro ciclo[31]	63
5.10	Cenário para coleta de resultados[31]	64

5.11	Arquivo de configuração da lista branca	70
5.12	Arquivo de configuração dos serviços	70
5.13	Servidores de produção do cenário[31]	71
5.14	Resultado do 1º ciclo	72
5.15	Gráfico dos testes colhidos no 1º ciclo	74
5.16	Regras para controle de tráfego e log no iptables	75
5.17	Lógica da estratégia autônoma do segundo ciclo[31]	76
5.18	Log gerado no firewall	77
5.19	Endereços após filtros realizado pelo monitor	77
5.20	Arquivo de configuração dos Honeypots	78
5.21	Plano de ações criado pelo analisador/planejador	78
5.22	Resultado do 2º ciclo	79
5.23	Gráfico dos testes colhidos no 2º ciclo	79
5.24	Resultado do 1º ciclo	82
5.25	Gráfico dos testes do 1º ciclo	83
5.26	Resultado do 2º ciclo	84
5.27	Gráfico dos testes do 2º ciclo	85

Lista de Tabelas

5.1	Lógica fuzzy aplicada no 1º ciclo	73
5.2	Lógica fuzzy aplicada no 2º ciclo	80
5.3	Lógica fuzzy aplicada no 1º ciclo	82
5.4	Lógica fuzzy aplicada no 2º ciclo	85

Lista de Siglas

ACID	Atomicidade, Consistência, Isolamento e Durabilidade.
DDoS	Distributed Denial of Service.
DNS	Domain Name System.
DoS	Denial of Service.
ECA	Evento-Condição-Ação.
HIDS	Host Based Intrusion Detection System.
ICMP	Internet Control Message Protocol.
IDE	Integrated Development Environment.
IDS	Intrusion Detection Systems.
IGMP	Internet Group Management Protocol.
IP	Internet Protocol.
IPv4	Internet Protocol version 4.
IPv6	Internet Protocol version 6.
LB	Listas Brancas
LC	Listas Cinzas
LN	Listas Negras
MAPE-K	Monitoring, Analysis, Planning, Execution and Knowledge loop.
mDNS	Multicast Domain Name System.
SGBD	Sistemas de Gerenciamento de Banco de Dados.
SSH	Secure Shell.
TCP	Transmission Control Protocol.
UDP	User Datagram Protocol.

Sumário

Lista de Figuras	9
Lista de Tabelas	11
1 Introdução	15
1.1 Objetivos	16
1.2 Organização do Trabalho	17
2 Segurança em Redes	19
2.1 O que é um Ataque	19
2.2 Classificação dos Ataques	19
2.3 Classificação de ataques conforme objetivos	20
2.4 Classificação de ataques conforme a origem	20
2.5 Classificação de ataques conforme a severidade	21
2.6 Formas de Ataques	22
2.6.1 Ataques automatizados	22
2.6.2 Ataques manuais	23
2.7 Ferramentas de segurança	23
2.7.1 Firewalls	23
2.7.2 IDS	23
2.7.3 Honeypots	24
2.7.4 Como funciona um Honeypot	25
2.7.5 Vantagens de um Honeypot	25
2.7.6 Desvantagens de um Honeypot	25

2.7.7	Tipos e Níveis de Honeypots	25
2.8	Conclusão	26
3	Computação Autônoma	27
3.1	Propriedades de Sistemas Autônomicos	28
3.2	Arquitetura de um Sistema Autônomico	30
3.3	Modelo MAPE-K	31
3.3.1	Monitoramento	32
3.3.2	Análise e Planejamento	33
3.3.3	Análise e Planejamento Baseados em Regras ECA	33
3.3.4	Execução de Ações de Reconfiguração	34
3.4	Conclusão	35
4	O Mecanismo AutonomociSec	36
4.1	Framework Autônomico	36
4.2	Primeiro Ciclo Autônomico	40
4.3	Segundo Ciclo Autônomico	43
4.4	Discussão	46
4.5	Conclusão	47
5	Avaliação do Mecanismo Autônomico	49
5.1	Implementação do <i>Framework</i> Autônomico	49
5.2	Avaliação de Desempenho	52
5.2.1	As Métricas de Avaliação de Desempenho	53
5.3	Conclusão	86
6	Conclusões	87
	Referências Bibliográficas	89

1 Introdução

Com o crescimento do tamanho e da complexidade dos sistemas computacionais atuais, dado a grande quantidade de elementos de hardware e software que os compõe, resulta na inevitável incapacidade humana de os gerenciarem. Computação Autônoma[11] trata tal problema transferindo grande parte das responsabilidades administrativas para o próprio sistema, criando-se assim, softwares capazes de se auto-gerenciarem a partir de diretivas de alto nível fornecidas previamente pelo administrador.

Tal complexidade tornam difícil de analisar e avaliar esses sistemas. Para isso, em [22] define nove métricas que são capazes de avaliar Sistemas Autônimos. A Qualidade de serviço, o Custo, a Granularidade/Flexibilidade, Evitar Falhas, o Grau de Autonomia, a Adaptação, o Tempo para se adaptar e tempo de reação(Time to adapt and Reaction Time), a Sensibilidade(Sensitivty) e Estabilização(Stabilisation). Entretanto, apenas três foram o suficiente para avaliar o AutonomicSec[31], são: o Tempo para se adaptar e tempo de reação, essa métrica se preocupar com a reconfiguração do sistema e adaptação, o tempo de adaptar é a medida do tempo que um sistema leva para se adaptar a uma mudança no ambiente, e o tempo de reação envolve parcialmente o tempo de adaptação, isto é, o tempo entre o momento que um elemento do ambiente muda e o sistema reconhece à mudança, então decide sobre o que será necessário para reconfigurar para que o sistema reage a mudança no ambiente, a outra métrica é o Grau de Autonomia, ela diminui o grau de previsibilidade do ambiente e analisa como o sistema reage a coisas que ele não foi projetado. Um grau de previsibilidade também poderia comparar esses recursos, e por último, a sensibilidade, dependendo da natureza da atividade, há alguns Sistemas Autônimos que sofrem constantemente alterações na sua configuração, ficando desagradável esse ciclo, para isso, a sensibilidade irá atuar, ou seja, ela irá nivelar esse processo de reconfiguração, para que esse ciclo fique nivelado, encontre o tempo ideal para que não fique muito custoso a curto período de tempo e que não leve muito tempo para que a reconfiguração seja realizada.

Como já temos as três métricas[22] selecionadas, a primeira métrica(Tempo para se adaptar e tempo de reação) faz referência a métrica genérica assim pode-se dizer, em [1], essas métricas servem para avaliar tanto hardware quanto software, além disso, elas

resolvem problemas como por exemplo: suponha que um Servidor Web seja usado como parte de uma aplicação cliente/servidor. Estações de trabalho clientes são conectadas ao servidor de banco de dados através de uma rede LAN. Clientes trabalham de forma independente e alternam entre processamento local (“thinking”) e “waiting” por uma resposta do servidor.

Qual o tempo de resposta médio de uma transação de um determinado tipo, incluindo tempo de transmissão na LAN? Qual o throughput do servidor?

Essas perguntas são respondidas avaliando o sistema de acordo com as métricas que são: o Tempo de resposta, a Taxa de processamento, a Utilização, a Escalabilidade, a Disponibilidade, a Confiabilidade, o Custo, a Segurança e a Extensibilidade, dentre elas, uma possui afinidade com a métrica escolhida em [22], “o Tempo de respostas” que avalia intervalo de tempo entre requisição do usuário e a resposta do sistema, está relacionada ao “Tempo para de adaptar e tempo para reagir”. Em ambas as métricas, é avaliado o tempo de resposta a um pedido ou interação feita pelo usuário, por exemplo, em um modelo cliente-servidor, é o caso de um Servidor Web, onde o usuário faz uma requisição de um serviço, isso do lado do cliente, do lado servidor, ele recebe essa requisição e retorna com uma mensagem dizendo abriu um canal de comunicação e através desse canal fornece o pedido, no caso do *AutonomicSec*, o usuário tem ações maliciosas contra o sistema, partindo do princípio que toda interação com um *honeypot* é um ataque, logo, o sistema não pode levar muito tempo para reagir, pois corre o risco do atacante efetue por completo sua tarefa, ou descobre que na verdade, o que ele esta interagindo é um *honeypot*.

1.1 Objetivos

Esta monografia tem como objetivo geral propor uma avaliação de um mecanismo autônomo para segurança em rede baseado em metodologia de decepção. A intenção de nossa proposta é avaliar o *AutonomicSec* a partir de métricas definidas, devido a escassez de trabalhos publicados nessa área, dificultando de certa forma, a análise e avaliação de certos sistemas que sejam compostos por Computação Autônoma e, através disso, seja possível avaliar o desempenho e autonomia que o *AutonomicSec* possui. Para isso, considera-se os seguintes objetivos específicos:

- Mostrar de forma clara as três métricas selecionadas (grau de autonomia, tempo para se adaptar e tempo de reação, e sensibilidade) para avaliar o desempenho de Sistemas Autônômicos.
- Analisar as métricas genéricas selecionadas para avaliar o desempenho do AutonomicSec.
- Integrar as Métricas genéricas com as Métricas específicas para avaliar Sistemas autônômicos, para que juntas possam avaliar com maior precisão o AutonomicSec.
- Verificar a partir das métricas definidas específicas da CA, pode-se concluir que o AutonomicSec apresenta conceitos de Computação Autônômica.
- Mostrar o resultado da avaliação colhida pela sobre os testes no AutonomicSec.

1.2 Organização do Trabalho

O restante deste trabalho está estruturado da seguinte forma:

- O **Capítulo 2** são descritos conceitos básicos da área de segurança em redes de computadores e, ainda neste capítulo, os fundamentos conceituais sobre máquinas virtuais são apresentados, mostrando as principais possibilidades de uso de virtualização na área de segurança.
- O **Capítulo 3** descreve todos os fundamentos de CA (Computação Autônômica), mostrando suas propriedades, arquitetura e detalhes do ciclo autônômico, explicando a função de cada fase com informações pertinentes para o contexto de segurança de redes;
- O **Capítulo 4** mostra o mecanismo autônômico para segurança de redes, composta de um framework e dois ciclos autônômicos utilizados para fornecer autonomia a esta área. As funcionalidades e modelagem do framework são apresentadas, como também o contexto em que ela se aplica;
- O **Capítulo 5** descreve a avaliação de desempenho do mecanismo autônômico, sendo realizados testes para avaliação, e no final, mostra os resultados colhidos através da avaliação.

-
- O **Capítulo 6** mostra as conclusões encontradas após os testes, mostrando que o AutonomicSec apresenta propriedades da Computação Autônoma, a auto-gestão como principal propriedade de qualquer outro Sistema Autônomo.

2 Segurança em Redes

Junto com o surgimento da Internet, vieram uma série de inovações, benefícios e oportunidades, e também trouxe consigo diversos riscos a seus usuários e novas oportunidades a pessoas mal-intencionadas, se utilizando de fraudes e ataques virtuais. Surge, então a necessidade de prevenir contra essas ações maliciosas, criando-se barreiras para impedir que algum dano seja causado à nossa propriedade física ou intelectual no mundo virtual, podemos citar danos como: o roubo de informações confidências, que é muito rotineiro na web, onde o atacante costuma capturar de forma ilegal dados sigilosos de usuários e empresas para benefício próprio ou de outrem, uma das formas de se prevenir contra esse tipo de ameaça, é certificar que sua máquina possui atualizadas todos os software que fazem segurança no sistema, não acessar páginas que provavelmente estejam contaminas ou que possui armadilhas para capturar seus dados.

2.1 O que é um Ataque

Segundo [28], um ataque é uma ação nociva à segurança de um sistema que deriva de uma ameaça inteligente, sendo essa ameaça uma tentativa deliberada de evitar os serviços de segurança e violar a política de segurança de um sistema, podendo ser classificado, inicialmente quanto ao seu objetivo em passivo e ativo e também quanto à sua origem em interna e externa.

2.2 Classificação dos Ataques

Para auxiliar na compreensão dos riscos de ataque aos quais os sistemas digitais estão expostos, é necessário classificar os ataques conforme objetivo, origem e severidade. Conforme [28],[29] e [6], existe uma fronteira virtual erguida pelas entidades na forma de sua Política de Segurança. Uma política de segurança é um conjunto de regras que visa regulamentar a produção, acesso e tráfego de informações e recursos computacionais em uma organização e determinar formas de agir em caso de violação destas regras. Este

conjunto de regras é usado como limitador e para determinar o escopo das técnicas e ferramentas de segurança de uma rede.

As políticas de segurança surgem expressões “perímetro de segurança” e “domínio de segurança” como sinônimos que serão usados a seguir.

2.3 Classificação de ataques conforme objetivos

- Ataque Passivo - Ataques passivos são aqueles que não comprometem ou que tem influência no funcionamento do sistema, buscam apenas em obter informações de um sistema. Furtos de senhas, endereços de email, fraudes bancárias são exemplos de ataques passivos. As entidades que mais sofrem com esse tipo de invasão são instituições públicas e privadas (Universidades, Bancos, etc...).
- Ataque Ativo - Ataques ativos se preocupam em danificar fisicamente o sistema, comprometer os dispositivos de uma rede, seja através da desativação de serviços por completo ou danificar parcialmente seus serviços.

2.4 Classificação de ataques conforme a origem

- Ataque interno - Ataques internos são aqueles que são iniciados do lado de dentro do perímetro de segurança que é criado pelas políticas de segurança de uma organização. São considerados ataques internos todas as atividades que visam extrapolar o uso dos recursos computacionais aos quais teriam direitos de acesso regularmente. Geralmente, esses ataques são realizados por funcionários com más intenções contra a instituição a qual faz parte, ou vírus que contaminaram as máquinas dos usuários para depois atacar servidores.
- Ataque externo - Ataques externos são toda ameaça ao sistema de segurança vinda de fora do perímetro de segurança, essas ameaças são gerados por usuários não autorizados a entrar no sistema. No entanto, em uma rede corporativa é possível conceber-se diversos perímetros de segurança, e ataques vindos de outros setores, apesar de estarem partindo da mesma rede física, seriam considerados como ataques externos.

2.5 Classificação de ataques conforme a severidade

Quando o ataque obtêm sucesso, temos a classificação da sua severidade. A severidade é determinada pelo tempo gasto na recuperação e prejuízo que o ataque consegue causar ao sistema afetado. O grau de severidade não é uma informação quantitativa, mas sim qualitativa, e ligada ao objetivo principal da entidade atacada. Um ataque de baixa severidade para uma entidade pode ser severidade crítica para outra. Em [26], define 3 perguntas quando o administrador de sistema está construindo as políticas de segurança:

- Qual o principal objetivo do sistema em relação ao negócio da entidade?
- Quanto tempo a entidade pode funcionar em caso de interrupção dos serviços?
- De todos os serviços disponibilizados, quais são os mais importantes perante os objetivos da entidade?

De posse destas informações é possível determinar quais são as prioridades no caso de falhas múltiplas e contabilizar os danos sofridos em caso de ataques.

- **Baixa Severidade** - São ataques que causam pouco ou nenhum impacto para entidade, não atrapalham o funcionamento da entidade. Um ataques que causasse a deleção de arquivos, mais os mesmos estão em backup prontos para serem recuperados em pouco tempo e a brecha que permitiu seu acontecimento fechada, seria considerado de baixa severidade.
- **Alta Severidade** - Ataques de alta severidade são aqueles que dificultariam o funcionamento da entidade, e para reparar o prejuízo levará tempo e recursos para o conserto. Queda de servidores de arquivos, epidemias de vírus e interrupções no acesso a internet são considerados ataques de alta severidade.
- **Ataques Críticos ou Incapacitantes** - São todos aqueles ataques que causam grande prejuízo a atividades da entidade. Os ataques críticos afetam diretamente o negócio da entidade, e como tal, variam de cenário para cenário. Uma empresa financeira cujo cadastro de clientes furtados, uma entidade de segurança nacional que tivesse seus servidores invadidos são exemplos de ataques críticos.

2.6 Formas de Ataques

Uma vez conhecendo os tipos de ataques é necessário saber como são feitos para poder finalmente proteger os sistema contra os mesmos. Entender as formas de ataque e as ferramentas utilizadas são uma necessidade para se conseguir gerar ferramentas e técnicas de prevenção a novas ações.

Duas formas de ataques são definidos: ataques manuais e ataques automatizados.

Ataques automatizados são mais comuns e responsáveis pela maioria das invasões, enquanto ataques manuais são mais perigosos, devido à maneira como acontecem, exige bastante experiência de quem vai executar, e geralmente que está do outro lado realizando a invasão é o hacker, que maliciosamente tentar bular o perímetro de segurança do sistema.

2.6.1 Ataques automatizados

Ataques automatizados são aqueles que não é necessário muita atenção do ser humana para serem efetuados, podendo serem executados através de script e software específicos para invasão.[23]

Existem algumas formas de ataques automatizados: vírus, worms, cavalo de tróia.

- Vírus - são seções de códigos maliciosos, que para agir precisa ser inserido entre o código que será invadido, é considerado ataque automatizado, pois sua capacidade de replicação não depende do atacante, mas sim do atacado, quanto mais sistemas interagirem com o atacado, mais infecções serão efetuada.
- Worms - diferem dos vírus, ele não precisa de um hospedeiro, ele já vem por completo, ele se propaga através de mensagens de correio, no meio de arquivos baixados da internet pelo usuários, são considerados ataques automatizados pelo mesmo motivo dos vírus, ele se propaga independente da atividade do atacante, e sim do atacado.
- Cavalo de Tróia - são software aparentemente inofensivos, mais camuflão seu verdadeiro objetivo, que depois de está dentro do sistema, procuram abrir portas para que o atacante possa efetuar o restante do ataque, essas portas são chamadas de *BackDoor*.

2.6.2 Ataques manuais

Diferem dos ataques automatizados pela motivação e em perfil do executante. Um ataque manual o invasor escolhe cuidadosamente seu alvo e um objetivo antes de realizar a atividade maliciosa, varia desde simples pichação até fraudes bancárias. Uma vez escolhendo seu alvo, o invasor incessantemente irá em busca de alguma brecha até que consiga encontrá-la. Os atacantes manuais geralmente constroem suas próprias ferramentas, depois utilizarem em uma ataque e obter sucesso, será disponibilizado na internet.

2.7 Ferramentas de segurança

Ferramentas de segurança vem para agregar ainda mais as políticas de segurança já criadas, fornecendo ao ambiente de rede maior segurança na manipulação e administração de dados.

Um administrador pode dispor de três ferramentas básicas para integrar a segurança exposta: *firewalls*[4], sistemas de detecção de intruso-IDS[5] e *honeypots*[26], apresentados a seguir.

2.7.1 Firewalls

Firewalls faz alusão a “parede de fogo”. Na verdade, o nome é usado para dar idéia de uma parede que separa dois ambientes, interno e externo, essa ideia nos dar a informação de que o ambiente externo é inseguro, já o interno é seguro, essa ferramenta foi criada para administrar e oferecer mais segurança ao tráfego que ocorre entre as redes. O firewall define uma política de acesso à rede que obriga a todos os pacotes a passarem pelo *firewall* antes de entrar na rede. Dessa forma, pacotes passam por filtros, caso forem suspeitos eles são bloqueados, impedidos de chegarem nos seus destinos. Assim, o *firewall* mantém seguro a rede de ações maliciosas.

2.7.2 IDS

Sistemas de Detecção de Intrusão(IDS) tem como finalidade descobrir se houve uma tentativa de invasão à rede se houve comprometimento de algum elemento dessa

rede. Em caso positivo, o IDS gera alarmes para informar sobre ações que violem a políticas de segurança de uma organização. O IDS difere do escopo do *firewall* e serve como complemento à essa ferramenta. Ao passo que os *firewalls* situam-se às margens das redes, na transição de uma rede e outra, os IDS precisam estar depois do *firewall*, ou melhor, dentro da rede, para que possa analisar os pacotes que conseguiram passar pelo *firewall*, caso encontre algo suspeito, irá alertar o administrador do sistema para que possa tomar as devidas providências necessárias para coibir a ação maliciosa.

Como citado em [26] que faz uma analogia, um *firewall* é a porta de um cofre, o IDS é o sensor de movimento que monitora a sala do cofre. A porta do cofre protege seu interior do meio externo, mas o sensor de movimento, ao detectar uma presença na sala do cofre, dispara os alarmes apropriados.

2.7.3 Honeypots

A principal medida de defesa que podemos usar contra um ataque cibernético, é poder conhecer os passos do atacante, como ele age e atua. No mundo da informática existe muito poucas informações de como são efetuados os ataques e como eles acontecem. Se não for possível conseguir prever o que o inimigo pode fazer contra o sistema, não tem como implementar meios para se proteger.

Hoje existem algumas ferramentas para analisar o comportamento dos inimigos, com elas podemos saber o que ocorre após uma invasão de um sistema e qual a atitude de um invasor depois que ele consegue comprometer o sistema. Saber quais os passos dele até a invasão, como ele age e conhecer seus objetivos. Uma dessas ferramentas é o *Honeypot*[26] que traduzindo significa “pote de mel”. Essa ferramenta usada na sua melhor forma, conseguiu obter informações relevantes e primordiais do invasor.

Desde já, muitas empresas tem buscado ferramentas de segurança para garantir melhor confiabilidade no tráfego de suas informações. Logo vem ganhando um número cada vez maior de adeptos: a do estudo dos agressores e de seus *modi operandi*(modos de operação). Honeypots permite descobrir como um invasor mal-intencionado “trabalha” e ao mesmo tempo identifica suas ferramentas - e o mais importante - seu alvo.

2.7.4 Como funciona um Honeypot

Simulando um simples ataque, o primeiro contato do hacker com o *honeypot*, ele realiza um *telnet* para uma determinada máquina, o fake Server desta máquina irá emular o *telnet* e responde ao comando capturando as informações do atacante[4].

2.7.5 Vantagens de um Honeypot

Os Honeypots foram criados para serem comprometido. Com base nisso, ninguém que esteja fora da rede, ao interagir com a máquina(*honeypot*), passará a ser um invasor. Partimos do princípio que qualquer tentativa de se comunicar com o sistema é uma sondagem, varredura ou ataque de estranhos. Os *Honeypots* aprimora a detecção ao reduzir o número de falsos positivos, pois as tecnologia tradicionais sobrecarregam o trabalho dos administradores. Os honeypots também coletam informações sobre a identidade dos atacantes. Outra grande vantagem do *honeypots* é que, como ele não é um sistema que pertença a produção, pode ser retirado da rede a qualquer momento para análise dos dados.

2.7.6 Desvantagens de um Honeypot

Como os *honeypots* são sistemas isolado, além disso, é também uma ferramenta com a qual os atacantes podem interagir, o que é algo extremamente arriscado se mal implementado, o risco caso ocorra, é que o *honeypot* pode servir de trampolim para invadir outras redes que não esteja a espera desse tipo de ataque.

2.7.7 Tipos e Níveis de Honeypots

Os *honeypots* possuem dois tipos que são: *honeypots* de pesquisas e honeypots de produção, já os níveis são: baixa e alta interação.

- Quanto aos tipos de Honeypots:

- *Honeypots* de pesquisa: são programadas para obter o máximo possível de informações dos atacantes, ele não toma nenhuma prevenção, apenas capturam informações das ferramentas utilizadas.

- *Honeypots* de produção: nesse caso o *honeypot* logo após ser comprometido, ele toma alguma providência, ou seja, ele toma alguma reação de contra medida, geralmente, os *honeypots* de produção possui um elemento que é a distração e dispersão, essas técnicas confundem e ganham mais tempo para que o invasor não desconfie que a máquina para o qual estava interagindo, é na verdade uma *honeypot* .

- Quanto aos níveis de *Honeypots*:

- *Honeypots* de baixa interação: esse tipo de *honeypot* emulam serviços e sistemas operacionais, não permitindo que o atacante interaja com o sistema.
- *Honeypots* de alta interação: são compostos por sistemas operacionais e serviços reais e permitem que o atacante interaja com o sistema.

- Baseados na Implementação:

- *Honeypots* virtuais - são softwares que emulam serviços, servidores e tráfego de rede. Estes software são configurados para abrirem portas de conexão e responderem a requisições externas. O grau de resposta varia de software para software. O *Honeyd* é uma ferramenta de implementação de *honeypots* virtuais, e é utilizado no *AutonomicSec*.
- *Honeypots* reais - são implementações de sistemas operacionais e serviços válidos, sobre os quais são implementadas modificações que aprimoram o nível de geração de alertas e históricos de registros do sistema.

2.8 Conclusão

Uma vez vistas as formas de ataques e a facilidade que podem realiza-las, deve-se construir e preparar com muita atenção as políticas de segurança que é um item obrigatório a todas entidades que possuem negócio na internet.

Firewalls e *IDS's* são a opção mais comum para aumentar o nível de segurança em sistemas digitais; no entanto; até mesmo essas ferramentas possuem suas fragilidades, e para auxiliar na diminuição destas falhas, pode-se sugerir as ferramentas como os *honeypots*.

3 Computação Autônoma

O termo “Computação Autônoma” surgiu em outubro de 2001, em um manifesto produzido pela IBM, alertava para dificuldade do gerenciamento de softwares complexos como o principal obstáculo para futuras inovações na indústria de TI. O documento destaca o conceito de Computação Autônoma (do inglês *Autonomic Computing - AC*) – sistemas computacionais capazes de se auto-gerenciarem dado um conjunto abstrato de objetivos pré-definidos pelo administrador. Além disso, o termo traz uma conotação biológica com o sistema autônomo humano, onde a principal função é manter as funções vitais para que o organismo humano se mantenha equilibrado. Da mesma forma, um sistema autônomo deve prover mecanismos de mudanças em seus componentes que podem provocar alterações no comportamento do sistema, de modo que haja o mínimo de intervenção do administrador.

O principal objetivo da AC é auto-gerenciamento. Para atingir esse objetivo, quatro propriedades são essenciais: auto-cura (*self-healing*), auto-otimização (*self-optimizing*), auto-proteção (*self-protecting*) e auto-configuração (*self-configuring*) e ciência de contexto como mostra a figura 3.1. Essas propriedades costumam ser chamadas de propriedades *self-**.

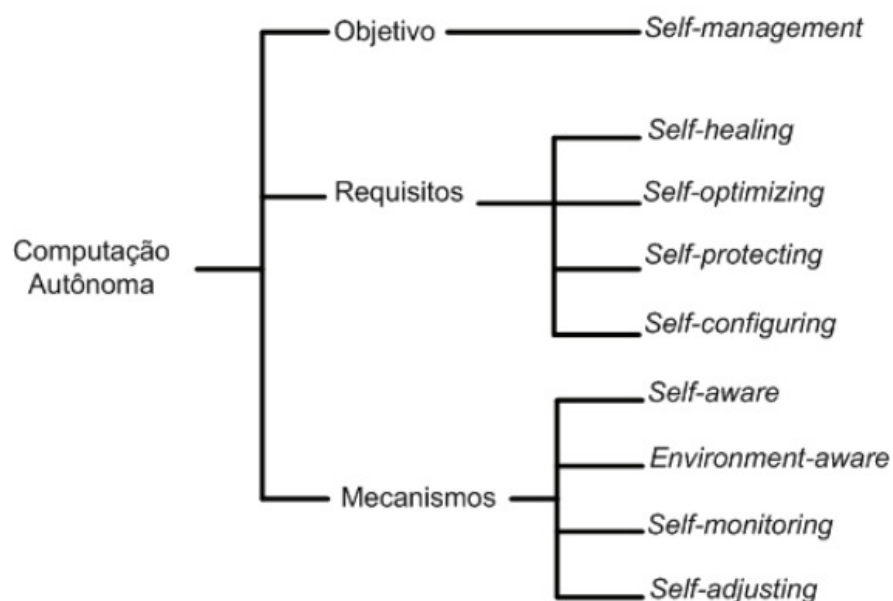


Figura 3.1: Propriedades gerais para Computação Autônoma[12]

Auto-cura é a propriedade do sistema que fica responsável por recuperar a configuração original, quando falhas forem detectadas, voltando a manter o sistema em condições normais antes do ataque.

Auto-otimização consiste na capacidade do sistema em manter ajustado a utilização e uso dos recursos, afim de manter a demanda dos usuários.

Auto-proteção refere-se à capacidade do sistema de defender-se de ataques. Para tanto, o sistema deve previamente ter conhecimento sobre potenciais ameaças, para poder prover mecanismos para tratá-las.

Auto-configuração fica responsável por configurar automaticamente o sistema às novas circunstâncias percebidas em virtude do seu próprio funcionamento ou como apoio a processo de auto-cura, auto-otimização ou auto-proteção.

Por último, ciência de contexto. Esse mecanismo permite que sistemas computacionais tenham conhecimento dos componentes que os formam, seus estados de suas conexões com outros componentes, bem como recursos disponíveis no ambiente e o estados dos mesmos.

3.1 Propriedades de Sistemas Autônômicos

A essência da CA é o auto-gerenciamento. Para implementá-lo, o sistema deve ao mesmo tempo estar atento a si próprio e ao seu ambiente. Desta forma, o sistema deve conhecer com precisão a sua própria situação e ter consciência do seu ambiente operacional em que atua. Do ponto de vista prático, conforme em [10], o termo computação autônômica tem sido utilizado para denotar sistemas que possuem as seguintes propriedades:

- **Autoconsciência (self-awareness):** O sistema conhece a si próprios: seus componentes e inter-relações, seu estado e comportamento;
- **Consciência do contexto (context-aware):** O sistema deve ser ciente de contexto de seu ambiente de execução e ser capaz de reagir a mudanças em seu ambiente;
- **Auto-otimização (self-optimizing):** O sistema é capaz de detectar degradações de desempenho e de realizar funções para auto-otimização;

- **Auto-proteção (self-protecting)**: O sistema é capaz de detectar e proteger seus recursos de atacantes internos e externos mantendo sua segurança e integridade geral;
- **Autocura (self-healing)**: É a propriedade do sistema que assegura sua recuperação efetiva e automática, quando falhas são detectadas. Entretanto, ao contrário de técnicas de tolerância a falhas tradicionais, auto-cura requer não só o mascaramento da falha, mas também a identificação do problema e seu reparo imediato, sem interrupção do serviço e com o mínimo de intervenção externa;
- **Aberto (open)**: O sistema deve ser portátil para diversas arquiteturas de hardware e software e, conseqüentemente, deve ser construído a partir de protocolos abertos e padronizados;
- **Capacidade de antecipação (anticipatory)**: O sistema deve ser capaz de antecipar, na medida do possível, suas necessidades e comportamentos considerando seus contextos e de se auto-gerenciar de forma pro-ativa;
- **Autoconfiguração (self-configuring)**: É a característica do sistema que o permite ajustar-se automaticamente às novas circunstâncias percebidas em virtude do seu próprio funcionamento ou como apoio a processos de auto-cura, auto-otimização ou auto-proteção;

As propriedades de autoconfiguração, auto-otimização, auto-cura e autoproteção são suficientes para realizar a visão original da Computação Autônômica[25]. Para a incorporação das propriedades de auto-otimização e auto-cura, mecanismos de autoconsciência, consciência de contexto e autoconfiguração devem ser requisitos do sistema.

Mais recentemente a comunidade de AC tem discutido o que realmente pode ser auto-gerenciamento, uma vez que alguns serviços isolados propiciam momentos em que um sistema pode gerenciar a si próprio. Serviços podem ser encontrados, por exemplo, em um módulo de otimização de consulta de um SGBD (Sistema de Gerenciamento de Banco de Dados), um gerente de recursos de um sistema operacional ou software de roteamento de uma rede. No entanto, o termo AC é aplicado para designar sistemas em que mudanças de políticas ocorrem para refletir o ambiente de execução corrente. Ou seja, mudanças devem ocorrer dinamicamente. Além disso, para distinguir sistemas autônomos

de sistemas puramente adaptativos (como, por exemplo, alguns sistemas multimídia que se adaptam a flutuações da largura de banda disponível), os primeiros devem exibir mais de uma característica de gerenciamento[11].

3.2 Arquitetura de um Sistema Autônomo

Arquitetura de sistemas autônomos visam formalizar um quadro de referência que identifica as funções comuns e estabelece os alicerces necessários para alcançar a autonomia. Em geral, essas arquiteturas apresentam soluções para automatizar o ciclo de gerenciamento de sistemas, conforme mostrado na figura 3.2, o qual envolve as seguintes atividades:

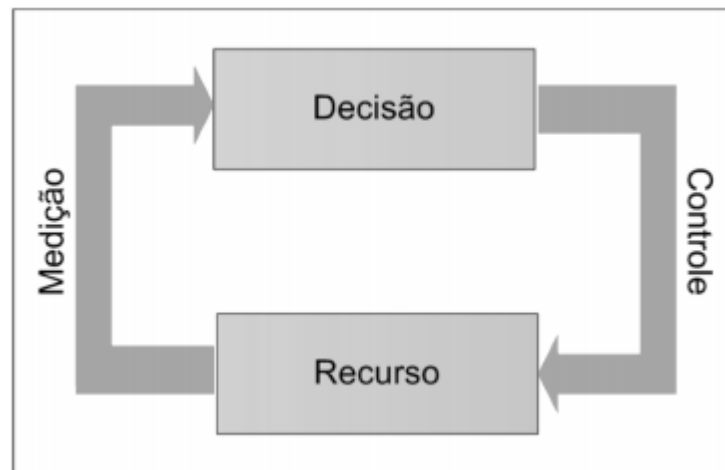


Figura 3.2: Ciclo de gerenciamento de sistemas[7]

- **Monitoramento ou Medição:** Coleta, agrega, correlaciona e filtra dados sobre recursos gerenciados;
- **Análise e Planejamento:** Analisa os dados coletados e determina se devem ser realizadas mudanças nas estratégias utilizadas pelo recurso gerenciado;
- **Controle e Execução:** Escalona e executa as mudanças identificadas como necessárias pela função de análise e decisão.

3.3 Modelo MAPE-K

Em 2003 a IBM propôs uma versão automatizada do ciclo de gerenciamento de sistemas chamado de MAPE-K[15], representando na figura 3.3. Este modelo está sendo cada vez mais utilizado para interrelacionar os componentes arquiteturais dos sistemas autônomicos. De acordo com essa arquitetura, um sistema autônomico é formado por um conjunto de elementos autônomicos.

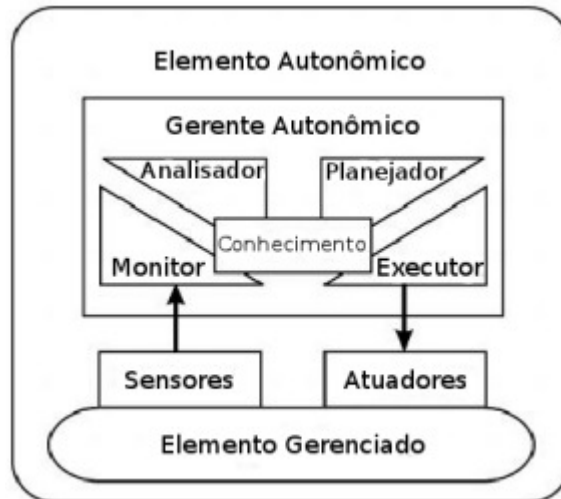


Figura 3.3: MAPE-K: Ciclo de gerenciamento autônomico[11]

Um elemento autônomico contém um único gerente autônomico que representa e monitora um ou mais elementos gerenciados. Cada elemento autônomico atua como um gerente responsável por promover a produtividade dos recursos e a qualidade dos serviços providos pelo componente do sistema no qual está instalado.

No ciclo MAPE-K autônomico figura 3.2, o elemento gerenciado representa qualquer recurso de software ou hardware o qual é dado o comportamento autônomico através do acoplamento de um gerenciador autônomico, podemos ser por exemplo um servidor Web ou bando de dados, um componente de software específico em um aplicativo (por exemplo, o otimizador de consulta em um banco de dados), o sistema operacional, um conjunto de máquinas em um ambiente de rede, etc.

Os sensores (sensors) são responsáveis por coletar informações do elemento gerenciado. Estes dados podem ser os mais diversos possíveis, por exemplo, o tempo de resposta das requisições dos clientes, caso o elemento gerenciado seja um servidor Web. As informações coletadas pelo sensores são enviadas aos monitores (monitors) onde são interpretados, preprocessadas e colocadas em um nível mais alto de abstração, para então

serem enviadas para etapa seguinte no ciclo, a fase de análise e planejamento (analyze and planing). Nesta fase, tem-se como produto uma espécie de plano de trabalho, que consiste de um conjunto de ações a serem executadas pelo executor (execute).

O componente responsável por fazer as alterações no ambiente é chamado de atuador (effectors). Somente os sensores e atuadores possuem acesso direto ao elemento gerenciado. Durante todo ciclo de gerenciamento autônomo, pode haver a necessidade de uma tomada de decisão, dessa forma, faz-se necessário também a presença de uma base de conhecimento (knowledge), sendo que esta é comumente mais explorada na fase de análise e planejamento[14].

Este modelo é implementado utilizando dois ou mais ciclos de gerenciamento autônomo, um ou mais ciclos de controle local e um global. Os ciclos de controle local tratam apenas de estados conhecidos do ambiente local, sendo baseado no conhecimento encontrado no próprio elemento gerenciado. Por esta razão, o ciclo local é incapaz de controlar o comportamento global do sistema. O ciclo global, por sua vez, a partir de dados provenientes dos gerenciadores locais ou através de um monitoramento a nível global, podem tomar decisões e atuar globalmente no sistema. No entanto, a implementação das interações entre os diversos níveis existentes vai depender das necessidades e das limitações da aplicação.

3.3.1 Monitoramento

O monitoramento corresponde a primeira fase do ciclo autônomo MAPE-K. Nesta etapa, sensores são utilizados com a finalidade de obter dados que reflitam mudanças de comportamento do elemento gerenciado ou informações do ambiente de execução que sejam relevantes ao processo de auto-gerenciamento. Os sensores são dispositivos de hardware ou software que estão diretamente ligados ao componente que se deseja monitorar.

O tipo de informação coletada bem como o modelo do(s) monitor(es) empregados são específicos para cada tipo de aplicação. Contudo, alguns requisitos são comuns entre eles, tais como a filtragem, a escalabilidade, a dinamicidade e a tolerância a falhas [21]. Um monitor deve prover filtragem, pois a quantidade de dados coletados pode crescer muito rapidamente e consumir recursos de transmissão, processamento e armazenamento. Grande parte desses dados podem ser de pouca relevância e, portanto,

descartá-los não implicaria em prejuízo. Considerando-se que um sistema pode crescer indefinidamente, contendo inúmeros objetos sob monitoramento, a tarefa dos monitores não pode consumir recursos e reduzir o desempenho do sistema. Em contrapartida, as mudanças no ambiente ou no comportamento do sistema não podem afetar o serviço de monitoramento [13]. Além disso, falhas podem ocorrer e o monitor pode apresentar algum comportamento atípico ou mesmo parar de funcionar completamente. Dessa forma, devem ser levadas em consideração provisões como redundância e mecanismos de recuperação de falhas dos monitores.

3.3.2 Análise e Planejamento

A análise vem após o monitoramento no ciclo de gerenciamento MAPE-K, tendo como fase seguinte o planejamento. Em geral estas duas fases são geralmente implementadas em um único componente. O processo de análise e planejamento é essencial para autonomia do sistema, pois é nele que são geradas as decisões sobre quais modificações serão realizadas no sistema, que corresponde ao elemento gerenciado.

A fase de análise e planejamento recebe como entrada os eventos e seus dados gerados pelo sistema de monitoramento e gera como saída um conjunto de ações, também chamado de plano de ações. Estas ações correspondem às operações de reconfiguração que, de acordo com o mecanismo de tomada de decisão, devem ser executadas no sistema a fim de manter o equilíbrio do sistema considerando seus objetivos. Muitas técnicas podem ser empregadas na fase de análise e planejamento, entre as quais se destacam o uso de funções de utilidade, aprendizado por reforço e regras Evento-Condição-Ação (ECA), sendo esta última detalhada a seguir.

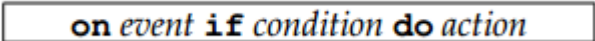
3.3.3 Análise e Planejamento Baseados em Regras ECA

Regras do tipo ECA determinam as ações a serem realizadas quando um evento ocorre desde que certas condições sejam satisfeitas. ECA rules são especificações declarativas de regras ou regulamentações que determinam o comportamento de componentes de aplicações [7]. ECA rules consistem de eventos, condições e ações.

Para cada evento é definido um conjunto de regras que podem gerar uma ou mais ações. Esses eventos também podem satisfazer as condições de diferentes

regras e algumas dessas regras podem gerar ações que conflitam entre si. Nem sempre esses conflitos podem ser detectados durante a escrita das regras, sendo muitas vezes descobertos em tempo de execução.

Em ECA rules o evento especifica quando as regras deverão ser acionadas. A condição é a parte a ser verificada, podendo esta ser satisfeita ou não. E por fim, a ação, a qual representa a operação a ser realizada caso a condição seja satisfeita. ECA rules são definidas da seguinte forma:



```
on event if condition do action
```

Figura 3.4: Definição de uma regra do tipo ECA

3.3.4 Execução de Ações de Reconfiguração

Na etapa de execução do ciclo MAPE-K são realizadas reconfigurações no sistema de forma a restabelecer seu equilíbrio. A finalidade da reconfiguração é permitir que um sistema evolua (ou simplesmente mude) incrementalmente de uma configuração para outra em tempo de execução, introduzindo pouco (ou, idealmente, nenhum) impacto sobre a execução do sistema. Desta forma, os sistemas (ou aplicações) não necessitam ser finalizados, ou reiniciados, para que haja as mudanças.

A autoconfiguração (self-configuring) é a característica do sistema autônomo que o permite ajustar-se automaticamente às novas circunstâncias percebidas em virtude do seu próprio funcionamento, de forma a atender a objetivos especificados pelos processos de autocura, auto-otimização ou autoproteção [7].

O processo de reconfiguração é realizado pelos executores através dos atuadores. Executores recebem como entrada um plano de ações gerado na etapa de análise e planejamento e utiliza os atuadores pertinentes para implementar as ações de reconfiguração descritas no plano. As reconfigurações devem ser realizadas dinamicamente, sem impor a necessidade de parar e/ou reiniciar o sistema.

3.4 Conclusão

Nesse capítulo foi visto que o auto-gerenciamento é a característica necessária para um sistema ser classificado como autonômico. Foi mostrado as principais propriedades de sistemas autonômicos e que para um sistema alcançar o auto-gerenciamento é necessário provê as propriedades de auto-configuração, autotomização, autoproteção, autocura, autoconsciência e ciência de contexto.

Também apresentou-se a arquitetura de um sistema autonômico e, em seguida, o modelo MAPE-K, proposto originalmente pela IBM e que é a referência mais utilizada para o desenvolvimento de sistemas autonômicos. Por fim, foi descrito as fases que envolvem o ciclo do gerenciamento autonômico, mostrando detalhes particulares de cada uma.

4 O Mecanismo AutonomociSec

Neste capítulo apresentamos o framework AutonomicSec[31], em uma visão de mais alto nível, mostramos sua arquitetura, topologia e funcionamento. Logo depois, o primeiro e segundo ciclo autonômico é apresentado, quais os elementos que os compõe, e uma sucinta discussão sobre o tema.

O nosso trabalho é inspirado na ideia de CA e sua utilização para resolver problemas em segurança de redes. Então temos primeiramente um framework autonômico com a finalidade de ser reutilizado em ambientes de redes por sistemas de segurança. Em seguida utilizamos o próprio framework para mostrar dois ciclos autonômicos que utilizam como base *honeypots*. O conjunto composto pelo framework e pelos dois ciclos autonômicos denominamos de Mecanismo Autonômico para Segurança de Redes baseado em Decepção, nome inspirado na ideia de Metodologias de Decepção (Seção 2.7.3), e o chamamos de *AutonomicSec* (*Autonomic Security* - Segurança Autonômica). Nas próximas seções apresentamos detalhes da arquitetura, topologia e funcionamento da proposta desse mecanismo.

4.1 Framework Autonômico

O nosso *framework* é organizado seguindo o modelo MAPE-K para fornecer a possibilidade de alcançar integração, cooperação, inteligência e, principalmente, autonomia aos sistemas de segurança de redes. Neste modelo gerentes autonômicos realizam as atividades de sensoriamento do ambiente de execução, análise de contexto, planejamento e execução de ações de reconfiguração dinâmica. A arquitetura do *framework* pode ser visto na figura 4.1.

Como visto na arquitetura do framework, sensores são componentes de software que podem coletar dados de qualquer natureza, sendo o tipo desses dados que determina o próprio tipo do sensor. O número de sensores é extensível, podendo haver crescimento na quantidade de dados sensoreados. Cada sensor necessita passar para o AutonomicSec, além das suas informações coletadas, o tipo (identificador) do monitor no qual é

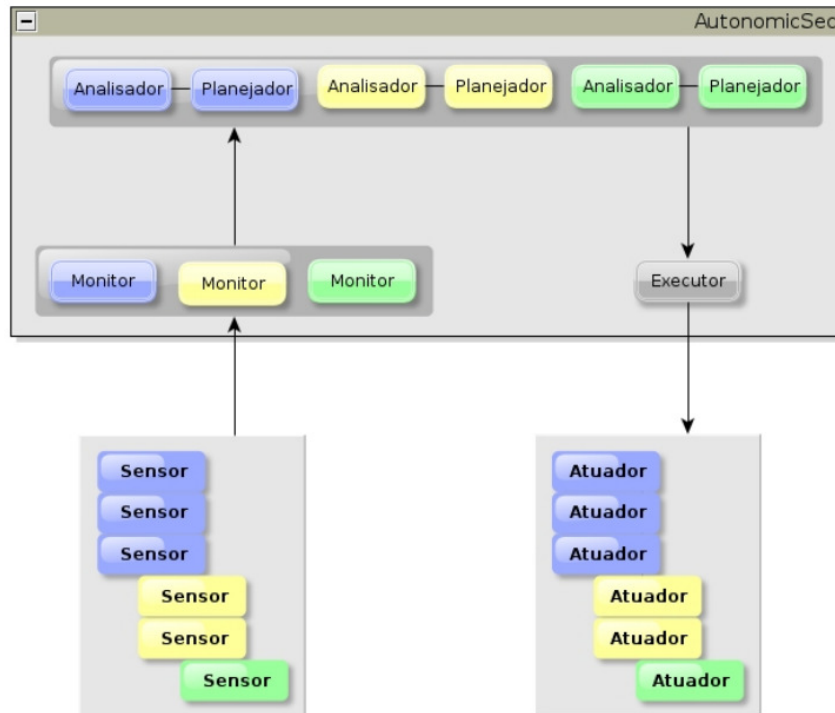


Figura 4.1: Arquitetura do *Framework*[31]

responsável por tratar o dado coletado. Portanto, a comunicação remota entre sensores e AutonomicSec transfere dois campos: o próprio dado coletado e o identificador do monitor que tem a capacidade de tratar esses dados. Com isso, é possível ter vários tipos de sensores ligados a vários tipos de monitores. Porém, cada sensor necessariamente deve possuir uma ligação com pelo menos um monitor, caso contrário os dados sensoreados serão descartados pelo AutonomicSec.

Essa maneira nos quais sensores e monitores se comunicam fornece possibilidade de crescimento e organização para o sistema. Por exemplo, tendo como base a Figura 4.1, uma rede possui três sistemas de segurança onde é possível coletar dados para, a partir desses, realizar alguma tomada de decisão, são eles: três sensores A, dois sensores B e um sensor C. Cada sensor envia para o AutonomicSec o dado coletado e o seu tipo. Então o AutonomicSec fica responsável por passar os dados coletados pelos sensores do tipo A para o monitor A, dos sensores do tipo B para o monitor B e do sensor C para o monitor C. Em cada monitor é criada um fila de requisições dos sensores contendo os dados que devem ser filtrados. Caso não haja nenhuma informação a ser tratada por um monitor, este componente (processo) ficará em modo de espera (wait).

Internamente ao AutonomicSec, logo que ele é inicializado, é feita uma ligação entre componentes, através da passagem da referência (nome) de suas instâncias.

Chamamos isso de gerenciamento de registros de componentes. O monitor inicialmente é registrado no servidor AutonomicSec, que esperará por requisições sockets de sensores, os quais enviarão dados coletados através dessas conexões. O analisador e planejador são implementados como um único componente, assim ele deve ser registrado em pelo menos um monitor. Por fim, o executor, único no sistema, é registrado em todos os componentes de análise e planejamento. Dessa forma o monitor sabe que deve encaminhar os dados relevantes para um determinado analisador/planejador, que por sua vez irá passar um plano de ações de reconfiguração para serem tomados pelo executor, como pode ser visto na Figura 4.2.

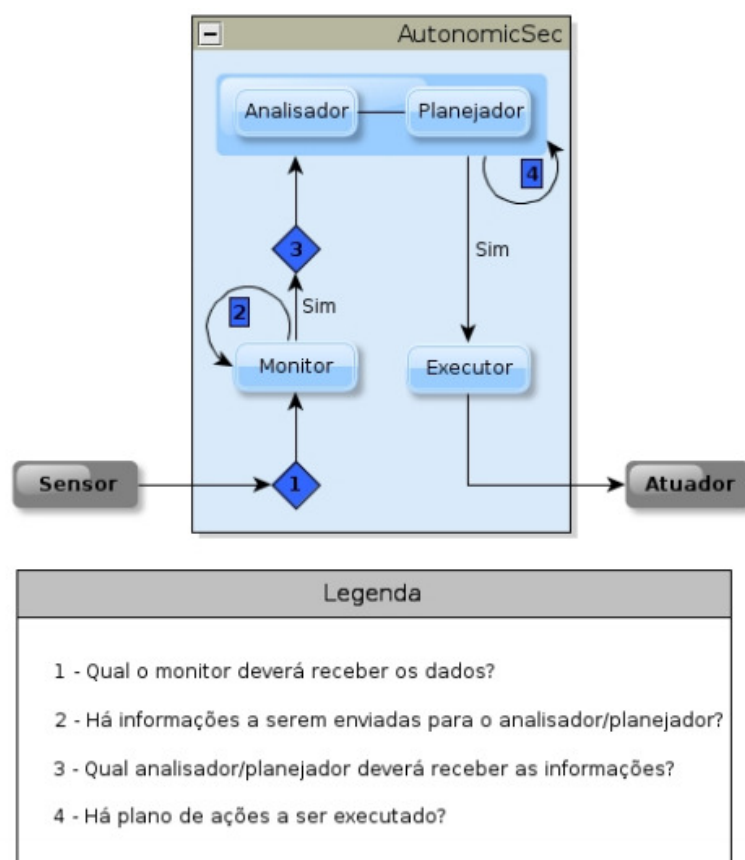


Figura 4.2: Comunicação entre componentes do *Framework*[31]

O plano de ações resultante do processamento realizado pelo componente de análise e planejamento é baseado no que chamamos de estratégia autônômica. Através desta última são geradas as decisões sobre quais modificações serão realizadas no elemento gerenciado. Estas modificações correspondem às operações de reconfiguração que, de acordo com o mecanismo de tomada de decisão, devem ser executadas. A estratégia autônômica adotada pela fase de análise e planejamento deve levar em consideração os objetivos do ciclo a qual ela faz parte.

O executor é único para todo o sistema e processa as ações em série, sendo estas as características fundamentais para este componente, pois as ações de reconfiguração no elemento gerenciado podem ser conflitantes. Igualmente aos monitores, no componente de análise e planejamento é criada uma fila, mas esta é contida de dados filtrados e considerados relevantes. E no executor também é criada uma fila, contendo plano de ações a serem executadas. Logo, quando não há nenhum dado a ser processado pela fase de análise e planejamento ou plano de ações a ser tomado pelo executor, estes entram em modo de espera.

O executor tem a função de simplesmente executar os planos de ações enviados pelos analisadores/planejadores. Ele aplica ações no elemento gerenciado através dos atuadores. Estes, por sua vez, interagem diretamente com o elemento gerenciado. Os atuadores podem modificar, por exemplo, a configuração de algum sistema de segurança da rede, com o objetivo de aumentar o nível de dificuldade para a concretização de ataques. Como visto na Figura 4.1, podem existir vários tipos de atuadores e em quantidade escalável, considerando que o elemento gerenciado é a rede e não sistemas isolados. Diferentemente da ligação que existe entre sensores e monitores, os planos de ações podem ser executados através de qualquer atuador, independente do seu tipo ou da sua quantidade, dependendo apenas da tomada de decisão realizada pelo componente de análise e planejamento.

A abordagem que utilizamos para a nosso framework foi tentar seguir fielmente o modelo MAPE-K, considerando todos seus componentes e a base de conhecimento (knowledge) foram utilizados para construir o *AutonomicSec*. Esta última representada pela estratégia autônômica. Consideramos e acreditamos que nosso framework possa ser utilizado em diversos ambientes de rede para a integração e cooperação entre sistemas, mais especificamente entre sistemas de segurança de redes.

Para mostrar a viabilidade de utilização do framework, mostraremos os dois ciclos autônômicos focados em segurança de redes, que estendem todos seus componentes. O processo de criação de ciclos autônômicos através do reuso do framework é descrito no próximo capítulo. Os ciclos propostos são detalhados a seguir.

4.2 Primeiro Ciclo Autônomo

Com a utilização do framework, mostraremos o primeiro ciclo autônomo. Este ciclo tem a funcionalidade de gerar regras de firewall baseadas em logs de honeypots. Como visto na Figura 4.3, os componentes sensores são representados por honeypots, ou seja, os dados sensoreados são logs a nível de rede gerados por honeypots de baixa interatividade. Esses logs registram interações com o honeypot na camada de rede. Então qualquer interação com um honeypot gera log e esse é enviado para o componente monitor, no AutonomicSec.

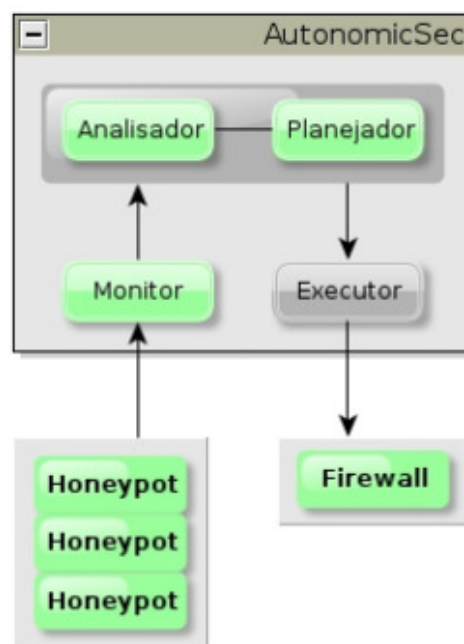


Figura 4.3: Arquitetura do Primeiro Ciclo Autônomo[31]

O monitor recebe os dados sensoreados e realiza uma filtragem do que é relevante para ser enviado à fase de análise e planejamento. Os dados recebidos são: endereço IP de origem (possível intruso), IP de destino (*honeypot* que recebeu interação), porta de origem, porta de destino (informa o serviço), timestamp (horário e data) e protocolo. Este último podendo ser: *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP), *Internet Control Message Protocol* (ICMP) ou *Internet Group Management Protocol* (IGMP). Para esse ciclo autônomo são descartados: *timestamp*, porta de origem e interações com o protocolo IGMP, pois não são informações utilizadas pela estratégia autônoma. Dessa maneira, são enviadas apenas informações úteis.

Nós chamamos a tomada de decisão realizada no componente de análise e planejamento de **estratégia autônoma baseada em intenções**. As regras de firewall geradas por ela são de acordo com a intenção do intruso ao interagir com um honeypot. Caso a interação com o honeypot seja a um determinado serviço, o atacante terá acesso bloqueado a todos os serviços equivalentes da rede de produção. Como visto na Figura 4.4, os honeypots emulam dispositivos e serviços equivalentes aos servidores de produção. Em suma, é criada uma honeynet, sendo esta uma imagem da rede de produção. Esse processo é realizado de maneira transparente para qualquer usuário, inclusive o próprio intruso, tendo como base sua intenção.

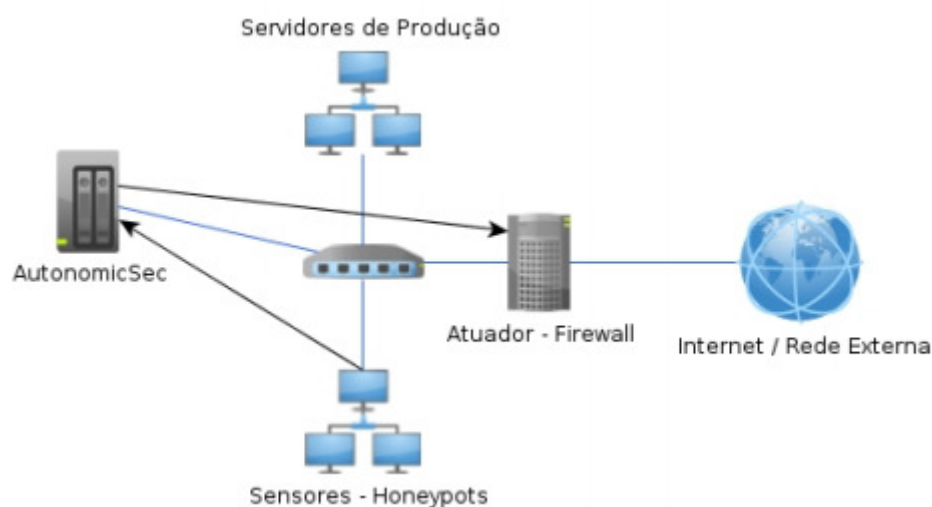


Figura 4.4: Topologia do Primeiro Ciclo Autônomo[31]

A exemplo de uso da estratégia autônoma baseada em intenções, um usuário mal intencionado está objetivando realizar um ataque ao servidor Web da rede. Ele interage primeiramente com o *honeypot* que emula o respectivo serviço de porta 80, utilizada normalmente por servidores Web. Com isso, será gerada uma regra a ser aplicada pelo executor, através do atuador, na tabela de encaminhamento em uma aplicação de *firewall*. Esta regra modifica as permissões de acesso do endereço do intruso, de maneira que ele seja rejeitado em qualquer tentativa de acesso (ou ataque) a todos serviços Web de porta 80 que possam existir na rede de produção. Portanto, nossa estratégia é aplicar regras para controle de tráfego entre redes distintas, associadas a modificação das permissões de acesso de usuários externos a serviços da rede interna, como pode ser visto na Figura 4.5. No caso de interação com um honeypot com o uso do protocolo ICMP, utilizado pelo *ping*, o intruso tem acesso bloqueado nesse protocolo a todos servidores de

produção da rede, não levando-se em consideração a porta. Nós chamamos o conjunto de endereços bloqueados dessa forma de **lista cinza**.

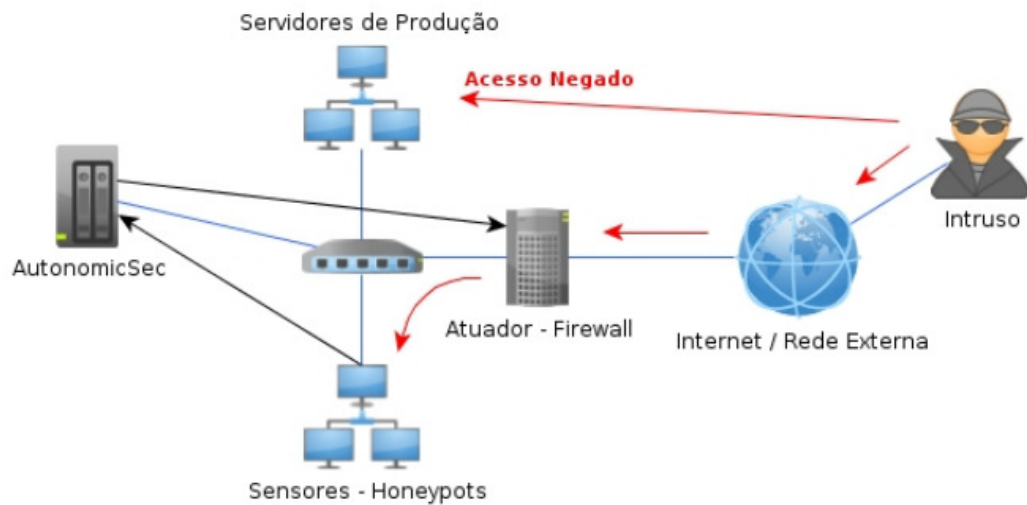


Figura 4.5: Lista Cinza[31]

Uma outra tomada de decisão feita por nossa estratégia autônoma baseada em intenções é a verificação da ocorrência de endereços, a fim de inspecionar a existência de uma quantidade elevada de regras geradas por uma determinada origem. Se houver ocorrências acima de um valor especificado na configuração deste ciclo, o endereço não terá permissão negada apenas para determinados serviços contidos em regras geradas por ele, mas a toda rede. Nesse caso consideramos que sua intenção não é apenas atacar um serviço específico, mas sim qualquer vulnerabilidade que haja na rede. Como visto na Figura 4.6 por exemplo, o intruso tem acesso negado a toda a rede se interagir com três serviços distintos de honeypots, passando a integrar o que chamamos de lista negra. Dessa forma, é feita uma otimização nas regras do firewall, excluindo todas regras de bloqueio a serviços específicos geradas anteriormente e criando apenas uma de bloqueio total a rede.

A estratégia autônoma necessita de algumas informações para funcionar corretamente, sendo uma delas a de identificação dos servidores de produção da rede, na qual deve conter o endereço, protocolo (TCP, UDP ou ICMP) e porta utilizada pelo serviço. Uma outra informação que a estratégia deve conhecer são os endereços considerados sempre legítimos na rede, nos quais fazem parte do que chamamos de **lista branca**. Nela são informados endereços que não geram regras de *firewall*, pois são considerados confiáveis, mesmo que interajam com um *honeypot* e gerem log no componente sensor.

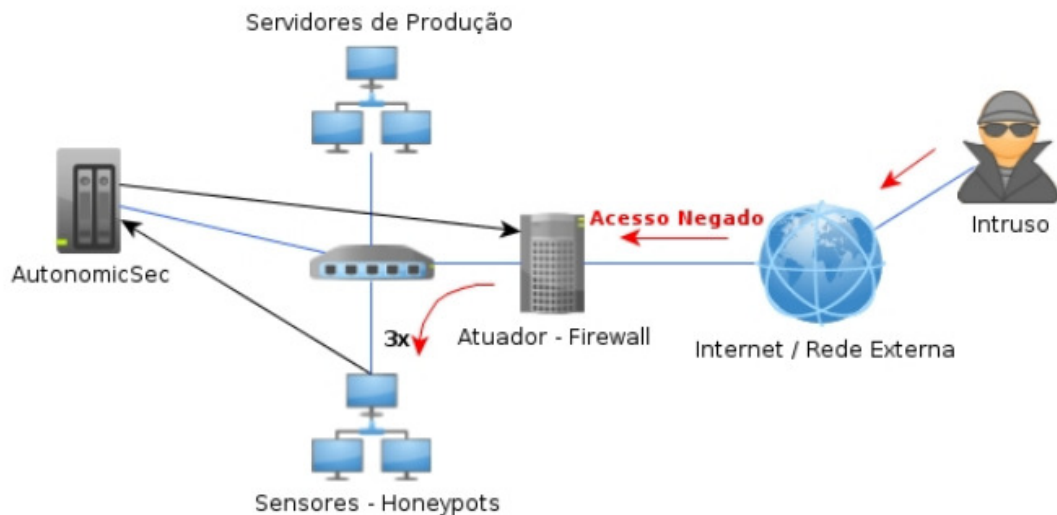


Figura 4.6: Lista Negra[31]

Ao final deste ciclo obtemos as seguintes propriedades autônomicas: autoconfiguração, pois este ciclo do *AutonomicSec* reconfigura dinamicamente as regras de controle de tráfego da aplicação de *firewall*; autoaprendizagem para o próprio *firewall*, porque as regras são implementadas nele a fim de que possa adquirir conhecimento sobre novos usuários mal intencionados e assim bloqueá-los, de acordo com sua intenção de ataque; auto-otimização para as próprias regras, visto que no caso da inserção de um endereço na lista negra, há uma redução na sua quantidade, retirando as que rejeitam para serviços específicos e adicionando apenas a que bloqueia para toda rede interna; por fim, autoproteção, sendo esta a propriedade implícita a ser atingida pela rede e objetivo deste primeiro ciclo autônomo.

4.3 Segundo Ciclo Autônomo

O segundo ciclo autônomo também é proposto utilizando o *framework*, como uma extensão. Ele é responsável por manipular dinamicamente honeypots virtuais comprometidos com base em logs de *firewall*. Como pode ser visto na Figura 4.7, o sensor é representado pelo firewall e o atuador pelo hipervisor dos *honeypots* virtuais. O uso de *honeypots* combinado com a tecnologia de virtualização em segurança de redes vem sendo explorado a fim de obter uma melhor utilização dos recursos oferecidos [19]. A implantação de uma *honeynet* implica em alocar vários computadores, com sistemas operacionais e serviços distintos, o que pode significar um alto custo de implantação e

operação. Nesse contexto, máquinas virtuais podem ser usadas para construir *honeynets* virtuais. Para esta situação a virtualização traz vantagens, como o menor custo de implantação e gerência.

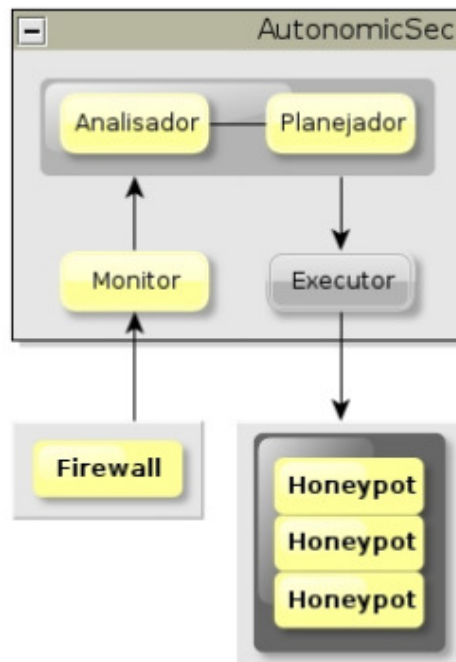


Figura 4.7: Arquitetura do Segundo Ciclo Autônomo[31]

Um honeypot é considerado comprometido quando o intruso que interage com ele descobre que está sendo enganado, ganha domínio e passa a utilizá-lo como intermediador para a realização de outros ataques [29] [20] [2]. Normalmente *honeypots* comprometidos, quando dominados (invadidos), são usados como nós integrantes de uma *botnet* e utilizados para prejudicar outros sistemas. Dessa forma, o problema de *honeypots* comprometidos é um risco que diminui o potencial da ideia de metodologias de decepção para a área de segurança, no qual merece atenção e deve ser tratado.

Neste ciclo, o sensoriamento são em logs gerados por uma aplicação de *firewall* para verificar a ocorrência de tráfego de saída originados a partir de *honeypots*. O *firewall* deverá está configurado para registrar no log tentativas de início de conexão que utilizam os protocolos TCP, UDP e ICMP que excedam uma determinada quantidade. Ou seja, o *honeypot* só poderá receber conexões UDP e ICMP, como também responder conexões TCP que tenham sido iniciadas a partir de um usuário externo. A configuração inserida no *firewall* é responsável por determinar a quantidade limite de conexões que podem ser iniciadas a partir do *honeypot* em um intervalo de tempo, por exemplo, dez conexões por hora. Dessa maneira é possível identificar uma máquina que está sendo utilizada para

intermediar a realização de ataques, pois um *honeypot* não deve ser instalado e mantido de forma que inicie interações (conexões). Portanto, se um *honeypot* iniciar conexões acima de um limite especificado, ele é considerado comprometido. Esse mecanismo de identificação de honeypots comprometidos já é implementado no *Honeywall* e *sessionlimit*, então tomamos como base na geração dos logs para os sensores deste ciclo.

Os sensores enviam para o *AutonomicSec* os logs gerados pelo *firewall*, nas ocorrências explicadas acima, que as repassam ao monitor responsável. Este, por sua vez, extrai o endereço do nó da rede que gerou a ocorrência e o envia para o componente de análise e planejamento. Portanto a filtragem realizada pela fase de monitoramento neste ciclo é simples, havendo a necessidade de extrair apenas o campo de endereçamento. Dessa maneira é possível identificar um *honeypot* comprometido, se o log tiver sido gerado pelo endereço de um e isso é feito pela estratégia autônoma deste ciclo.

Então na próxima etapa, a de análise e planejamento, a estratégia autônoma verifica se o endereço recebido pertence a alguma máquina virtual da *honeynet*. Caso isso ocorra, a ação a ser executada através do atuador, que neste ciclo é representado pelo hipervisor dos honeypots virtuais, como visto na Figura 4.8, é a substituição dos *honeypots* comprometidos. Essa substituição é dinâmica e representa o desligamento da máquina virtual e inicialização de outra idêntica com as mesmas características, porém, sem estar sob domínio do intruso. Consideramos que qualquer código malicioso instalado inicialmente pelo usuário mal intencionado em um *honeypot* para ser utilizado como ponte para a realização de outros ataques não terá efeito e, assim, não será mais dominado, visto que a máquina virtual de decepção será desativada junto com qualquer atividade maliciosa realizada anteriormente. Portanto, o plano de ações neste ciclo é a realocação dinâmica de *honeypots*, substituindo os comprometidos por suas réplicas não infectadas.

Para o correto funcionamento deste ciclo, informações da *honeynet* necessitam ser conhecidas, a saber: endereço da máquina do hipervisor responsável pelo gerenciamento da virtualização (pode haver mais de um), endereços dos *honeypots* para a identificação dos mesmos (quando gerarem log de tráfego de saída no *firewall*) e localização das imagens de cada máquina virtual que representa a réplica não infectada por códigos maliciosos. Esta última é criada inicialmente antes do *honeypot* ser colocado em atividade, não havendo a possibilidade de está comprometida.

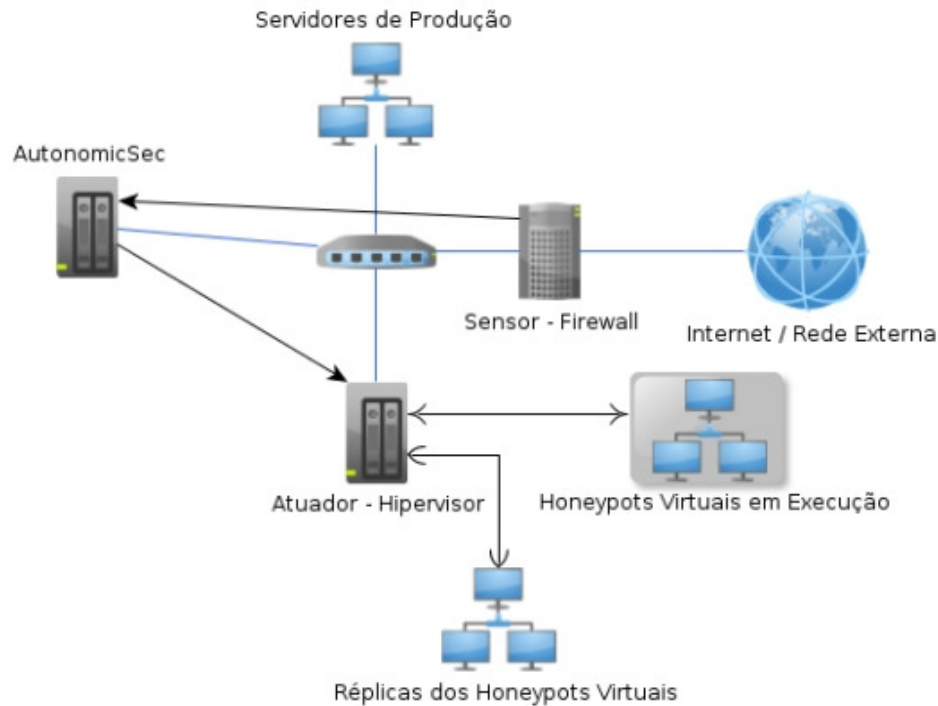


Figura 4.8: Topologia do Segundo Ciclo Autônomo[31]

Ao final deste ciclo obtemos as seguintes propriedades autônomicas: autoconfiguração, pois este ciclo do *AutonomicSec* reconfigura dinamicamente a *honeynet*, através da manipulação dos *honeypots* virtuais; autoproteção para a rede como um todo, partindo do princípio que metodologias de decepção não serão utilizadas para a realização de ataques à rede externa e também interna; por último, autocura para os próprios *honeypots*, desabilitando o infectado e inserindo um que não esteja, sendo este o processo de cura. Contudo, consideramos que um mecanismo de decepção deve ser, na verdade, utilizado para a proteção de redes na fase de detecção de intrusão e não um meio para a realização de ataques, sendo este segundo ciclo focado em conseguir uma solução que garanta isso.

4.4 Discussão

Para introduzir um novo ciclo é necessário estender os três principais componentes do *AutonomicSec*: monitor, analisador/planejador e o plano de ação, que é o código a ser processado pelo executor. Como também devem ser implementados os componentes sensor e atuador. O sensor deve ser criado como um cliente socket coletando dados e os enviando ao monitor. O atuador, também externo ao *framework*, poderá ser

qualquer aplicação responsável pela reconfiguração do elemento gerenciado. O monitor deve ser implementado de maneira a filtrar o que é relevante para a tomada de decisão, passando dados úteis para o componente de análise e planejamento. Este último, por sua vez, possuirá o algoritmo da estratégia autonômica. E o plano de ações, terá o código processado pelo executor.

Ressaltamos que nosso *framework* é criado com o objetivo de ser aplicado em redes, para cooperação entre sistemas de segurança, uma vez que foi pensado com intuito de resolver problemas nesses ambientes. Isso não significa que ele não pode ser estendido ou adaptado para problemas em outras áreas. Com o reuso dele o primeiro ciclo autonômico foi implementado, tendo como intenção a melhor utilização dos logs gerados por *honeypots*. Ele foi inspirado em CA e objetiva alcançar a característica de autoaprendizagem, utilizando regras ECA, através da geração automática de novas regras de *firewall* e possível identificação de ataques de último dia. O segundo ciclo autonômico foi desenvolvido, também com o reuso do *framework*, para contornar o problema de *honeypots* comprometidos, através da manipulação dinâmica de máquinas virtuais. Ele possui uma abordagem ativa para o problema em questão, utilizando-se para isso a substituição dos *honeypots* virtuais comprometidos.

Reiteramos ainda que esses dois ciclos autonômicos propõem soluções para problemas particulares na área de metodologias de decepção e que as propriedades básicas de CA são alcançadas por ele: autoconfiguração, autocura, autoaprendizagem, auto-otimização e autoproteção. Além disto, este trabalho (*framework* e ciclos autonômicos) mostra que é possível obter: integração e cooperação entre sistemas de segurança, que foram inicialmente desenvolvidos para trabalharem isoladamente; inteligência, através da implantação de estratégias autonômicas que dinamizam o processo de proteção; e autonomia, para alcançar auto-segurança na rede de maneira que os sistemas consigam se auto-gerenciar.

4.5 Conclusão

Este capítulo apresentou o AutonomicSec, o nosso mecanismo autonômico para segurança de redes baseado em decepção. Em seguida foi detalhada toda a ideia de nosso trabalho, através da explicação da arquitetura, topologia e seu funcionamento.

Para isso, dividimos o mecanismo nos três tópicos: framework autonômico, primeiro ciclo autonômico e segundo ciclo autonômico, sendo que os dois ciclos autonômicos são uma extensão do framework. Além disso, foi possível observar o principal contexto em que o AutonomicSec é aplicável. E ressaltamos que o framework permite, através de seu reuso, ser estendido para criar outros ciclos autonômicos.

5 Avaliação do Mecanismo Autônomo

Este capítulo descreve a avaliação do *framework*. Para isso, desenvolvemos um protótipo do mecanismo autônomo. através da implementação do *framework* e sua extensão para os dois ciclos autônômicos. Posteriormente, realizamos as avaliações de desempenho nos ciclos em cenário comum de uso e, em seguida, mostramos os resultados obtidos sobre as três métricas de avaliação previamente selecionadas. Ao explicar os resultados alcançados, mostramos o que resultou da avaliação de desempenho do mecanismo autônomo.

5.1 Implementação do *Framework* Autônomo

A fim de tornar a ideia do *framework* autônomo utilizável, para ser estendido em aplicações de segurança, o *AutonomicSec* foi implementado utilizando tecnologia Java[18] e o Ambiente Integrado de Desenvolvimento - *Integrated Development Environment* (IDE) - Eclipse[8]. A Figura 5.1 mostra o diagrama de componentes, na qual é possível observar como os componentes são distribuídos (sistema descentralizado).

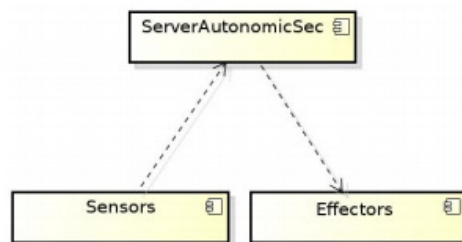


Figura 5.1: Diagrama de componentes do *Framework*[31]

Os componentes de sensoriamento (*sensors*) devem ser implementados junto a sistemas de segurança, para serem utilizados na coleta de dados em uma extensão. Da mesma maneira que os componentes atuadores (*effectors*) precisam ser implementados junto a outros sistemas de segurança, sendo que estes, no entanto, sofrerão reconfigurações. Como explicado anteriormente na Seção 4.1, ressaltamos que ao implementar cada sensor é necessário passar para o *AutonomicSec* dois campos: o próprio dado coletado e o identificador do monitor que tem a capacidade de tratar esses dados. O componente

ServerAutonomicSec é implementado em um servidor, como o próprio nome sugere, de maneira a ficar responsável pela execução do restante do ciclo.

A nossa implementação preza por simplicidade, com o objetivo de facilitar a utilização de CA, através do uso do modelo MAPE-K. Com isso, o *framework* impõe uma arquitetura a ser seguida, de maneira que em todas as extensões, ou ciclos, cada componente execute funcionalidades que sejam de sua responsabilidade. Com isso, o diagrama de classe pode ser visto na Figura 5.2.

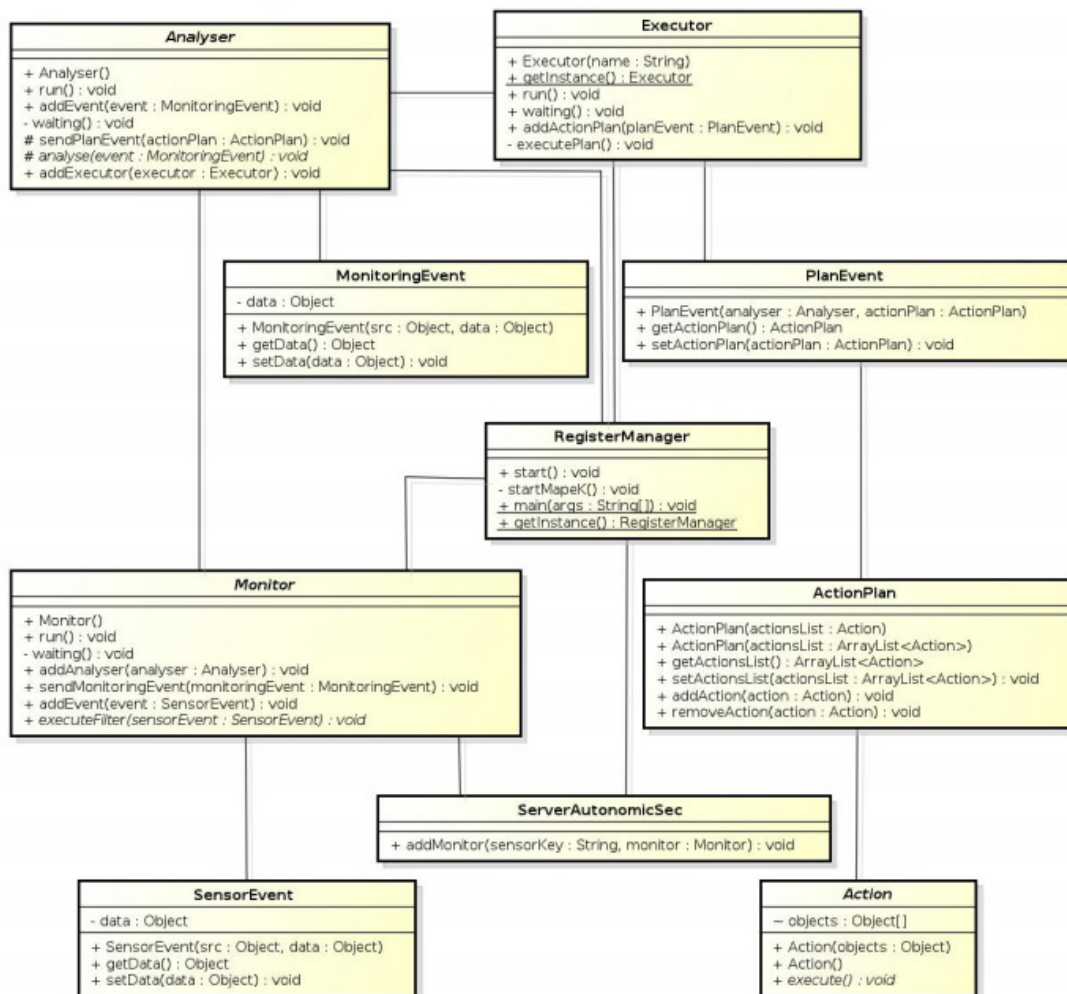


Figura 5.2: Diagrama de classe do Framework[31]

A classe `RegisterManager` inicializa o framework, juntamente com todos os processos, representados em Java por threads, que a compõem, através do método `startMapeK()`, são eles: `ServerAutonomicSec`, `Monitor`, `Analyser` e `Executor`. A funcionalidade desta classe é a criação do registro, ou associação, entre componentes, onde monitores são registrados no servidor, analisadores/planejadores nos monitores, que por sua vez são associados ao executor, de maneira que cada componente saiba para onde

enviar o resultado de seu processamento, criando assim a ideia de ciclo. Além disso, esta classe implementa o padrão de projeto *Singleton*[9], através do método `getInstance()`, garantindo que apenas um objeto possa ser instanciado a partir dessa classe.

A classe `ServerAutonomicSec` é responsável pela criação do servidor socket, onde sensores irão conectar para o envio de dados sensoreados a monitores. Dessa maneira, através do método `addMonitor(String sensorKey, Monitor monitor)`, devem ser adicionados todos os monitores da aplicação. Além disso, essa classe implementa a estrutura de decisão para a escolha do monitor correto que irá tratar os dados recebidos dos sensores. Essa tomada de decisão é feita de acordo com o parâmetro `sensorKey`, utilizado ao instanciar monitores e adicioná-los no servidor, como também passado por sensores para serem associados a esses últimos.

A filtragem nos dados recebidos por sensores é de responsabilidade da classe `Monitor`, realizada pelo método `executeFilter(SensorEvent sensorEvent)`. Ela é instanciada como processo (herda de `Thread`), onde eventos sensoriados são depositados em uma fila por sensores, com o uso do método `addEvent(SensorEvent event)`. Quando esta fila está vazia, o processo entra em estado de espera (`waiting()`), sendo notificado para entrar em execução quando é inserido algum evento nessa fila. Após os filtros serem realizados, as informações úteis serão enviadas para todos analisadores registrados em um monitor. O envio é feito pelo método `sendMonitoringEvent(MonitoringEvent monitoringEvent)` e o registro pelo `addAnalyser(Analyser analyser)`.

Na classe `Analyser`, instanciada também como processo, o método `analyse` é o responsável por executar a estratégia autônômica utilizada pelo ciclo, o que normalmente necessita acessar dados em arquivos de configuração e/ou uma base de conhecimento (`knowledge`) para tomada de decisões. O executor é conhecido por esta classe através do método `addExecutor(Executor executor)`, no qual determina o fluxo do ciclo. A manipulação da fila contendo dados filtrados pelo monitor é realizada por dois métodos: `addEvent(MonitoringEvent event)`, para a inserção, e pelo próprio `analyse`, para remoção, quando for processar a estratégia autônômica. Caso a fila esteja vazia, a `thread` irá esperar, com a chamada do método `waiting()`.

Ainda na classe `Analyser`, o método `sendPlanEvent(ActionPlan actionPlan)` envia eventos para o executor. Um evento é composto por uma ou mais ações (`Action`), resultado do processamento realizado pela estratégia autônômica, e

que formam um plano de ações. Esse último é manipulado através da classe `ActionPlan`, na qual o analisador pode adicionar ou remover ações em uma lista.

Por fim, a classe `Executor`, que também herda de `Thread`, realiza a execução de todas as ações enviadas a ela. Em um ambiente de redes, o componente executor normalmente aplica ações de reconfiguração nos sistemas através da passagem de parâmetros para scripts. Esses scripts representam os atuadores e são usados para modificar, inserir ou remover configurações em sistemas de segurança de redes. No caso dos atuadores do `AutonomicSec`, os parâmetros são passados a eles remotamente, através do protocolo `Secure Shell (SSH)`[30] e, como os sistemas de segurança executam em sistema operacional Linux, os scripts são, na verdade, escritos em `Shell Script Bash`[3].

5.2 Avaliação de Desempenho

Mais recentemente, a comunidade de CA tem discutido o que realmente pode ser considerado auto-gerenciamento, uma vez que alguns serviços isolados propiciam momentos em que um sistema pode gerenciar a si próprio. Serviços como esses podem ser encontrados, por exemplo, em um módulo de otimização de consulta de um sistema operacional ou o software de roteamento de uma rede. No entanto, o termo CA é aplicado para designar sistemas em que mudanças de políticas ocorrem de forma em tempo de execução. Ou seja, mudanças devem ocorrer dinamicamente. Além disso, para distinguir sistemas autônomos de sistemas puramente adaptativos (como por exemplo, alguns sistemas multimídias que se adaptam a flutuações da largura de banda disponível), os primeiros devem exibir mais de uma característica de gerenciamento [11].

O objetivo deste trabalho é avaliação de desempenho do `AutonomicSec` através de seu grau de autônomo, avaliar o sistema através de medições, análise quantitativa, e métricas de desempenho definidas por McCann[22] por meio de três previamente escolhida, que são: o Grau de Autonomia, o Tempo para se adaptar e Tempo de Reação e por último a Sensibilidade. Existe uma relação com as métricas de avaliação de desempenho gerais[1][32], essas métricas avaliam tanto a parte física, como o hardware, quanto a parte lógica, como o software. Diante dos resultados obtidos, é que chegaremos ao nível de autonomia encontrado no Mecanismo Autônomo baseado em metodologia de decepção.

5.2.1 As Métricas de Avaliação de Desempenho

As métricas de avaliação de desempenho servem para mostrar a quantidade e qualidade dos serviços prestados por uma entidade, objetivando a maximização da eficiência, maximização da utilização de um recurso, minimização do custo, minimização do tempo de resposta [1]. Além disso, uma boa avaliação de desempenho, pode-se encontrar possíveis problemas que causariam grandes transtornos.

As técnicas de avaliação de desempenho de sistemas obtêm suas informações através do próprio sistema ou através de um modelo representativo do sistema. Quando as informações são capturados do próprio sistema, ou seja, a captura das informações são baseadas em medições feitas sobre o sistema real ou sobre o protótipo do mesmo, elas passam por um algum processo de avaliação, entre elas, citadas em [1] estão:

- **Protótipos:** A prototipação é feita simplificação de um sistema ou artefato, mas preservando a funcionalidade do mesmo, possui custo elevado, devido os recurso serem construído para serem utilizados.
- **Benchmarks:** O benchmark são programas usados para testar o desempenho de um software, hardware ou sistema computacional, são utilizados para efeito de comparação, além disso, o próprio benchmark não pode influenciar nos resultados obtidos.
- **Coleta de Dados:** Na coleta de dados a análise é feita a partir de dados reais do sistema, geralmente fica difícil de ser executada devido ao alto custo envolvido na operação.

Além das informações serem obtidas através do próprio sistema, podem ser requerida através de um modelo representativo do sistema. Isso acontece por meio de Modelos Analíticos e por Simulações.

- **Modelagem Analítica:** A modelagem analítica ela utiliza teoria de filas, petri e statecharts, pode ser aplicada a sistemas ainda em fase de projeto, requer boa formação matemática, muitas simplificações do modelo podem ser necessárias.
- **Simulação:** Simula-se o comportamento de um sistema real a partir de um modelo do mesmo, em geral, é possível construir um modelo mais próximo da realidade

do que quando se usa a modelagem analítica, os resultados são exatos, emprega técnicas flexíveis, de baixo custo e ampla aceitação.

A escolha das métricas mais adequadas seguiram critérios como eliminar redundâncias de tempo de espera da retorno, que no caso seria o tempo de resposta de um possível teste com a métrica, devem ser realísticas e mensuráveis, ou seja, devem ser capazes de realizá-la, o resultado deve ser alcançável e medido. A figura 5.3 mostra a tabela com os critérios para selecionar a técnica de avaliação.

A técnica escolhida para realizar a avaliação do *AutonomicSec* foi a **Prototipação**, pelo fato de ser viável de realizá-la, de alcançar seus resultados, além disso, o poder de convencimento é a mais alta entre todas, realizaremos a avaliação utilizando um protótipo do *AutonomicSec*, uma versão simplificada do sistema original, mas preservando suas funcionalidades originais.

	Modelagem Analítica	Simulação	Aferição
1. Estágio	Qualquer	Qualquer	Após prototipação
2. Tempo	Pouco	Médio	Variado
3. Ferramentas	Analistas	Linguagens de programação	Instrumentação
4. Precisão	Pouca	Moderada	Variada
5. Comparações	Fáceis	Moderadas	Difíceis
6. Custo	Baixo	Médio	Alto
7. Poder de convencimento	Baixo	Médio	Alto

Figura 5.3: Critérios para seleção da Técnica de Avaliação[32]

A representação abaixo mostra como esta organizada as técnicas de avaliação de desempenho citadas anteriormente, figura 5.4.

Depois que já tem escolhido a técnica de avaliação de desempenho que iremos utilizar, temos agora a escolha das métricas que farão a avaliação do sistema. As métricas são mostrados na figura 5.5, a última coluna mostra como chegar

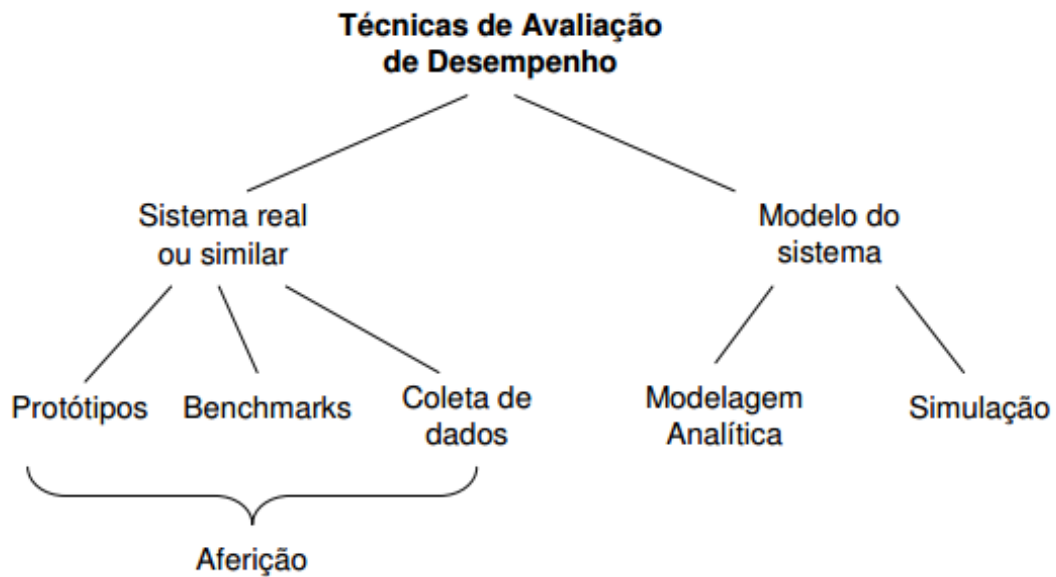


Figura 5.4: Taxonomia das técnicas de avaliação de desempenho[32]

as métricas. Existem várias métricas, as mais usadas são: Tempo de respostas, Taxa de processamento (*Throughput*), Utilização, Escalabilidade, Disponibilidade (falhas ou sobre-cargas), Confiabilidade, Custo, Segurança (confidencialidade, integridade,...), Extensibilidade.

Dentre as métricas mostradas, uma possuem semelhança com a métrica de McCann[22] citadas no capítulo 1, o “Tempo de resposta” relaciona-se com a “Tempo para se adaptar e Tempo de reação”.

Avaliação do *AutonomicSec*

Observações relevantes sobre a avaliação:

- 2 Fatores:
 - Número de testes para a métrica Tempo para se adaptação e tempo de reação: 1 a 10. Foram realizadas dez interações, onde cada interação compreende o tempo que o sistema leva para reagir a um ataque.
 - Faixa de pacotes para avaliar a métrica Sensibilidade: 5, 10, 15 e 20. Foram escolhidas uma faixa de valores que vai definir a quantidade de trocas de máquina, ou melhor, rollback que ocorrerá a cada valor definido na faixa, por exemplo, a quantidade de rollback que ocorrerá quando o número de pacotes for definido em 5, depois 10, depois 15 e por último 20.

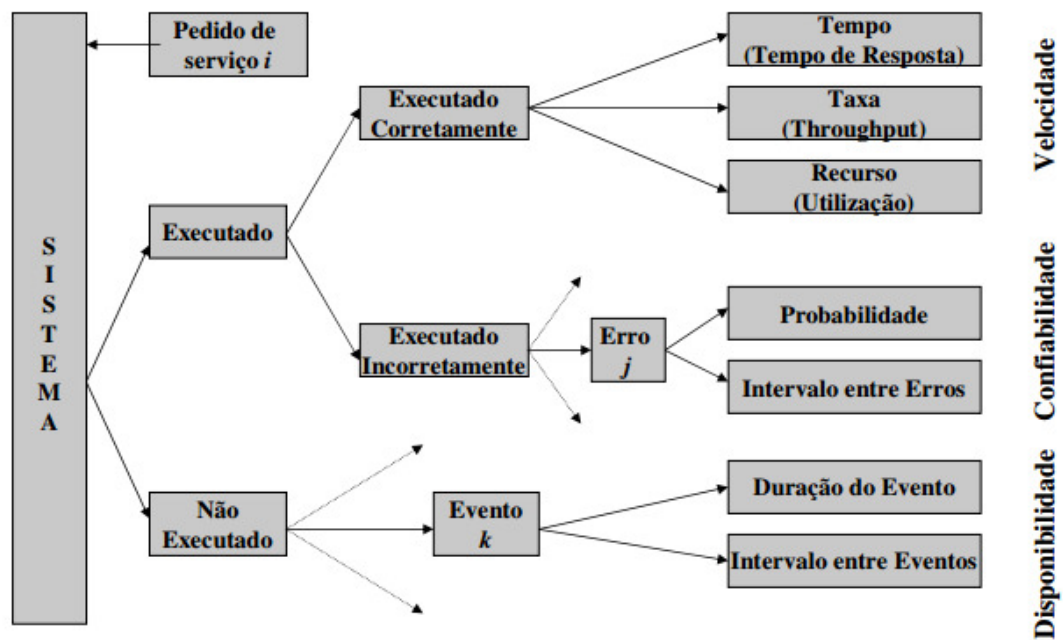


Figura 5.5: Métricas de Desempenho[32]

- Configuração das máquinas:
 - *Firewall*: Pentium 4, 1GB de RAM.
 - *AutonomicSec*: Core i3, 4GB de RAM.
 - *Servidor de produção*: Pentium 4, 4GB de RAM.

- Procedimentos Utilizado:
 - Configuração da rede.
 - Configuração do sistema Autônômico - *AutonomicSec*.
 - Configuração do *Firewall*.
 - Configuração do *Honeyd*.
 - Programa que realiza contagem do tempo em milissegundos.

- 1° ciclo Autônômico:

Tem-se k interação direcionados para os servidores de produção.

Variável de saída - Tempo para se adaptar a uma mudança no ambiente.

Para 1 a 10 interação.

- 2 ° ciclo Autônômico: Tem-se k interação direcionados para os servidores de produção, para cada interação, o tempo de 1hora é calculado.

Variável de saída - número de ocorrências na troca de máquinas comprometidas por cada interação com um dos servidores de produção.

Para 5, 10, 15, e 20 testes com pacotes.

A área de interesse para essa avaliação, pode ser visto na figura 5.6, quando o mecanismo autônomo é inicializado, ele já começa no estado normal, quando sofre uma tentativa de invasão, passa para o estado comprometido, no primeiro ciclo, o atuador gera regras de *iptables* onde será implantando no *firewall*, já no segundo ciclo, o *AutonomicSec* identifica a máquina comprometida, e efetua a troca da máquina por uma imagem que é sua réplica. O tempo final vai ser $tf = t1-t0$, que é avaliado na métrica Tempo para se adaptar e tempo de reação, e esse tempo vai está variando de acordo com o número de pacotes limitados pelas regras de *iptables* na segunda métrica, a Sensibilidade.

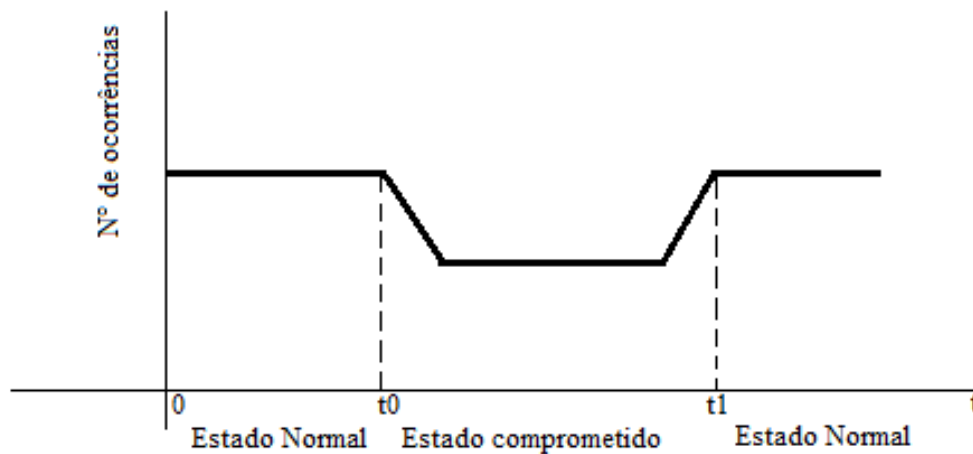


Figura 5.6: Estados alcançado pelo *AutonomicSec*[11]

1º Métrica: Grau de Autonomia

O Grau de Autonomia está relacionada com a previsibilidade que o sistema possui frente a situações que exijam respostas imediatas, nessa métrica, o autor Huebscher [11] criador dos quatro “nã níveis” ou melhor, faixas de classificação, meramente classifica os sistemas autônomo, em relação ao seu grau de autonomia, e com base nessa classificação que levarei para o *AutonomicSec*.

A IBM propôs em Computação Autônomo um conjunto de níveis, modelo que se estende desde o nível 1: Básico, para o nível 5: Autônomo. Resumidamente, nível

1 define pelo qual os elementos do sistema são geridos por pessoal altamente qualificado que utilizam ferramentas de monitoramento e, em seguida, fazem as mudanças de forma manual. IBM acredita que esse nível é o lugar onde a maioria dos sistemas de TI estão hoje. Nível 2 é conhecido como Gerenciado(Managed). Este nível mostra ferramentas de monitoramento de informações do sistema de uma forma inteligente o suficiente para reduzir a carga administrativa do sistemas. Nível 3 é intitulado preditiva em que monitoramento mais inteligente que o nível 2 é realizado a reconhecer o comportamento do sistema em padrões e sugerir ações aprovadas e realizadas por uma equipe de TI. O nível 4 é chamado de Adaptivo, o sistema usa os tipos de ferramentas disponíveis para o Nível 3 do sistema de pessoal, mas é mais capaz de agir. Interação humano é minimizado e espera-se que o desempenho esteja preparado para atender os acordos de nível de serviço (SLAs). Finalmente, o nível autônomo, é o nível completo, é o nível 5, em que os sistemas e os componentes são gerenciadas de forma dinâmica por regras e políticas de negócios, liberando assim, a equipe de TI para se concentrar em manter em constante mudança necessidades de negócios, e se dedicam a outras áreas mais emergências. O problema com este modelo está como os meios para definir computação autônoma onde ele é muito estreito (com foco em um determinado tipo de sistemas tradicionais de apoio às empresas) e a diferença entre os níveis é vago. Por isso propomos uma alternativa que acreditamos melhor representar os níveis de auto-gestão que descobrimos.

Foram definidas quatro elementos Autônomicos (e não níveis) que traz para o foco da pesquisa, em que grau, e em que arquitetura, que o foco tem sido aplicado, no caso do *AutonomicSec*. Os elementos são listados abaixo:

- Suporte: Descreve-se isso como um trabalho que se concentra em um aspecto particular ou componente de uma arquitetura para ajudar a melhorar o desempenho do arquitetura completa usando Autônomico. Por exemplo, o foco em gestão de recursos ou de rede ferramentas de suporte ou administração que contêm propriedades de auto-gestão.
- Núcleo: Isto é onde a função de auto-gestão está dirigindo o núcleo da aplicação si. Ou seja, se o foco do aplicativo é fornecer dados multimídia através de uma rede e o trabalho descreve uma solução end-to-end, incluindo gerenciamento de rede e tela, áudio, etc, em seguida, descreve-se o auto-gestão como núcleo. No entanto, o

trabalho realizado neste âmbito não está tomando as metas baseadas humanas de alto nível (por exemplo, negócios ou SLAs) em conta.

- Autônomo: Existe ainda, quando uma solução de ponta a ponta completa é produzida, mas normalmente a solução diz respeito a inteligência mais emergente e baseado em agentes tecnológicos, onde o sistema de auto-adapta ao ambiente para superar desafios que pode produzir falha, mas não mede o seu desempenho e adaptação como ele realiza sua tarefa para melhor atender algum tipo de objetivos performativas. Muitas pesquisas na área de sistemas baseados em agentes podem entrar nesta categoria, como já neste trabalho estamos interessados apenas em aplicações de tecnologia baseada em agente que impulsionam a auto-gestão de um sistema de computação que realiza mais de uma aplicação. Por exemplo, o Pathfinder é interessante para nós, porque o "operador" deve se auto-administrar-se a realizar as tarefas do buggy em extrema terrenos, e essas tarefas são muitas e variadas, mas um sistema baseado em agentes cuja único foco é conseguir uma meta específica do aplicativo não está dirigindo o sistema em um forma de auto-gestão e, portanto, não está incluído neste trabalho.
- Autônomo: A este nível, o trabalho diz respeito a uma arquitetura completa e descreve o trabalhar em termos de que a arquitetura. No entanto, o interesse é de alto nível humano metas baseadas, por exemplo, metas de negócios ou SLAs são levados em conta. Este nível é o mesmo que o Nível 5: Nível autônomo e que envolve um sistema que reflete a seu próprio desempenho e se adapta de acordo com as mudanças que ocorrem em seu ambiente de desenvolvimento.

Temos uma figura 5.7 mostra a classificação em que colocamos maior parte do trabalho que trata da Avaliação do Grau Autônomo de um sistema. Os sistemas são classificados pelo grau de autenticidade (na horizontal) e a parte do circuito MAPE-K centrada em (verticalmente). Enquanto algumas das escolhas são altamente subjetiva, tentamos colocá-los em uma categoria que destaca a grande contribuição de um projeto.

Chegamos a conclusão que o AutonomicSec[31] está classificado como uma Arquitetura Completa no modelo MAPE-K, como sendo um sistema que apresenta características fundamentais para serem considerados Sistemas Autônomo que são: auto-configuração, auto-proteção, auto-otimização e autocura, diminuindo a intervenções externas, e o mais importante das classificações, o seu Grau de Autonomia, diante das

classificações feitas pelo McCann e Husbeshcher sobre os sistemas apresentados na figura 5.7, concluímos que o AutonomicSec está presente no nível Autônomo apresentado, devido ao fato de se adaptar de acordo com as mudanças provocadas em seu ambiente, ou seja, a medida que os ataques direcionados a rede de honeypots ocorram, independente da proporção dos ataques, o AutonomicSec mantém a segurança interna de sua rede, impedindo que a invasão consiga chegar aos servidores de produção, sendo assim, o Primeiro Ciclo reagindo de forma a bloquear o IP do atacante, para que não possa mais realizar ataques direcionados a rede de honeypots, e o Segundo Ciclo reagindo de forma a realizar a troca de máquinas virtuais(honeypots) caso forem comprometidas por um ataque.

Grau de Autonomia MAPE-K	Suporte	Núcleo	Autonomia	Autônomo
Arquitetura Completa <i>(Full architecture)</i>	Rutherford et al. 2002; [34]	Badger 2004; [35]	Kenyon 2001; [36]	Kephart and Chess 2003; [37] AutonomicSec [31]
Gerenciamento Autônomo <i>(Autonomic management)</i>	Paulson 2002; [38]	Bhatti and Knight 1999; [39]	Chakeres and Belding-Royer 2004; [40]	Todos acima contem algum elementos gerenciável
Monitoração <i>(Monitoring)</i>	Agarwala et al. 2006; [41]	Garlan et al. 2001; [42]		Todos acima contem algum elementos gerenciável
Planejamento de modelos <i>(Planning models)</i>	Zenmyo et al. 2006 [43]		Jennings 2000; [44]	
Políticas <i>(Policies)</i>	Batra et al. 2002; [45]			
Arquitetura de Modelos <i>(Architecture models)</i>	Oreizy et al. 1998; [46]	Garlan and Schmerl 2002b; [47]		Dashofy et al. 2002b; [48]
Conhecimento <i>(Knowledge)</i>	Walsh et al. 2004; [49]			Bhola et al. 2006; [50]

Figura 5.7: Categorização de trabalho autônomo descrito[31]

2º Métrica: Tempo para se adaptar e tempo de reação

Tempo para se adaptar e tempo de reação está relacionado com a sensibilidade e custo, ou seja, essa métrica se preocupa com a reconfiguração do sistema e a forma como ela vai reagir, o tempo para se adaptar leva em conta a medida do tempo que um sistema leva para se adaptar a uma mudança no ambiente, dessa forma, ela medeia entre a identificação do problema onde será necessário até que a mudança seja efetuada de forma segura levando o sistema a seu estado normal e o tempo de reação envolve parcialmente o tempo de adaptação, isto é, o tempo entre o momento que um elemento do ambiente mudou e o sistema reconhece essa mudança, decide sobre o que reconfigurar necessário para reagir à mudança no ambiente [22]. No *AutonomicSec*, será analisado os dois ciclos autônômicos. No primeiro ciclo, foi realizado 10 testes, para simular o tempo que o *AutonomicSec* leva para se adaptar e o tempo que leva para reagir a esse ataque. A figura 4.6 mostrada na seção 4.2 representa o cenário de teste, onde o atacante de fora da rede, tenta invadir uma máquina de dentro da rede usando o protocolo ICMP, utilizando o *ping*.

Testes - 1º Ciclo *AutonomicSec*

Primeiramente, o atacante de fora da rede tenta invadir uma das máquinas de produção, entretanto, propositalmente existem honeypots colocados de forma estratégica dentro da rede, levando o atacante a interagir com um dos honeypots. Com base nessa interação, o 1º ciclo é avaliado, com essa interação, o *firewall* captura o tráfego de pacotes que fica intencional nessa região, logo após, o *firewall* gera logs que alertam o *AutonomicSec*, indicando que está ocorrendo uma invasão.

O monitoramento no primeiro ciclo é baseado em tempo, visto que o componente sensor faz uma verificação por novos registros no log do honeypot a cada período de tempo estabelecido por parâmetro. No *AutonomicSec*, estabelecemos a periodicidade em cinco segundos. Os dados sensoreados são logs de honeypots de baixa interatividade emulados com a ferramenta *Honeyd*, nos quais contém interações a nível de rede com qualquer usuário que acesse ou requisição serviços. A lógica do algoritmo do Sensor pode ser vista na Figura 5.8. Os dados contendo o log e o identificador do monitor são serializados para serem enviados. Lembrando que essa lógica que é explicada aqui, é a mesma aplicada no primeiro ciclo *autonomicSec* na terceira métrica, a Sensibilidade.

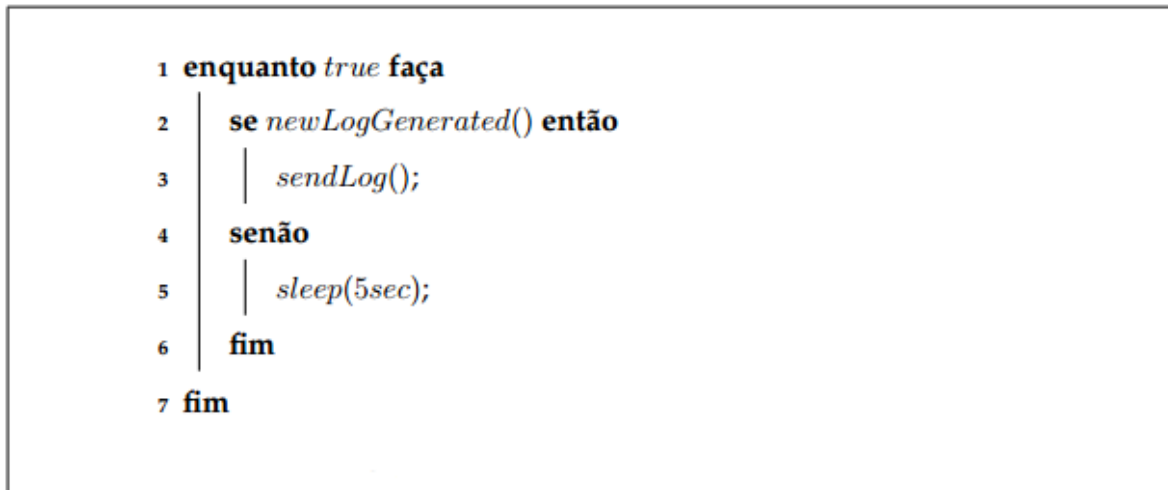


Figura 5.8: Lógica do sensor no primeiro ciclo autônomo[31]

O componente *Monitor*, ao receber os logs, executa três métodos sobre o conteúdo. O primeiro é responsável por descartar todos os campos desnecessários para a estratégia autônoma. O segundo, exclui as linhas que contém os protocolos não utilizados também pela estratégia, dentre eles o IGMP, deixando apenas as que contém TCP, UDP, e ICMP. O último método envia as informações finais para a próxima fase do ciclo como um evento (*sendMonitoringEvent(filteredLog)*), de acordo com os analisadores/planejadores registrados para recebê-los, após o processamento (*filtragem*) realizados pelos métodos anteriores.

Ao receber o evento contendo informações a serem processadas, o componente de análise e planejamento executa a estratégia autônoma baseada em intenções. Essa, por sua vez, é baseada em *ECA Rules* e a lógica da sua implementação pode ser vista na Figura 5.9.

Para cada evento recebido do monitor, as interações são separadas e armazenadas em um vetor, a fim de que possam ser analisadas individualmente. Para cada interação, é executado um método (*checkWhiteList(interaction, fileWhiteList)*) responsável por verificar se o campo do endereço IP, em sua versão quatro - *Internet Protocol version 4* (IPv4), de origem está contido na lista branca, sendo esta armazenada em um arquivo e que tem sua localização passada como parâmetro. O conteúdo desse arquivo é composto por uma lista de endereços considerados usuários legítimos da rede e, por isso, a estratégia autônoma nunca gera regras que possam bloqueá-los. Após essa verificação, caso o IP não esteja na lista branca, é executado o método que checa os serviços contidos na rede (*checkServices(interaction, fileServices)*), retornando verdadeiro

```

1 para cada event faça
2   interactions[] = separatesInteractions(event);
3   para cada interaction faça
4     se !checkWhiteList(interaction, fileWhiteList) então
5       se checkServices(interaction, fileServices) então
6         services = examineServices(interaction, fileServices);
7         rules = generatesRules(interaction, services);
8         insertActionPlan(rules);
9       fim
10    fim
11  fim
12  se actionPlan! = null então
13    sendPlanEvent(actionPlan);
14  fim
15 fim

```

Figura 5.9: Lógica da estratégia autônoma do primeiro ciclo[31]

se houver pelo menos um serviço que funcione na mesma porta do *honeypot* que sofreu a interação. Dois parâmetros são passados para esse método: a própria interação e a localização do arquivo que contém informações dos serviços. Essas informações são, na verdade, configurações dos serviços que o administrador deseja proteger, e segue um formato contendo os seguintes campos:

- **IP:** Endereço IP do servidor;
- **Protocolo:** Protocolo utilizado pelo serviços, tendo como opção: TCP, UDP ou ICMP;
- **Porta:** Porta utilizada pelo serviço, sendo uma maneira de identificá-lo;
- **Comentários:** Usado para o administrador do sistema reconhecer mais facilmente os dispositivos.

Em uma configuração que especifique a utilização do protocolo ICMP, deverá ser informado o campo porta, porém ela será descartada no momento da criação da regra de *firewall*. O protocolo ICMP pertence a camada de rede do modelo TCP/IP [17] e não se utiliza de portas [27], por isso o descarte deste campo pela estratégia autônoma. Portanto, deverá ser informada alguma porta neste campo apenas para fins de formatação do arquivo de configuração.

Depois de passar por essas condições, é executado um método *examineServices(interaction, fileServices)* que faz algo semelhante ao anterior, no entanto, ele retorna para *services* o(s) endereço(s) de todos os *serviços* de produção que estejam no arquivo de configuração *fileServices* nos quais funcionem com mesma porta e protocolo. Em seguida *services* e a própria interação são passados para a criação e/ou exclusão das regras, de acordo com as listas cinza e negra, explicadas na Seção 4.2. Elas então são inseridas em um plano de ações, para que dessa forma, várias possam ser enviadas ao executor.

Depois, é verificado se a tomada de decisão gerou alguma regra, para que o analisador/planejador não envie um evento vazio ou nulo (*null*) para o executor. Na verdade, os eventos são as próprias regras, utilizadas como parâmetros na chamada do *Shell Script*. Esse script é escrito em *Bash* e, para este ciclo, ele é o atuador, responsável por reconfigurar um *firewall iptables*. Esse atuador também cria um relatório com horário e data de regras aplicadas, para que o administrador do sistema possa visualizar o que está sendo feito. A comunicação entre o *AutonomicSec* e atuador neste ciclo é via SSH. As figuras 5.8 e 5.9 servem também de base de explicação para o primeiro ciclo com a métrica Sensibilidade.

Resultados

Os resultados que apresentamos são organizados de maneira a mostrar a configuração aplicada na rede, das três máquinas: o *AutonomicSec*, *Firewall* e *Servidor de produção/honeypots* necessárias para montar o sistema autônomo.



Figura 5.10: Cenário para coleta de resultados[31]

Nesse cenário, temos a primeira máquina que é o *firewall*, com duas interfaces, *eth0*, *eth1*, logo depois temos uma única máquina que encontra-se tanto o servidor de produção, como os servidores *honeypots*, e por fim a máquina que controla o sistema autônomo, o *AutonomicSec*. As principais configurações da rede está detalhada abaixo:

- **Firewall**
 - Interface da rede externa (**eth0**): 192.168.27.83
 - Interface da rede interna - **DMZ (eth1)**: 192.168.0.1
- **AutonomicSec**: 192.168.0.10
- **Honeynet Virtual**
 - Hipervisor: 192.168.0.50
 - Máquina Virtual 1 - Honeypot de alta interatividade: 192.168.0.201
 - Máquina Virtual 2 - Honeypot de alta interatividade: 192.168.0.202
- **Servidores de Produção**
 - Servidor Web(Apache),ICMP: 192.168.0.120
 - Servidor Web(IIS),ICMP: 192.168.0.121
 - Servidor SSH,ICMP: 192.168.0.130

Os teste foram realizados nos dias 24 de Novembro até 07 de Dezembro, utilizando a faixa de rede (192.168.27.2-91), sendo os testes realizados em laboratório, em uma rede interna ao laboratório, ou seja, uma *Intranet*, devido a isso, o tráfego do *AutonomicSec* não era comprometido pelo fluxo da rede externa ao laboratório.

Inicialmente, em um sensor deste primeiro ciclo autônomo, geram vários registros no log, nos quais selecionamos os que foram realizados os teste para serem explorados e que podem ser vistos abaixo, onde começa na linha 1 a 17. O log segue o formato da ferramenta *Honeyd*.

1° - Log *honeypot* enviado para o *AutonomicSec*

```
1 ----- honeyd log started -----
2 2013-12-07-08:49:21.2668 udp(17) - 192.168.0.1 5353 224.0.0.251 5353: 70
3 2013-12-07-08:49:23.2639 udp(17) - 192.168.0.1 5353 224.0.0.251 5353: 70
4 2013-12-07-08:50:20.7028 igmp(2) - 224.0.0.22 192.168.0.10: 40
```

5 2013-12-07-09:51:55.9232 icmp(1) - 192.168.0.10 192.168.0.120: 8(0): 84
6 2013-12-07-10:21:55.9237 icmp(1) - 192.168.27.20 192.168.0.120: 8(0): 84
7 2013-12-07-10:30:56.8741 icmp(1) - 192.168.27.91 192.168.0.121: 8(0): 84
8 2013-12-07-11:31:57.8735 icmp(1) - 192.168.27.2 192.168.0.130: 8(0): 84
9 2013-12-07-12:12:19.1486 icmp(1) - 192.168.27.5 192.168.0.120: 8(0): 84
10 2013-12-07-12:42:20.1467 icmp(1) - 192.168.27.10 192.168.0.130: 8(0): 84
11 2013-12-07-12:53:02.3228 icmp(1) - 192.168.27.4 192.168.0.120: 8(0): 84
12 2013-12-07-13:23:22.3231 icmp(1) - 192.168.27.8 192.168.0.130: 8(0): 84
13 2013-12-07-14:43:32.3275 icmp(1) - 192.168.27.15 192.168.0.130: 8(0): 84
14 2013-12-07-14:53:43.3268 icmp(1) - 192.168.27.18 192.168.0.120: 8(0): 84
15 2013-12-07-15:23:24.3254 icmp(1) - 192.168.27.31 192.168.0.130: 8(0): 84
16 2013-12-07-16:13:34.4250 tcp(6) - 192.168.0.20 1331 192.168.0.130 22
17 2013-12-07-17:23:24.4267 tcp(6) - 192.168.0.91 1550 192.168.0.121 80

2° - AutonomicSec

18 Recebe logs honeypots...

19 MonitorFW recebeu:

20 2013-12-07-08:49:21.2668 udp(17) - 192.168.0.1 5353 224.0.0.251 5353: 70

21 AnalyzerFW recebeu:

22 udp 192.168.0.1 5353 224.0.0.251 5353

23 Lendo service...

24 Lendo whitelist...

25 MonitorFW recebeu:

26 2013-12-07-08:49:23.2639 udp(17) - 192.168.0.1 5353 224.0.0.251 5353: 70

27 AnalyzerFW recebeu:

28 udp 192.168.0.1 5353 224.0.0.251 5353

29 Lendo service...

30 Lendo whitelist...

31 MonitorFW recebeu:

32 2013-12-07-09:51:55.9232 icmp(1) - 192.168.0.10 192.168.0.120: 8(0): 84
33 2013-12-07-10:21:55.9237 icmp(1) - 192.168.27.20 192.168.0.120: 8(0): 84
34 2013-12-07-10:30:56.8741 icmp(1) - 192.168.27.91 192.168.0.121: 8(0): 84
35 2013-12-07-11:31:57.8735 icmp(1) - 192.168.27.2 192.168.0.130: 8(0): 84
36 2013-12-07-12:12:19.1486 icmp(1) - 192.168.27.5 192.168.0.120: 8(0): 84
37 2013-12-07-12:42:20.1467 icmp(1) - 192.168.27.10 192.168.0.130: 8(0): 84
38 2013-12-07-12:53:02.3228 icmp(1) - 192.168.27.4 192.168.0.120: 8(0): 84
39 2013-12-07-13:23:22.3231 icmp(1) - 192.168.27.8 192.168.0.130: 8(0): 84
40 2013-12-07-14:43:32.3275 icmp(1) - 192.168.27.15 192.168.0.130: 8(0): 84
41 2013-12-07-14:53:43.3268 icmp(1) - 192.168.27.18 192.168.0.120: 8(0): 84
42 2013-12-07-15:23:24.3254 ICMP(1) - 192.168.27.31 192.168.0.130: 8(0): 84
43 2013-12-07-16:13:34.4250 tcp(6) - 192.168.0.20 1331 192.168.0.130 22
44 2013-12-07-17:23:24.4267 tcp(6) - 192.168.0.91 1550 192.168.0.121 80

45 AnalyzerFW recebeu:

46 icmp 192.168.0.10 192.168.0.120
47 icmp 192.168.27.20 192.168.0.120
48 icmp 192.168.27.91 192.168.0.121
49 icmp 192.168.27.2 192.168.0.130
50 icmp 192.168.27.5 192.168.0.120
51 icmp 192.168.27.10 192.168.0.130
52 icmp 192.168.27.4 192.168.0.120
53 icmp 192.168.27.8 192.168.0.130
54 icmp 192.168.27.15 192.168.0.130
55 icmp 192.168.27.18 192.168.0.120
56 icmp 192.168.27.31 192.168.0.130
57 tcp 192.168.27.20 192.168.0.130
58 tcp 192.168.27.91 192.168.0.121

59 Lendo service...

60 Lendo whitelist...

61 ActionFW recebeu:

```
62 iptables -A FORWARD -p tcp -i eth1 -m state --state NEW -m limit --limit 10/hour
--limit-burst 10 -j ACCEPT
63 iptables -A FORWARD -p tcp -i eth1 -m state --state NEW -m limit --limit 1/hour
--limit-burst 1 -j LOG --log-prefix "Honeypot Comprometido--log-level 7
64 iptables -A FORWARD -p tcp -i eth1 -m state --state NEW -j DROP
65 iptables -A FORWARD -p tcp -i eth1 -m state --state RELATED -j ACCEPT
66 iptables -A FORWARD -p udp -i eth1 -m state --state NEW -m limit --limit 10/hour
--limit-burst 10 -j ACCEPT
67 iptables -A FORWARD -p udp -i eth1 -m state --state NEW -m limit --limit 1/hour
--limit-burst 1 -j LOG --log-prefix "Honeypot Comprometido--log-level 7
68 iptables -A FORWARD -p udp -i eth1 -m state --state NEW -j DROP
69 iptables -A FORWARD -p icmp -i eth1 -m state --state NEW -m limit --limit 10/hour
--limit-burst 10 -j ACCEPT
70 iptables -A FORWARD -p icmp -i eth1 -m state --state NEW -m limit --limit 1/hour
--limit-burst 1 -j LOG --log-prefix "Honeypot Comprometido--log-level 7
71 iptables -A FORWARD -p icmp -i eth1 -m state --state NEW -j DROP
72 iptables -A FORWARD -i eth1 -m state --state NEW -m limit --limit 10/hour
--limit-burst 10 -j ACCEPT
73 iptables -A FORWARD -i eth1 -m state --state NEW -m limit --limit 1/hour
--limit-burst 1 -j LOG --log-prefix "Honeypot Comprometido--log-level 7
74 iptables -A FORWARD -i eth1 -m state --state NEW -j DROP
75 iptables -A FORWARD -s 192.168.27.20/24 -d 192.168.0.120/24 -p icmp -i eth1 -o eth0
-j REJECT
76 iptables -A FORWARD -s 192.168.27.91/24 -d 192.168.0.121/24 -p icmp -i eth1 -o eth0
-j REJECT
77 iptables -A FORWARD -s 192.168.27.2/24 -d 192.168.0.130/24 -p icmp -i eth1 -o eth0
-j REJECT
78 iptables -A FORWARD -s 192.168.27.5/24 -d 192.168.0.120/24 -p icmp -i eth1 -o eth0
-j REJECT
79 iptables -A FORWARD -s 192.168.27.10/24 -d 192.168.0.130/24 -p icmp -i eth1 -o eth0
-j REJECT
80 iptables -A FORWARD -s 192.168.27.4/24 -d 192.168.0.120/24 -p icmp -i eth1 -o eth0
-j REJECT
```

```
81 iptables -A FORWARD -s 192.168.27.8/24 -d 192.168.0.130/24 -p icmp -i eth1 -o eth0
-j REJECT
82 iptables -A FORWARD -s 192.168.27.15/24 -d 192.168.0.130/24 -p icmp -i eth1 -o eth0
-j REJECT
83 iptables -A FORWARD -s 192.168.27.18/24 -d 192.168.0.120/24 -p icmp -i eth1 -o eth0 -j REJECT
84 iptables -A FORWARD -s 192.168.27.31/24 -d 192.168.0.130/24 -p icmp -i eth1 -o eth0 -j REJECT
85 iptables -A FORWARD -s 192.168.27.20 -d 192.168.0.130/24 -p tcp -dport 22 -j REJECT
86 iptables -A FORWARD -s 192.168.27.91 -d 192.168.0.121/24 -p tcp -dport 80 -j REJECT
```

Nas linhas 1 a 17, temos o log gerado pelo *honeypot*, que logo depois o monitor recebe essas informações para serem processadas e enviadas para o analisador, foram registradas conexões *multicast* nas linhas 2, 3 e 4, sendo que a duas primeiras linhas, mostra conexão *Multicast Domain Name System*(mDNS), não sendo relevante para essa avaliação, e a segunda com o uso do protocolo IGMP. Nas linhas 5 a 15 são conexões ICMP e provavelmente de um ping, são dez pacotes, em direção aos *honeypots* de baixa interatividade com endereços 192.168.0.120, 192.168.121 e 192.168.0.130. As linhas 16 e 17 são tentativas de conexão a servidores SSH e Web, respectivamente.

As linhas 5 a 15 representam o primeiro evento, 16 o segundo e, por fim, 17 o terceiro evento. Pois, como o sensoriamento/monitoramento é baseado em tempo e ocorre a cada cinco segundos neste ciclo, essas foram ocorrências em tempos distintos.

O *AutonomicSec* recebe o log do *honeypot*, linha 2, ao chegar no componente do monitoramento, na linha 20, o log passa por um filtro que organiza as informações e descarta dados desnecessários. Dessa forma, quando chega na fase análise e planejamento, as informações serão usadas para uma tomada de decisão, na qual utiliza os arquivos *fileWhiteList* e *fileSevices*, vistos nas figuras 5.11 e 5.12.

Na lista branca adicionamos os endereços dos hosts da rede que consideramos confiáveis por não representarem uma ameaça. Então inicialmente colocamos os endereços do firewall, *AutonomicSec* e da máquina no qual tem o hipervisor das máquinas virtuais (linhas 1 a 3). Em seguida, os endereços das máquinas virtuais que estão os *honeypots* de alta interatividade (4 e 5) e também os endereços dos servidores de produção (6 a 7).

No arquivo de configuração, que contém as informações dos servidores de produção com seus respectivos serviços, inserimos os dados dos três que compõem nosso

```
1 192.168.0.1
2 192.168.0.10
3 192.168.0.50
4 192.168.0.201
5 192.168.0.202
6 192.168.0.120
7 192.168.0.121
8 192.168.0.130
```

Figura 5.11: Arquivo de configuração da lista branca

```
1 192.168.0.120 tcp 80 web
2 192.168.0.120 icmp 0 icmp-web
3 192.168.0.121 tcp 80 web
4 192.168.0.121 icmp 0 icmp-web
5 192.168.0.130 tcp 22 ssh
```

Figura 5.12: Arquivo de configuração dos serviços

cenário: Web (linhas 1 a 4) e SSH (5). Nas linhas 2 e 4 repetimos o ICMP para os dois servidores, pois os dois respondem a requisições desse protocolo. Essa configuração dos servidores de produção do nosso cenário pode ser representada na Figura 5.13.

O componente monitoramento continua a receber os logs, referente as linhas 5 a 17, como de rotina, ele filtra as informações relevantes e as envia para os componentes de análise e planejamento, mostradas nas linhas 46 a 58, que realiza a tomada de decisão, ou seja, encontra algo suspeito, toma uma medida, obtém os endereços que supostamente estão realizando ataques direcionados aos servidores de produção, são eles: 192.168.0.10, *.20, *.91, *.2, *.5, *.10, *.4, *.8, *.15, *.18 e *.31, com o uso do protocolo ICMP, e 192.168.27.20 e *.91 com o protocolo TCP, acessando os serviços SSH e Web, respectivamente. A ação tomada pela fase de análise e planejamento, pode ser visto nas

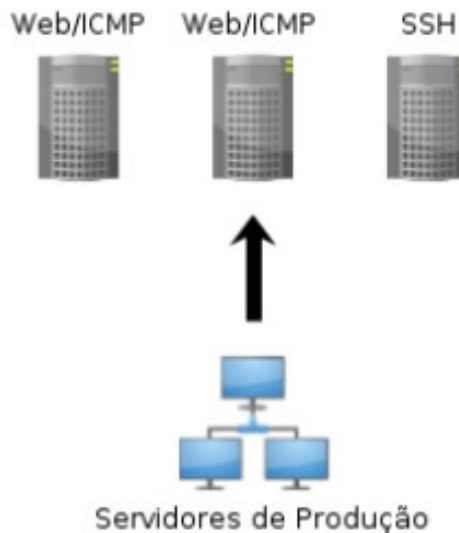


Figura 5.13: Servidores de produção do cenário[31]

linhas 62 a 86, as linhas 62 a 74 são as regras padrões carregadas no *firewall*, mostrada na figura 5.16, as linhas 75 a 86 são as regras geradas pela tentativa de invasão aos servidores de produção. A seguir os objetivos das regras e os motivos nos quais as geraram são explicadas:

- **Linha 75:** Pacotes encaminhados pelo *firewall* com origem 192.168.27.20 destino 192.168.0.120/24 com protocolo ICMP são rejeitadas. Essa regra foi gerada porque houve interação do IP 192.168.27.20 com o protocolo ICMP de algum *honeypot*;
- **Linhas 76 a 84:** Tem a mesma ocorrência da linha 75.
- **Linha 85:** Pacotes encaminhados pelo firewall com origem 192.168.27.20e destino 192.168.0.130/24 com protocolo TCP e porta de destino 22 são rejeitados. Essa regra foi gerada porque houve interação do IP 192.168.27.130 com serviço de porta 22 que utiliza protocolo TCP de algum *honeypot*;
- **Linha 86:** Pacotes encaminhados pelo firewall com origem 192.168.27.91 e destino 192.168.0.121/24 com protocolo TCP e porta de destino 80 são rejeitados. Essa regra foi gerada porque houve interação do IP 192.168.27.91 com serviço de porta 80 que utiliza protocolo TCP de algum *honeypot*;

No entanto, quando a regra da linha 86 é gerada, antes de ser realmente carregada para ser aplicada, um método verifica as ocorrências do endereço de origem em

todas regras anteriormente criadas. Caso sejam encontradas mais de uma determinada quantidade, as ocorrências anteriores são excluídas e, logo após, é criada uma nova regra que bloqueia qualquer encaminhamento de pacote originado pelo endereço que excedeu as três ocorrências. Essa quantidade é um valor passado por parâmetro e que para nosso cenário utilizamos três (3), pois consideramos que mais de três ocorrências significa que o usuário quer realmente realizar atividade maliciosa na rede. Os endereços que são bloqueados dessa maneira, para toda rede, são os que compõem a lista negra.

Neste primeiro ciclo, o tempo médio para que o *AutonomicSec* consiga se adaptar a uma mudança em seu ambiente é pequeno, isto é, se manteve até o quarto teste, quando seu tempo de adaptação elevou-se para 19 segundos, por questões adversas, uma delas é o sistema perceber que o ambiente sensoriado sofreu ataque mais não consiga aplicar as regras criadas com essa detecção, os testes podem ser vistos na figura 5.14.

Testes	TIME	SRC	DST
1	9s 5ms	192.168.27.20	192.168.0.120
2	13s	192.168.27.91	192.168.0.121
3	9s 1ms	192.168.27.2	192.168.0.130
4	19s 7ms	192.168.27.5	192.168.0.120
5	9s 4ms	192.168.27.10	192.168.0.130
6	10s 1ms	192.168.27.4	192.168.0.120
7	11s 6ms	192.168.27.8	192.168.0.130
8	10s 3ms	192.168.27.15	192.168.0.130
9	11s 5ms	192.168.27.18	192.168.0.120
10	12s 4ms	192.168.27.31	192.168.0.130

Média = 9,525 s

Figura 5.14: Resultado do 1º ciclo

Os testes no primeiro ciclo mostrados na figura 5.14, representa uma parcela pequena no tempo médio que o *AutonomicSec* leva para se adaptar a uma mudança em seu ambiente sensorado, o menor tempo 9s 1ms, e o maior 19s 7ms foram descartados afim de encontrar a média, essa estratégia de excluir o menor e o maior valor, herda de uma área bastante conhecida, em Sistemas Distribuído, é natural descartar o menor e maior valor, pois geralmente, pode haver alguma ocorrência para alcançar esse valor, seja por um distúrbio da rede, ou por uma queda no tráfego ou congestionamento do mesmo, podem acabar gerando esses valores baixo e altos, por esta questão, é necessário excluir o menor e o maior valor para que possa encontrar a média mais próxima do real funcionamento do sistema, essa técnica também será aplicada nos resultados subsequentes, os ataques partiram de um conjunto de endereços SRC, com destinos aos servidores de

produção DST, variando entre: 192.168.0.120, 121 e 130. Além disso, mostramos também uma representação gráfica desses dados onde podem ser observados na figura 5.15.

Observando a tabela 5.1 abaixo, seja o conjunto universo $U=\{1,2,3,4,5,6,7,8,9,10\}$, representando as interações, e considerando os seguintes conjuntos “Fuzzy” :

$A=\{\text{Normal}\}$, $B=\{\text{Acima do Normal}\}$ e $C=\{\text{Suspeito}\}$, para os quais atribuímos os graus de pertinência dos elementos do conjunto U na seguinte tabela, lembrando que a interação 3 e 4 foram descartas na tabela:

Interações	Normal	Acima do Normal	Suspeito
1	1	0	0
2	0	1	0.8
3	-	-	-
4	-	-	-
5	1	0	0
6	0	1	0
7	0	1	0.2
8	0	1	0
9	0	1	0.2
10	0	1	0.8

Tabela 5.1: Lógica fuzzy aplicada no 1º ciclo

Pode-se concluir:

- O tempo para se adaptar e tempo de reação é o momento em que o ambiente sensoreado (*honeypot*) sofre uma mudança, isto é, o sistema percebe uma tentativa de ataque direcionada aos servidores de produção, e realiza uma tomada de decisão, a qual tenta impedir que esse intruso consiga obter êxito em sua tarefa, com isso, é gerada regras de *iptables* que serão implantadas no *Firewall* do *AutonomicSec*, que leva pouco tempo avaliado por esta métrica, a entrar em uso.
- Nesse gráfico representado na figura 5.15, mostra o tempo alcançado em cada fase do teste, a média chegou a 9,525s, onde pode-se perceber no gráfico.
- Essa avaliação mostra em relação a média, que: em caso do tempo que o sistema leva

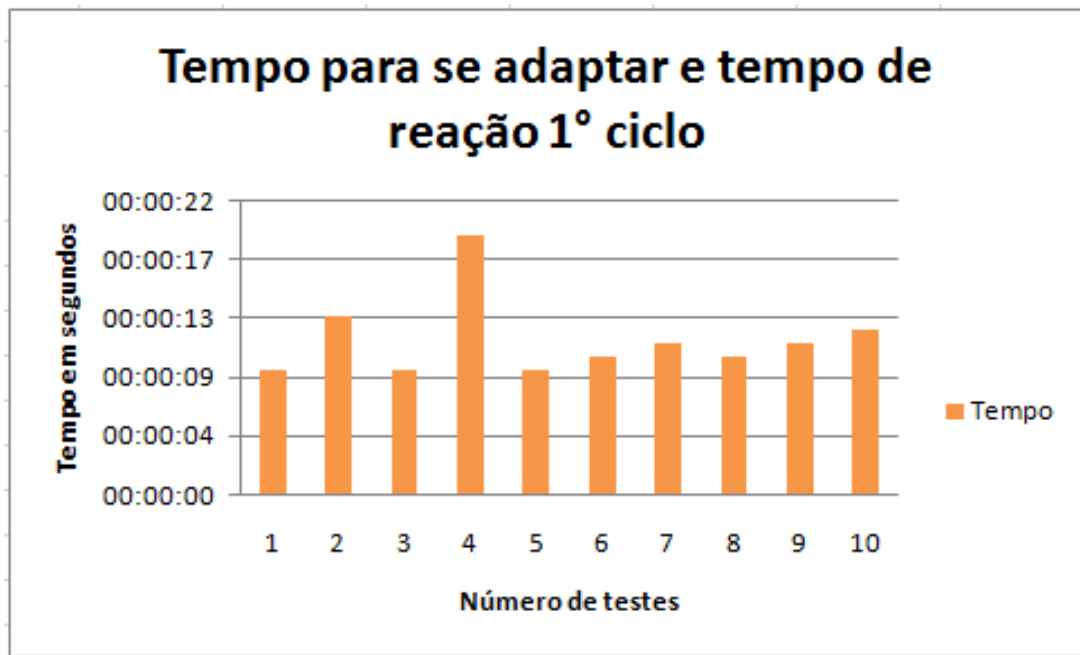


Figura 5.15: Gráfico dos testes colhidos no 1º ciclo

para se adaptar a uma tentativa de invasão exceda a média, o administrador deve ficar mais atento as ocorrências da recuperação do sistema, pois o sistema por mais que seja seguro, o atacante pode burlar as medidas de segurança.

- De acordo com a tabela fuzzy, não teve nenhuma ocorrência suspeito.

Testes - 2º Ciclo AutonomicSec

Para esse ciclo, o cenário é mostrado na figura 4.8, agora os ataques são direcionados para os *honeypots* que estão na rede, este ciclo também serializa os dados no sensor para serem enviados, tem monitoramento baseado em tempo e utiliza *ECA Rules* para tomada de decisão, como no anterior. Os dados sensoreados são logs gerados pelo *iptables*, através de regras inseridas em sua configuração, vistas na Figura 5.16.

As linhas 3 e 4 estabelecem o registro no log, quando um dispositivo da rede interna inicia uma conexão com algum outro externo. No exemplo da Figura 5.16, a cada dez novas conexões por hora (linha 3) é criada uma ocorrência no log por hora (linha 4). Essa ocorrência é considerada um alerta pelo *iptables e syslog*, devido está sendo logada em prioridade de nível sete, com o parâmetro *-log-level 7*. Para identificar uma ocorrência no log criada por uma dessas regras, basta realizar uma busca pela palavra-chave *Honeypot Comprometido*. Uma regra específica para o protocolo TCP é necessária (linha 2), para

```
1 ...
2 iptables -A FORWARD -p tcp -i eth1 -m state --state ESTABLISHED,RELATED -j
  ACCEPT
3 iptables -A FORWARD -i eth1 -m state --state NEW -m limit --limit 10/hour
  --limit-burst 10 -j ACCEPT
4 iptables -A FORWARD -i eth1 -m state --state NEW -m limit --limit 1/hour --limit-burst
  1 -j LOG --log-prefix "Honeypot Comprometido " --log-level 7
5 ...
6 iptables -A FORWARD -i eth1 -m state --state NEW -j ACCEPT
```

Figura 5.16: Regras para controle de tráfego e log no iptables

que não sejam criadas ocorrências de conexões iniciadas da rede externa, através do seu estado *ESTABLISHED,RELATED*. O estado *related* ocorre quando uma conexão é nova, mas de alguma maneira é relacionado com outra já estabelecida anteriormente e o *established* é para pacotes de uma mesma conexão já estabelecida. Essas principais regras foram baseadas nas utilizadas pelo *Honeywall* e *Sessionlimit*.

Quando as linhas chegam no monitor, ele extrai o endereço IP que gerou a ocorrência e o envia para o componente de análise e planejamento, como melhor explicado anteriormente na Seção 4.3. Nessa próxima fase é verificado se o endereço IP pertence a algum *honeypot* em um arquivo de configuração que contém informações da *honeynet* virtual. Caso seja, no mesmo arquivo de configuração são extraídos os demais campos para serem enviados como evento ao executor. A lógica do algoritmo da estratégia autônoma aplicada neste ciclo pode ser vista na figura 5.17.

Os campos contidos no arquivo de configuração (*fileHoneynet*) são listados abaixo:

- **IP do Hypervisor:** Endereço da máquina do hipervisor responsável pelo gerenciamento da virtualização;
- **IP do honeypot:** Endereço do *honeypot*, para sua identificação quando gerarem ocorrências;
- **Nome:** Possui o nome da máquina virtual no qual está funcionado o *honeypot*;

```

1 para cada event faça
2   | occurrences[] = separatesoccurrences(event);
3   para cada occurrence faça
4     | se checkHoneypots(ip, fileHoneynet) então
5       |   | fields = extractFields(ip, fileHoneynet);
6       |   | insertActionPlan(fields);
7       |   fim
8     fim
9     se actionPlan! = null então
10    |   | sendPlanEvent(actionPlan);
11    |   fim
12 fim

```

Figura 5.17: Lógica da estratégia autônômica do segundo ciclo[31]

- **Número de sequência:** Número de sequência que conta a quantidade de vezes que um determinado honeypot virtual é substituído. Também utilizado pelo algoritmo do atuador para a criação de novo nome para a máquina virtual, concatenando-o com o nome anterior, por exemplo: Ubuntu3, Ubuntu4, Ubuntu5;
- **Localização da máquina:** Localização das imagens da máquina virtual que representa a réplica não infectada por códigos maliciosos.

Para os nossos testes utilizamos honeypots de alta interatividade, sendo instalado neles sistemas operacionais reais. Os honeypots são considerados de alta interatividade por serem máquinas virtuais que oferecem serviços e recursos de sistemas reais, e não emulados. Usamos como hipervisor o *VirtualBox*[34] e a ferramenta *VBoxManage* [33] para manipular as máquinas virtuais através de *Shell Script Bash*, sendo esse último a representação dos atuadores no ciclo. A comunicação entre o *AutonomicSec* e atuador neste ciclo é via SSH(Secure Shell). As figuras 5.16 e 5.17 também servem como base de explicação para o segundo ciclo autônômico com a métrica Sensibilidade, ou seja, a mesma lógica de raciocínio é empregado na avaliação com a segunda métrica.

Resultados

Neste ciclo, são inseridas regras no *iptables* que criam logs. Com isso, foram gerados os registros no log visto na figura 5.18 os quais foram identificados pela palavra-chave *Honeypot Comrpomtido* e, logo depois, passados para o monitor responsável por tratá-los.

```
1 ...
2 Dec 7 17:47:59 firewall-desktop kernel: [ 7459.494175] Honeypot Comrpomtido
IN=eth1 OUT=eth0 SRC=192.168.0.201 DST=192.168.27.20 LEN=60
TOS=0x00 PREC=0x00 TTL=63 ID=41750 DF PROTO=ICMP SPT=60909 LEN=40
3 Dec 7 18:51:27 firewall-desktop kernel: [11267.716532] Honeypot Comrpomtido
IN=eth1 OUT=eth0 SRC=192.168.0.10 DST=224.0.0.251 LEN=58 TOS=0x00
PREC=0x00 TTL=63 ID=32374 DF PROTO=UDP SPT=35161 DPT=53 LEN=38
4 Dec 7 20:33:19 firewall-desktop kernel: [17379.672687] Honeypot Comrpomtido
IN=eth1 OUT=eth0 SRC=192.168.0.202 DST=192.168.27.20 LEN=84
TOS=0x00 PREC=0x00 TTL=63 ID=0 DF PROTO=ICMP TYPE=8 CODE=0
ID=1433 SEQ=1
5 ...
```

Figura 5.18: Log gerado no firewall

Ao chegar no monitor, o log é filtrado e extraído apenas o endereço de origem de cada registro. Então os endereços vistos na Figura 5.19 são passados para a etapa de análise e planejamento do ciclo.

```
1 192.168.0.201
2 192.168.0.10
3 192.168.0.202
```

Figura 5.19: Endereços após filtros realizado pelo monitor

A estratégia deste ciclo podemos verificar se o nó da rede que gerou o registro é um honeypot. Caso seja, campos do arquivo de configuração (Figura 5.20), usado na verificação para saber se é um honeypot, são enviados como parâmetro para o executor.

```
1 192.168.0.201 Ubuntu 0 /home/honey/Ubuntu-virgem/Ubuntu.vdi
2 192.168.0.202 Debian 2 /home/honey/Ubuntu-virgem/Debian.vdi
```

Figura 5.20: Arquivo de configuração dos Honeypots

As ações geradas pela estratégia da fase de análise e planejamento são vistas na Figura 5.21. Na primeira linha, o atuador, quando receber o plano de ações, em sequência irá: desligar a máquina virtual com nome Ubuntu, excluir dados relacionados a ela, criar uma nova a partir da réplica ou imagem `/home/honey/Ubuntu-virgem/Ubuntu.vdi` com nome `Ubuntu1` e inicializá-la. O mesmo processo é realizado na segunda linha, no entanto, será feita a substituição da máquina Debian2 pela Debian3, com imagem localizada em `/home/honey/Ubuntu-virgem/Debian.vdi`. Lembramos que esse atuador é escrito em Shell Script Bash e as tarefas realizadas por ele para a manipulação das máquinas virtuais utiliza a ferramenta `VBoxManage`.

```
1 Ubuntu 1 /home/honey/Ubuntu-virgem/Ubuntu.vdi
2 Debian 3 /home/honey/Ubuntu-virgem/Debian.vdi
```

Figura 5.21: Plano de ações criado pelo analisador/planejador

Neste segundo ciclo, o tempo médio para que o *AutonomicSec* consiga reagir a uma mudança em seu ambiente é grande, isto é, a média encontrada nessa avaliação, aproxima-se em torno de 5.066min, nesse ciclo, quanto menor o tempo de reação do *AutonomicSec* a realizar uma troca de máquina, mais rápido a operação é realizada afim de não levantar suspeita para o atacante. Os dados colhidos são mostrados na figura 5.22.

Os testes no segundo ciclo mostrados na figura 5.22, representa o tempo que foi realizado para cada troca de máquina, isto é, o tempo que leva para troca a máquina(*honeypot*) comprometida pela sua réplica virgem, frisando que mesmo que o atacante instale algo suspeito na máquina comprometida, a sua réplica não herdará nenhum arquivo, pois a máquina virgem representa o estado normal, sem infecções. Além disso, o menor tempo 3min 54s 8ms e o maior tempo 7min 3s 7ms foram descartados afim de encontrar a média, os ataques partiram de um conjunto de endereços SRC, com destinos

Testes	TIME	SRC	DST
1	3min 54s 8ms	192.168.27.20	192.168.0.201
2	7min 3s 7ms	192.168.27.91	192.168.0.201
3	6min 0s 9ms	192.168.27.2	192.168.0.201
4	4min 31s 9ms	192.168.27.5	192.168.0.201
5	5min 20s 2ms	192.168.27.10	192.168.0.201
6	4min 35s 0ms	192.168.27.4	192.168.0.201
7	4min 40s 1ms	192.168.27.8	192.168.0.201
8	5min 47s 3ms	192.168.27.15	192.168.0.201
9	4min 5s 3ms	192.168.27.18	192.168.0.201
10	6min 3s 2ms	192.168.27.31	192.168.0.201

Média = 5,066min

Figura 5.22: Resultado do 2º ciclo

ao *honeypot* 192.168.0.201, com isso, os foi realizado testes com o segundo *honeypot*, devido ser outro sistema operacional, e bastando apenas um modelo para alcançar os resultados, mostramos também uma representação gráfica desses dados onde podem ser observados na figura 5.23.

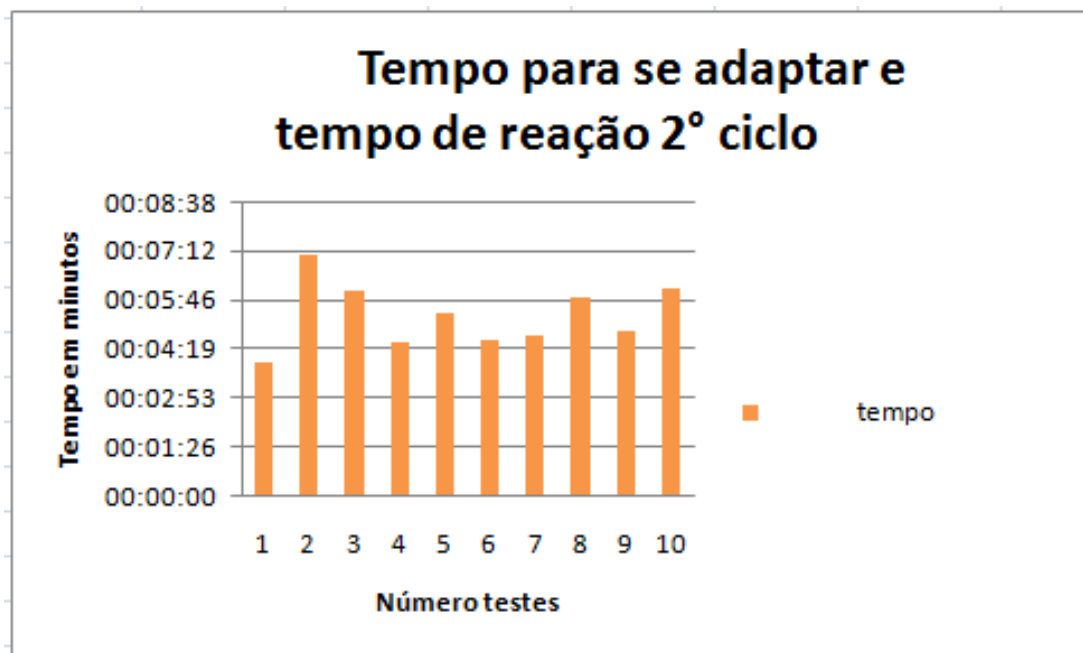


Figura 5.23: Gráfico dos testes colhidos no 2 ciclo

Observando a tabela 5.2 abaixo, seja o conjunto universo $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, representando as interações, e considerando os seguintes conjuntos “Fuzzy” :

$A = \{\text{Normal}\}$, $B = \{\text{Acima do Normal}\}$ e $C = \{\text{Suspeito}\}$, para os quais atribuímos os graus de pertinência dos elementos do conjunto U na seguinte tabela, lembrando que a interação 1 e 2 foram descartas na tabela:

Interações	Normal	Acima do Normal	Suspeito
1	-	-	-
2	-	-	-
3	0	1	0.2
4	1	0	0
5	1	0	0
6	1	0	0
7	1	0	0
8	1	0	0
9	1	0	0
10	0	1	0.2

Tabela 5.2: Lógica fuzzy aplicada no 2º ciclo

Pode-se concluir:

- O tempo médio alcançado para realizar a troca de máquina, leva em torno de 5.066min, isto é, quando um *honeypot* for comprometido, o sistema troca esse *honeypot* invadido, por uma réplica que fica em um local segura a espera de ser chamada para troca.
- O tempo de rollback ou melhor, a troca de máquinas não pode exceder o tempo médio nesse sistema, pois levantaria atenção do administrador, que deverá procurar algo suspeito para que esse tempo ultrapasse a normalidade, dessa forma, serve como alerta para o administrador que algo está errado.
- O tempo de rollback não pode ser muito grande, pois levantaria suspeita do atacante para algo está demorando.
- De acordo com a tabela fuzzy aplicada no 2º ciclo, não teve nenhuma ocorrência suspeita, caso ocorra alguma ocorrência acima do valor normal, por exemplo: um tempo 10min, seria necessário alertar o administrador do sistema.

3º Métrica: Sensibilidade

A terceira métrica à avaliar o *AutonomicSec* é a Sensibilidade, essa métrica na realidade, depende da natureza da atividade, há geralmente alguma forma de atraso no retorno de alguma parte do ambiente do sistema autônomo, sofrido por alguma mudança

em seu ambiente. Logo, se um sistema é muito sensível ao ambiente, potencialmente pode causar constantes alterações na sua configuração, ficando muito custoso essa sensibilidade não controlada. No *AutonomicSec*, essa sensibilidade é avaliada pelo número de pacotes que passa pelo *firewall*, que é controlado pelas regras de *iptables* [16], que é responsável por controlar os alertas caso haja uma invasão, criamos uma faixa de pacotes para avaliar com essa métrica o *AutonomicSec*, realizamos testes com o limite de pacotes: 5, 10, 15 e 20, essa faixa não pode ser muito baixa, pois não seria ideal para defesa do *firewall* reagir dessa forma, qualquer pacote que passasse pelo firewall ele imediatamente bloquearia, isto é, a idéia é deixar brechas para que haja interação do atacante com os *honeypots*, e nem muito alto, pois levaria um tempo suficiente para o atacante consiga o que deseja.

Testes - 1° Ciclo AutonomicSec

Afim de realizarmos os testes com no primeiro ciclo, a partir de um computador fora da rede interna do *AutonomicSec*, no caso de interação com um *honeypot* com o uso protocolo ICMP(Internet Control Message Protocol) utilizando o *ping*, a reação do *AutonomicSec* é bloqueiar o intruso nesse protocolo a todos servidores de produção da rede, não levando em consideração a porta.

É importante frisar, que a avaliação da Sensibilidade sobre o *AutonomicSec*, segue o mesmo modelo do funcionamento lógico mostrado na métrica Tempo para se adaptar e tempo de reação no primeiro ciclo, mostrado na figura 5.8, os dados contendo o log e o identificador do monitor são serealizados para serem enviados a fase de análise e planejamento. Depois que esses chegam a essa fase, os dados filtrados passam por uma estratégia autônômica, verificando o IP encontrado não esteja na lista branca, mostrados na figura 5.9.

O cenário de coleta de resultados é o mesmo neste ciclo, mostrado na figura 5.10, a única mudança realizada está nas regras de *iptables*, no número de pacotes que determina o alerta caso um endereço de origem ultrapassar 5, 10, 15 e 20 pacotes direcionados aos servidores de produção, reduzindo o número de testes de 10 para 4 nessa métrica. Ao final é gerada as mesmas regras *iptables* bloqueando o atacante aos servidores de produção, mostrados nos logs do *honeypot* e *AutonomicSec*, nas linhas 75, 76, 77 e 78.

Resultados

A figura 5.24 abaixo mostra os resultados realizados com a Sensibilidade no primeiro ciclo, o tempo pouco variou, nesse caso, não é tão importante o tempo que realiza a troca de máquinas, mas sim o número de pacotes que influencia o tempo de reação do mecanismo autônomo. Nesse teste mostrado na figura 5.24, ataques de origem externa a rede, gera ataques utilizando protocolo ICMP, através do *ping* em SRC, destinados aos servidores de produção em DST, com endereços 120, 121 e 130.

PACOTES	TIME	SRC	DST
5	13s 4ms	192.168.27.20	192.168.0.120
10	12s 0ms	192.168.27.91	192.168.0.121
15	12s 9ms	192.168.27.2	192.168.0.130
20	13s 1ms	192.168.27.5	192.168.0.130
Média pacotes = 10			

Figura 5.24: Resultado do 1º ciclo

Na figura 5.25 mostra-se o gráfico dos os resultados do primeiro ciclo, a média de pacotes é mantida entre 10 e 15, o número ideal de pacotes para esse ciclo é 10, pois não pode ser muito grande, permitindo que o atacante tenha tempo suficiente para concluir sua tarefa, e nem muito pequena, pois afetaria a sensibilidade do mecanismos autônomo, tornando muito sensível a qualquer interação com os *honeypots*.

Observando a tabela 5.3 abaixo, seja o conjunto universo $U = \{5, 10, 15, 20\}$, representando os Pacotes, e considerando os seguintes conjuntos “Fuzzy” : $A = \{\text{Normal}\}$, $B = \{\text{Acima do Normal}\}$ e $C = \{\text{Suspeito}\}$, para os quais atribuímos os graus de pertinência dos elementos do conjunto U na seguinte tabela:

Pacotes	Normal	Acima do Normal	Suspeito
5	0	0.2	0.8
10	1	0	0
15	1	0	0
20	0	0.2	0.8

Tabela 5.3: Lógica fuzzy aplicada no 1º ciclo

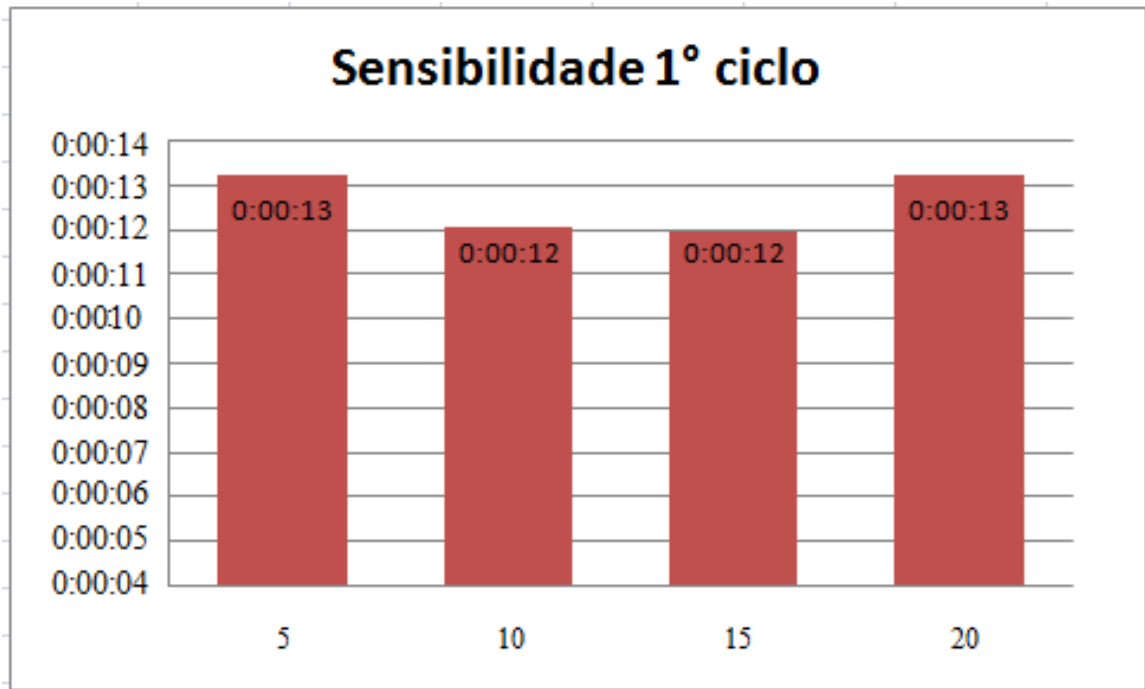


Figura 5.25: Gráfico dos testes do 1º ciclo

Pode-se concluir:

- O número de pacote médio para serem realiza a troca de máquinas chega a 10, dessa forma, o número não é tão pequeno para gerar tomada de decisão pelo componentes *Analyser/plan*.
- Essa média serve para detectar ataques manuais, como ataques automatizados, pois em uma situação de ataques automatizados, o nível de pacotes transmitidos é enorme, ocorrendo uma inversão no ataque manual, onde o invasor tenta o mínimo possível de interações com a vítima para não levantar muita suspeita.
- A sensibilidade tem influências diretamente no custo e reação a intrusões, o primeiro, leva a ser muito custoso, quando a sensibilidade torna o *AutonomicSec* sensível as interações, causando uma enorme troca de máquinas em um curto período de tempo, o segundo, influência na reação na tomada de decisão realizada pelo atuador quando se checka o log do *iptables*, tendo uma ocorrência muito maior do que o normal, tendo pouco tempo para que o sistema captura informações do atacante.
- De acordo com a tabela fuzzy aplicado no 1º ciclo, houve duas ocorrências próximo suspeito.

Testes - 2º Ciclo AutonomicSec

Neste ciclo, os dados também passam pelo mesmo processo do segundo ciclo na métrica Tempo para se adaptar e tempo de reação. Dessa forma, o monitoramento baseado em tempo e utiliza *ECA Rules* para tomada de decisão. Segue as mesmas regras de *iptables* inseridas na configuração do *firewall* que pode ser visto na figura ???. O *honeypot* torna-se comprometido quando há tráfego de saída, essas regras foram baseadas no *Honeywall* e *Sessionlimit*. Por fim, quando o sensor captura através do log do *iptables*, e encontra a seguinte informação, “*Honeypot Comprometido*”, o sensor enviar para o *AutonomicSec* toda linha extraída do respectivo log do *iptables*, ao chegar no monitor, o log é filtrado e extraído apenas o endereço de origem de cada registro, sendo mostrado na figura 5.19, esses endereços são passados para fase de análise e planejamento, que verifica se esses endereços pertencem a algum *honeypot*, caso seja verdade, as ações geradas pelo executor pega esse endereço e substitui por sua respectiva máquina virgem, mostrada na figura 5.20.

Resultados

Os resultados são mostrados nesse segundo ciclo, podem ser vistos na figura 5.26, onde analisou-se a quantidade de trocas de máquinas que seria realizadas em um tempo contante, em uma hora, nesse intervalo de tempo, e conseqüentemente alternando o número de pacotes, verificou-se que a medida que o número de pacotes aumenta, a quantidade de troca de máquinas decresce, isto é, o número de pacotes influencia na quantidade de máquinas sendo trocadas, ou seja, o número de pacotes é proporcional a quantidade de substituições sendo realizadas.

PACOTES	TIME (hora)	Número de troca de máquinas	SRC	DST
5	1	17	192.168.27.20	192.168.0.201
10	1	13	192.168.27.91	192.168.0.201
15	1	10	192.168.27.2	192.168.0.201
20	1	9	192.168.27.5	192.168.0.201

Média de pacotes = 10

Figura 5.26: Resultado do 2º ciclo

Fazendo uma análise mais profunda do resultado dos dados colhidos nesta métrica, onde pode ser visto na figura 5.27, o gráfico mostra a quantidade de máquina que houve por tempo constante de uma hora, variando apenas o número de pacotes, a

medida que o número de pacotes aumentam, o número de pacotes diminui, ou seja, o número de pacotes é proporcional a quantidade de troca de máquinas que são realizadas, devido a métrica sensibilidade afetar diretamente no tempo de reação do atuador.

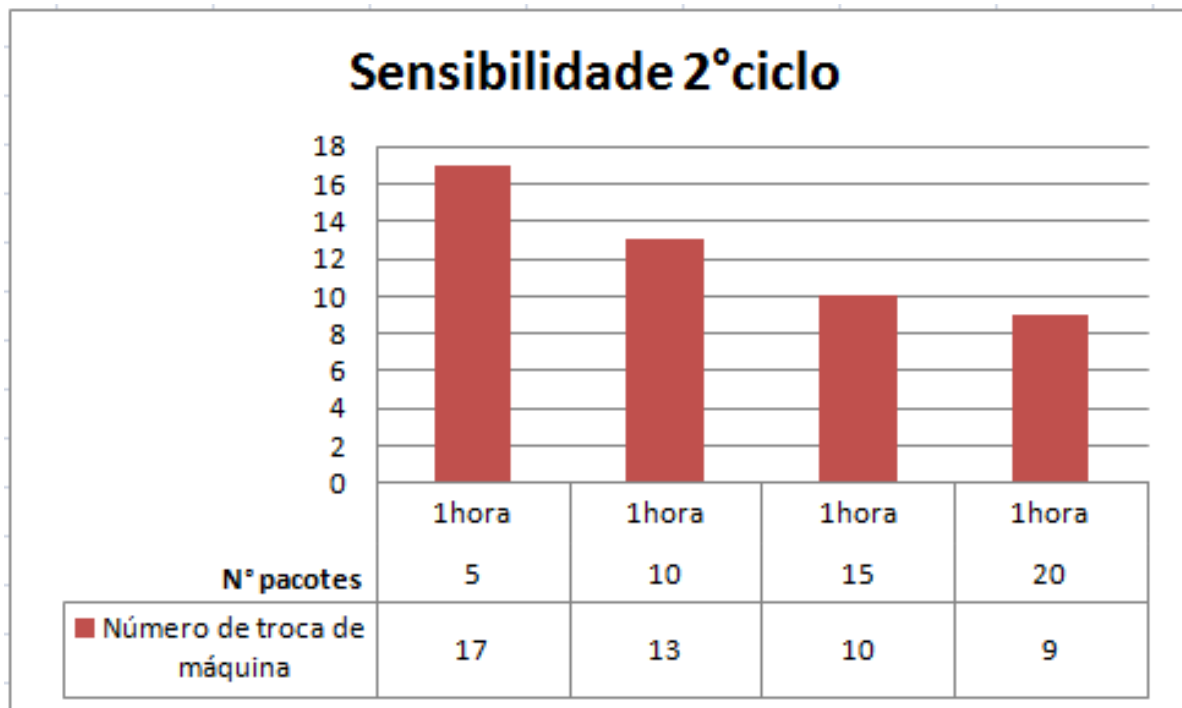


Figura 5.27: Gráfico dos testes do 2º ciclo

Observando a tabela 5.4 abaixo, seja o conjunto universo $U = \{5, 10, 15, 20\}$, representando os Pacotes, e considerando os seguintes conjuntos “Fuzzy” : $A = \{\text{Normal}\}$, $B = \{\text{Acima do Normal}\}$ e $C = \{\text{Suspeito}\}$, para os quais atribuímos os graus de pertinência dos elementos do conjunto U na seguinte tabela:

Pacotes	Normal	Acima do Normal	Suspeito
5	0	0.8	1
10	0	1	0.2
15	1	0	0
20	1	0	0

Tabela 5.4: Lógica fuzzy aplicada no 2º ciclo

Pode-se concluir:

- O número de pacotes influencia na sensibilidade do *AutonomicSec*, afetando no tempo da troca de máquinas.
- O número de pacotes médio para o segundo ciclo, foi encontrado um nível de 10 pacotes, para que a tomada de decisão do *analyser/plan* seja o suficiente para detectar uma ação de intrusão.
- Essa média serve para detectar ataques manuais, como ataques automatizados, pois em uma situação de ataques automatizados, o nível de pacotes transmitidos é enorme, ocorrendo uma inversão no ataque manual, onde o invasor tenta o mínimo possível de interações com a vítima para não levantar muita suspeita.
- De acordo com os resultados observados na fuzzy aplicada no 2º ciclo, houve uma ocorrência suspeita com o pacote 5, localizado na primeira linha, devida a essa ocorrência, o sistema mostra algo suspeito está ocorrendo, deve o administrador tomar medidas que irão buscar a falha.

5.3 Conclusão

Neste capítulo apresentamos a avaliação do *AutonomicSec*, diante da reação de três métricas: *Grau de Autonomia*, *Tempo para se adaptar e tempo de reação* e por fim, a Sensibilidade, onde foram selecionadas sobre critérios que atendessem a funcionalidade do *AutonomicSec*.

Como visto nos resultados, é possível avaliar mecanismos autonômicos utilizando essas três métricas, ao final da avaliação de desempenho, obter resultados satisfatório como o sistema deve reagir em certas situações que se deparar, através do apoio da lógica Fuzzy, os resultados podem ser mais bem avaliados, mostrando a classificação em algo *Normal*, *Acima do Normal* e *Suspeito*.

6 Conclusões

Com o crescimento do tamanho e da complexidade que os sistemas computacionais atuais estão chegando, dado a grande quantidade de elementos de hardware e software que os compõe, resulta na incapacidade humana de os gerenciarem. Devido a isso, a Computação Autônômica entre em ação, transfere grande parte das responsabilidades administrativas para o próprio sistema. Tornando-os cada vez mais complexos para os seres humanos gerenciarem.

A idéia por trás dessa avaliação de desempenho é mostrar o quão bem o *AutonomicSec* foi desenvolvido, e mostrar a melhor forma de gerenciá-lo, levando a diminuição da pouca ou nenhuma intervenção humana para manter-se. Foi mostrado os valores médios encontrados através das avaliações das métricas de *Grau de Autonomia*, *Tempo para se adaptar e tempo de reação* e *Sensibilidade* para manter o nível de funcionamento rápido e eficiente do *AutonomicSec*, que seja capaz de reagir a situações cotidianas de seu funcionamento. As três métricas selecionadas, mostram que são capazes e suficientes para avaliar Sistema Autônômicos.

As médias resultantes da avaliação de cada ciclo, mostram que se forem ultrapassadas em tempo de execução, deve alertar ao administrador do sistema que algo é suspeito, ou seja, esta fora da normalidade. Entretanto, o resultado de uma avaliação usando essas três métricas variam de sistemas autônômicos para sistemas.

Contribuições

Neste contexto, as principais contribuições desse trabalho são:

- A avaliação de desempenho de um mecanismo autônômico diante da ausência de materias na literatura que avaliam o desempenho de sistemas autônômicos.
- Modelo de avaliação de sistema autônômicos proposto por esse trabalho.
- A avaliação do mecanismo autônômico, através da realização de testes com os dois ciclos autônômicos em cenário comum de uso. Nesta avaliação, detalhamos os

resultados obtidos em cada componente correspondente a uma fase do ciclo MAPE-K. Ainda mostramos, através desses resultados, o que o mecanismo autônomo agrega para a segurança de redes;

- Para contextualizar essas principais contribuições, ainda exploramos os tópicos de pesquisa fazendo:
 - Uma apresentação sobre Computação Autônoma, na qual mostramos sua origem e conceitos, principais propriedades, arquitetura, modelo MAPE-K e suas fases;
 - Uma visão geral do estado atual da segurança das redes de computadores.
 - Uma apresentação sobre Avaliação de Desempenho, na qual mostramos a escolha ideal da métrica para avaliar um sistema autônomo;

Lições Aprendidas

- Um sistema que envolva Computação Autônoma às redes de computadores ou a segurança destas não vem substituir o papel do administrador de redes ou do responsável por essas áreas no segmento de Tecnologia da Informação. Acreditamos que a CA vem fornecer recurso a mais para esse profissional, para que possa garantir maior qualidade no serviço prestados;
- A construção de um trabalho avaliativo que é bem fundamentado, organizado e claro, leva a uma boa compreensão do leitor até mesmo dos leitores leigos sobre o assunto.
- Para uma boa Avaliação de um software, é necessário uma busca aprofundada de outras avaliações, para servir de base para o que será desenvolvido;
- Em relação aos conhecimentos técnicos aprendidos, afirmamos que um estudo aprofundado sobre uma ferramenta desejável a ser utilizada, tendo foco ou não em segurança, é necessário uma boa avaliação, para que possa encontrar mecanismos ideais para um bom funcionamento.

Referências Bibliográficas

- [1] ALMEIDA, Jussara M., *Análise e Modelagem de Desempenho de Sistemas de Computação*. <http://homepages.dcc.ufmg.br/~jussara/anamodes/anamodes.html>, Acessado em Setembro/2013.
- [2] ASSUNCAO, M. F. A., *Honeypots e honeynets: aprenda a detectar e enganar invasores*, Visual Books, Florianópolis, 2009.
- [3] BASH, *gnu project - free software foundation*, Disponível em: <http://www.gnu.org/software/bash>. Acessado em Setembro/2013.
- [4] BATISTA, Thais Vasconcelos, Pereira, Lucas Silva, *Metropole Digital - Curso Segurança em Redes de Computadores*. http://www.metropoledigital.ufrn.br/aulas_avacado.html, Acessado em Novembro/2013.
- [5] BORGES, P. C.; Coutinho, R. T., *Análise de sistema de detecção de intrusos em redes de computadore.*, 133f. 2007. Monografia (Trabalho de Conclusão de Curso) - Universidade de Franca, Franca, 2007. Disponível em: <http://snort.org.br/arquivos/Monografia-pedro.pdf>. Acessado Novembro/2013.
- [6] CAMPANA, C., *Ferramentas de Segurança*, <http://www.rnp.br/newsgen.html>. Acessado em Novembro/2013.
- [7] CORRÊA, S., Cerqueira, R., *Computação Autônoma: Conceitos, Infraestruturas e Soluções em Sistemas Distribuídos*, In: Capítulo 4 - Minicurso 27º SBRC, 2009.
- [8] ECLIPSE - *the eclipse foundation open source community website*, Disponível em: <http://www.eclipse.org>. Acessado em Setembro/2013.
- [9] FREEMAN, E., *Use a Cabeça! Padrões de Projetos - Design Patterns*. Alta Books, 2009.
- [10] HARIRI, S., B. Khargharia, H. Chen, J. Yang, Y. Zhang, M. Parashar, and H. Liu, *The autonomic computing paradigm*, Cluster Computing, 9:5-17, January 2006.

- [11] HUEBSCHER, M. C. and McCann, J. A., *A survey of autonomic computing—degrees, models, and applications.*, ACM Comput. Surv., 40, 3, Article 7, 2008.
- [12] HORN, P., *Autonomic Computing: IBM Perspective on the State of Information Technology*, IBM T.J. Watson Labs, NY, 15th October 2001. Presented at AGENDA 2001, Scottsdale, AR., Disponível em <http://www.research.ibm.com/autonomic/2001>.
- [13] GOODLOE, A. and L. Pike. Monitoring distributed real-time systems: A survey and future directions. Technical Report NASA/CR-2010-216724, NASA Langley Research Center, 2010. Disponível em <https://www.cs.indiana.edu/~lepik/pub/survey.pdf>. Acessado em Setembro/2013.
- [14] GONCALVES, Jesseildo F., *Introdução à computação autônoma e sua aplicação em ambientes computacionais distribuídos*, Monografia do curso de Bacharelado em Ciência da Computação da Universidade Federal do Maranhão, São Luís, MA, 2010.
- [15] IBM. *An architectural blueprint for autonomic computing*, 2003.
- [16] NETFILTER, *Regras de Iptables*. <http://www.netfilter.org>, Acessado em Setembro/2013.
- [17] K. W. Ross and J. F. Kurose. *Redes de Computadores e a Internet: Uma abordagem top-down*. Addison Wesley, 3 edition, 2006.
- [18] ORACLE e JAVA - *tecnologias*, Disponível em: <http://www.oracle.com/br/technologies/java/index.html>. Acessado em Agosto/2013.
- [19] LAUREANO, M. A. P. and Maziero, C. A., *Virtualização: Conceitos e aplicações em segurança*, In SBSeg 08: Anais dos Minicursos do Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. SBC, 2008.
- [20] MARCELO, A. and Pitanga, M. *Honeypots: a arte de iludir hackers*, Brasport, Rio de Janeiro, 2003.
- [21] MANSOURI-SAMANI, M., *Monitoring of distributed systems*, PhD Thesis Imperial College, 1995.

- [22] MCCANN, J. A. and Huebscher, M. C., *Evaluation Issues in Autonomic Computing. In Proceedings of Grid and Cooperative Computing Workshop, pages, 597-608*, Springer Berlin/Heidelberg, 2004.
- [23] MEINELL, C., *The Überhacker II: More ways to break into a computer*, <http://www.happyhacker.org/tuh.shtml>. Acessado em Novembro/2013.
- [24] MULTICAST, *dns*, Disponível em: <http://www.multicastdns.org>. Acessado em Setembro 2013.
- [25] MURCH, Richard, *Autonomic Computing*, Prentice Hall PTR, 1ªed., March, 2004;
- [26] RAVANELLO, Anderson Luiz; HIJAZI, Houssan Ali; MAZZORANA, Sidney Miguel, *Honeypos e Aspectos Legais*, Dissertação - Especialização em Redes e Segurança - Programa de Pós-Graduação em Informática Aplicada, Pontifícia Universidade Católica do Parana, Curitiba-PR, 2004.
- [27] Rfc, 792 - internet control message protocol. Disponível em: <http://tools.ietf.org/html/rfc792>. Acessado em Setembro 2013.
- [28] SHIREY Internet Security Glossary, <http://www.ietf.org/rfc/rfc2828.txt>, Acessado em Novembro/2013.
- [29] SPITZNER, L., *Honeypos: Definitions and value of honeypots*, In SANS Annual Conference, 2002.
- [30] SSH communications security, *ssh, secure shell, data-in-transit*, Disponível em: <http://ssh.com>. Acessado em Agosto/2013.
- [31] TELES, Ariel Soares, *AutonomicSec: Um Mecanismo autônômico para Segurança de Redes baseado em Decepção. Master's Thesis*, PPGEE/UFMA, 2012.
- [32] TEIXEIRA, Mario Meireles, *Avaliação de desempenho*. <http://www.deinf.ufma.br/mario/pos/avalades/avalades.html>, Acessado em Setembro/2013.
- [33] VM, *virtualbox*, Disponível em: <http://www.virtualbox.org>. Acessado em Novembro/2013.
- [34] VIRTUALBOX, *VirtualBox Corporation*, Disponível em: <http://www.virtualbox.org/manual/ch08.html>. Acessado em: 24 jan. 2012.

- [35] RUTHERFORD, M. J., Anderson, K., Carzaniga, A., Heimbigner, D., and Wolf, A. L. *Reconfiguration in the Enterprise Javabean component model*, In Proceedings of the IFIP/ACM. Working Conference on Component Deployment, Berlin, Germany. 67-81, 2002.
- [36] BADGER, L. *Self-regenerative systems (SRS) program abstract*, 2004.
- [37] KENYON, H. S. *Battlefield cognizance tool points to future*, SIGNAL Magazine, 2001.
- [38] KEPHART, J. O. and Chess, D. M. *The vision of autonomic computing*. Computer 36, 41-50, 2003.
- [39] PAULSON, L. D. *Computer system, heal thyself*. Computer, 20-22, 2002.
- [40] BHATTI, S. N. and Knight, G. *Enabling QoS adaptation decisions for Internet applications*. Computer Networks 31, 7, 669-692, 1999.
- [41] CHAKERES, I. D. and Belding-Royer, E. M. *Aodv routing protocol implementation design*. In ICDCSW 04: Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W7. IEEE Computer Society, Washington, DC, USA, 698-703, 2004.
- [42] AGARWALA, S., Chen, Y., Milojicic, D., and Schwan, K. *QMON: QoS- and Utilityaware monitoring in enterprise systems*. In Proceedings of the 3rd IEEE International Conference on Autonomic Computing (ICAC). Dublin, Ireland, 2006.
- [43] GARLAN, D., Schmerl, B., and Chang, J. *Using gauges for architecture-based monitoring and adaptation*. In Working Conference on Complex and Dynamic Systems Architecture, Brisbane, Australia, 2001.
- [44] ZENMYO, T., Yoshida, H., and Kimura, T. *A self-healing technique based on encapsulated operation knowledge*. In Proceedings of 3rd IEEE International Conference on Autonomic Computing (ICAC). Dublin, Ireland, 25-32, 2006.
- [45] JENNINGS, N. R. *On agent-based software engineering*. Artificial Intelligence, 277-296, 2000.
- [46] BATRA, V. S., Bhattacharya, J., Chauhan, H., Gupta, A., Mohania, M., and Sharma, U. *Policy driven data administration*, In Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks, 2002.

- [47] OREIZY, P., Medvidovic, N., and Taylor, R. N. *Architecture-based runtime software evolution*, In ICSE 98: Proceedings of the 20th international conference on Software engineering. IEEE Computer Society, Washington, DC, USA, 177-186,1998.
- [48] D. and Schmerl, Garlan, B. Model-based adaptation for self-healing systems. In Proceedings of the first workshop on Self-healing systems,2002.
- [49] DASHOFY, E. M., van der Hoek, A., and Taylor, R. N. *Towards architecture-based self-healing systems*, In Proceedings of the first workshop on Self-healing systems, 2002.
- [50] WALSH, W. E., Tesauro, G., Kephart, J. O., and Das, R. *Utility functions in autonomic systems*, In Proceedings of the First International Conference on Autonomic Computing. 70-77,2004.
- [51] BHOLA, S., Astley, M., Saccone, R., and Ward, M. *Utility-aware resource allocation in an event processing system*, In Proceedings of 3rd IEEE International Conference on Autonomic Computing (ICAC). Dublin, Ireland, 55-64,2006.