UNIVERSIDADE ESTADUAL DO CEARÁ

CENTRO DE CIÊNCIAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

DAVI TELES FRANÇA

UM SISTEMA INTELIGENTE PARA AMBIENTES SIMULADOS DE TREINAMENTO EM SEGURANÇA DA INFORMAÇÃO

FORTALEZA – CEARÁ

2016

DAVI TELES FRANÇA

UM SISTEMA INTELIGENTE PARA AMBIENTES SIMULADOS DE TREINAMENTO EM SEGURANÇA DA INFORMAÇÃO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Orientador: Marcial Porto Fernandez

Co-Orientador: André Luiz Moura dos Santos

FORTALEZA – CEARÁ

2016

Deve ser gerada através do preenchimento do Formulário Eletrônico de Elaboração da Ficha Catalográfica, disponível no link: http://www.uece.br/biblioteca/index.php/entrega-de-trabalho.

X000x Sobrenome, Nome do 1º autor. (citado na folha de rosto)
Título principal: subtítulo./Nome completo do 1º autor, Nome completo do 2º autor, Nome completo do 3º autor; orientação [de]. – Local: ano.
Nº de folhas.: il.(se houver ilustração); 30 cm.

Inclui bibliografias: f.(nº da folha em que se encontra)
Trabalho de Conclusão de Curso (Graduação em) – Universidade Estadual do Ceará – (UECE).

1. Assunto. 2. Assunto. 3. Assunto. I. Sobrenome, Nome do 2º autor. II. Sobrenome, Nome do 3º autor. III. Sobrenome, Nome do orientador (orient.). IV. Universidade Estadual do Ceará – UECE. V. Título.

CDU

ATA DA NONAGÉSIMA DEFESA PÚBLICA DE DISSERTAÇÃO DE MESTRADO

Ao vigésimo quinto dia do mês de agosto de dois mil e dezesseis, no miniauditório do prédio de Pesquisa e Pós-Graduação em Computação, do Mestrado Acadêmico em Ciência da Computação – MACC, realizou-se a sessão pública de defesa da dissertação de **DAVI TELES FRANÇA**, aluno regularmente matriculado no Mestrado Acadêmico em Ciência da Computação – MACC, intitulada: **"UM SISTEMA INTELIGENTE PARA AMBIENTES SIMULADOS DE TREINAMENTO EM SEGURANÇA DA INFORMAÇÃO".** A Banca Examinadora reuniu-se no horário de 15h às 17 horas, sendo constituída pelos Professores Doutores: **Marcial Porto Fernandez** (Orientador/UECE); **André Luiz Moura dos Santos** (Co-Orientador/UECE); **Gustavo Augusto Lima de Campos**(UECE); **José Neuman de Sousa**- Universidade Federal do Ceará-UFC. Inicialmente o mestrando expôs seu trabalho e a seguir foi submetido à arguição pelos membros da Banca, dispondo cada membro de tempo para tal. Finalmente a Banca reuniu-se em separado e concluiu por considerar o mestrando APROVADO, por sua dissertação e sua defesa pública. Eu, **Professor Dr. Marcial Porto Fernández** , Orientador e Presidente da Banca, lavrei a presente Ata que será assinada por mim e demais membros da Banca. Fortaleza, 25 de agosto de 2016.

Prof. Dr. Marcial Porto Fernández
(Orientador – UECE)

Prof. Dr. André Luiz Moura dos Santos
(Co-Orientador/UECE)

Prof. Dr. Gustavo Augusto Lima de Campos
(UECE)

Prof. Dr. José Neuman de Sousa
(UFC)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

AGRADECIMENTOS

Primeiramente a Deus que permitiu que tudo isso acontecesse, ao longo de minha vida, e não somente nestes anos como universitária, mas que em todos os momentos é o maior mestre que alguém pode conhecer.

Aos meus pais, pelo amor, incentivo e apoio incondicional.

Obrigada meus irmãos e sobrinhos, que nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente!

A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. a palavra mestre, nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos.

“É melhor lançar-se à luta em busca do triunfo mesmo expondo-se ao insucesso, que formar fila com os pobres de espírito, que nem gozam muito nem sofrem muito; E vivem nessa penumbra cinzenta sem conhecer nem vitoria nem derrota.”

(Franklin Roosevelt)

RESUMO

O ensino da Segurança da Informação é um desafio, pois exige um ambiente controlado, passível de corrompimentos de arquivos e diretórios, corrupção de configurações de sistema operacional, elementos de rede, entre outros. Assim, ambientes computacionais simulados, onde profissionais e estudiosos dessa área possam treinar as mais diferentes técnicas dessa se mostram os mais aptos a serem utilizados para o ensino da Segurança Informação. O Shellter, uma rede social dedicado ao ensino e prática da segurança da informação, visa suprir as necessidades dos educadores, tutores, alunos e entusiastas dessa área. Através de desafios, trilhas de ensino, matérias para estudos, fóruns para compartilhamento de informações e simulações fornece uma arena completa para o aprendizado *hands-on*. No ambiente simulado, um dos desafios é motivar os usuários a explorar os seus conhecimentos, assim como adquirir novos, para cumprir os desafios que serão apresentados. E os usuários desse ambiente simulado possuem diferentes perfis (experiências, conhecimentos, etc.). Dessa forma, é necessário escolher o desafio conforme o conhecimento e capacidade desses usuários para mantê-los motivados a continuar estudando e praticando Segurança da Informação. Este trabalho propõe um sistema para ambientes simulados de segurança da informação formado por um conjunto de agentes inteligentes para motivação e acompanhamento de usuários nesses ambientes, além da implementação de um protótipo e avaliação do mesmo.

Palavras-chave: Segurança. Agentes Inteligentes. Motivação. Simulação.

ABSTRACT

The information security education is a challenge because it requires the educator's knowledge in other areas of computer science, due to the fact that security is an intermediate area. In addition, study and practice of information security is also a challenge because it requires a controlled environment, prone to corruptions files and directories, operating system files, network elements, among others. Thus Shellter, a social network dedicated to the teaching and practice of information security, aims to meet the needs of educators, tutors, students and enthusiasts of this area. Through challenges, educational tracks, materials for studies, information sharing forums and simulations to provide a full arena for learning *hands-on*. In the simulated environment, one of the challenges is to motivate users to explore their knowledge, as well as acquire new ones, to meet the challenges that will be presented. This work presents a multi-agent system in simulated environment to motivate the teaching of information security.

Keywords: Information Security. Intelligence Agent. Motivation. Simulation

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A capacitação de recursos humanos em segurança da informação enfrenta vários desafios. Como em várias outras áreas, o treinamento em segurança cibernética necessita de um preparo de forma prática (NAGARAJAN et al., 2012). Apenas o aprendizado teórico não é o suficiente, pois deixa lacunas na formação profissional. Além disso, o treinamento em segurança da informação necessita de um ambiente simulado e controlado pela possibilidade de criar situações realistas, mas que não causem impactos no mundo real e que ofereça a possibilidade de analisar o desenvolvimento dos estudantes.

Para praticar segurança, muitas vezes, é necessário invadir redes e sistemas computacionais utilizando-se de técnicas que podem danificar os alvos se a ação não for realizada com perícia. Praticar técnicas de segurança da informação em sistemas de terceiros, mesmo que para fins educacionais, é ilegal. Atacar sua própria rede ou realizar exercícios em uma rede compartilhada, como em uma universidade, pode acabar danificando os equipamentos e causar grandes prejuízos. A maneira mais adequada, portanto, para aprender e praticar segurança cibernética é um ambiente computacional isolado especialmente criado para tanto.

Isso se deve ao fato de que a segurança da informação, além de toda a complexidade exigida, trabalha intimamente com as outras áreas da computação, ou seja, trabalha de maneira direta com os conhecimentos dessas áreas. Utiliza conhecimentos específicos de redes de computadores, conhecimentos específicos de banco de dados, etc.

Jogos de simulação são amplamente utilizados no ensino e treinamento de profissionais e no desenvolvimento organizacional, e o uso de simulações com objetivo de treinamento pode ser visto já nos anos 1600, no uso de jogos de guerra com o propósito de melhorar as estratégias do exército e marinha (ALLEN, 1994). As simulações começavam com um cenário, que se desenvolvia durante a partida, quando times representando diferentes governos começavam a atacar e reagir às situações.

Sistemas de ataque e defesa simulados já provaram ser uma ótima fonte de aprendizado, porém, para um treinamento eficiente, os ambientes devem ser bem definidos. Treinamentos usando o jogo para conduzir o aprendizado tem-se mostrado bastante eficientes.

A gamificação, isto é, o uso de técnicas motivacionais derivadas dos jogos, ajuda os estudantes a se dedicar mais, aumentando assim o seu aprendizado. Ela usa a dinâmica e a mecânica dos jogos, como a recompensa e a classificação por determinadas ações realizadas pelo usuário de determinado sistema (computacional ou não) que não seja um jogo (GROH, 2012).

Ela também reforça que, mesmo fora de um jogo, é necessário utilizar as técnicas de nível correto com os usuários de níveis compatíveis. Isso porque é importante não desestimular o usuário no cenário em que a técnica de gamificação é utilizada justamente para estimular.

O aprendizado pela prática pode ser em um ambiente no qual o estudante pode acessar um ambiente simulado e assim ter experiências em um ambiente virtual, como um jogo. Pela emulação, é possível uma melhor análise de resultados e acompanhamento de desenvolvimento individual e em equipe. O objetivo das emulações em segurança da informação é ensinar as habilidades necessárias para proteger sistemas computacionais, como técnicas de defesa e teste de penetração, por meio de um jogo.

Motivar alunos para o aprendizado é um desafio antigo de profissionais da educação. Um aluno sente-se motivado a participar das atividades de aprendizagem caso acredite que, com seus conhecimentos, talentos e habilidades, poderá adquirir novos conhecimentos, dominar um conteúdo, melhorar suas habilidades, etc. (BZUNECK, 2009).

O tema motivação ligado à aprendizagem está sempre em evidência nos ambientes escolares, impelindo professores a se superar ou fazendo-os recuar, chegando à desistência nos casos mais complexos. Ela tem um papel muito importante nos resultados que os professores e alunos almejam (MORAES, 2007).

Em ambientes de ensino virtuais, ou seja, ambientes de ensino não-presenciais, motivar alunos para aprender se torna um desafio maior, pois o educador não consegue identificar os sentimentos expressados pelos alunos em relação ao conhecimento que está sendo transmitido. Nestes ambientes, o educador pode identificar sentimentos dos alunos pelas expressões faciais, conversas pessoais, testes direcionados, entre outros índices.

O Shellter, apresentado na Seção 3.3, é uma rede social para o aprendizado de segurança da informação. Ele foi concebido com o objetivo de suprir as necessidades e lacunas do ensino da segurança da informação. Com ambientes de aprendizado teórico e prático, suportados por técnicas de virtualização e computação em nuvens, o Shellter oferece sistemas computacionais completos para resolução dos mais diversos desafios em segurança da informação, assim como a simulação de ambiente para treinamento de situações reais de segurança da informação.

Um dos grandes desafios do Shellter é motivar os seus usuários a aprender segurança da informação, a desafiar seus limites, a buscar novos conhecimentos e a vencer novos desafios. O Shellter inspirou este trabalho. Ele faz parte da arquitetura geral desta rede social. O sistema

multiagentes proposto se encaixa no ambiente simulado do Shellter.

Assim, este trabalho definirá um sistema de agentes inteligentes que motive e acompanhe o aprendizado em ambientes simulados virtuais de segurança da informação. Com isso, será feita uma definição do sistema e seus requisitos, além do estabelecimento da arquitetura e fluxos de interações dos agentes. Também será feita a validação do sistema, por meio da implementação e testes do protótipo de um dos agentes, visto que as arquiteturas dos agentes do sistema são semelhantes.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Definir um sistema multiagentes para motivação de usuários em ambientes de simulação e/ou laboratórios virtuais para o ensino da segurança da informação. Esse sistema deve incluir técnicas de gamificação para o acompanhamento.

1.1.2 Objetivos Específicos

1. Desenvolver a arquitetura do sistema proposto;
2. Implementar um dos agentes da arquitetura proposta. Assim como testá-lo e validá-lo: definir o modelo racional do agente e definir a classificação utilizada pelo agente.
3. Integrar a arquitetura proposta com a arquitetura do Shellter.

2 TRABALHOS RELACIONADOS

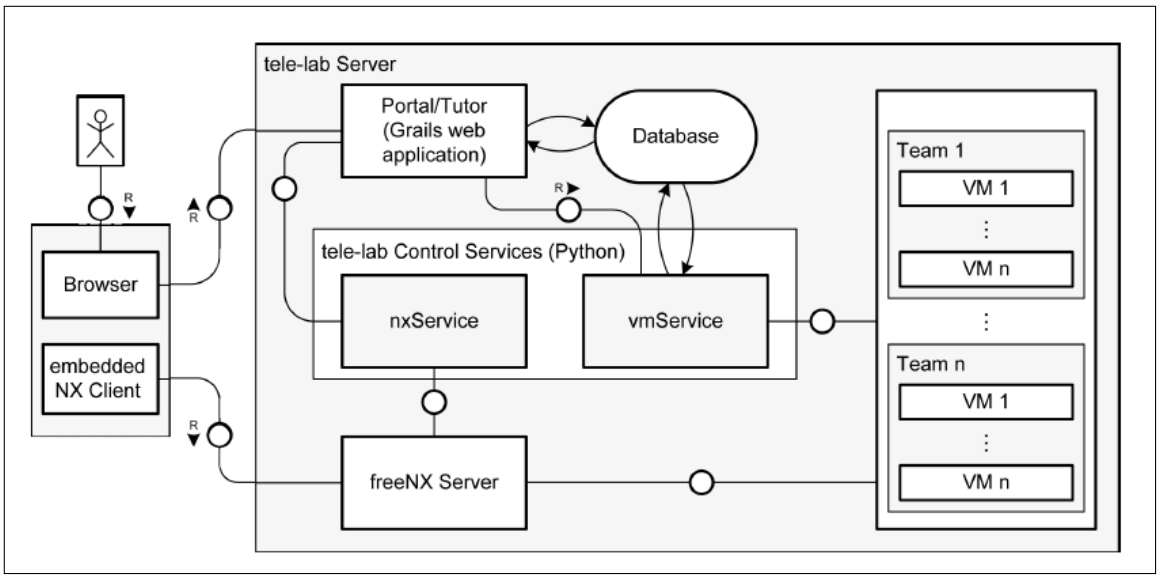
Nesta Seção serão apresentados trabalhos relacionados à temática de simulação e treinamento em ambientes virtuais.

2.1 TELE-LAB

O projeto Tele-Lab oferece um sistema *hands-on* para o treinamento de segurança da informação. Para isso, utiliza um ambiente virtual remoto, ou seja, na web, acessível em qualquer lugar (WILLEMS; MEINEL, 2011).

O sistema consiste de tutoriais em texto, em vídeo e exercícios práticos, além de um ambiente de treinamento constituído por máquinas virtuais. Os alunos realizam os exercícios práticos nas máquinas virtuais por acesso via SSH.

Figura 1 – Arquitetura Tele-Lab



Fonte – (WILLEMS; MEINEL, 2011).

A arquitetura na Figura 1 é formada por uma interface web (*Portal and Tutorial Enviroment*) responsável pela autenticação do usuário, navegação por lições, exposição de conteúdos e controle da requisição de máquinas virtuais para treinamento.

Um conjunto de máquina virtuais (*Virtual Machine Pool*), onde cada máquina virtual contém diferentes configurações para diferentes cenários oferecidos pelo sistema. Diferentes instâncias de um mesmo conjunto de máquinas virtuais configuradas para um mesmo exercício podem ser oferecidas a diferentes para times para que os exercícios sejam executados de maneira

isolada e transparente.

O banco de dados (*Database*) contém as informações dos usuários, o conteúdo das lições disponibilizadas pela *interface* web, assim como informações de máquinas virtuais e *templates* para exercícios.

Também existe um acesso ao ambiente de treinamento pelo *Remote Desktop Access Proxy*, que é feito por *softwares* clientes instalados nas máquinas dos usuários. A *interface* de administração (*Administration Interface*) oferece uma *interface* web de controle de todo o sistema, desde gerenciamento de usuários até gerenciamento de treinamentos.

A *Tele-Lab Control Services* é responsável por conectar e servir de *interface* entre os outros componentes da arquitetura.

Para motivação, os alunos são convidados a assumir a perspectiva de um atacante, assim podem vivenciar experiências relacionadas as possíveis ameaças que podem enfrentar na vida pessoal ou na vida profissional. Além disso, os alunos podem acompanhar o seu progresso nos exercícios de treinamento.

2.2 SOFTICE

SOFTICE (GASPAR SARAH LANGEVIN, 2007) é um projeto que foca no ensino de sistemas operacionais pelo aprendizado *hands-on*. Mais especificamente, o foco é no ensino do *kernel* do Linux, ensinando o seu funcionando, definições e implementações.

Com isso, o estudante pode testar os seus conhecimentos, assim como implementar novos módulos e funcionalidades em um ambiente controlado.

No trabalho, é proposto uma abordagem para o ensino de sistemas operacionais. Essa abordagem leva em conta que o código do *kernel* do Linux é grande e complexo o suficiente para desmotivar alunos que queiram explorá-lo:

- Os alunos podem ficar assustados com a grande quantidade de código do *kernel*. Assim, se concentram apenas em pequenas mudanças, absorvendo apenas uma quantidade mínima de conhecimento referente as suas alterações no código;
- Mesmo as pequenas modificações, podem exigir um guia passo a passo de onde e o que modificar. Isto pode causar dependência nos alunos, não os incentivando a desenvolver habilidades de exploração de código, importante para o estudo do *kernel* do Linux.

Na abordagem proposta, os módulos do *kernel* do Linux são utilizados para enfrentar as problemáticas apontadas anteriormente. Utilizam-se da característica que os módulos se

acoplam ao *kernel* e podem possuir quaisquer tamanho e complexidade.

Assim, os estudantes programam os módulos como se fossem o *kernel* em si. Em termos pedagógicos, o foco do ensino são elementos específicos do *kernel* e não pequenas alterações (GASPAR SARAH LANGEVIN, 2007).

2.3 INSIGHT

O Insight (FUTORANSKY et al., 2009) é uma plataforma de simulação criada para projetar e simular ciber ataques. Os ataques fazem parte de cenários disponibilizados na plataforma, onde cada cenário contém diversos atores: dispositivos de rede, dispositivos de *hardware*, *softwares*, protocolos de rede, usuários, etc. Objetivo é simular ataques da perspectiva do atacante.

A plataforma fornece uma interface e ferramentas para criação de ataques. O objetivo é que os usuários reproduzam técnicas existentes, assim como se motivem a melhorá-las, além de criar suas próprias técnicas. Todos os ataques são executados de um *framework* de ataque dentro do Insight para garantir o isolamento e transparência do ambiente simulado.

O ambiente simulado foi projetado para ser executado em uma única máquina física, que também possui a função de gerenciar máquinas virtuais e todo tráfego de rede virtual entre elas.

Um modelo probabilístico é utilizado para decidir se um ataque terá sucesso ou não: baseado na técnica em conjunto com as configurações da máquina. Se o ataque obtiver sucesso, o Insight garante o controle da máquina para o usuário. Por exemplo, determinado *exploit* possui 83% de possibilidade de sucesso contra um *Windows XP* puro se a porta 21 estiver aberta, entretanto irá travar o sistema e não realizará o seu objetivo se o *Windows XP* possuir o pacote *SP1* instalado.

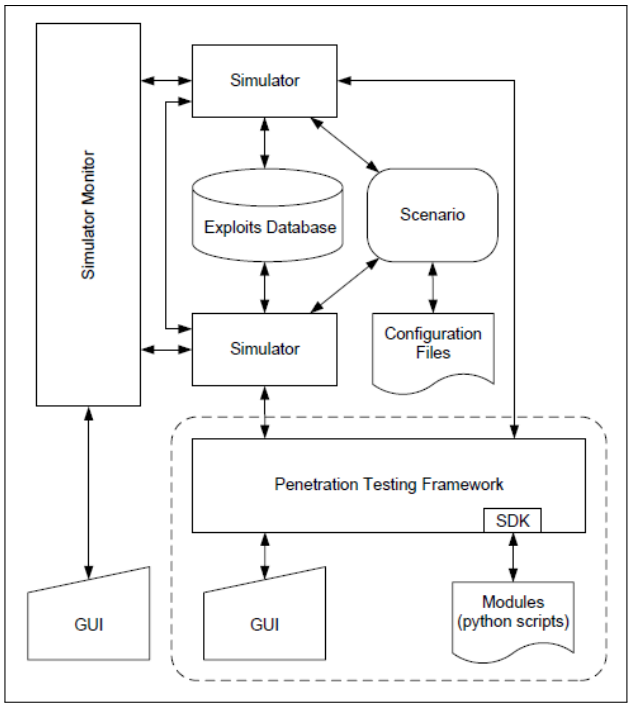
Um *exploit* é um pedaço de código que tenta comprometer um sistema computacional por meio de uma vulnerabilidade específica (FUTORANSKY et al., 2009).

No Insight, uma vulnerabilidade é um mecanismo para ganhar acesso de uma máquina virtual alvo. O *payload* do *exploit* escolhido para explorar uma vulnerabilidade é trocado por um ID ou uma *string* de controle e o modelo probabilístico analisa as chances de sucesso do *exploit*. Se obtiver sucesso, a aplicação vulnerável é instalada na máquina virtual, e o controle da mesma é passado ao usuário.

Da arquitetura do sistema, mostrada na Figura 2, o *Simulator Monitor* gerencia todos

os sistemas simulados. *Configuration Files* guarda configurações do sistema e de snapshots de máquinas virtuais de simulações realizadas para análise. O *Penetration Testing Framework* é o framework de ataque do Insight, ele realizada chamadas de sistema por meio de um canal implementado no simulador. Os ataques processados por este framework são escritos em *Python*.

Figura 2 – Arquitetura Insight



Fonte – (FUTORANSKY et al., 2009).

2.4 COMENTÁRIOS

Os trabalhos apresentados mostram ambientes virtuais para treinamento e ensino de usuários em diferentes áreas da computação, utilizando distintas abordagens. O *Tele-Lab* provê um amplo ambiente para treinamento em segurança da informação, o *SOFTICE* já provê um ambiente mais específico, focado no *kernel* do Linux e o *Insight* disponibiliza um ambiente simulado para testes com *exploits*.

O trabalho proposto nesta dissertação apresenta um ambiente mais completo para estudo teórico e prático, pois, além de propor diversos desafios, por meio da catalogação de técnicas, demonstrado na Seção 5.2, provê um monitoramento do desempenho emocional, descrito na Seção 4.2.2, e técnico no usuário, descrito na Seção 4.2.3 e utiliza técnicas de gamificação para premiar o usuário pelo seu desempenho.

Além do que, é possível oferecer um ambiente mais dinâmico, onde um agente joga com o usuário, descrito na sessão 4.2.4, promovendo um ambiente mais dinâmico e uma maior interação do usuário. E o usuário ainda conta com o acompanhamento de um agente tutor, descrito na sessão 4.2.5, que atua tutoreando de acordo com o desempenho e motivação do usuário junto ao ambiente.

O presente trabalho agrega valor a esses outros trabalhos, pois atua fornecendo inteligência computacional para as seguintes características: monitoramento do desempenho de usuário, pontuação e gamificação, identificação de técnicas, tutoreamento e classificação de usuário e agentes inteligentes para jogar com o usuário.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta Seção, será apresentada a fundamentação teórica que embasou o conteúdo para o desenvolvimento deste trabalho.

3.1 AGENTES INTELIGENTES

Um Agente Inteligente é um sistema computacional situado em um ambiente e capaz de tomar decisões próprias em ordem de cumprir uma série de objetivos (PADGHAM, 2005). Um agente inteligente se caracteriza por ser reativo, proativo e social (WOOLDRIDGE, 2002). Um agente inteligente é autônomo pela sua independência e tomada própria de decisões.

O ambiente no qual está contido um agente muitas vezes é o que o distingue de outros agentes. Agentes são utilizados em ambientes dinâmicos, ou seja, ambientes mais complexos. Esses ambientes não se mantêm estáticos em relação aos objetivos do agente. Assim, ele precisa racionar em relação ao estado do ambiente para tomar suas decisões. Esses ambientes tendem a ser imprevisíveis e não confiáveis.

São imprevisíveis no sentido de que o agente não consegue prever com total confiança os futuros estados que o ambiente pode assumir, pois ele não consegue informações completas sobre o ambiente. E não confiáveis, pois o agente não possui controle sobre o efeito de suas ações no ambiente. Essas ações podem falhar devido à imprevisibilidade de reação do ambiente em relação a elas.

Por exemplo, um agente controlando um robô que joga futebol faz planos baseado na posição atual da bola e de outros jogadores. Entretanto, o agente deve estar preparado para mudanças bruscas no ambiente, fazendo com que os seus planos precisem ser abandonados e novos planos de ação precisem ser feitos de acordo com essas mudanças. Isso caracteriza a reatividade de um agente.

A proatividade se dá quando o agente busca constantemente cumprir os seus objetos. Mesmo que um agente falhe em cumprir os seus objetivos, ele aprende com seus erros e se torna mais robusto. Assim, aumenta a possibilidade de cumprir os seus objetivos com sucesso.

Um agente racional percebe o ambiente por meio dos seus sensores, toma uma decisão baseado na sua função do agente e atua no ambiente pelos atuadores. Para que um agente seja considerado racional, é necessário definir o seu ambiente de tarefa.

Ele é formador por quatro componentes, o *Performance, Environment, Actuators,*

Sensors (PEAS) (RUSSELL, 2010). Definindo o *PEAS*, está sendo definido o ambiente de tarefa do agente.

O ambiente de tarefa de um agente mostra quais problemas o agente está tentando resolver, onde ele está resolvendo e como está tentando resolver.

- **Medida de Desempenho**: Que medidas de desempenho os agentes devem seguir como objetivos? As medidas de desempenho são importantes para que os agentes tenham objetivos para maximizar ou minimizar, ou seja, elas se referem a objetivos que se desejem alcançar de forma máxima ou de forma mínima;
- **Ambiente**: Quais ambientes os agentes irão atuar? O conhecimento do ambiente é importante para que o agente saiba onde ele está atuando;
- **Sensores**: Quais sensores estão disponíveis para que o agente possa perceber o ambiente? Os sensores são importantes para que o agente possa perceber o estado do ambiente e quais as consequências de suas ações no estado atual do ambiente;
- **Atuadores**: Como o agente pode interferir no ambiente? Os atuadores são responsáveis por realizar as ações tomadas pelo agente no ambiente e, assim, causar possíveis mudanças no estado atual do ambiente.

Para o ambiente em que o agente atua, algumas classificações serão necessárias para a definição (RUSSELL, 2010):

- *Completamente observável vs parcialmente observável*: Se o agente tem total acesso ao estado do ambiente em um determinado instante, ele é completamente observável, caso contrário, ele parcialmente observável;
- *Determinístico vs estocástico*: Se o próximo estado do ambiente é determinado pelo seu estado atual, ele é determinístico, caso contrário, se o próximo estado do ambiente não é determinado pelo seu estado atual, ele é estocástico;
- *Episódico vs sequencial*: Se o agente recebe uma percepção e executa uma ação única, independente de ações tomadas anteriormente e de estados futuros para aquela ação, o ambiente é episódico. Ou seja, a escolha da ação a ser tomada só se refere ao episódio da percepção. Caso contrário, se a escolha da ação leva em consideração ações já tomadas e possíveis consequências daquela ação em estados futuros, o ambiente é sequencial;
- *Estático vs dinâmico:* Se o ambiente puder alterar o seu estado enquanto o agente está tomando uma decisão de uma percepção referente ao estado mais recente, o ambiente é dinâmico, caso contrário, é estático, ou seja, o agente não precisa continuar a monitorar o

ambiente enquanto está tomando uma decisão;

- *Discreto vs contínuo:* Se um ambiente possui um número finito de estados distintos, ele é discreto. Caso contrário, é contínuo. O número de percepções e ações também pode ser levado em conta na classificação em discreto e contínuo;
- *Agente único vs multiagente:* se um ambiente possui mais de um agente, onde o comportamento de cada agente influencia na medida de desempenho de outros agentes, ele é multiagente. Caso contrário, ele é agente único. Em um ambiente de agente único, podem existir vários agentes no mesmo ambiente mas, se um agente não influencia na medida de desempenho de outro agente, ele é considerado agente único. Dentro da classificação em multiagente, ainda existe uma subclassificação importante para o contexto do trabalho, que é de *competitivo*, quando a maximização da medida de desempenho de um agente indica a minimização da medida de desempenho de outro agente e vice-versa. E a de *cooperativo*, que é quando a maximização da medida de desempenho de um agente indica a maximização da medida de desempenho de outro agente.

3.2 GAMIFICAÇÃO

A gamificação é capacidade de derivar uso da maneira de pensar, de mecânicas, diversão e vício dos jogos para outros contextos não relacionados a jogos para motivar pessoas a alcançar determinado resultado. A maioria dos sistemas são desenvolvidos focados no simples funcionamento, assumindo que os usuários irão usá-lo (GROH, 2012).

A gamificação traz o foco para os humanos, considerando que nem sempre eles se sentem motivados a cumprir suas tarefas e muitas vezes precisam de uma motivação, por isso ele tem como tem como base explorar comportamentos naturais do ser humano para alcançar seus objetivos (MUNTEAN, 2011).

Um exemplo básico de gamificação é premiar os usuários por realizar uma determinada tarefa. O motivo de se chamar gamificação é porque os jogos têm a capacidade de manter seus jogadores motivados por um longo período de tempo apesar das dificuldades que um jogo pode criar, e tem sido estudada e aplicada em diversas áreas com o objetivo de melhorar o compromisso dos usuários, experiências físicas e aprendizados (FLATLA et al., 2011).

3.3 SHELLTER

O Shellter (HACHEM, 2015) é uma rede social dedicada ao aprendizado de Segurança da Informação, idealizada e desenvolvida pelo Information Security Research Team (INSERT). O INSERT é um grupo de pesquisa em segurança da informação sediado na Universidade Estadual do Ceará (UECE).

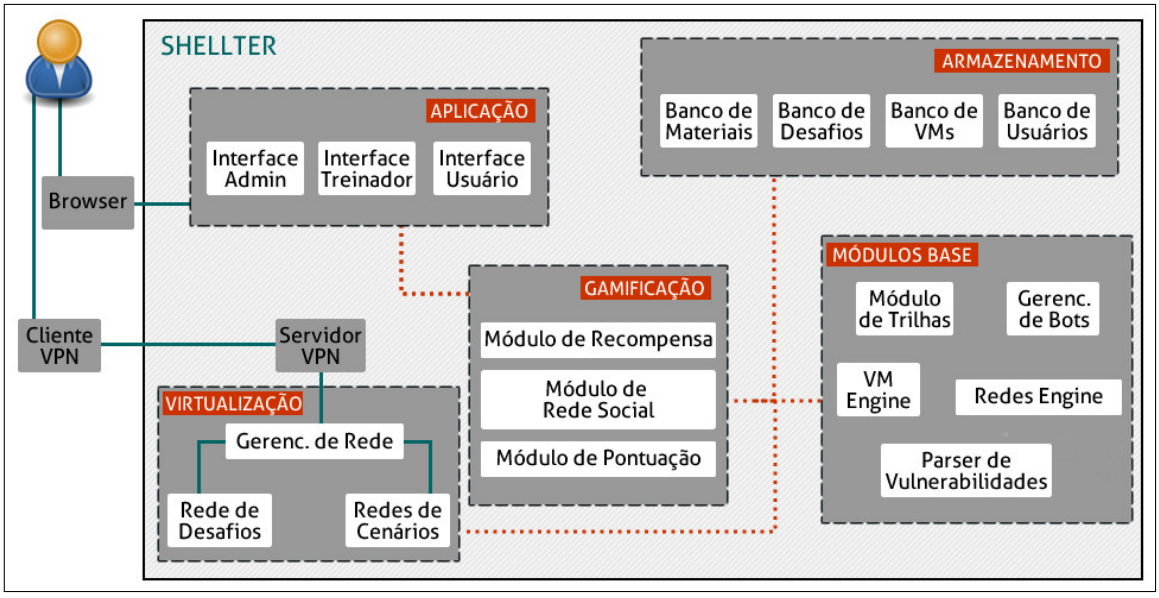
No Shellter, os usuários interagem do mesmo modo que usuários de jogos *multiplayer online*, formando times ou atuando individualmente em desafios para que, com o tempo, evoluam de *baby hacker* até *god hacker*. Em um ambiente controlado na nuvem, laboratórios e simulações virtuais com desafios de diferentes habilidades, juntamente com técnicas de gamificação, o Shellter forma um conjunto para o contínuo aprendizado em segurança.

O Shellter é um ambiente flexível e de possibilidades ilimitadas. Usuários de diferentes habilidades em áreas como sistema operacional, serviços web, programação e criptografia podem ser combinados em times para se desafiarem em competições modeladas conforme competições Capture the Flag (CTF).

Instrutores podem definir trilhas com intuito de capacitar alunos em uma ou mais áreas. Usuários podem buscar conhecimento em tutorias, fóruns, wikis e cursos disponibilizados pelo Shellter e testar habilidades individualmente.

Na Figura 3 temos a arquitetura do Shellter:

Figura 3 – Arquitetura Shellter



Fonte – (HACHEM, 2015).

Por meio da arquitetura do Shellter, é possível gerenciar usuários, cenários de simulado e sistemas computacionais, materiais para estudos e métricas para medir o desempenho de usuários, com o uso de técnicas de gamificação.

No Shellter, um usuário se conecta ao sistema de duas formas: por um *browser* ou um cliente Virtal Private Network (VPN). A interface fornecida pelo *browser* se conecta à camada de aplicação, onde o usuário pode acessar diversas funcionalidades: desafios, trilhas de ensino, formação de times para aprendizado colaborativo, materiais de estudos, gerenciamento de ambientes simulados, entre outros. Tudo isso de acordo com as permissões dele.

A camada de gamificação intermedeia os dados vindos dos módulos base, realizando os processamentos necessários (computando pontos, desempenho, medidas), para que possam ser armazenados de maneira correta na camada de armazenamento. Esses dados são apresentados ao usuário na camada de aplicação.

Os módulos bases também se comunicam diretamente com a camada de virtualização: os módulos VM Engine, Gerenciador de BOTs e Redes Engine são necessários ao gerenciamento e execução dos cenários de treinamento e simulação da camada de virtualização. O acesso do usuário pelo cliente VPN leva diretamente a essa camada, onde ele pode realizar exercícios simulados.

No Shellter, o aprendizado *hands-on* é maximizado. Os usuários terão acesso a materiais teóricos para estudo e a um demonstrativo prático do que está no material. Por exemplo, um usuário iniciante quer aprender sobre *buffer overflow*, uma técnica de engenharia reversa.

Ele acessa uma página com o conteúdo iniciante sobre o assunto, que fala sobre pilha de processos em um sistema operacional, já que este assunto é um pré-requisito para *buffer overflow* e usa como exemplo um código na linguagem de programação C.

Nessa mesma página, existirá um terminal Bourne Again SHell (BASH), simulando um sistema Unix, com funções limitadas, restrito a ferramentas que auxiliam o aprendizado prático do usuário naquele assunto específico. Ele será desafiado a reproduzir o exemplo dado. Assim, quando o usuário resolver o desafio da questão, irá passar para outro nível e ganhar pontuação pelo seu feito.

Os outros níveis irão seguir a mesma linha de raciocínio com diferentes tipos de desafios, questões, exemplos, proporcionando ao usuário vários desafios que o induzam a pensar e desenvolver suas expertises. Toda a evolução do usuário nos mais diferentes níveis de desafios, seus ganhos, acertos, conquistas, áreas de afinidade estarão no seu perfil no Shellter. Esse perfil

será responsável por mostrar as estatísticas do usuário em todo seu histórico de evolução do aprendizado.

Usuários também poderão testar suas habilidades, assim como adquirir novas no ambiente de guerra cibernética simulada do Shellter (Shellter's Cyber Warfare). Esse ambiente visa a proporcionar uma experiência real de uma rede de computadores.

Por meio de técnicas de virtualização, será possível simular diferentes cenários de ataque, defesa e ataque/defesa. O objetivo é proporcionar um ambiente real para o aprendizado *hands-on*.

No ambiente simulado, os usuários poderão jogar em equipe contra equipe:

- Atacando o ambiente da outra equipe;
- Defendendo o seu próprio ambiente;
- Atacando o ambiente da outra equipe e defendendo o seu próprio ambiente ao mesmo tempo.

Assim como essas equipes poderão jogar sozinhas contra os Agentes Inteligentes no Shellter (Shellter's Agents).

4 SISTEMA DE AGENTES INTELIGENTES PARA AMBIENTES SIMULADOS

Deixar os usuários motivados para aprender, motivados para superar os seus limites, motivados a buscar novos conhecimentos, motivados a buscar novas técnicas, é o objetivo do *Sistema de Agentes Inteligentes para Ambientes Simulados (SAI)*. Os agentes irão interagir com os usuários para mantê-los com vontade de aprender, querendo explorar a segurança da informação, assim como outras áreas da computação, já que a segurança da informação é uma área meio, ou seja, envolve direta ou indiretamente todas as áreas da computação.

Os agentes serão responsáveis por interagir com os usuários durantes os exercícios simulados. Durante esses exercícios, os agentes precisam auxiliar o aprendizado, assim como incitar o afloramento das expertises dos usuários. Para isso, existirão diferentes categorias de agentes, onde também serão classificados os usuários.

Para classificação em categorias, os agentes precisam de classificadores para uma taxonomia pré-definida. Essa taxonomia é baseada, inicialmente, em dados de conhecimento do usuário. Esses dados são originados das mais diferentes fontes, entre elas, por exemplo as redes sociais.

Com o uso de técnicas de Data Analysis, o histórico de conhecimento dos usuários será analisado para que a taxonomia possa ser construída. Essas técnicas são responsáveis por inspecionar, filtrar, canonicalizar e modelar os dados com o objetivo de agrupar usuários em diferentes categorias.

A partir dessa classificação inicial, será possível escolher o agente mais compatível com o perfil do usuário. Assim, o usuário terá um jogo mais justo e consequentemente um aprendizado otimizado, pois não serão escolhidos agentes mais fortes ou mais fracos.

Segundo (SALEN, 2003), um jogo é um sistema no qual os jogadores participam de um conflito artificial, definidor por regras, que ao final possui um resultado quantificado. Baseado nisso, o termo jogo é introduzido no *Sistema de Agentes Inteligentes*, pois, seguindo a definição apresentada, os usuários interagem com o sistema simulado em busca de vencer desafios e no final receber uma pontuação e/ou premiação por isso.

Agentes mais fortes, em relação ao nível de conhecimento do usuário, podem desestimular, pois irão facilmente atacar/defender ambientes e ganharão o exercício/jogo com facilidade. Agentes mais fracos não irão desafiar os usuários, fazendo com que os conhecimentos aplicados se tornem repetitivos e massivos.

Esses dois tipos de agentes desestimulam os usuários, já que deixaram o jogo muito

difícil ou muito fácil. O objetivo é que o usuário evolua gradualmente e com o tempo possa encarar desafios e cenários considerados difíceis.

A partir da escolha inicial dos agentes que competirão com os usuários, uma segunda fase se faz necessária: a evolução dos agentes durante o jogo. Os agentes precisam acompanhar a evolução do aprendizado dos usuários no ambiente simulado, já que estes estarão constantemente sendo desafiados a ampliar seus conhecimentos.

Esse acompanhamento é importante para que as habilidades dos usuários estejam sendo enfatizadas, colocadas à prova e que eles estejam estimulados para aprender novos conceitos, novas técnicas, novas ferramentas, assim como desenvolver suas próprias aptidões.

Nessa segunda fase, os agentes irão utilizar técnicas de inteligência artificial para que possam aprender em tempo real e assim cumprir uma série de objetivos:

- Evoluir de acordo com o usuário: os agentes, por meio de sensores, precisam capturar os dados de desempenho do usuário durante o jogo e assim decidir se, dentro da categoria a qual agente/usuário pertencem; devem elevar o nível da sua abordagem ou baixar o nível da abordagem ou permanecer no mesmo nível de agente. Deve ser considerada também a substituição do agente por outro agente de categoria superior ou categoria inferior, caso as mudanças de desempenhos dos usuários sejam consideráveis;
- Evoluir de acordo com o ambiente: o agente precisa mudar sua abordagem, sua tática, seus vetores de ataque/defesa de acordo com o que usuário for modificando o ambiente do jogo. Por exemplo: um usuário está treinando técnicas de defesa em um ambiente simulado. Existe um agente realizando técnicas de ataque de acordo com as vulnerabilidades presentes no ambiente. O usuário corrige uma das vulnerabilidades do ambiente, assim o agente de ataque precisa identificar essa correção, verificar se alguma nova vulnerabilidade surgiu com a correção da vulnerabilidade antiga, e refazer seus vetores de ataque e sua estratégia para ataque ao ambiente.

4.1 METODOLOGIA

Para que os objetivos deste trabalho sejam alcançados, a seguinte metodologia se faz necessária:

- Inicialmente será definida a arquitetura do *Sistema de Agentes Inteligentes para Ambientes*

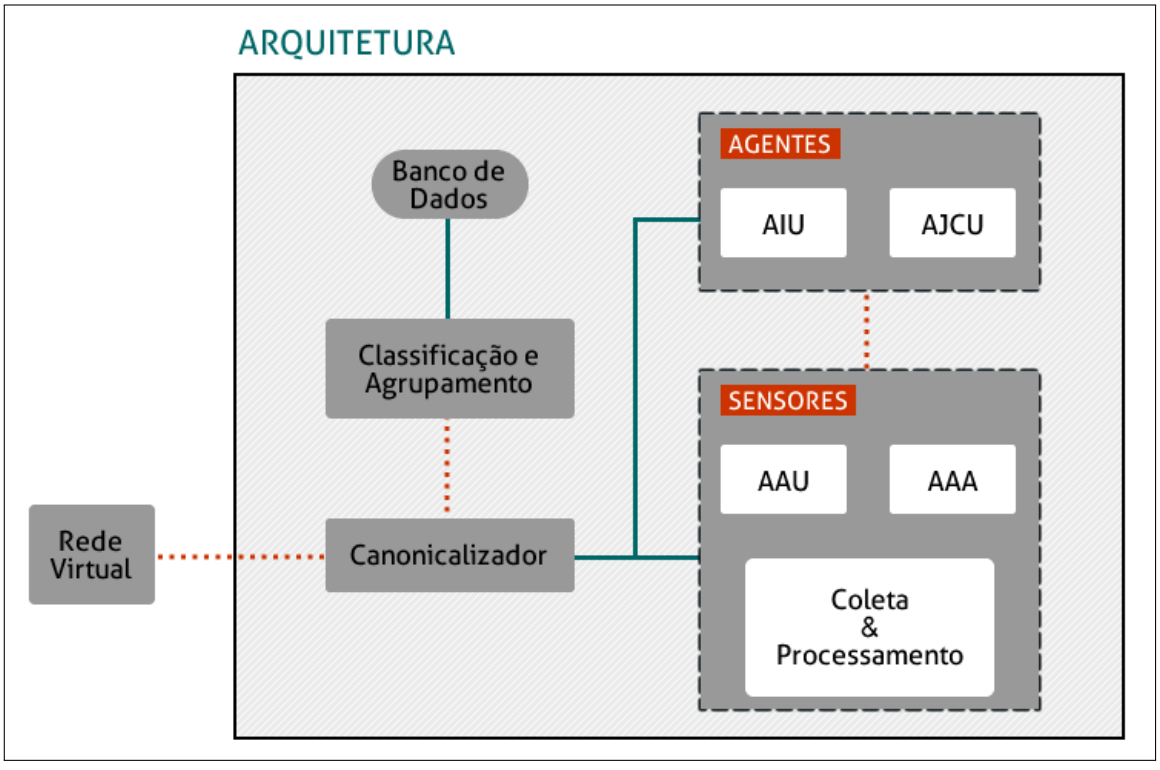
Simulados (SAI), detalhando cada um dos componentes e suas interações. Essa fase é necessária para que o escopo global da atuação dos agentes seja apresentado, assim como a importância de cada entidade para a arquitetura proposta;

- Em seguida, definir cada agente que faz parte do sistema. A definição da agente é importante para que a racionalidade de cada um deles seja justificada e o ambiente de tarefa de cada um deles sejam apresentadas. Nessa definição, será determinado o *PEAS*, Seção 3.1, de cada agente;
- Por fim, será implementado um dos agentes do *SAI*. Nessa etapa, para o agente escolhido, será definido o programa de agente e serão realizados testes e sua validação. O agente a ser implementando será o Agente responsável por Aprender sobre as Mudanças no Ambiente (AAA). Esse agente foi escolhido por possuir metodologia simular à dos outros agentes do trabalho. Ele será formalizado nas próximas sessões.

4.2 ARQUITETURA PROPOSTA

A Figura 4 mostra a arquitetura proposta para o Sistema de Agentes Inteligentes.

Figura 4 – Arquitetura Sistema de Agentes Inteligentes



Fonte – Elaborada pelo autor.

4.2.1 Classificação e Agrupamento

Na arquitetura proposta, a *engine* de Classificação e Agrupamento (CeA) é responsável por classificar os usuários nos mais diferentes níveis para que os agentes certos sejam escolhidos para interagir.

Inicialmente, antes de entrar no ambiente simulado, será pedido que o usuário compartilhe seus dados de uso do *Facebook*. Por meio desse compartilhamento, serão capturados os dados de interesse do usuário, além de uma análise do conteúdo de postagem dos usuários no que remete à segurança da informação.

Os dados de interesses se referem a páginas que o usuário curtiu e segue no *Facebook*. Desses dados, pode-se descobrir quais os principais interesses do usuário dentro da segurança da informação, devido ao conteúdo temático e postado pelas páginas.

Adicionalmente, as postagens do usuário ajudam a refinar a descoberta dos interesses. Elas mostram quais conteúdos eles estão interessados em compartilhar, além da interação com outros usuários e grupos de usuários. Além do *Facebook*, será pedido que o usuário informe suas contas nas mais variadas redes sociais de computação.

Por meio dessas redes sociais, é possível capturar dados de conhecimento do usuário pela participação, contribuição, compartilhamento e desenvolvimento nos mais diferentes projetos oferecidos por esses sites. Entre essas redes sociais, podemos mencionar *GitHub*, *Stackoverflow* e *Topcoder*.

O *GitHub*, por exemplo, pontua os seus usuários pela criação e participação em projetos *opensource*. O *Stackoverflow* oferece um espaço para que sejam feitas perguntas relacionadas a computação e pontua pela relevância da pergunta, assim como, pontua as respostas mais relevantes. O *Topcoder* oferece diferentes desafios de programação.

Além disso, o perfil de usuário do Shellter é analisado, ou seja, todo o aprendizado e desempenho nos outros ambientes do Shellter, pontos fortes e pontos fracos de cada membro são levados em conta na hora da classificação do usuário.

Por outros ambientes do Shellter, entendem-se as trilhas de ensino, questões de eventos de CTF disponíveis, minicompetições do Shellter, fóruns de discussão e quaisquer novos ambientes de ensino e aprendizado que possam ser desenvolvidos no Shellter.

Os dados coletados nesses ambientes são importantes para uma maior precisão na classificação de usuários em relação aos agentes que serão escolhidos para interagir no ambiente.

A *engine* de Classificação & Agrupamento irá utilizar técnicas de análise de dados

para escolher os agentes mais adequados. Após a escolha do nível dos agentes de interação, o usuário e sua equipe ganham acesso ao ambiente virtual e uma rede de monitoramento e comunicação entre os agentes se inicia.

4.2.2 Agente responsável por Aprender Sobre a Evolução do Usuário

Na *engine* de sensores, o Agente responsável por Aprender Sobre a Evolução do Usuário (AAU) monitora todas as atividades dos usuários. Esse monitoramento serve para identificar experiências afetivas e de emoção que influenciam no processo de aprendizado, uma vez que o nível de engajamento e o grau de motivação, medidas dessas experiências, afetam o processo de aprendizado (LESTER EUN Y. HA, 2013).

Este agente irá utilizar técnicas de computação afetiva para cumprir suas atividades. A computação afetiva é uma subárea da inteligência artificial, que reúne conhecimentos da psicologia e da ciência cognitiva. Por meio dela, é possível estudar como os sistemas computacionais podem detectar, classificar e responder às emoções humanas (FROZZA ANDREA APARECIDA KONZEN DA SILVA, 2011).

Diversos trabalhos (LESTER EUN Y. HA, 2013), (LONGHI MAGDA BERCHT, 2007), (FROZZA ANDREA APARECIDA KONZEN DA SILVA, 2011), (RODRIGUES, 2005) mostram técnicas de como modelar emoções para medir o grau de engajamento e motivação. Algumas emoções consideradas para medir o engajamento são ansiedade, tédio, confusão, curiosidade, animação, foco e frustração. No desenvolvimento desse agente, serão levantadas técnicas adequadas para esse tipo de modelagem.

Para que essas emoções sejam capturadas no ambiente, o agente utilizará *plugins* para perceber as ações dos usuários. Os *plugins* utilizados serão:

- *Plugin* de teclado;
- *Plugin* de vídeo;
- *Plugin* de áudio;
- *Plugin* de conteúdo.

O *plugin* de teclado é responsável por capturar todas as entradas de teclado executadas pelo usuário. Essas entradas informam a frequência de interação do usuário com o ambiente, acesso a diferentes desafios, número de tentativa de respostas, tempo de respostas, entre outros. *Keyloggers*, que são ferramentas que capturam entradas digitadas por um usuário, podem ser utilizadas neste *plugin*.

O *plugin* de vídeo captura as expressões faciais, gestos, postura e quaisquer outras expressões corporais do usuário enquanto utiliza o ambiente simulado. Essas expressões capturadas irão ajudar na identificação dos sentimentos expressados pelo usuário durante o uso do ambiente virtual.

Muitas expressões são capazes de traduzir emoções e sentimentos sem usar palavras, e a face é considerada o principal elemento das expressões, ou seja, os principais elementos da comunicação humana (FROZZA ANDREA APARECIDA KONZEN DA SILVA, 2011).

O *plugin* de áudio irá capturar os sons emitidos pelo usuário, entonação de voz e se ele está escutando alguma música no momento do exercício. A música favorece o raciocínio, evoca sentimentos e tem a capacidade de mudar estados de ânimo, atingindo as dimensões cognitiva e afetiva do ser humano (ROSAS, 2010).

Assim, será questionado ao usuário o seu gosto musical, qual estilo de música escuta para estudar, qual estilo de música escuta para momentos de distração, entre outros. Essas informações serão utilizadas para identificação do engajamento do usuário.

Por fim, o *plugin* de conteúdo será responsável por analisar o que o usuário está acessando durante o uso do ambiente simulado. Por acesso, entendem-se as páginas *webs* que estão sendo acessadas, os materiais que estão abertos, etc.

O objetivo é identificar se o usuário está focado em resolver os desafios do *SAI*. Se ele estiver utilizando o ambiente e estiver jogando um jogo ao mesmo tempo, por exemplo, o AAU irá identificar isso como um baixo grau de foco do usuário.

4.2.3 Agente responsável por Aprender sobre as Mudanças no Ambiente

Assim como o AAU, o Agente responsável por Aprender sobre as Mudanças no Ambiente (AAA) também faz parte da *engine* de sensores. Ele é responsável por monitorar as mudanças no ambiente computacional causadas pelo usuário e pelos agentes que interagem com os usuários. Esse agente será implementado neste trabalho.

Essas mudanças são necessárias para se fazer as mais diversas técnicas de ataque e defesa que muitas vezes modificam arquivos, abrem/fecham portas, mudam configurações, etc. O AAA tem como responsabilidade identificar essas mudanças e as suas relações com técnicas conhecidas de segurança da informação, assim como aprender novas técnicas.

Para que o AAA possa identificar as técnicas, ele irá utilizar a taxonomia de classificação, apresentada na Seção 5.1, e em seguida uma catalogação de técnicas conhecidas de

segurança da informação. Neste trabalho, foi catalogado um conjunto de técnicas na Seção 5.2. Assim como, as novas técnicas aprendidas por esse agente serão classificadas de acordo com a taxonomia pré-definida, seção 5.2.

Para capturar os dados, o AAA irá atuar por intermédio de *plugins* e *keyloggers*. Os *keyloggers* serão necessários para capturar dados de entrada de usuários e agentes, e os *plugins* são necessários para monitorar o uso e a modificação de elementos computacionais.

Essas modificações se referem a modificações em elementos computacionais virtuais: máquinas virtuais, serviços, aplicações, diretórios, arquivos, links virtuais, switchs, roteadores, firewall, configurações, etc. E por uso dos elementos computacionais se entende o acesso a recursos, como realizar somente a leitura de um arquivo de configuração.

O AAA irá possuir os mais diferentes tipos de *plugins* para as mais diferentes plataformas computacionais. Como o *SAI* atua em um ambiente virtual, inúmeros sistemas operacionais, aplicações e *appliances* de redes podem ser virtualizados.

Assim, os *plugins* serão desenvolvidos de acordo com a demanda dos cenários do ambiente virtual. Por exemplo, existirão *plugins* para ambientes *Linux Distribuições Debian*, *Arch Linux*, *FreeBSD*, *Windows*, *switchs Cisco*, entre outros. Os diferentes *plugins* são necessários para que as diferentes configurações de cada elemento sejam atendidas.

Por meio dos *plugins* e dos *keyloggers*, o AAA irá capturar dados para identificar técnicas utilizadas pelos usuários, com o objetivo de captar conhecimentos e experiências.

As informações coletadas e processadas pelo AAU e pelo AAA são passadas ao Agente responsável por Jogar Contra o Usuário (AJCU) e ao Agente responsável por Interagir com o Usuário (AIU).

4.2.4 Agente responsável por Jogar Contra o Usuário

O AJCU tem como responsabilidade realizar ações de ataque e defesa baseadas nas informações coletadas pelos agentes sensores. Essas informações permitem ao agente formular estratégias para jogar com o usuário/equipe:

- Quais técnicas utilizar para ataque/defesa em relação ao conhecimento do usuário/equipe;
- Quais técnicas utilizar para ataque/defesa em relação à evolução do usuário/equipe;
- Quais técnicas utilizar para ataque/defesa em relação às modificações no ambiente computacionais;
- Quais técnicas utilizar para ataque/defesa em relação às diversas combinações das relações

mencionadas.

Para que ele possa executar suas ações, serão necessários *plugins* para executar entradas junto ao ambiente e verificar as consequências dessas ações no ambiente:

- *Plugin* computacional;
- *Plugin* de redes.

O *plugin* computacional permite que o agente execute ações junto aos mais diferentes sistemas operacionais, com base nas estratégias formuladas, e verifique se as ações foram bem executadas e surtiram os efeitos esperados. O *plugin* de redes tem a mesma função do *plugin* computacional, mas é voltado para dispositivos de redes.

Esses *plugins* irão permitir que este agente seja análogo a um usuário do ambiente, podendo executar quaisquer movimentos que um usuário possa executar, por exemplo, executar comandos em *shell*, executar comandos em um *prompt*, utilizando e modificando recursos, clicando em ícones, etc.

4.2.5 Agente responsável por Interagir com o Usuário

O AIU também interage com o usuário com base nos dados coletados pelos agentes sensores, entretanto, o seu objetivo se difere do AJCU. O AIU tem como objetivo interferir no ambiente virtual para motivar os usuários, incentivá-los a adquirir novos conhecimentos, entre outras ações.

Ele atua como um tutor do usuário. Acompanhando-o durante as suas atividades e no cumprimento das suas tarefas procurando mantê-lo motivado, mesmo com as dificuldades encontradas na realização destas (KAMPFF ANTôNIO DA FONSECA DE LIRA, 2005).

Por meio dos dados dos agentes sensores, esse agente identificará se os usuários estão ansiosos, entediados, confusos, curiosos, animados, focados, frustrados, etc. (FROZZA ANDREA APARECIDA KONZEN DA SILVA, 2011).

A partir dessa identificação, o agente realizará alguma ação: seja no ambiente, seja diretamente junto ao usuário ou ambos. Essas ações podem ser dicas de como cumprir determinadas tarefas, *wizards* de ações para serem seguidas pelos usuários, modificações de configurações, indicação de materiais de estudos, indicação de tutoriais, entre outras.

Para atuar no ambiente, ele irá possuir os mesmos *plugins* do AJCU para que possa tutorear o usuário diretamente no ambiente. Por meio dos *plugins* ele poderá executar *softwares* próprios, voltados à tutoria do usuário (KAMPFF ANTôNIO DA FONSECA DE LIRA, 2005).

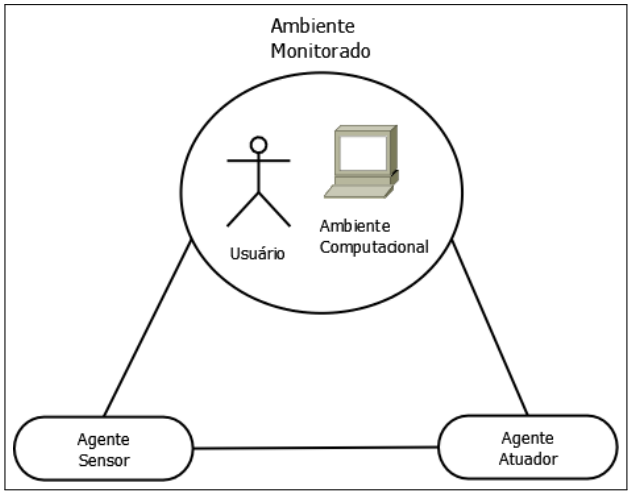
A *engine* de Coleta e Processamento tem a responsabilidade de capturar os dados do usuário no ambiente simulado para que o desempenho nesse ambiente seja agregado ao seu perfil do Shellter. Esses dados são importantes para que o usuário consiga maiores pontuações, premiações e outros benefícios no Shellter.

O canonicalizador tem a responsabilidade de canonicalizar os dados de uma entidade para a outra, ou seja, colocar o dado em padronização com o protocolo seguido por cada entidade da arquitetura. Por exemplo: dados de aprendizado passados do AAU para o AJCU, dados passados da C&A para AIU, dados passados da C&A para AJCU.

4.3 INTERAÇÃO ENTRE AGENTES

A escolha de agentes para a resolução do problema da motivação e acompanhamento de usuários em ambientes simulados, se deu, pois, esses são capazes de detectar mudanças no ambiente em que estão inseridos, raciocinar sobre essas mudanças e atuar proativamente executando tarefas para reduzir os efeitos negativos dessas na motivação. Para esta abordagem, os agentes tem a missão de perceber o ambiente em que estão inseridos e agir junto ao usuário e ao ambiente computacional para manter a motivação. A Figura 5 mostra o relacionamento entre agentes, usuários e ambiente computacional.

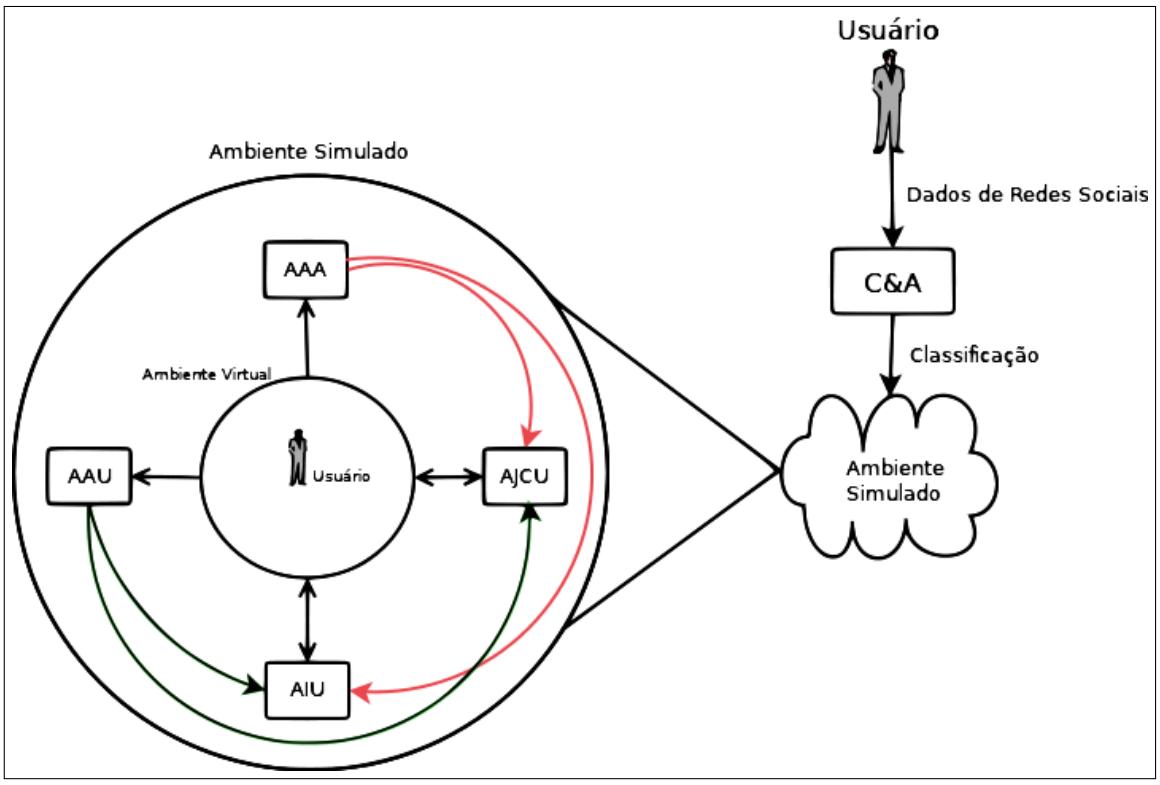
Figura 5 – Relacionamento Agentes - Usuário - Ambiente Computacional



Fonte – Elaborada pelo autor.

A troca de informações entre os agentes sensores, agentes atuadores e ambiente monitorado é uma atividade constante para se alcançar os objetivos do sistema. A Figura 6 mostra o fluxo de interação entre os agentes do *SAI*.

Figura 6 – Fluxo de comunicação entre agentes



Fonte – Elaborada pelo autor.

O usuário, antes de entrar no ambiente virtual, fornece os seus dados das redes sociais para que a Classificação e Agrupamento possa classificá-lo. O CeA passa a classificação para os agentes do *SAI*, e o usuário ganha acesso ao ambiente simulado.

No ambiente simulado, o usuário estará enfrentando os mais diferentes desafios de segurança da informação. Enquanto está no ambiente, os agentes sensores monitoram as atividades do usuário. O AAU monitora as emoções e expressões do usuário para medir o grau de engajamento e motivação, enquanto o AAA procede da mesma maneira com as modificações e uso do ambiente computacional para identificação de técnicas de segurança da informação.

As informações coletadas, analisadas e processadas pelos agentes sensores são passadas para os agentes atuadores. O AJCU joga contra o usuário, aplicando técnica de segurança da informação a partir das informações repassadas, assim como verifica se as técnicas estão sendo corretamente aplicadas.

Por outro lado, o AIU utiliza essas informações para buscar e aplicar as melhores formas de motivar o usuário, seja fornecendo materiais de estudos, ensinando técnicas, etc. Ele também verifica se o usuário demonstra interesse pelo material que é passado na tutoria do agente.

4.4 DEFINIÇÃO DOS AGENTES

Para definir os agentes que fazem parte da arquitetura do *SAI*, é necessário definir a racionalidade de cada um deles, ou seja, o *PEAS* (RUSSELL, 2010) de cada um deles.

4.4.1 AAU - Agente responsável por Aprender sobre a evolução do Usuário

O Agente responsável por Aprender Sobre a Evolução do Usuário atua no monitoramento das atividades dos usuários a fim de identificar o grau de motivação e engajamento no ambiente simulado. Assim, para esse agente, temos o seguinte *PEAS*.

4.4.1.1 Medida de desempenho

As medidas de desempenho que norteiam esse agente são:

- Maximizar a identificação do engajamento e comprometimento dos usuários;
- Maximizar a identificação do grau de motivação dos usuários.

Para isto, ele se utilizará de técnicas que captem as emoções passadas pelos usuários no ambiente computacional. Essas emoções captadas são utilizadas para definir o grau de engajamento e comprometimento do usuário com o ambiente.

A partir dessa captura, o grau de motivação do usuário poderá ser identificado. Vários trabalhos (LESTER EUN Y. HA, 2013), (LONGHI MAGDA BERCHT, 2007), (FROZZA ANDREA APARECIDA KONZEN DA SILVA, 2011), (RODRIGUES, 2005) mostram como definir e identificar o grau de motivação do usuário em ambiente computacional. As medidas referentes ao grau de engajamento e de comprometimento e ao grau de motivação serão definidas em trabalhos futuros.

4.4.1.2 Ambiente

O ambiente em que este agente atua é o usuário, ou seja, as ações que o usuário toma no ambiente de simulação. O estado do ambiente é formado pelo conhecimento, experiências e emoções que influenciam nas ações dos usuários. Esse estado pode ser modificado à medida que os usuários são desafios no ambiente de simulação.

Por exemplo, um usuário pode ter conhecimento e experiência de nível mediano em segurança da informação, mas se sente desmotivado e infeliz com os desafios, que ele considera muito fáceis, de um cenário no ambiente de simulação.

O AAU, identificando isso, comunica-se com os outros agentes, e os desafios do cenário mudam para um nível mais apropriado para o usuário. Em um primeiro instante, o usuário continua não se sentindo feliz, mas fica motivado a resolver os novos desafios do cenário.

Assim, esse ambiente é ***parcialmente observável***, pois o agente não possui total acesso ao estado do ambiente. Por exemplo, o agente não consegue perceber todas as emoções sentidas pelo usuário. A identificação das emoções de pessoas de personalidades mais fechadas serão um desafio maior.

Estocástico, pois o próximo estado do ambiente não é determinado por seu estado atual. Por exemplo, o usuário pode estar se sentindo empolgado após conseguir resolver uma série de desafios.

Em seguida, ele começa a se sentir desanimado após um desafio que não consegue resolver. Existe uma incerteza sobre os estados futuros, isto é, o conhecimento sobre os estados futuros é probabilístico.

Sequencial, porque a escolha de qual classificação o agente atribui para o usuário não é atômica, ou seja, ela depende de conhecimentos, experiências, emoções prévias dos usuários. Classificar o usuário também influencia em futuras classificações.

Por exemplo, um usuário classificado previamente como pouco engajado, pouco comprometido e pouco motivado pode ser direcionado a desafios que foquem na importância da segurança da informação e em como estudar a área pode ser interessante. Assim, o AAU considera uma sequência de percepções para tomar uma ação.

Dinâmico, pois o usuário adquire novos conhecimentos, novas experiências, novas percepções, ou seja, ele pode se alterar durante a deliberação do agente. Assim, o AAU precisa monitorar o ambiente durante uma tomada de ação.

Contínuo, pois os mais diferentes usuários, com as mais diferentes personalidades, conhecimentos, formações, etc. têm número infinito de estados distintos. Assim, não é possível contabilizar todos os estados que os usuários podem assumir no ambiente.

Multiagente, já que a medida de desempenho do AAU é formada por múltiplos objetivos, estes podem ser divididos para agentes menores, cada um responsável por um único objetivo. Esses agentes menores são cooperativos, pois a medida de desempenho de cada um deles maximiza a medida de desempenho dos outros agentes menores.

Em resumo, o AAU possui um ambiente *parcialmente observável*, *estocástico*, *sequencial*, *dinâmico*, *contínuo* e *multiagente*. Essa configuração de ambiente é considerada a

mais complexa de se lidar (RUSSELL, 2010).

4.4.1.3 Sensores

As percepções captadas por esse agente serão as ações dos usuários. Exemplos de ações consideradas pelo AAU:

- Frequência de interação com o ambiente;
- Tempo de respostas;
- Número de tentativas de resposta;
- Expressões faciais;
- Conteúdo acessado pelo usuário;
- Frequência de acesso a diferentes desafios.

Essas ações serão capturadas por *plugins* de teclado, vídeo, áudio e conteúdo. Esses *plugins* irão capturar as ações realizadas pelo usuário durante o uso do ambiente virtual.

4.4.1.4 Atuadores

Como o objetivo deste agente é maximizar identificações, ele não irá atuar diretamente junto ao ambiente. Ele atuará na classificação do usuário em relação ao grau de engajamento e comprometimento e em relação ao grau de motivação. Ou seja, a ação do AAU é a classificação do usuário.

Essas classificações influenciam o ambiente na escolha de desafios propostos, metodologias de ensino, entre outros, já que as informações colhidas e processadas por este agente serão repassadas para os outros agentes que interagem com o usuário. A partir disso, esses outros agentes irão utilizar as decisões do AAU para tomar suas próprias decisões de como atuar nos seus respectivos ambientes de tarefa.

Como exemplo, para este agente, podemos considerar um usuário que vem apresentando um bom desempenho ao resolver desafios em um cenário de ataque a uma rede doméstica. Entretanto, quando o usuário se depara com um servidor DNS BIND9, ele não consegue evoluir na tarefa de comprometer esse servidor. As suas técnicas de invasão no servidor começam a ficar repetitivas, mesmo que das outras vezes não tenham surtido efeito e o tempo de tentativa de resolução do desafio já esteja muito elevado.

O AAU, após uma série de percepções mais detalhadas, percebe que o usuário está ficando menos engajado e comprometido. Com isso, classifica a motivação do usuário em estado

decrescente e a transfere para o agente que interage com o usuário (AIU) para motivá-lo.

4.4.2 AAA - Agente responsável por Aprender sobre as mudanças no Ambiente

O Agente responsável por Aprender sobre as Mudanças no Ambiente monitora as mudanças do ambiente computacional causadas pelos usuários e pelos agentes que jogam com o usuário. Além disso, identifica as técnicas de segurança da informação utilizadas pelos usuários. Essas técnicas são formadas por um conjunto de passos que envolvem acesso e modificações de recursos.

4.4.2.1 Medida de desempenho

As medidas de desempenho desse agente são:

- Maximizar a identificação de alterações realizadas pelo usuário no ambiente simulado;
- Maximizar a identificação de alterações realizadas por agentes que jogam com os usuários no ambiente simulado.

Essas modificações se referem a modificações em elementos computacionais virtuais (máquinas virtuais, serviços, aplicações, diretórios, arquivos, etc.) e em elementos de redes virtuais (links virtuais, switchs, roteadores, firewall, configurações, etc.).

Também identificará o uso desses elementos. Por uso se entende como acesso a recursos. Realizar somente a leitura de um arquivo de configuração, por exemplo.

Por meio de modificações e o uso dos elementos virtuais, o AAA irá identificar técnicas utilizadas pelos usuários, com o objetivo de identificar conhecimentos e experiências. Além de tudo, também irá aprender novas técnicas de segurança da informação utilizadas pelos usuários.

4.4.2.2 Ambiente

O ambiente em que este agente atua são os elementos computacionais virtuais e os elementos virtuais de rede. Os estados que este ambiente pode assumir são os estados em que estão arquivos, diretórios, configurações, etc., ou seja, todos os recursos que compõem esse ambiente.

Esse ambiente é ***completamente observável***, pois o agente poderá identificar todos os acessos aos recursos de todos os elementos virtuais. Isso se dá, pois o AAA irá possuir total

acesso ao controle e gerenciamento do ambiente virtual.

Estocástico, pois o próximo estado do ambiente não é determinado pelo estado atual do ambiente. Por exemplo, um usuário que está defendendo um ambiente e muda uma página de uma aplicação web para protegê-la de ataques de *SQL Injection*.

Entretanto, isso não quer dizer que o atacante não poderá identificar outra falha de *SQL Injection* na mesma aplicação. Existe uma incerteza sobre os estados futuros, isto é, o conhecimento sobre estados futuros é probabilístico.

Sequencial, como o agente tem a missão de identificar técnicas de segurança da informação, sequência de ações, onde uma depende da outra, caracterizam as mais diferentes técnicas de ataque/defesa. E, como o estado do ambiente é considerado pelo estado dos seus arquivos, configurações, etc., o ambiente é sequencial.

Dinâmico, porque o ambiente será constantemente alterado, fazendo com que uma ação que o AAA esteja prestes a executar não se torne mais válida devido a novas mudanças no estado do ambiente, independentes do estado atual do mesmo.

O ambiente é ***Contínuo***, apesar de possuir um número finito de arquivos. A combinação de estados que os arquivos podem assumir é inviável de se contabilizar em uma base de dados, pois ocuparia muito espaço e se tomaria muito tempo de processamento.

Multiagente, já que a medida de desempenho do AAA é formada por múltiplos objetivos, estes podem ser divididos para agentes menores, cada um responsável por um único objetivo. Esses agentes menores são cooperativos, pois a medida de desempenho de cada um dos objetivos maximiza a medida de desempenho dos outros objetivos.

Em resumo, temos um ambiente *completamente observável*, *estocástico*, *sequencial*, *dinâmico*, *contínuo* e *multiagente*.

4.4.2.3 Sensores

Os sensores utilizados pelo AAA serão na forma de *plugins*, que, instalados em cada um dos elementos virtuais, identificam as alterações e os usos dos recursos, fazendo com que o agente perceba o ambiente. Assim como uma base de dados de técnicas conhecidas de segurança da informação será utilizada para identificação das técnicas utilizadas pelos usuários.

Caso seja identificada uma nova técnica, a classificação interna desse agente irá aprender essa nova técnica, catalogá-la e classificá-la. A classificação será de acordo com a taxonomia interna do agente, seção 5.1. Em seguida, essa técnica será adicionada ao banco de

dados de novas técnicas aprendidas pelo AAA.

Esse banco de dados de novas técnicas aprendidas é importante, pois servirá de contribuição do *SAI* para a comunidade de segurança da informação. Serão desenvolvidos e implantados os procedimentos necessários no que diz a respeito à propriedade intelectual das novas técnicas utilizadas no ambiente.

4.4.2.4 Atuadores

O objetivo desse agente é maximizar identificações, por isso ele não atuará diretamente junto ao ambiente. Sua atuação será identificar mudanças no ambiente e técnicas, classificá-las, e repassá-las para os agentes que jogam e interagem com o usuário.

Essa classificação influencia nas técnicas de ataque/defesa a serem utilizadas pelos agentes que jogam com o usuário irão utilizar e na atuação dos agentes que interagem com os usuários.

Como exemplo para a atuação do AAA, temos um ambiente defendido pelo agente que joga com o usuário (AJCU), onde este protege o ambiente contra ataques a switchs virtuais *CISCO*.

Entretanto, um usuário atacante consegue entrar na rede por meio desses switchs, mesmo que o AJCU tenha utilizado suas melhores técnicas de defesa desse elemento de rede contra as técnicas de ataques já conhecidas.

O AAA percebe todos os movimentos do usuário, percebe que é uma nova técnica de ataque a switchs Cisco. Aprende a nova técnica e passa para a classificação. Essa nova técnica ficará disponível para uso do AJCU, após a classificação do nível da mesma.

4.4.3 AJCU - Agente responsável por Jogar Contra o Usuário

O Agente responsável por Jogar Contra o Usuário é o agente que joga com os usuários no ambiente simulado. Sendo responsável por atacar e defender o ambiente computacional virtual e o ambiente de rede virtual situado dentro de um cenário do ambiente simulado, ele poderá atuar nos seguintes contextos:

- Somente atacar o ambiente que os usuários estão defendendo;
- Somente defender o ambiente que os usuários estão atacando;
- Defender o ambiente que os usuários estão atacando e atacar o ambiente que os usuários estão defendendo ao mesmo tempo.

4.4.3.1 Medida de desempenho

As medidas de desempenho do AJCU têm como objetivo principal vencer o jogo contra os usuários. Para isso, diversos objetivos menores devem ser considerados:

- Menor tempo de ação;
- Melhor escolha de técnica;
- Melhor estratégia;
- Melhor execução de técnicas.

É importante ressaltar que, para cada um dos objetivos, o AJCU deve fazer suas escolhas de acordo com o nível do agente que está jogando contra o usuário.

Por exemplo, o AJCU não deve sempre escolher a melhor técnica já existente para proteger uma rede, independente dos usuários que estão jogando com ele. É preciso analisar qual técnica melhor se adapta ao nível do agente e ao dos usuários que estão no jogo, assim como o contexto do ensino que o cenário da simulação está inserido.

Em um cenário em que um usuário iniciante ataca uma rede, é necessário protegê-la, mas deixando algumas brechas. O objetivo é que os usuários iniciantes possam conseguir invadir a rede e tomar conhecimento da falha explorada, para que, nas suas atividades de proteção de rede, não seja executada a mesma falha.

As técnicas utilizadas por esse agente serão de ataque e defesa, dependendo do contexto do cenário do ambiente de simulação. Como consequência, técnicas de outras áreas da computação serão necessárias.

4.4.3.2 Ambiente

O ambiente em que este agente atua é o mesmo do AAA, ou seja, nos elementos computacionais virtuais e nos elementos de redes virtuais. Os estados considerados por esse agente também são os mesmos do AAA (estados dos arquivos, diretórios, configurações, etc., ou seja, todos os recursos que compõem esse ambiente).

Importante ressaltar que as ações dos usuários e do AJCU são nos elementos virtuais do ambiente. Não existe um contato entre esse agente e os usuários, ou seja, não existe uma ação do AJCU diretamente para o usuário, assim como não existe uma ação que o usuário faça que modifique o agente como *software* instalado no SAI.

Assim, o ambiente do AJCU possui as mesmas propriedades do ambiente do AAA:

completamente observável, *estocástico*, *sequencial*, *dinâmico*, *contínuo* e *multiagente*.

4.4.3.3 Sensores

As percepções desse agente, em sua maioria, vêm do AAA. A partir dos dados recebidos, que contêm informações sobre os usuários (técnicas utilizadas) e do ambiente virtual (modificações), o AJCU poderá escolher qual a melhor ação ou qual melhor conjunto de ação para tomar, a fim de maximizar sua medida de desempenho.

Os dados recebidos do AAA são de suma importância para que se possa escolher a técnica de ataque/defesa mais apropriada para ser aplicada no ambiente. A técnica escolhida será de acordo com o nível do agente e do usuário atuantes no ambiente, dados da *engine* de Classificação & Agrupamento.

Além dessas percepções, o AJCU também leva em consideração o cenário e o desafio para melhor tomar sua decisão. O AJCU poderá identificar se o usuário está em níveis acima ou abaixo do seu conhecimento e solicitar uma revisão da classificação utilizada para a *engine* de *Classificação e Agrupamento*.

O AJCU também usará sensores para identificar a correta criação, modificação e configuração de arquivos, criação e modificações de diretórios, ou quaisquer outras ações de modificação que ele faça no ambiente. As ações desse agente precisam ser revisadas junto ao ambiente para se tenha a certeza de que foram corretamente aplicadas.

4.4.3.4 Atuadores

O AJCU atua de forma dinâmica no ambiente: modificando arquivos, modificando configurações, abrindo e fechando portas, instalando e utilizando *softwares*, varrendo redes, analisando tráfego, etc. Ele irá atacar e/ou defender um ambiente virtual dependendo do cenário em que estiver inserido.

A ação ou o conjunto de ações tomadas pelo agente levará em conta as suas percepções. O banco de dados de técnicas conhecidas de segurança da informação e o banco de dados de técnicas aprendidas serão utilizados pelo agente. Um terceiro banco de dados também será utilizado, é o banco de dados de técnicas desenvolvidas.

Esse banco de dados se refere a técnicas criadas pelo próprio AJCU, após aprender com o uso das outras técnicas e a interação com o ambiente. É importante que esse agente esteja sempre tentando aprender e desenvolver novas técnicas de segurança da informação.

Novas técnicas desenvolvidas pelo próprio agente poderão ser ensinadas para os usuários. Esse aprendizado diferenciado contribuirá para a motivação dos usuários e para a contribuição do *SAI* para a comunidade de segurança da informação.

Para esse agente, podemos usar como exemplo um cenário em que um usuário de nível mediano ataca uma rede e um AJCU defende esta rede. O usuário faz uma varredura do perímetro da rede e identifica cinco portas abertas. Das cinco portas, duas delas estão com o nome e a versão do serviço desprotegidas. Ou seja, uma porta contém o **Serviço 1** na versão **3.4**, e a outra porta está rodando o **Serviço 2** na versão **1.3**.

O AAA percebe essa varredura e seus resultados, faz sua tomada de ação e passa para o agente jogador. O AJCU percebe a informação da AAA, percebe o cenário e o desafio do usuário (explorar os dois serviços) para tomar a sua decisão de qual ação tomar.

AJCU decide atualizar a versão do Serviço 1, para corrigir possíveis vulnerabilidades, e não faz nada em relação ao Serviço 2. O agente decidiu assim, pois com base no nível do usuário e nos serviços expostos, o Serviço 2 é mais difícil de explorar do que o Serviço 1.

Entretanto, a dificuldade de explorar o Serviço 2 está de acordo com o nível do usuário, com o nível do agente e com o nível do desafio proposto. Dessa maneira, o AJCU fez um movimento válido de defesa, fechando o serviço mais fácil e fora do nível do jogo, e deixando para o usuário a tarefa de expandir os seus conhecimentos para explorar a vulnerabilidade do Serviço 2.

4.4.4 AIU - Agente responsável por Interagir com o Usuário

O Agente responsável por Interagir com o Usuário interage com o usuário no ambiente virtual simulado incentivando-o a adquirir novos conhecimentos, explorar suas expertises, ensino técnicas de segurança da informação, entre outros. Esse agente é o responsável direto pela motivação do usuário no *SAI*.

Ele se difere do AJCU, pois interage diretamente com o usuário, fazendo sugestões, recebendo avaliações, dando dicas, ensinando técnicas, entre outras ações.

4.4.4.1 Medida de desempenho

O objetivo desse agente é motivar o usuário, assim a suas medidas de desempenho são:

- Maximizar o engajamento e o comprometimento do usuário;

- Maximizar a motivação do usuário.

Para isso, ele irá atuar como um tutor inteligente do usuário junto ao ambiente. Para alcançar esse objetivo, ele precisa considerar os seguintes objetivos menores:

- Maximizar o comprometimento do usuário;
- Maximizar o aprendizado do usuário;
- Maximizar a experiência do usuário;
- Maximizar a curiosidade do usuário;
- Entre outros.

Os objetivos menores se referem a emoções do usuário e serão definidos em trabalhos futuros.

4.4.4.2 Ambiente

O ambiente em que o AIU atua é o mesmo que o do AAU, isto é, o usuário. Considera também as ações que o usuário toma junto ao ambiente de simulação. O estado considerado também é o mesmo: conhecimento, experiência e emoções do usuário, ou seja, o que influencia nas ações do usuário no ambiente simulado.

Assim, as propriedades do ambiente do AIU são as mesmas propriedades do ambiente do AAU: *parcialmente observável*, *estocástico*, *sequencial*, *dinâmico*, *contínuo* e *multiagente*.

4.4.4.3 Sensores

Os dados passados pelo AAU e pelo AAA são utilizados na maioria das percepções do AIU. A partir desses dados, esse agente percebe o grau de engajamento, comprometimento e motivação do usuário, e assim pode tomar as ações necessárias para maximizar essas medidas de desempenho, se necessário.

O agente também percebe se o usuário seguiu as medidas para motivação passadas por ele. Alguns exemplos dessa percepção podem ser a finalização de um tutorial sugerido pelo agente, o aumento da pontuação do usuário em determinada área da segurança da informação após intervenção do agente incentivando o usuário a explorar melhor a área em questão (o que mostra engajamento e comprometimento), entre outros.

Esse agente também receberá avaliações do usuário sobre sua motivação no ambiente simulado por meio de questionários.

4.4.4.4 Atuadores

O AIU pode atuar de diversas maneiras para motivar o usuário:

- Sugerindo tutoriais de uso de ferramentas;
- Sugerindo matérias para estudo teórico e prático;
- Mostrando *wizards* de ações que podem ser tomadas no ambiente virtual;
- Realizando demonstrações de técnicas;
- Etc.

As ações do AIU são amplas e serão melhor definidas em trabalhos futuros. O importante é que essas ações sejam voltadas para a motivação do usuário. Ele deve saber qual a melhor forma de atuar para motivar de acordo com o desafio do cenário no ambiente simulado em que o usuário está apresentando níveis decrescentes de motivação.

Por exemplo, um usuário que está há muito tempo tentando realizar um desafio e se mostrando confuso ao repetir as mesmas ações sem sucesso diversas vezes. O AAU detecta esse cenário, e passa ao AIU para este analisar qual seria a melhor ação para que o usuário saia desse estado.

O AIU, por sua vez, analisa o estado do ambiente e o desafio alvo e escolhe qual melhor forma de motivar. Neste caso, a motivação é para o usuário buscar novos conhecimentos para resolver o desafio. Assim, ele pode sugerir os materiais para estudo corretos, para que o usuário adquira o conhecimento necessário para resolver o desafio.

É importante também que o AIU saiba a hora certa de agir. Ele deve levar em consideração, além das percepções passadas, o tempo correto para executar a ação. O usuário não deve receber ações demais ou ações de menos desse agente. O usuário deve aprimorar sua capacidade de ser proativo.

A Figura 7 mostra um resumo do *PEAS* dos agentes do *SAI*.

Figura 7 – *PEAS* agentes *SAI*

	Ambiente de Tarefa	Observável	Determinístico	Episódico	Estático	Discreto
AAU	Usuário	Parcialmente	Estocástico	Sequencial	Dinâmico	Contínuo
AAA	Elementos Virtuais	Completamente	Estocástico	Sequencial	Dinâmico	Contínuo
AJCU	Elementos Virtuais	Parcialmente	Estocástico	Sequencial	Dinâmico	Contínuo
AIU	Usuário	Parcialmente	Estocástico	Sequencial	Dinâmico	Contínuo

Fonte – Elaborada pelo autor.

5 IMPLEMENTAÇÃO DO PROTÓTIPO AAA

O agente do *SAI* que será implementado neste trabalho será o Agente responsável por Aprender sobre as mudanças no Ambiente - AAA. A escolha deste se justifica devido a sua importância para o funcionamento da arquitetura proposta, pois influencia para que todos os outros agentes e *engine* possam também atuar de forma efetiva.

Além disso, a arquitetura dos outros agentes do SAI são semelhantes à arquitetura do AAA.

A *engine* de Classificação e Agrupamento utiliza o AAA para classificar o usuário já que as ações que executa no ambiente, conhecimentos e experiências aplicadas servirão para futuras classificações. O AJCU utiliza os dados passados pelo AAA para analisar, planejar e executar a técnica de ataque e defesa no ambiente de simulação.

O AIU, por sua vez, atua tutoreando os usuários por meio dos dados passados pelo AAA e pelo AAU. O AAU sofre uma influência indireta do AAA, porque depende da classificação do usuário e do nível dos desafios presentes no ambiente.

Assim, é possível ver que o AAA tem uma grande influência na arquitetura do *SAI* e seu funcionamento.

Para a implementação desse agente, inicialmente será necessária a definição da taxonomia de classificação das técnicas de segurança utilizadas no ambiente e a catalogação destas. Em seguida, será definido o programa de agente, por meio da sua formalização, do desenvolvimento do programa do agente, testes e validação.

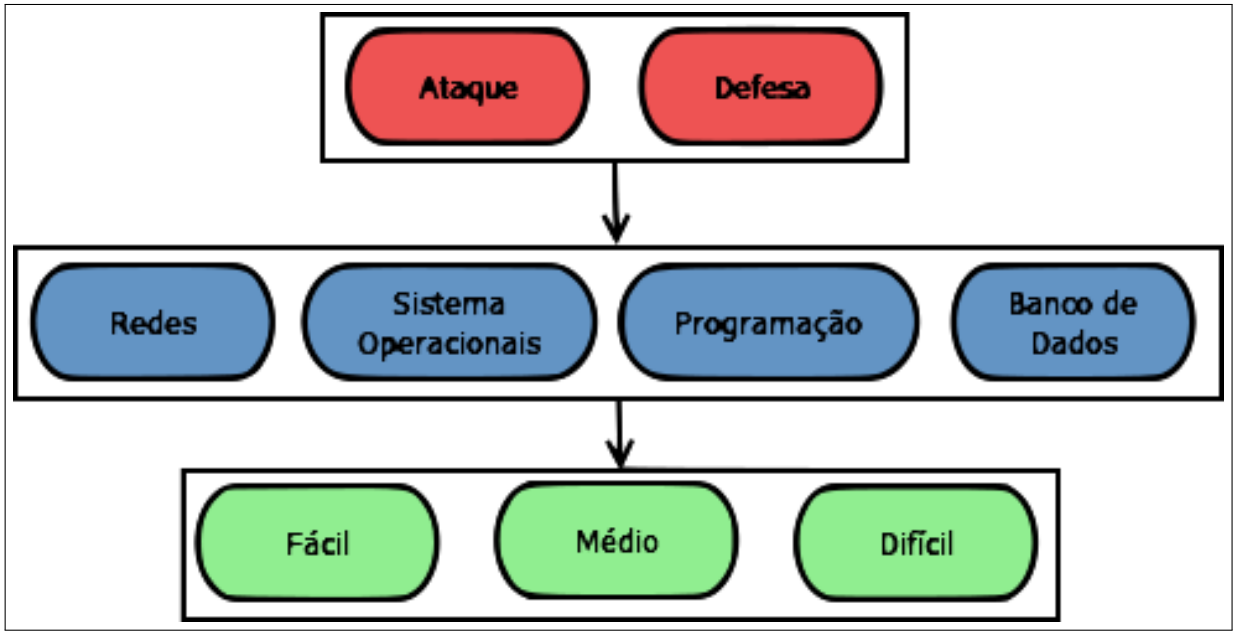
5.1 TAXONOMIA DE CLASSIFICAÇÃO DAS TÉCNICAS DE SEGURANÇA DA INFORMAÇÃO

A taxonomia utilizada para classificação é mostrada na Figura 8.

Inicialmente a técnica é classificada como técnica de ataque ou técnica de defesa. Essa classificação depende do objetivo do usuário ao utilizar a técnica: atacar ou defender um ambiente computacional. Independente de que, no desafio como um todo, o usuário tenha liberdade para atacar e defender o ambiente.

Se o usuário estiver em uma simulação em que precise defender o seu ambiente computacional e ao mesmo tempo atacar outro ambiente computacional, o AAA irá avaliar cada técnica do usuário em unicidade. Assim como em ambientes em que o objetivo seja ataque ou

Figura 8 – Classificação de Técnicas



Fonte – Elaborada pelo autor.

defesa.

Em seguida, a técnica é classificada de acordo com a área da segurança da informação em que está sendo aplicada:

- **Redes**: refere-se a redes de computadores, área da ciência da computação que estuda conexão entre os mais diferentes elementos computacionais. De browsers Web, telefones celulares, redes domésticas, redes empresariais, carros, redes de sensores, entre outros, as redes de computadores são uma ampla área com grande abrangência (KUROSE, 2005);
- ***Sistema Operacional (SO)***: refere-se à área da ciência da computação que estuda o principal *software* de um computador, aquele que controla todos os recursos do computador, além de prover a interligação de *hardware* e *softwares* que serão instalados no *SO*;
- **Programação**: refere-se a técnicas de programação de computadores e redes de computadores. Esta área refere-se a técnicas relacionadas à escrita de programas para controle e gerência de computadores e redes de computadores. Esses programas podem se apresentar na forma de *scripts*, sistemas, entre outros.
- **Banco de Dados (BD)**: refere-se à área da ciência da computação que estuda o armazenamento de dados. Técnicas de armazenamento, consultas, disponibilidade, segurança, entre outras. São técnicas que podem ser encontradas na área de banco de dados.

Por fim, a classificação em fácil, médio e difícil remete à divisão simples do nível das técnicas.

5.2 CATALOGAÇÃO DE TÉCNICAS

Para a catalogação das técnicas de segurança, será necessário criar um identificador para cada uma delas. Ele será formado pela letra inicial de cada um dos critérios da taxonomia em que ela está inserida. Além disso, cada combinação de critérios da taxonomia terá numeração própria, por exemplo:

- A primeira técnica catalogada como *Ataque - Redes - Difícil* terá o identificador *ARD1*. A segunda técnica classificada como *Ataque - Redes - Difícil* será identificada como *ACD2*. Assim como a décima técnica catalogada como *Defesa - Programação - Fácil* terá o identificador *DPF10*.

A ordem utilizada na numeração será de acordo com a ordem em que cada técnica for catalogada.

Além do identificador, cada técnica será associada a duas bases de dados consolidadas de segurança da informação, o *CVE* e o *EXPLOIT-DB*.

- **Common Vulnerabilities and Exposures (CVE)**: é um dicionário internacional de uso e conhecimento público sobre vulnerabilidades e exposições de segurança da informação. O *CVE* atua desde 2000 como referência na área. Ele se apresenta no seguinte formato: *CVE-2003-0132*. Um vazamento de memória no Apache 2.0.44 permite atacantes causar ataques de negação de serviço, por meio do consumo excessivo de memória;
- **Exploit-DB**: É um repositório de *exploits* e provas de conceitos (*POC*) desses *exploits* criado e mantido pelo Offensive Security, uma das maiores organizações de segurança da informação (SECURITY, 2015b). Um *exploit* é um pedaço de código que tenta comprometer um sistema computacional por meio de uma vulnerabilidade específica (FUTORANSKY et al., 2009).

As duas bases de conhecimento já são correlacionadas, pois o *Exploit-DB* já faz a ligação do *exploit* ao *CVE*. Por exemplo: o *exploit* de ID 9, *Apache HTTP Server 2.x Memory Leak Exploit* se refere ao *CVE-2003-0132*. Entretanto, nem todo *exploit* possui uma *CVE* e vice-versa.

Cada técnica catalogada no AAA será analisada e referenciada a um *exploits* e/ou a um *CVE*. A análise levará em conta se a técnica é de ataque ou defesa:

- **Técnicas de ataque:** o usuário irá utilizar um *exploit* ou um conjunto de passos relacionados a um *exploit* para atacar o ambiente;
- **Técnicas de defesa:** o usuário irá utilizar um conjunto de passos para proteger o seu

ambiente de uma vulnerabilidade referenciada por um *exploit* e/ou uma *CVE*.

Em ambas as situações de análise, novas técnicas poderão ser executadas pelos usuários, e, assim, o AAA irá aprender essas novas técnicas. Caso não seja encontrado um *exploit* e/ou um *CVE* para uma técnica catalogada, a ela será adicionada uma descrição. Com isso, a Figura 9 mostra os componentes da catalogação:

Figura 9 – Catalogação de Técnicas

Fonte – Elaborada pelo autor.

5.3 PROGRAMA DO AGENTE

O AAA tem uma arquitetura de *software* centralizada para maior gerenciamento dos seus *plugins*. Além de prover para eles escalabilidade e resiliência. Outro objetivo é diminuir o consumo de recursos dos *plugins* das máquinas clientes. Com isso, temos a arquitetura do AAA na Figura 10.

Da arquitetura é possível identificar duas entidades:

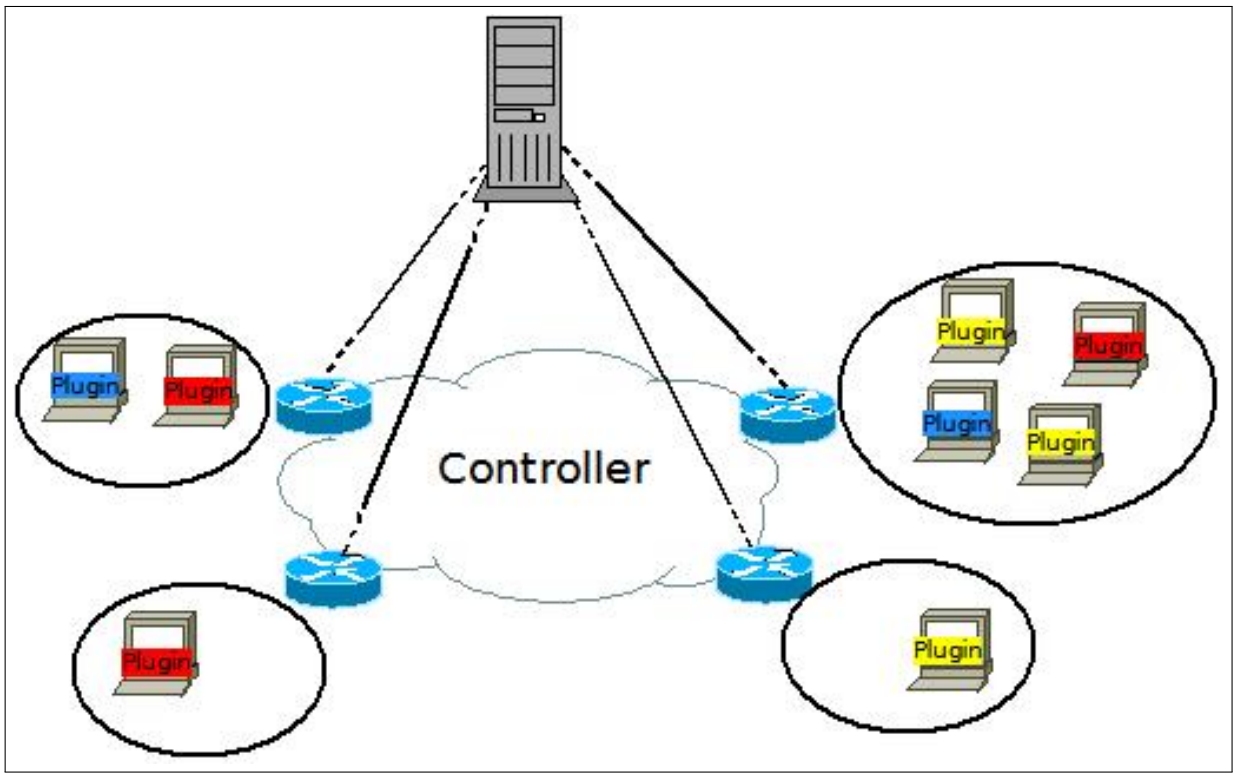
- ***Controller***: Responsável por toda a inteligência, gerenciamento de plugins, identificação de técnicas, processamento de informações e classificação de usuários;
- ***Plugin***: Responsável pela coleta de informação no ambiente computacional.

No *controller* é que estará implementada a racionalidade do AAA. Nele, serão recebidos os dados dos *plugins* (sensores), será tomada a decisão, baseada nos seus objetivos, e será aplicada uma atuação (classificação). Além disso, ele gerencia e distribui os *plugins* nos ambientes de acordo com a demanda.

O controle dos *plugins* se dará por uma tabela que irá controlar o endereço IP, sistema operacional e versão do ambiente e se os *plugins* de monitoramento estão ativos ou não. A cada novo ambiente monitorado, basta adicionar uma entrada na tabela, assim como para remover um ambiente do monitoramento basta marcar o *plugins* como inativo.

Adicionalmente, o *controller* monitora se um *plugin* está ativo ou não. Seja porque o

Figura 10 – Arquitetura de Software AAA



Fonte – Elaborada pelo autor.

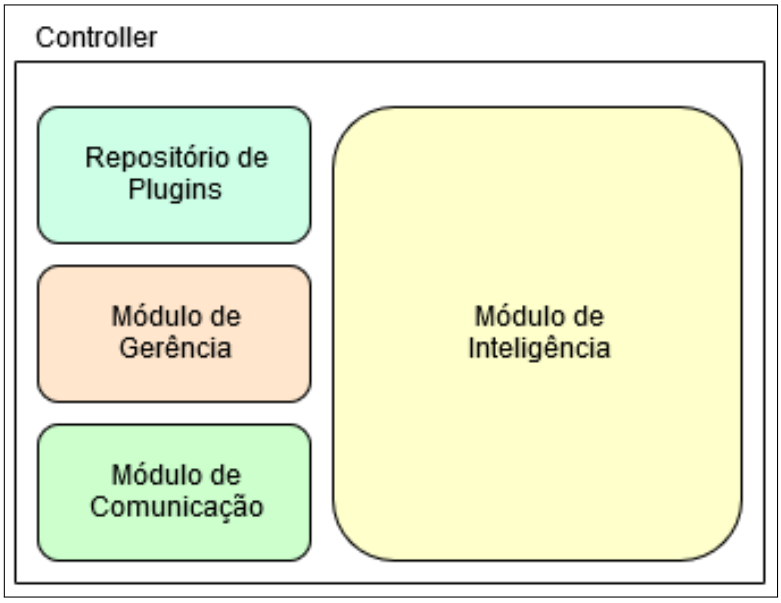
ambiente em que *plugin* está instalado foi encerrado ou por algum erro no próprio. Em caso de erros, um outro *plugin* é enviado e instalado no ambiente para substituição.

O *controller* possui três módulos principais, conforme a Figura 11. Esses módulos são responsáveis por todo o funcionamento do agente. Cada um deles será descrito a seguir.

- **Repositório de *Plugins***: é um repositório com os mais diferentes plugins direcionados para os mais diferentes ambientes. Cada ambiente, dependendo da sua plataforma, arquitetura e versão do sistema operacional terá o *plugin* mais adequado. Onde um *plugin* poderá ser utilizado para uma ou mais plataformas;
- **Módulo de Gerência**: tem como responsabilidade gerenciar os *plugins* nos mais diferentes ambientes. Controla a adesão, exclusão, ativação, desativação e configurações dos sensores. Neste módulo também estarão incluídas as informações da classificação das técnicas de segurança do usuário;
- **Módulo de Comunicação**: controla a comunicação entre os *plugins* e o controller. Recebe os dados coletados, envia novos sensores para os ambientes, envia comandos e a monitora a atividade dos mesmos;
- **Módulo de Inteligência**: é o responsável por interpretar e processar os dados vindo dos

sensores e tomar as decisões. Neste módulo estará a modelagem do agente.

Figura 11 – Arquitetura Controller



Fonte – Elaborada pelo autor.

Essas características permitem ao AAA prover as seguintes requisitos não-funcionais:

- **Escalabilidade**: Com a centralização e distribuição de *plugins* no *controller* o AAA pode prover uma maior escalabilidade de *plugins*;
- **Resiliência**: Como os *plugins* possuem características de monitoramento e colhimento de informações, deixando as tomadas de decisões e gerenciamento para o *controller*, eles podem facilmente ser substituídos sem causar prejuízos. Com isso, o AAA possui capacidade de cura em relação a alguma inatividade dos seus sensores;
- **Consumo de recursos:** Dado a divisão de papéis das entidades do AAA, o consumo de recursos nos ambientes de treinamento é minimizado. Como nos mesmos só é necessário a coleta de informações, não sendo necessário nenhum processamento em relação a elas, o AAA somente exige o consumo de recurso estritamente necessário nas máquinas clientes.

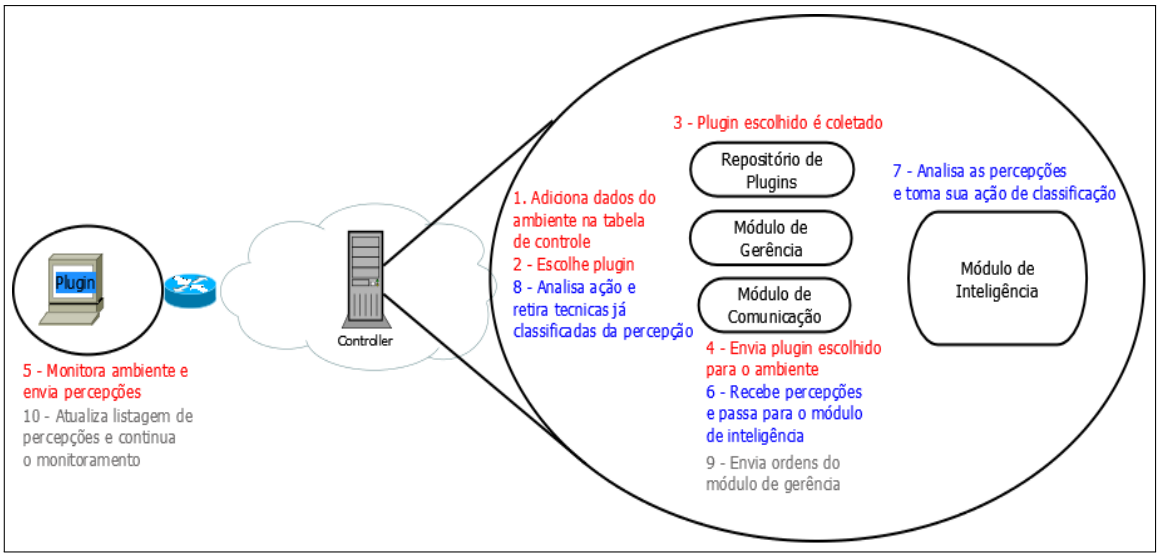
Assim, a Figura 12 mostra o funcionamento do sistema do agente em relação a coleta e processamento de dados. O *controller* envia o *plugin* adequado para um determinado sistema computacional para monitorar as técnicas utilizadas pelo usuário no ambiente. Para isso, inicialmente, o módulo de gerência do *controller* adiciona uma entrada na tabela de controle o endereço IP, sistema operacional e versão do ambiente alvo e escolhe o *plugin* adequado para isso no repositório de *plugins*.

O *plugin* é enviado para o ambiente através do módulo de comunicação, via *FTP*

ou *SCP*, onde começa a monitorar o ambiente. O *plugin* possui, na sua base de dados local, um conjunto de técnicas que ele irá monitorar. Essas são gerenciadas pelo módulo de gerência. Assim, o *plugin* envia para o módulo de comunicação as suas percepções através de requisições *Rest*. As percepções recebidas são passadas para o módulo de inteligência do *controller*, onde serão analisadas pelo programa, Seção 5.6, de formalização do AAA.

Após a atuação de classificação do usuário, o módulo de gerência verifica quais técnicas foram identificadas no ambiente. Caso alguma técnica seja identificada, o módulo de gerência irá comandar que o *plugin* não monitore mais aquela técnica em específico. Essa ação é tomada para minimizar o consumo de recursos do *plugin* no ambiente computacional.

Figura 12 – Funcionamento do sistema do agente



Fonte – Elaborada pelo autor.

5.4 DEFINIÇÃO INICIAL DE TÉCNICAS

O programa agente irá capturar os comandos realizados pelo usuário no ambiente simulado para que, baseado no seu catálogo de técnicas já existentes, possa identificar as técnicas de segurança da informação. O catálogo de técnicas consideradas para a validação deste agente será definido nesta Seção.

Foram escolhidas as seguintes vulnerabilidades para serem validadas:

5.4.1 Acesso root via SSH

Essa vulnerabilidade refere-se à configuração que permite acesso via Secure Shell (SSH) com o usuário *root* em ambientes Linux baseados em Debian.

Essa configuração caracteriza-se como uma vulnerabilidade, pois, como o usuário *root* é um usuário padrão em todas essas distribuições, técnicas de acesso ssh via *brute force* trabalham mais facilmente, pois focam em quebrar senhas deste usuário.

Ou seja, um atacante já possui a informação de que o usuário *root* é um usuário válido no sistema, assim só precisa focar em quebrar a senha deste usuário. Não se preocupando em achar usuários válidos em um determinado sistema.

Quebrar uma senha não é uma tarefa trivial, dependendo do tamanho e da complexidade da mesma. Entretanto, se o atacante tiver, também, que procurar usuários válidos para comprometer um sistema, aumentará muito o esforço gasto para a invasão. Além disso, o usuário *root*, por padrão, é o usuário com acesso total ao sistema operacional.

Por exemplo: um atacante possui o IP válido de servidor Linux alvo. Ele realiza uma varredura nas portas abertas nesse servidor e constata que a porta 22, referente ao protocolo SSH, está aberta. Assim, o mesmo tenta realizar uma conexão SSH com o usuário *root* e não recebe uma recusa de conexão, mas um pedido para entrar com uma senha válida para o usuário.

Assim o atacante identifica que o servidor alvo permite conexão SSH com o usuário *root* e pode aplicar as técnicas para quebra de senha no servidor utilizando esse usuário.

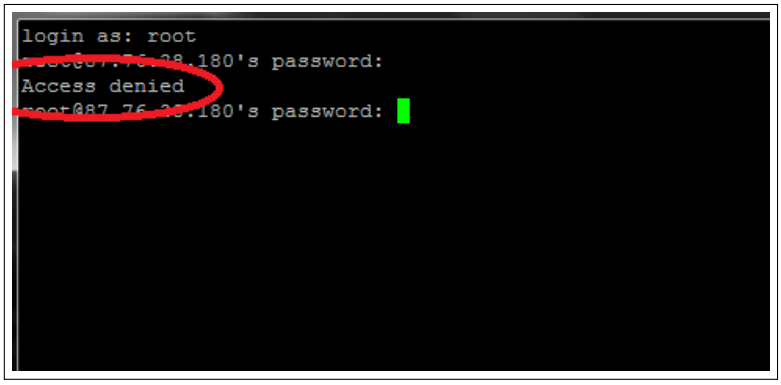
Não existe um *exploit* ou um *CVE* associado diretamente a essa vulnerabilidade, pois ela pode ser explorada de diferentes maneiras. Entretanto existem diversos *exploits* ou *CVEs* que utilizam essa vulnerabilidade como meio para conseguir o seu objetivo. Podemos mencionar os seguintes: *EDB-ID 17462*, *EDB-ID 387* e *EDB-ID 349 - CVE 2001-0144*

Essa técnica será analisada em dois cenários: ataque e defesa. Para o ataque, será considerada a descrição acima sobre o uso da técnica. Para a defesa, Figura 13, será considerado o usuário que seta a configuração para não permitir acesso do usuário *root* por SSH.

Para a identificação dessa técnica, o AAA irá monitorar o arquivo *sshd_config* localizado no diretório */etc/ssh/*. Dentro desse arquivo, o parâmetro *PermitRootLogin* será considerado, onde a configuração *yes* permite o acesso, e *no* não permite o acesso.

Esta técnica será classificada e catalogada como *ataque, redes, fácil - ARF1* ou *defesa, redes, fácil - DRF1*.

Figura 13 – Acesso SSH desativado para usuário *root*



5.4.2 Senhas fracas em banco de dados MySql e PostgreSql

Essa vulnerabilidade refere-se a utilizar senhas padrões ou senhas fracas em servidores de banco de dados. Para este trabalho, serão considerados os bancos de dados MySql (ORACLE, 2015) e PostgreSql (GROUP, 2015). Eles foram escolhidos por serem os bancos de dados *open sources* mais utilizados do mundo (ENGINES, 2015) até a data de publicação deste trabalho.

Utilizar senhas padrões facilita o trabalho de atacantes, pois os mesmos já sabem qual usuário e senha utilizar para obter acesso privilegiado. Essa falha aparece não só em banco de dados, mas também nos mais diferentes sistemas, *frameworks*, *API*s e quaisquer outros softwares que necessitem definir senhas padrões.

Assim como, utilizar senhas fracas também facilita o trabalho de atacantes, pois os mesmos podem utilizar dicionários de senhas para obter acesso privilegiado por meio de usuário padrão. Dicionários de senhas são listas de senhas mais utilizadas por diferentes usuários. Existem diversas fontes para se obter grandes e complexos dicionários de senhas (SECURITY, 2015a), (SCOWL, 2015).

Para o MySql, essa vulnerabilidade é explorada em *EDB-ID 19092 CVE 2012-2122*, além de diversos outros a utilizam para explorar outra vulnerabilidade. Entre os quais, podemos mencionar: *EDB-ID: 20718 CVE 2001-0407*, *EDB-ID 21725 CVE 2002-1809* e *EDB-ID 21726*.

Para o PostgreSql, ela é explorada e demonstrada em um artigo (LEIDECKER, 2015), publicado no *Exploit-DB*. Além disso, outros *exploits* utilizam a mesma para explorar outras vulnerabilidades, dentre os quais podemos mencionar: *EDB-ID 19875 CVE 2000-1199*, *EDB-ID 25076 CVE 2005-0245* e *EDB-ID 946*.

Essa técnica será analisada em dois cenários: ataque e defesa. Para o ataque,

será considerado o cenário em que o atacante tenta obter acesso a um dos bancos explorando a vulnerabilidade. E, para defesa, será considerado o cenário em que o usuário que está defendendo o ambiente altera a senha padrão e não utiliza uma senha fraca.

Para o MySql, o usuário padrão é o *root*, e a senha padrão é vazia, ou seja, sem senha. Para o PostgreSql, o usuário padrão é o *postgres*, e a senha padrão também é vazia. Com isso, o AAA identificará que essa vulnerabilidade existe ou foi consertada das seguintes maneiras:

- **MySql**: Inicialmente irá aplicar o comando *mysql -u root*. Se conseguir acesso ao banco, a vulnerabilidade existe. Se não, irá aplicar o comando *mysql -u root -p $dicionario_de_senhas*. O termo *$dicionario_de_senhas* indica o dicionário de senhas escolhidas para testar a vulnerabilidade.
- **PostSql**: Inicialmente irá aplicar o comando *sudo -u postgres psql postgres*. Se conseguir acesso ao banco, a vulnerabilidade existe. Se não, irá aplicar o comando *sudo -u postgres psql postgres -W $dicionario_de_senhas*. O termo *$dicionario_de_senhas* indica o dicionário de senhas escolhidas para testar a vulnerabilidade.

Essa técnica será classificada e catalogada como *ataque, so, fácil - ASF1* ou *defesa, so, fácil - DSF1*.

5.4.3 Banner Grabbing Apache Tomcat

Banner Grabbing refere-se a uma técnica em que se inicia uma conexão a um servidor ou serviço de rede e grava-se a informação que é retornada no início da conexão. Esses dados podem referir-se ao nome da aplicação, versão, sistema operacional, entre outros (FARROW, 2003).

O Apache Tomcat foi o servidor de aplicação mais utilizado no ano de 2015 (SALNIKOV-TARNOVSKI, 2015) para o Java, a linguagem de programação mais utilizada de 2015 (CASS, 2015).

Essa vulnerabilidade é explorada e demonstrada em (HARDENING. . . , 2009). Ela é referenciada pelos seguintes *exploits*: *EDB-ID 12343 CVE 2010-1157*, *EDB-ID 31551 - CVE 2005-4703*, *EDB-ID 21605*, *EDB-ID 20131 CVE 2000-0759*, *EDB-ID 21412 - CVE 2002-2006*, entre outros.

Para um atacante, existem muitas maneiras de se realizar um *banner grabbing* no *apache tomcat*. Uma delas é pelo comando *telnet*, por exemplo: *telnet ip_servidor_porta* e analisando o retorno da conexão feita. Outra forma é acessando uma página não existente da

aplicação, a página de erro padrão mostra a versão desse servidor.

Para um defensor do ambiente computacional, é possível evitar o *banner grabbing* alterando o arquivo *server.xml*. Nesse arquivo, ele deve adicionar o parâmetro *server="Banner"* na tag <*Connector* />. O que o defensor colocar como parâmetro será mostrado ao iniciar uma nova conexão.

Esta técnica será classificada e catalogada como *ataque, so, fácil - ASF2* ou *defesa, so, fácil - DSF2*.

Figura 14 – Banner Grabbing Telnet Apache Tomcat

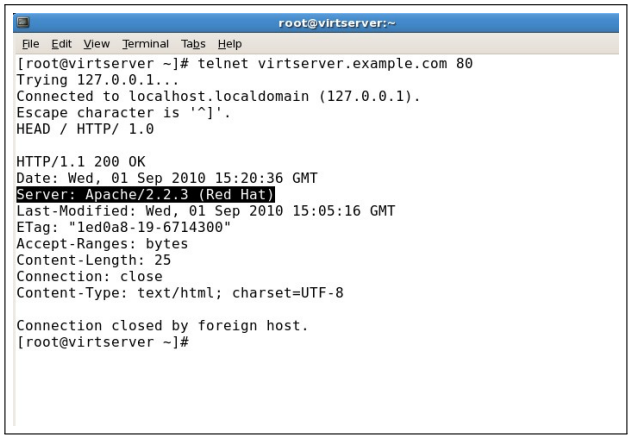
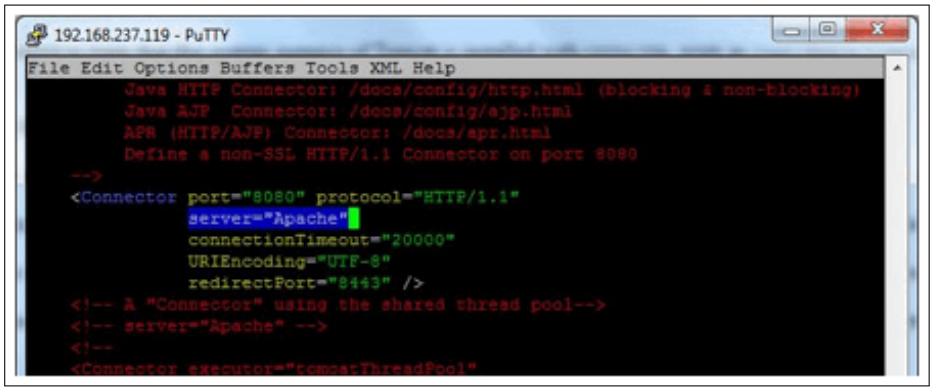


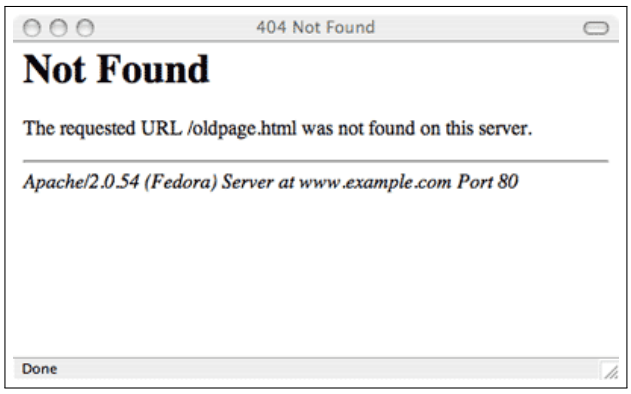
Figura 15 – Server.xml Apache Tomcat



Existe uma forma complementar e mais complexa para um defensor evitar o *banner grabbing*. No caso, para evitar que alguma informação seja extraída ao aparecer uma página de erro não tratada no *apache tomcat*. Para que isso seja feito, é necessário extrair o arquivo *ServerInfo.properties* de *catalina.jar*, que é um dos arquivos de compilação do servidor.

Uma das maneiras de fazer essa extração é por meio do comando: *jar xf catalina.jar org/apache/catalina/util/ServerInfo.properties*. Conforme a Figura 17, podem-se apagar as infor-

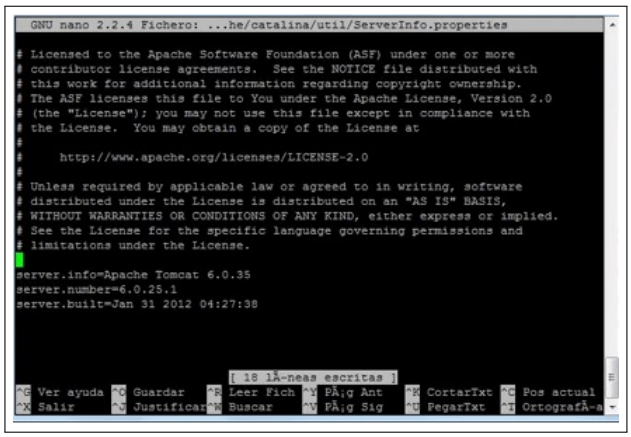
Figura 16 – Pagina de Erro Apache Tomcat



mações referentes a nome e versão do servidor. Após as alterações, é possível fazer a reintegração do arquivo pelo comando: *jar uf catalina.jar org/apache/catalina/util/ServerInfo.properties*.

Essa técnica será classificada e catalogada como *defesa, so, médio - DSM1*.

Figura 17 – ServerInfo Apache Tomcat



5.5 COMBINAÇÃO DE TÉCNICAS

O AAA não classificará os usuários apenas de maneira estática, ou seja, não classificará as técnicas do usuário a partir apenas de situações estáticas quando a alterar um arquivo de configuração, por exemplo. As técnicas utilizadas podem ser combinadas para formar uma técnica mais complexa e robusta.

Técnicas complexas de ataques e defesa, em termos gerais, são combinações de outras técnicas, simples ou complexas. Com isso, o AAA terá que identificar técnicas complexas a partir de combinações de técnicas simples. Para isso, ele se utilizará de uma árvore de decisões para realizar a combinação entre as técnicas.

Dessas combinações, é possível gerar novas classificações voltadas para subáreas da segurança da informação. As subáreas consideradas serão:

- **Crypto**: refere-se à criptografia, ou seja, o estudo de sistemas ”matemáticos” para resolver dois tipos de problemas de segurança: privacidade e autenticação. Um sistema com privacidade e autenticação evita a extração de informações por terceiros não autorizados a partir de mensagens transmitidas por um canal público, garantindo, assim, ao remetente que a mensagem será lida apenas pelo receptor pretendido (DIFFIE; HELLMAN, 2006);
- **Web**: refere-se à exploração de serviços *web*. Essa classificação abrangerá aplicações, servidores de aplicações, banco de dados, entre outros, desde que o elemento computacional alvo esteja dentro de um contexto web. O contexto web será considerando quando o usuário quiser comprometer uma aplicação web ou proteger a mesma, por meio de intervenções nos elementos que a compõem;
- **Forense**: refere-se a técnicas de computação forense. Nessa área, conhecimentos de coleta, organização, classificação e análise de evidências digitais são utilizados com o objetivo de investigar soluções para problemas relacionados a incidentes computacionais;
- **Reverse**: refere-se a técnicas de engenharia reversa. Esta, em segurança da informação, é um conjunto de técnicas de análise de *software* com o objetivo de entender seu funcionamento e seus componentes;
- **Hardenning**: refere-se a técnicas de blindagem de um sistema. Essa área ajuda a proteger um sistema computacional de usuários mal-intencionados que procuram brechas em um sistema, geralmente deixadas por informações e configurações padrões de *softwares* instalados no mesmo.

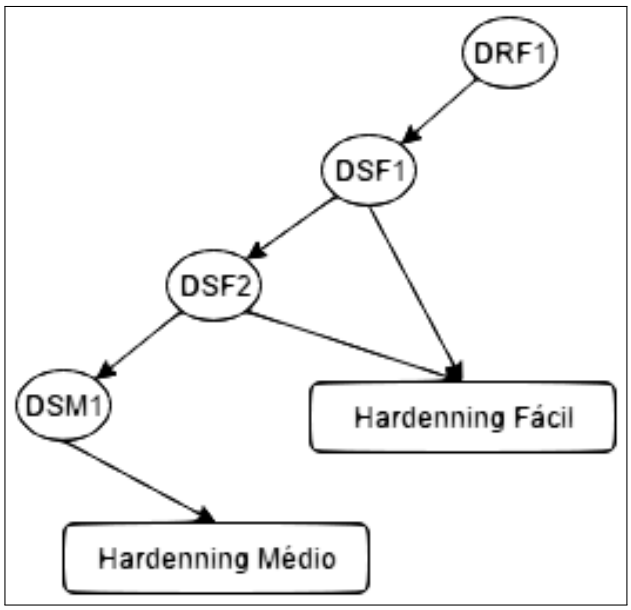
Das técnicas já definidas, podemos ter as seguintes árvores de decisões:

Observando a Figura 18, a partir da combinação das técnicas de defesa definidas neste trabalho, podemos ter uma nova classificação das técnicas utilizadas pelo usuário. Combinando as técnicas *DRF1 + DSF1 + DSF2*, o AAA identifica que o usuário está utilizando técnicas iniciais de blindagem de sistema e classifica essa combinação como *Hardenning Fácil*.

Desse modo, ainda na combinação anterior, se for adicionada a técnica DSM1, o AAA classifica essa nova combinação em *Hardenning Médio*. E, se o usuário se utilizar das técnicas *DRF1 + DSF1 + DSM1*, o AAA classificará a combinação em *Hardenning Fácil*, dado que a técnica *DSF2* não está mais presente na combinação.

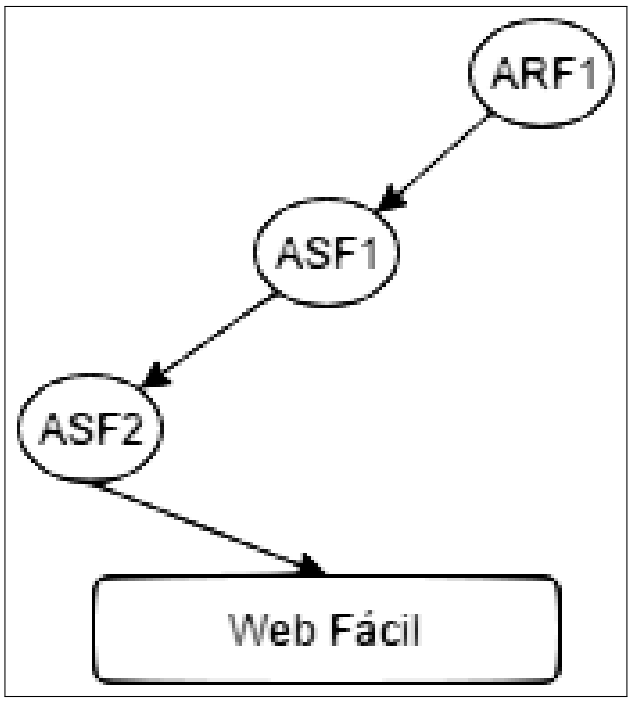
Essas classificações das combinações acima seguem a análise de técnicas complexas

Figura 18 – Árvore de Decisões - Defesa



de *hardenning* de sistemas computacionais. Com isso, cada subárea da segurança da informação que será utilizada neste trabalho será dividida nos seguintes níveis: fácil, médio e difícil.

Figura 19 – Árvore de Decisões - Ataque



Já na Figura 19, a partir das técnicas de ataque apresentadas, da combinação de *ARF1 + ASF1 + ASF2*, o AAA classificará as técnicas do usuário atacante como *Web Fácil*.

5.6 FORMALIZAÇÃO

Para a formalização do agente, utilizaram-se as técnicas definidas na Seção 5.4. A partir dessas técnicas, pode-se definir a inteligência do AAA. Inicialmente, foi desenvolvido um código para cada classificação de técnica catalogada por usuário. Com isso, para cada nova classificação um novo código associado deveria ser criado.

Entretanto, essa solução inicial não se mostrava escalável, além de dificultar a manutenção do código, pois para atualizações gerais seria necessário modificar todos os códigos existentes. Além do que, constatou-se que a gerência e a execução dessa solução seriam dificultadas, pois, para cada novo código, deveria ser adicionada ao controle do AAA uma nova chamada a ser executada e, para remoção de código, deveria ser removida uma chamada a ser executada pelo AAA. Assim, pensando em uma nova solução escalável, de fácil manutenção e gerência, desenvolveu-se a solução descrito no Algoritmo 1:

Algoritmo 1: Algoritmo de formalização AAA

```
Data: Lista de percepções - Dados coletados pelos plugins
      Lista de estados esperados - Percepção esperada para classificação do usuário
início;
for lista de percepção do
    for lista de estados esperado do
        if Cod percepção == Cod Percepcao esperada then
            if Percepção != Estado Esperado then
                classifica usuário como FALSE para técnica analisada;
            else
                classifica usuário como TRUE para técnica analisada;
            end
        end
    end
end
retorna lista de classificações do usuário;
```

Do código do Algoritmo 1, o AAA recebe as seguintes entradas:

- **Lista de percepções:** é uma lista de percepções por usuário recebidas dos *plugins*. Essa listagem não remete aos dados coletados diretamente pelos *plugins*. Eles são, inicialmente, tratados pelo AAA, no módulo de inteligência, para evitar erros e já vêm com a técnica

que será avaliada de acordo com determinada percepção;

- **Lista de estado esperados:** é uma lista com os estados esperados para a classificação do usuário em relação às técnicas que serão avaliadas.

E o AAA retorna a seguinte saída:

- **Lista de classificação do usuário:** classificação do usuário em relação às técnicas avaliadas.

A formalização proposta acima é escalável, pois o programa do agente é independente de novas técnicas que venham a ser catalogadas, assim como é independente de técnicas que venham a ser retiradas. Além disso, a manutenção e a gerência do programa do agente são facilitadas, dado que só é necessário realizar alterações em um único código.

Criou-se também a Figura 20, mostrando as técnicas catalogadas aqui, a partir dos seguintes dados:

1. Nome;
2. Classificação;
3. Estado Esperado para classificação.

Para a modelagem definida do AAA, seguem dois exemplos de cenários considerados. Esses dois exemplos são apresentados para facilitar o entendimento da modelagem definida:

5.6.1 Cenário 1

O plugin do AAA está monitorando três técnicas em um ambiente em que o usuário está atacante e defendendo. As técnicas monitoradas são: ARF1, ABF1 e DSF3. Com isso, o plugin envia as percepções para o *controller*. No *controller*, esses dados são recebidos pelo módulo de comunicação e passados para o módulo de inteligência, onde os dados são tratados para que possam ser analisados.

Após o tratamento, temos a seguinte listagem de percepções:

- **Arquivo de Percepção exemplo 1**:

  ARF1 - [ssh root@192.168.3.78 && ssh root@192.168.3.78 « $dicionario_de_senhas]

  ABF1 - [mysql -u root && mysql -u root $dicionario_de_senhas && sudo -u postgres psql postgres && sudo -u postgres psql postgres -W $dicionario_de_senhas]

  DSF3 - [server.info=” && server.number=”]

Para as percepções consideradas, temos os Arquivos de Estado Esperado:

- **Arquivo de Estado Esperado exemplo 1**:

Figura 20 – Definição de técnicas catalogadas de acordo com a formalização do agente

Nome	Classificação	Estado Esperado
Acesso Root SSH	Ataque, Redes, Fácil - ARF1	Usuário Atacante tentar um ataque usando o comando: ***ssh root@enderecoDoAlvo*** || ***ssh root@enderecoDoAlvo << $dicionario_de_senhas***
Acesso Root SSH	Defesa, Redes, Fácil - DRF1	Usuário Defensor modificiar o parametro ***PermitRootLogin***, do arquivo ***/etc/ssh/sshd_config***, para para ***NO***
Senhas e Usuários padrões + senhas fracas em BD MySql e PostGres	Ataque, Banco de Dados, Fácil - ABF1	Usuário tentar um ataque utilizando os comandos: ***mysql -u root*** || ***mysql -u root $dicionario_de_senhas ||*** ***sudo -u postgres psql postgres*** || ***sudo -u postgres psql postgres -W $dicionario_de_senhas***
Senhas e Usuários padrões + senhas fracas em BD MySql e PostGres	Defesa, Banco de Dados, Fácil - DBF1	Plugin não conseguir acesso utilizando os comandos: ***mysql -u root*** || ***mysql -u root $dicionario_de_senhas ||*** ***sudo -u postgres psql postgres*** || ***sudo -u postgres psql postgres -W $dicionario_de_senhas***
Banner Grabbing Apache Tomcat	Ataque, Sistema Operacional, Fácil - ASF1	Usuário tentar um ataque utilizando o comando: ***telnet endereco_do_servidor porta_do_servico*** Ou acessar ***páginas de erro do servidor***
Banner Grabbing Apache Tomcat	Defesa, Sistema Operacional, Fácil - DSF2	Usuário defensor modificar o parametro ***server="Banner_Desejado"*** na tag ***<Connector />*** no arquivo ***servers.xml*** do Apache Tomcat
Banner Grabbing Apache Tomcat	Defesa, Sistema Operacional, Fácil - DSF3	Usuário deve extrair o arquivo ***ServerInfo.properties*** de ***catalina.jar***, apagar informações do servidor Apache Tomcat e recompilar o ***catalina.jar. Os parametros considerados são: server.info='' && server.number=''***.

ARF1 - [ssh root@enderecoDoAlvo || ssh root@enderecoDoAlvo « $dicionario_de_senhas]

ABF1 - [mysql -u root || mysql -u root $dicionario_de_senhas || sudo -u postgres psql postgres || sudo -u postgres psql postgres -W $dicionario_de_senhas]

DSF3 - [server.info=" && server.number="]

Para as entradas acima, o AAA realiza a sua ação e toma a sua decisão. Com isso, seguem as atuações dadas pelo AAA para cada um dos casos:

- Atuação AAA exemplo 1:

  ARF1 -TRUE

  ABF1 - TRUE

  DSF3 - TRUE

5.6.2 Cenário 2

No segundo cenário, o AAA está monitorando outro ambiente, onde cinco técnicas estão sendo sensoriadas: ARF1, DRF1, DSF3, ABF1 e DSF2. O caminho das informações coletadas até o tratamento para as listas de entrada do AAA é o mesmo do cenário 1, dado que este é o procedimento padrão do AAA, ou seja:

- *Plugin* do AAA percebe, monitora o ambiente e envia as percepções para o *controller*;
- No *controller*, os dados são recebidos pelo módulo de comunicação e passados para o módulo de inteligência;
- No módulo de inteligência, os dados são tratados para que se possam gerar as listas de percepções.

Assim, é gerada a seguinte lista de percepções:

- **Arquivo de Percepção exemplo 2**:

ARF1 - [ssh root@192.168.3.78]

DRF1 - [PermitRootLogin NO]

ABF1 - [sudo -u userRandom psql bd_user_authentication

DSF3 - [server.info='Apache Tomcat 6.0.35' && server.number='6.0.25.1']

DSF2 - [<Connector ... server="Apache"... />]

Para as percepções consideradas, temos os Arquivos de Estado Esperado:

- **Arquivo de Estado Esperado exemplo 2**:

ARF1 - [ssh root@enderecoDoAlvo || ssh root@enderecoDoAlvo « $dicionario_de_senhas]

DRF1 - [PermitRootLogin NO]

ABF1 - [mysql -u root || mysql -u root $dicionario_de_senhas || sudo -u postgres psql postgres || sudo -u postgres psql postgres -W $dicionario_de_senhas]

DSF3 - [server.info=" && server.number="]

DSF2 - [<Connector ... server= ... />]

Para as entradas acima, o AAA realiza a sua ação e toma a sua decisão. Com isso, seguem as atuações dadas pelo AAA para cada um dos casos:

- Atuação AAA exemplo 2:

ARF1 - TRUE

DRF1 - TRUE

ABF1 - FALSE

DSF3 - FALSE

DSF2 - FALSE

6 VALIDAÇÃO DO PROTIPO AAA

Nesta Seção serão apresentados os testes feitos para a validação da formalização do AAA. Os testes irão validar a inteligência do AAA, ou seja, irão testar a capacidade do agente classificar as técnicas de segurança da informação catalogadas no seu banco de dados. Para este trabalho, serão utilizados nos testes as técnicas definidas na Figura 20.

Serão realizadas três baterias de testes:

- **Primeira bateria:** foram feitos testes abrangendo as combinações dos formatos de entradas aceitas por cada uma das técnicas da Figura 20. Ou seja, foram criadas entradas para testar a capacitada do agente classificar corretamente as técnicas utilizando um único parâmetro e classificar técnicas utilizando uma combinação de parâmetros, para as técnicas que assim permitirem.
- **Segunda bateria:** na segunda bateria, foram feitos os mesmos testes da primeira e foram adicionadas novas combinações de parâmetros para testar a capacidade do agente de lidar com diferentes situações.
- **Terceira bateria:** na terceira bateria de testes, foram feitos os mesmos testes da segunda, onde criou-se entradas para enganar a inteligência do agente.

A base de dados utilizada nos testes foi definida manualmente para permitir validações mais exaustivas.

6.1 AMBIENTE DE TESTES

O ambiente escolhido para testes foi o *Linux Mint 17.3 64 Bits*, pois refere-se a distribuição linux baseada em Debian. Para a codificação dos serviços que estarão rodando no ambiente para identificação das técnicas, será utiliza o *Python 3.4*.

O *Python* foi escolhido por se tratar de uma linguagem nativa nas distribuições recentes linux baseadas em Debian e possuir funcionalidades de scripts, ou seja, possuem o poder de executar um conjunto de tarefas pré-determinadas em diferentes ambientes, dentre eles sistemas operacionais.

Para a implementação das regras de atuação e aprendizado do agente será utilizado *Prolog*.

6.2 VALIDAÇÃO

Para validação do agente, inicialmente, fizeram-se testes com a sua formalização. Assim, foram criadas listagem simuladas de percepções do AAA. As percepções escolhidas foram relacionadas com as técnicas definidas neste trabalho, Seção 5.2. As listas utilizadas para simulação encontram-se no Anexo B. Assim como o código desenvolvido para a formalização do agente encontra-se no Anexo A.

Para os testes do agente, pensou-se em diferentes situações para que se tivesse efetividade, pois o AAA atuará em ambientes de treinamentos e encontrará diversas circunstâncias. Assim, é importante que o agente seja capaz de enfrentar essas adversidades. Por exemplo, para uma mesma classificação, criaram-se cinco diferentes situações.

Além disso, muitas classificações dependem da combinação de uma tupla de dados, tal qual a técnica DSF3, que depende da combinação da modificação de dois parâmetros. E outras técnicas podem ser executadas com uma variedade diferente de comandos, por exemplo, a técnica ABF1 pode ser executada de quatro diferentes maneiras, de acordo com o que foi definido.

6.2.1 Primeira Bateria

A Tabela 1 abaixo mostra os resultados da primeira bateria de testes da ação do agente considerando percepção utilizada, classificação esperada e resultado obtido. Da tabela de resultados, a primeira coluna indica as percepções utilizadas para a simulação. A segunda coluna indica o estado esperado para aquela percepção, ou seja, dado o seu conteúdo da mesma, a classificação esperada. E a terceira coluna indica o resultado dos testes com o agente, a atuação do AAA para a percepção do ambiente.

Tabela 1 – Resultados da primeira bateria de testes da simulação

Percepção	Classificação Esperada	Resultado Obtido
ARF1 - [ssh root@192.168.3.78 && ssh root@192.168.3.78 « $dicionario_de_senhas]	ARF1 - TRUE	ARF1 - TRUE
ARF1 - [ssh root@192.168.3.78 « $dicionario_de_senhas]	ARF1 - TRUE	ARF1 - TRUE
ARF1 - [ssh root@192.168.3.78]	ARF1 - TRUE	ARF1 - TRUE
ARF1 - [ ]	ARF1 - FALSE	ARF1 - FALSE
ABF1 - [mysql -u root	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root && mysql -u root $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root && sudo -u postgres psql postgres	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root && sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root $dicionario_de_senhas && sudo -u postgres psql postgres]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root $dicionario_de_senhas && sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root $dicionario_de_senhas && sudo -u postgres psql postgres && sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [sudo -u postgres psql postgres]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [sudo -u postgres psql postgres && sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root && mysql -u root $dicionario_de_senhas && sudo -u postgres psql postgres && sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [ ]	ABF1 - FALSE	ABF1 - FALSE
DSF3 - [server.info=”]	DSF3 - TRUE	DSF3 - TRUE
DSF3 - [server.number=”]	DSF3 - TRUE	DSF3 - TRUE
DSF3 - [server.info=” && server.number=”]	DSF3 - TRUE	DSF3 - TRUE
DSF3 - [server.info=’Apache Tomcat 6.0.35’]	DSF3 - FALSE	DSF3 - FALSE
DSF3 - [server.number=’6.0.25.1’]	DSF3 - FALSE	DSF3 - FALSE
DSF3 - [server.info=’Apache Tomcat 6.0.35’ && server.number=’6.0.25.1’]	DSF3 - FALSE	DSF3 - FALSE
DSF3 - [server.info=’Apache Tomcat 6.0.35’ && server.number=”]	DSF3 - FALSE	DSF3 - FALSE
DSF3 - [server.info=” && server.number=’6.0.25.1’]	DSF3 - FALSE	DSF3 - FALSE
DSF3 - [ ]	DSF3 - FALSE	DSF3 - FALSE
DRF1 - [PermitRootLogin YES]	DRF1 - FALSE	DRF1 - FALSE
DRF1 - [PermitRootLogin NO]	DRF1 - TRUE	DRF1 - TRUE
DRF1 - [ ]	DRF1 - FALSE	DRF1 - FALSE

Fonte – Elaborado pelo autor.

Os resultados mostram que o agente classificou todas as percepções corretamente. Isso pode ser observado na coluna *Resultado Obtido*. O sucesso dos resultados se dá na maneira em que recebe as percepções e a maneira em que ele as processa.

Em uma percepção da lista de entrada, o primeiro dado considerado é o identificador da técnica. Com esse identificador, o agente pode encontrar qual o estado esperado correto a ser comparado na lista de estados esperados. Em seguida, o dado considerado é o que está entre os colchetes, *[ ]*. Esses se referem ao que o *plugin* do agente percebeu do ambiente.

Para detectar se a classificação precisa da combinação de um ou mais parâmetros e/ou comandos, o agente procura pelos separadores *&&* e ||. Eles se referem a operadores lógicos de percepções considerados pelos agentes. O operador *&&* indica que a classificação depende da

combinação de dados. E o operador || mostra que a classificação depende de um ou outro dado de parâmetros e/ou comandos.

Assim, diversas técnicas foram testadas com diferentes combinações de entradas para cada uma delas. Por exemplo a técnica Defesa, Sistema Operacional, Fácil - DSF3 foi testada de nove maneiras. Inicialmente, testou-se separadamente cada um dos parâmetros. As informações do servidor foram retiradas, *server.info=”* e *server.info=”*, respectivamente. Em seguida, combinaram-se esses dois parâmetros, ambos não possuíam quaisquer informações do servidor. A partir disso, novas combinações foram feitas com esses parâmetros para que se pudesse autenticar a capacidade de classificação do AAA.

6.2.2 Segunda Bateria

Na segunda bateria de testes, Tabela 2, procurou-se introduzir novas situações. Ou seja, foram feitos os mesmos testes da primeira bateria e adicionadas novas ocorrências de combinações. Nessa fase, procurou-se utilizar variáveis diferentes dentro dos parâmetros. Por exemplo, a técnica Ataque, Redes, Fácil - ARF1, na primeira bateria, só foi testada com o usuário *root*. Agora, adicionou-se o usuário *admin* para se ter mais testes de validação da identificação desta técnica.

Os resultados da segunda bateria podem ser visualizados na Tabela 2.

Tabela 2 – Resultados da segunda bateria de testes da simulação

Percepção	Classificação Esperada	Resultado Obtido
ARF1 - [ssh root@192.168.3.78 && ssh root@192.168.3.78 « $dicionario_de_senhas]	ARF1 - TRUE	ARF1 - TRUE
ARF1 - [ssh root@192.168.3.78 « $dicionario_de_senhas]	ARF1 - TRUE	ARF1 - TRUE
ARF1 - [ssh root@192.168.3.78]	ARF1 - TRUE	ARF1 - TRUE
ARF1 - [ssh admin@192.168.3.78 && ssh admin@192.168.3.78 « $dicionario_de_senhas]	ARF1 - FALSE	ARF1 - FALSE
ARF1 - [ssh admin@192.168.3.78 « $dicionario_de_senhas]	ARF1 - FALSE	ARF1 - FALSE
ARF1 - [ssh admin@192.168.3.78]	ARF1 - FALSE	ARF1 - FALSE
ARF1 - [ ]	ARF1 - FALSE	ARF1 - FALSE
ABF1 - [mysql -u root	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root && mysql -u root $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root && sudo -u postgres psql postgres	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root && sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root $dicionario_de_senhas && sudo -u postgres psql postgres]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root $dicionario_de_senhas && sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root $dicionario_de_senhas && sudo -u postgres psql postgres && sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [sudo -u postgres psql postgres]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [sudo -u postgres psql postgres && sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u root && mysql -u root $dicionario_de_senhas && sudo -u postgres psql postgres && sudo -u postgres psql postgres -W $dicionario_de_senhas]	ABF1 - TRUE	ABF1 - TRUE
ABF1 - [mysql -u admin	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [mysql -u admin && mysql -u admin $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [mysql -u admin && sudo -u postgres admin adminbd	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [mysql -u admin && sudo -u postgres admin adminbd -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [mysql -u admin $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [mysql -u admin $dicionario_de_senhas && sudo -u postgres admin adminbd]	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [mysql -u admin $dicionario_de_senhas && sudo -u postgres admin adminbd -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [mysql -u admin $dicionario_de_senhas && sudo -u postgres admin adminbd && sudo -u postgres admin adminbd -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [sudo -u postgres admin adminbd]	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [sudo -u postgres admin adminbd && sudo -u postgres admin adminbd -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [sudo -u postgres admin adminbd -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [mysql -u admin && mysql -u admin $dicionario_de_senhas && sudo -u postgres admin adminbd && sudo -u postgres admin adminbd -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - FALSE
ABF1 - [ ]	ABF1 - FALSE	ABF1 - FALSE
DSF3 - [server.info=”]	DSF3 - TRUE	DSF3 - TRUE
DSF3 - [server.number=”]	DSF3 - TRUE	DSF3 - TRUE
DSF3 - [server.info=” && server.number=”]	DSF3 - TRUE	DSF3 - TRUE
DSF3 - [server.info=’Apache Tomcat 7’]	DSF3 - FALSE	DSF3 - FALSE
DSF3 - [server.number=’6.0’]	DSF3 - FALSE	DSF3 - FALSE
DSF3 - [server.info=’Tomcat’ && server.number=’6.0.25.1’]	DSF3 - FALSE	DSF3 - FALSE
DSF3 - [server.info=’Apache’ && server.number=”]	DSF3 - FALSE	DSF3 - FALSE
DSF3 - [server.info=” && server.number=’6.0.25.1’]	DSF3 - FALSE	DSF3 - FALSE
DSF3 - [ ]	DSF3 - FALSE	DSF3 - FALSE
DRF1 - [PermitRootLogin YES]	DRF1 - FALSE	DRF1 - FALSE
DRF1 - [PermitRootLogin NO]	DRF1 - TRUE	DRF1 - TRUE
DRF1 - [ ]	DRF1 - FALSE	DRF1 - FALSE
DSF2 - [<Connector ... server= ... />]	DSF2 - TRUE	DSF2 - TRUE
DSF2 - [<Connector ... server="Apache"... />]	DSF2 - FALSE	DSF2 - FALSE
DSF2 - [<Connector ... server="Tomcat"... />]	DSF2 - FALSE	DSF2 - FALSE
DSF2 - [<Connector ... server="Apache Tomcat"... />]	DSF2 - FALSE	DSF2 - FALSE

Fonte – Elaborado pelo autor.

6.2.3 Terceira Bateria

Para a terceira, e última, bateria de testes, foram feitos os mesmos testes da segunda bateria, com a introdução de novas situações mostradas na Tabela 3.

Tabela 3 – Resultados da terceira bateria de testes da simulação

Percepção	Classificação Esperada	Resultado Obtido
DSF3 - [server.info='NIGNX Server' && server.number='1.10.1']	DSF3 - FALSE	DSF3 - TRUE
ABF1 - [mysql -u usuarioSistema	ABF1 - FALSE	ABF1 - TRUE
ABF1 - [mysql -u usuarioSistema && mysql -u usuarioSistema $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - TRUE
ABF1 - [mysql -u usuarioSistema && sudo -u postgres usuarioSistema bancoSistema	ABF1 - FALSE	ABF1 - TRUE
ABF1 - [mysql -u usuarioSistema && sudo -u postgres usuarioSistema bancoSistema -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - TRUE
ABF1 - [mysql -u usuarioSistema $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - TRUE
ABF1 - [mysql -u usuarioSistema $dicionario_de_senhas && sudo -u postgres usuarioSistema bancoSistema]	ABF1 - FALSE	ABF1 - TRUE
ABF1 - [mysql -u usuarioSistema $dicionario_de_senhas && sudo -u postgres usuarioSistema bancoSistema -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - TRUE
ABF1 - [mysql -u usuarioSistema $dicionario_de_senhas && sudo -u postgres usuarioSistema bancoSistema && sudo -u postgres usuarioSistema bancoSistema -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - TRUE
ABF1 - [sudo -u postgres usuarioSistema bancoSistema]	ABF1 - FALSE	ABF1 - TRUE
ABF1 - [sudo -u postgres usuarioSistema bancoSistema && sudo -u postgres usuarioSistema bancoSistema -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - TRUE
ABF1 - [sudo -u postgres usuarioSistema bancoSistema -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - TRUE
ABF1 - [mysql -u usuarioSistema && mysql -u usuarioSistema $dicionario_de_senhas && sudo -u postgres usuarioSistema bancoSistema && sudo -u postgres usuarioSistema bancoSistema -W $dicionario_de_senhas]	ABF1 - FALSE	ABF1 - TRUE

Fonte – Elaborado pelo autor.

Os testes adicionais feitos na última bateria obtiveram resultados em que o agente não consegue classificar corretamente a técnica utilizada. Fazendo a análise desses resultados, é possível identificar o porquê de o agente não conseguir classificá-las corretamente. A primeira linha mostra que o usuário substituiu as informações do servidor Apache Tomcat por informações referentes a outro servidor de aplicação, o ***NGINX***.

Essa tática de defesa utilizada é uma válida, pois o usuário defensor está tirando a atenção de um atacante para um serviço que não está instalado na máquina, no caso, o ***NGINX***. Com essa tática, o defensor está evitando o ***banner grabbing*** no Apache Tomcat, mas o agente não consegue identificar este tipo de situação.

Além disso, é possível ver que para tentativas de *brute force* em banco de dados *mysql* e *postgresql* por meio de usuários válidos dos respectivos sistemas, que não seja o usuário *root* não será classificado pelo agente. Isso se deve pelo fato de que a técnica catalogada não abrange esse tipo de situação válida.

6.3 TESTES DE PERFORMANCE

Após a validação do protótipo do agente, foram feitos testes de performance para verificar o desempenho da formalização. Esses testes são importantes porque o agente atuará em um sistema escalável.

Dada a arquitetura do AAA, definida na Seção 5.3, cada *plugin* ativo representa um ambiente computacional monitorado. Assim, o AAA precisa analisar as percepções de cada *plugin* separadamente das percepções de outros *plugins*.

Entretanto, é importante pontuar que um usuário não está associado, exclusivamente, a um *plugin*. O usuário pode estar atuando em mais de um ambiente computacional. Por exemplo, ele pode estar treinando técnicas de defesa em uma rede virtual de máquinas virtuais. Assim, será necessário um *plugin* para cada uma das máquinas virtuais, mesmo que somente um usuário esteja ativo nessa rede virtual.

Dessa maneira, foram feitos testes de tempo de execução para diferentes números de *plugins*. Cada execução do teste representa a tomada de ação do agente em relação à percepção de um *plugin*. A Figura 21 mostra a tabela de tempos obtidos de atuação, dado o número de execuções.

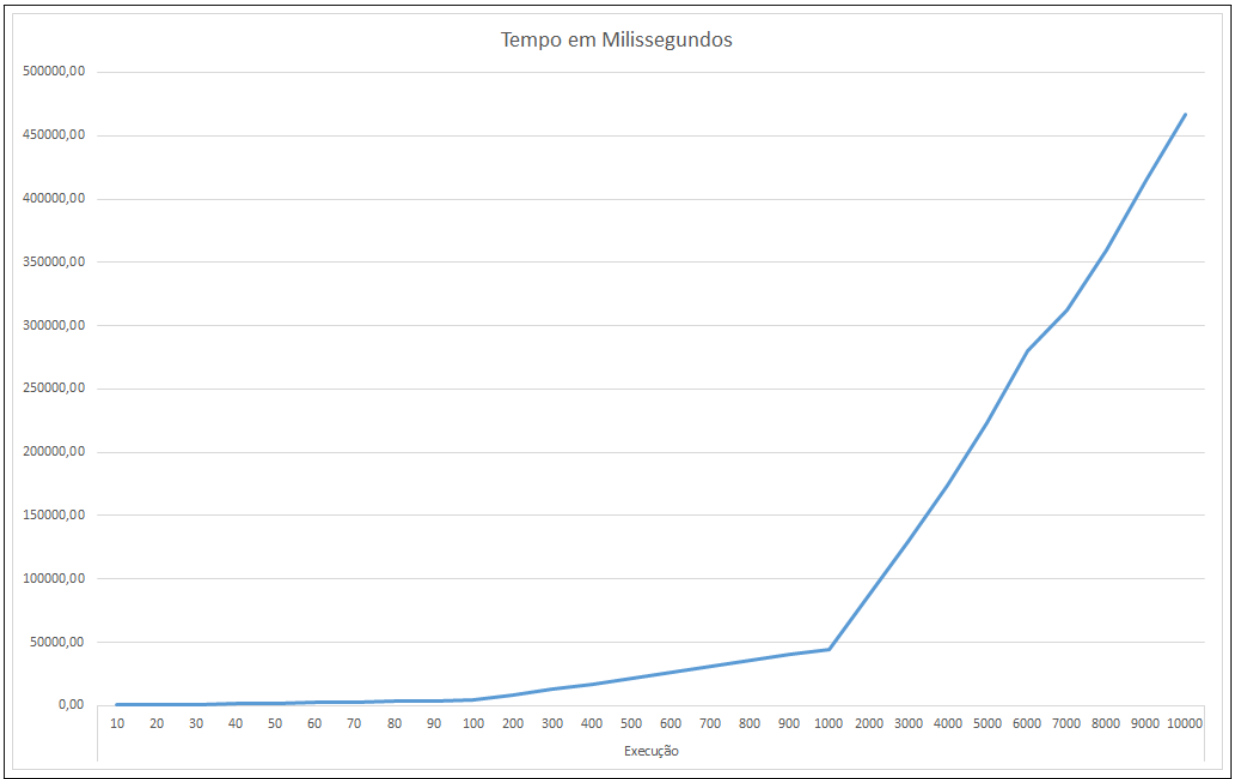
Figura 21 – Testes de Performance

		Tempo	
		Milissegundos	Segundos
Execução	10	639,18	0,64
	20	700,92	0,70
	30	1155,28	1,16
	40	1634,76	1,63
	50	2174,08	2,17
	60	2451,95	2,45
	70	2890,32	2,89
	80	3344,14	3,34
	90	4012,42	4,01
	100	4426,65	4,43
	200	8724,12	8,72
	300	12796,48	12,80
	400	17089,33	17,09
	500	21756,25	21,76
	600	26702,99	26,70
	700	30639,36	30,64
	800	36009,69	36,01
	900	40363,96	40,36
	1000	44021,77	44,02
	2000	87427,10	87,43
	3000	129162,46	129,16
	4000	174125,77	174,13
	5000	222905,87	222,91
	6000	280402,29	280,40
	7000	312092,03	312,09
	8000	359947,62	359,95
	9000	414468,72	414,47
	10000	466369,42	466,37

Fonte – Elaborada pelo autor.

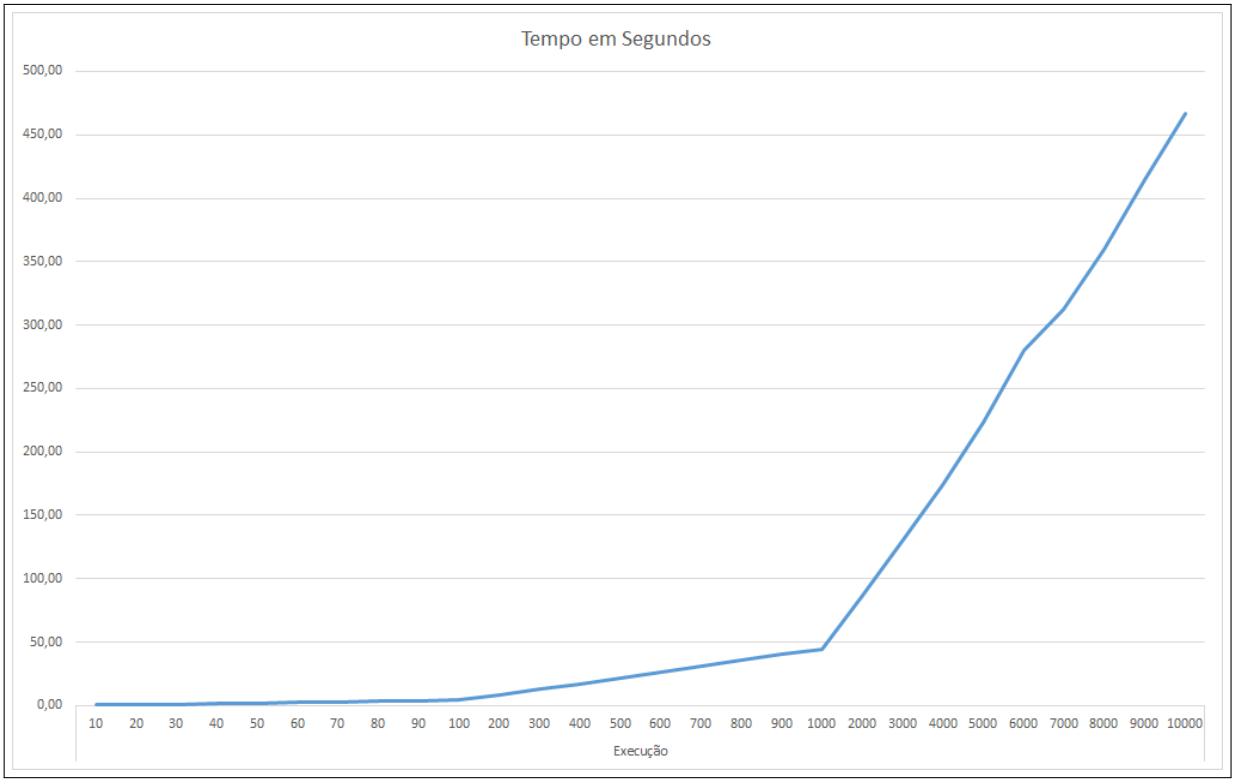
Dos resultados obtidos, as Figuras 22 e 23 mostram o gráfico da performance em milissegundos e segundos respectivamente. Conforme esperado, em ambos os casos, o algoritmo apresenta complexidade $\Theta(n^2)$.

Figura 22 – Gráfico Testes de Performance em Milissegundos



Fonte – Elaborada pelo autor.

Figura 23 – Gráfico Testes de Performance em Segundos



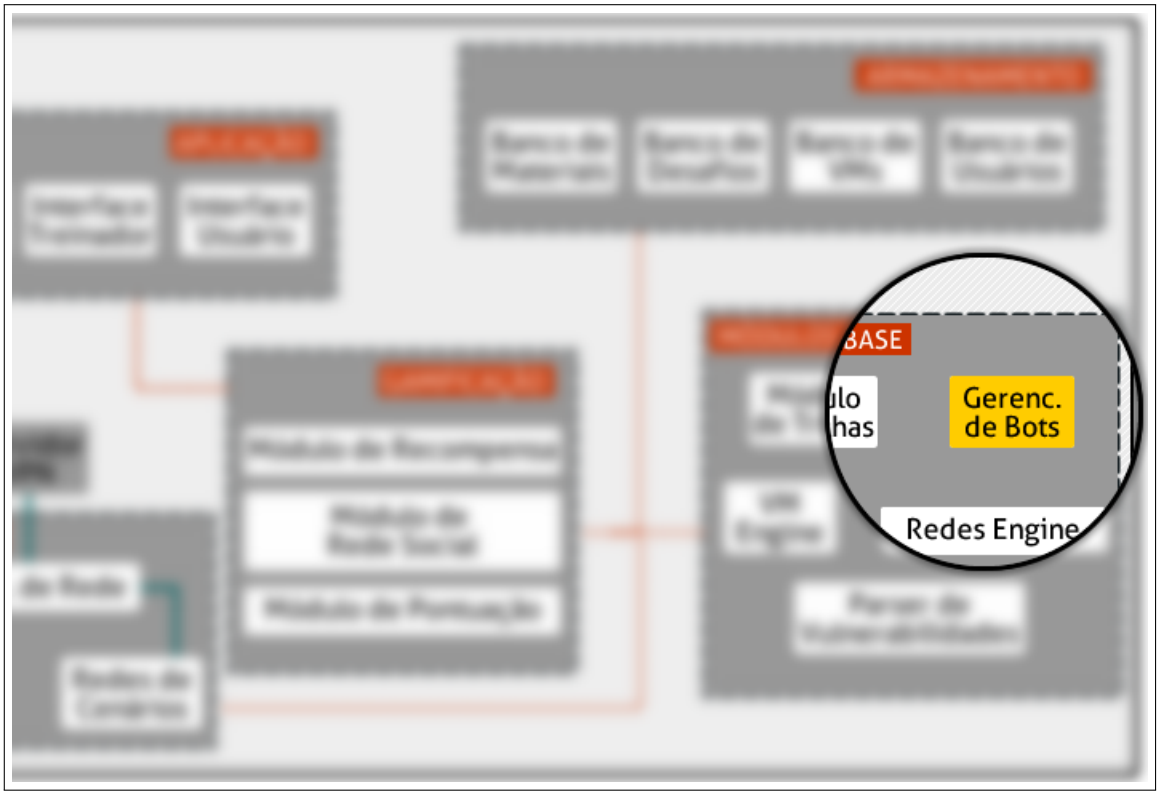
Fonte – Elaborada pelo autor.

Essa complexidade esperada vem do Algoritmo 1, em que se tem um laço *For* dentro do outro, representando a Lista de Percepções e a Lista de Estados Esperados respectivamente. Assim, visto que a primeira lista contém N percepções e que, para cada uma delas, existe um item, representando o estado esperado, na segunda lista, teremos $N \times N$ operações.

7 INTEGRAÇÃO COM O SHELLTER

A arquitetura proposta do *SAI* estará contida na arquitetura do Shellter mostrada na Figura 3. A Figura 24 mostra em qual módulo as duas arquiteturas irão se integrar. O *SAI* integrará os módulos base do Shellter, no módulo Gerenciamento de Bots, interagindo diretamente com o módulo de virtualização do Shellter, atuando nos exercícios simulados/laboratórios virtuais em que for requisitado.

Figura 24 – Arquitetura *SAI* no Shellter



Fonte – Elaborado pelo autor.

O módulo de virtualização é responsável por virtualizar, executar e gerenciar os sistemas computacionais necessários para o aprendizado *hand-ons* no Shellter. Ele possui dois tipos de redes: rede de desafios remotos e rede de cenários. A rede de desafios remotos é responsável por suportar requisitos para desafios que necessitem de um ambiente remoto (por exemplo, desafios de segurança em aplicações *web*) para que possa ser completado com sucesso.

A rede de cenários é responsável por suportar os requisitos para os exercícios simulados. Essas redes serão capazes de simular os elementos de rede e os elementos computacionais necessários para que a simulação seja o mais semelhante ao ambiente real. O *SAI* irá atuar nesses

dois ambientes, monitorando os usuários que estiverem conectados a essas redes.

O *SAI* também irá comunicar-se com o módulo de armazenamento, mais especificamente, com o banco de dados de usuários. Essa comunicação se faz necessária para que a *engine* de Classificação & Agrupamento possa capturar os dados dos usuários que interagirão com o ambiente virtual, e, por meio da sua modelagem matemática e técnicas de mineração de dados apropriada, definir uma classificação para cada um desses usuários, assim como uma classificação adequada para uma equipe.

Dessa comunicação com o módulo de armazenamento, o *SAI* ainda alimentará o banco de dados dos usuários, pelos dados coletados e processados pela *engine* de Coleta e Processamento. Assim o usuário será classificado e receberá pontos e premiações pelo seu desempenho no ambiente simulado, assim como as equipes também receberão pontuações e premiações.

Essa classificação se difere da classificação aplicada pela *engine* de Classificação & Agrupamento, pois esta classifica o usuário de maneira que sejam escolhidos os agentes mais adequados para interagir com ele no ambiente simulado. A classificação da *engine* de Coleta & Processamento se dá com o objetivo de agregar as conquistas do usuário no Shellter e assim melhorar o seu posicionamento no *rank* geral do Shellter, fornecer premiações e habilidades, entre outros.

Em (MAGALHAES, 2015), é apresentada uma ferramenta capaz de auxiliar na criação de cenários para treinamentos simulados em segurança da informação, assim como gerenciá-los. Este trabalho também se integra ao Shellter nos módulos Bases, na VM Engine e Redes Engine.

Assim, seguindo os fluxos apresentados em (HACHEM, 2015) e (MAGALHAES, 2015), um usuário administrador se loga no Shellter por meio da interface gráfica, cria um cenário para ser simulado no ambiente virtual e envia as chaves VPN de acesso para os usuários interessados em interagir com o ambiente. Esses usuários se conectam ao ambiente por meio das suas chaves VPNs e aguardam a simulação iniciar.

O *SAI* começa a agir no momento em que os usuários se conectam ao ambiente virtual do Shellter e esperam o início da simulação. Ele requisita dados do banco de dados de usuário para que possa fazer a classificação e agrupamento do usuário e da equipe como um todo, e assim escolher os agentes de níveis mais adequados para interagirem com o usuário e sua equipe.

Ao final da simulação, os dados de desempenho do usuário e sua equipe são tratados para que possam ir para o seu perfil de usuário e perfil de equipe no Shellter.

8 CONCLUSÕES E TRABALHOS FUTUROS

O sistema apresentando neste trabalho, o Sistema de Agentes Inteligentes para Ambientes Simulados para treinamento em segurança da informação, auxilia na motivação de estudantes e entusiastas, pois provê um acompanhamento desses nos ambientes simulados. O sistema atua em diversas frentes, onde classifica o usuário para escolher os desafios do mesmo nível que o usuário. Além de escolher o agente de mesmo nível para jogar. Isso com o objetivo de proporcionar uma partida que não desestimule, pois a não classificação pode causar desentusiasmo devido ao problema de desafios e/ou agentes muito fáceis ou muito difíceis, relatados no início do capítulo 4.

Ainda mais, o SAI provê um acompanhamento durante a simulação e treinamento para que o usuário permaneça estimulado. O AAU monitora as expressões e emoções do usuário para capturar sinais de motivação classificando-as. O AAA percebe e classifica as técnicas de segurança da informação utilizadas no ambiente para que possa pontuar o usuário de acordo com o seu desenvolvimento.

Esses dois agentes atuam com o foco de monitorar e classificar o desempenho emocional e técnico do usuário, para que, além da classificação, possam passar informações para que os outros dois agentes do sistema possam atuar para mudar possíveis casos de desmotivação.

Dos outros dois agentes, o AJCU é o responsável por jogar contra o usuário, proporcionando a simulação de jogador real atacante ou defendendo o ambiente. Esse agente se utiliza das informações na CeA, AAU e AAA para escolher as técnicas e movimentos para aplicar durante a simulação. E o AIU atua tutoriando o usuário, ou seja, baseado nos dados passados pela CeA, AAU e AAA. Atua dando dicas, tutoriais, entre outras coisas para que o usuário possa desenvolver novas conhecimentos e continuar com vontade de aprender e praticar segurança da informação.

Para cada um dos quatro agentes do sistema, definiu-se a sua racionalidade por meio do PEAS, conforme visto na Seção 4.4.

Para validação do sistema, implementou-se o AAA, devido a sua arquitetura ser semelhante aos outros agentes do trabalho, além da sua importância no sistema. Para este agente, inicialmente definiu-se a taxonomia e arquitetura de catalogação das técnicas de segurança da informação que serão classificadas pelo mesmo. Assim, validou-se o mesmo por meio da formalização proposta na Seção 5.6 e dos testes feitos na Seção 6.

Com isso, podemos listar as contribuições feitas neste trabalho:

- Definição do Sistema de Agentes Inteligentes para Ambientes Simulados para o treinamento em segurança da informação;
- Definição da *engine* de Classificação e Agrupamento;
- Definição do *PEAS* do *AAU - Agente responsável por Aprender sobre a evolução do Usuário*;
- Definição do *PEAS* do *AJCU - Agente responsável por Jogar Contra o Usuário*;
- Definição do *PEAS* do *AIU - Agente responsável por Interagir com o Usuário*;
- Definição do *PEAS* do *AAA - Agente responsável por Aprender sobre mudanças no Ambiente*;
- Definição de uma taxonomia de classificação de técnicas de segurança da informação;
- Catalogação de técnicas de segurança da informação;
- Formalização do *AAA*;
- Prototipagem, implementação, testes e validação do *AAA*;
- Integração do SAI a arquitetura do Shellter.

Dadas as contribuições apresentadas, este trabalho conseguiu cumprir os seus objetivos, visto que conseguiu a definição de um sistema de agentes, assim como a sua arquitetura e fluxo de interações. Ainda, definiu-se a racionalidade de cada um dos agentes do sistema, por meio do estabelecimento do *PEAS*. Para a validação do sistema, implementou-se o protótipo de um dos agentes do sistema, o *AAA*, dado a sua semelhança com a arquitetura dos outros agentes do SAI, e validou-se o mesmo por meio de testes.

8.1 TRABALHOS FUTUROS

Dadas as contribuições do trabalho e a complexidade do ***Sistema de Agentes Inteligentes para Ambientes Simulados*** de treinamento em segurança da informação, os seguintes trabalhos futuros se fazem necessário:

- Definição da arquitetura, formalização e implementação do *AAU - Agente responsável por Aprender sobre a evolução do Usuário*;
- Definição da arquitetura, formalização e implementação do *AJCU - Agente responsável por Jogar Contra o Usuário*;
- Definição da arquitetura, formalização e implementação do *AIU - Agente responsável por Interagir com o Usuário*;
- Para o agente implementado neste trabalho:

  - Catalogação de técnicas: definição, catalogação e classificação de mais técnicas para serem analisadas pelo agente;
  - Adição de inteligencia para que o agente identifique movimentos de ilusão do usuário, ou seja, quando o usuário utiliza definições falsas para tirar a atenção de atacante e/ou defensor;
  - Testar a implementação do agente com outras linguagens de programação;
  - Melhorar a medida de desempenho do mesmo, fornecendo novas soluções de implementações para serem testadas em diversas linguagens de programação.
- Implementação e validação da integração do SAI com o Shellter.

REFERÊNCIAS

ALLEN, T. **War games**. [S.l.]: McGraw-Hilly, 1994.

BZUNECK, J. A. As crenças de auto-eficácia e o seu papel na motivação do aluno. In: **A Motivação do Aluno: Contribuições da Psicologia Contemporânea**. [S.l.: s.n.], 2009. p. 116–133.

CASS, S. **The 2015 Top Ten Programming Languages**. 2015. Disponível em: <http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages>.

DIFFIE, W.; HELLMAN, M. New directions in cryptography. **IEEE Trans. Inf. Theor.**, IEEE Press, Piscataway, NJ, USA, v. 22, n. 6, p. 644–654, set. 2006. ISSN 0018-9448. Disponível em: <http://dx.doi.org/10.1109/TIT.1976.1055638>.

ENGINES, D. **DB Engines Ranking**. 2015. Disponível em: <http://db-engines.com/en/ranking>.

FARROW, R. Identifiable fingerprints in network applications. **Login: The Usenix Magazine - 12th USENIX Security Symposium**, v. 28, n. 6, p. 31–45, 2003.

FLATLA, D.; GUTWIN, C.; NACKE, L.; BATEMAN, S.; MANDRYK, R. Calibration games: Making calibration tasks enjoyable by adding motivating game elements. In: **UIST 11: Proceedings of the 24th annual ACM symposium on User interface software and technology**. Santa Barbara, California, USA: [s.n.], 2011. p. 403–412.

FROZZA ANDREA APARECIDA KONZEN DA SILVA, J. N. C. S. B. L. K. W. M. L. M. K. M. P. B. A. B. d. C. J. L. B. L. S. R. Agentes pedagógicos emocionais atuando em um ambiente virtual de aprendizagem. **Revista Renote - Novas Tecnologias na Educação**, v. 9, n. 1, 2011.

FUTORANSKY, A.; MIRANDA, F.; ORLICKI, J.; SARRAUTE, C. Simulating cyber-attacks for fun and profit. In: **Proceedings of the 2Nd International Conference on Simulation Tools and Techniques**. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009. (Simutools '09), p. 4:1–4:9. ISBN 978-963-9799-45-5. Disponível em: <http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5773>.

GASPAR SARAH LANGEVIN, J. S. C. G. A. Softice: Facilitating both adoption of linux undergraduate operating systems laboratories and students' immersion in kernel code. **SYSTEMICS, CYBERNETICS AND INFORMATICS**, v. 5, n. 3, p. 30–35, 2007.

GROH, F. Gamification: State of the art definition and utilization. In: **RTMI**. [S.l.: s.n.], 2012.

GROUP, P. G. D. **PostgreSQL**. 2015. Disponível em: <http://www.postgresql.org/>.

HACHEM, S. R. F. Graduação, **SHELLTER - UM AMBIENTE DE APRENDIZADO EM SEGURANÇA DA INFORMAÇÃO COM ABORDAGEM PRÁTICA**. 2015.

HARDENING and messing with win32 Apache Tomcat. 2009. Disponível em: <https://www.exploit-db.com/papers/12878/>.

KAMPFF ANTôNIO DA FONSECA DE LIRA, D. S. R. F. d. J. L. G. L. C. C. F. N. C. F. M. M. B. A. J. C. Relação entre o perfil do usuário e a escolha do perfil do tutor. **Revista Renote - Novas Tecnologias na Educação**, v. 3, n. 1, 2005.

KUROSE, K. W. R. J. F. **Redes de Computadores e a Internet - Um abordagem top-down**. [S.l.]: Pearson Addison Wesley, 2005. 625 p.

LEIDECKER, N. **Having Fun With PostgreSQL**. 2015. Disponível em: <https://www.exploit-db.com/papers/13084/>.

LESTER EUN Y. HA, S. Y. L. B. W. M. J. P. R. J. L. S. J. C. Serious games get smart: Intelligent game-based learning environments. **AI Magazine**, v. 34, n. 4, p. 31–45, 2013.

LONGHI MAGDA BERCHT, P. A. B. M. T. Reconhecimento de estados afetivos do aluno em ambientes virtuais de aprendizagem. **Revista Renote - Novas Tecnologias na Educação**, v. 5, n. 2, 2007.

MAGALHAES, R. L. Graduação, **UMA FERRAMENTA PARA CRIAÇÃO AUTOMATIZADA DE CENÁRIOS EM LABORATÓRIOS VIRTUAIS PARA UTILIZAÇÃO NO ENSINO DE SEGURANÇA CIBERNÉTICA**. 2015.

MORAES, S. V. C. R. Motivação do aluno durante o processo de ensino-aprendizagem. In: **Revista Eletrônica de Educação**. [S.l.: s.n.], 2007. p. 116–133.

MUNTEAN, C. I. Raising engagement in e-learning through gamification. **The 6o International Conference on Virtual Learning ICVL**, 2011.

NAGARAJAN, A.; ALLBECK, J.; SOOD, A.; JANSSEN, T. Exploring game design for cybersecurity training. In: **Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2012 IEEE International Conference on**. [S.l.: s.n.], 2012. p. 256–262.

ORACLE. **MySQL**. 2015. Disponível em: <http://www.mysql.com/>.

PADGHAM, M. W. L. **Developing Intelligent Agent Systems**. [S.l.]: Wiley, 2005. 1-5 p.

RODRIGUES, M. C. L. M. L. Sti-i: Sistemas tutoriais inteligentes que integram cogniÇÃo, emoÇÃo e motivaÇÃo. **Revista Brasileira de Informática na Educação**, v. 13, n. 1, 2005.

ROSAS, D. P. A. B. F. W. **A IMPORTÂNCIA DA MÚSICA EM OBJETOS DE APRENDIZAGEM**. 2010. Disponível em: <http://www.nuted.ufrgs.br/wordpress/wp-content/uploads/2010/Artigos/Importancia.pdf>.

RUSSELL, P. N. S. J. **Artificial Intelligence: A Modern Approach**. [S.l.]: Prentice Hall, 2010. 1132 p.

SALEN, E. Z. K. **Rules of Play Game Design Fundamentals**. [S.l.]: MIT Press, 2003. 11 p.

SALNIKOV-TARNOVSKI, N. **Most popular Java EE containers: 2015 edition**. 2015. Disponível em: <https://plumbr.eu/blog/java/most-popular-java-ee-containers-2015-edition>.

SCOWL. **Word List SCOWL**. 2015. Disponível em: <http://wordlist.aspell.net/>.

SECURITY, D. **Crack Station**. 2015. Disponível em: <https://crackstation.net/buy-crackstation-wordlist-password-cracking-dictionary.htm>.

SECURITY, O. **Exploit Database**. 2015. Disponível em: <https://www.offensive-security.com/>.

WILLEMS, C.; MEINEL, C. **Practical Network Security Teaching in an Online Virtual Laboratory**. 2011.

WOOLDRIDGE, M. **An Introduction to MultiAgent Systems**. [S.l.]: Wiley, 2002. 23 p.

ANEXOS

ANEXO A – Código Prolog Formalização AAA

```
aaa(ArqsEnt,ArqDes,ArqSai):-
sensor(ArqsEnt,1,P,ArqDes,Des),
classificador(P,Des,A),
atuador(A,ArqSai).

/*************************************************/

sensor([],_,[],ArqDes,Des):-
sensor1(ArqDes,Des).
sensor([ArqEnt|ArqsEnt],N,[percept(N,P)|RP],ArqDes,Des):-
Nacum is N+1,
sensor1(ArqEnt,P),
sensor(ArqsEnt,Nacum,RP,ArqDes,Des).

sensor1(Arq,P):-
see(Arq),
read(Info),!,
ler(Info,[],P),
seen.
sensor1(_,[]):-
seen.

ler(P,L,L):-
P = end_of_file,!.
ler(P,Laux,L2):-
concatena(Laux,[P],Lauxnovo),
read(OutroP),
ler(OutroP,Lauxnovo,L2).

/*************************************************/

classificador(P,Des,A):-
```

```
33      ver(P,S),
34      proximo(Des,S),
35      acao(A).
36
37      /*************************************************/
38
39      ver([],[]).
40      ver([percept(N,P)|Rpercepts],Lpredicados):-
41      ver1(percept(N,P),Lpredicados1),
42      ver(Rpercepts,Lpredicados2),
43      concatena(Lpredicados1,Lpredicados2,Lpredicados).
44
45      ver1(percept(1,[1,ServerInfo,2,ServerNumber]),[1,server_info(
           ServerInfo),2,server_number(ServerNumber)]):-!.
46      ver1(percept(1,[1,ServerInfo]),[1,server_info(ServerInfo),2,no
           ]):- !.
47      ver1(percept(1,[2,ServerNumber]),[1,no,2,server_number(
           ServerNumber)]):- !.
48      ver1(percept(1,[]),[1,no,2,no]):- !.
49      ver1(percept(2,[Server2]),[3,server2(Server2)]):- !.
50      ver1(percept(2,[]),[3,no]):- !.
51      ver1(percept(3,[Server3]),[4,server3(Server3)]):- !.
52      ver1(percept(3,[]),[4,no]).
53
54      /*************************************************/
55
56      :-dynamic server_info/1.
57      :-dynamic server_number/1.
58      :-dynamic server2/1.
59      :-dynamic server3/1.
60      :-dynamic modificado_server_info/0.
61      :-dynamic modificado_server_number/0.
62      :-dynamic modificado_root_login/0.
63      :-dynamic modificado_server3/0.
```

```
64
65      proximo(Des,Linfo):-
66      inserir_base(Des),
67      verifica(Linfo),
68      limpar_base.
69
70      inserir_base([]).
71      inserir_base([Pred|Rpreds]):-
72      assertz(Pred),
73      inserir_base(Rpreds).
74
75      verifica([]).
76      verifica([N,Info|Rinfo]):-
77      verifica1(N,Info),
78      verifica(Rinfo).
79
80      verifica1(1,Server_Info):-
81      retract(Server_Info),!,
82      asserta(Server_Info).
83      verifica1(1,_):-
84      asserta(modificado_server_info).
85      verifica1(2,Server_Number):-
86      retract(Server_Number),!,
87      asserta(Server_Number).
88      verifica1(2,_):-
89      asserta(modificado_server_number).
90      verifica1(3,Server2):-
91      retract(Server2),!,
92      asserta(Server2).
93      verifica1(3,_):-
94      asserta(modificado_root_login).
95      verifica1(4,Server3):-
96      retract(Server3),!,
97      asserta(Server3).
```

```
98    verifica1(4,_):-
99    asserta(modificado_server3).
100
101   limpar_base:-
102   abolish(server_info/1),
103   abolish(server_number/1),
104   abolish(server2/1),
105   abolish(server3/1).
106
107   /*************************************************/
108
109   acao(A):- faca_todas(A).
110
111   /*************************************************/
112
113   :- dynamic lista_mensagens/1.
114
115   faca_todas(_):-
116   asserta(lista_mensagens([])),
117   faca(M),
118   retract(lista_mensagens(LM)),
119   concatena(LM,[M],LM1),
120   asserta(lista_mensagens(LM1)),
121   fail.
122   faca_todas(A):-
123   retract(lista_mensagens(A)).
124
125   /*************************************************/
126
127   faca(classificacao('Banner Grabbing Apache Tomcat',
         modificado_server_info, 'DSM')):-
128   retract(modificado_server_info).
129   faca(classificacao('Banner Grabbing Apache Tomcat',
         modificado_server_number, 'DSM')):-
```

```
130     retract(modificado_server_number).
131     faca(classificacao('Root Login By SSH',modificado_root_login,'
            ARF1')):-
132     retract(modificado_root_login).
133     faca(classificacao('Banner Grabbing Apache Tomcat',
            modificado_server, 'DSF3')):-
134     retract(modificado_server3).
135
136     /*************************************************/
137     atuador(A,Arq):-
138     tell(Arq),
139     escrever(A),
140     told.
141
142     escrever([]).
143     escrever([Classificacao|Classificacoes]):-
144     write(Classificacao),nl,
145     escrever(Classificacoes).
146
147     /*************************************************/
148     concatena([],L,L).
149     concatena([X|Y],Z,[X|YZ]):-
150     concatena(Y,Z,YZ).
```

ANEXO B – Listas de Percepções Simuladas

```
1  ARF1 - {ssh root@192.168.3.78}
2  ARF1 - {ssh root@192.168.3.78 << $dicionario_de_senhas}
3  ARF1 - {ssh root@192.168.3.78 || ssh root@192.168.3.78 <<
       $dicionario_de_senhas}
4  BF1 - {mysql -u root}
5  BF1 - {mysql -u root $dicionario_de_senhas}
6  BF1 - {sudo -u postgres psql postgres}
7  BF1 - {sudo -u postgres psql postgres -W $dicionario_de_senhas}
8  BF1 - {mysql -u root || mysql -u root $dicionario_de_senhas}
9  BF1 - {mysql -u root || sudo -u postgres psql postgres}
10 BF1 - {mysql -u root || sudo -u postgres psql postgres -W
       $dicionario_de_senhas}
11 BF1 - {mysql -u root $dicionario_de_senhas || sudo -u postgres
       psql postgres}
12 BF1 - {mysql -u root $dicionario_de_senhas || sudo -u postgres
       psql postgres -W $dicionario_de_senhas}
13 BF1 - {sudo -u postgres psql postgres || sudo -u postgres psql
       postgres -W $dicionario_de_senhas}
14 DSF3 - {server.info=''}
15 DSF3 - {server.number=''}
16 DSF3 - {server.info='' && server.number=''}
```

```
1  DRF1 - [PermitRootLogin NO]
2  DRF1 - [PermitRootLogin YES]
3  DRF1 - []
4  DSF3 - [server.info='Apache Tomcat 6.0.35']
5  DSF3 - [server.number='6.0.25.1']
6  DSF3 - [server.info='Apache Tomcat 6.0.35' && server.number='
       6.0.25.1']
```

```
1  ARF1 - [ssh root@192.168.3.78 && ssh root@192.168.3.78 <<
       $dicionario_de_senhas]
2  ABF1 - [sudo -u postgres psql postgres && sudo -u postgres psql
        postgres -W $dicionario_de_senhas]
```

```
3     ARF1 - [ssh admin@192.168.3.78]
4     DRF1 - [PermitRootLogin NO]
5     DSF3 - [server.info='Apache Tomcat 6.0.35' \&\& server.number='
          ']
6     DSF3 - [server.info='Apache Tomcat 6.0.35']
7     DSF3 - [server.number='']
```

```
1     ARF1 - [ssh root@192.168.3.78 && ssh root@192.168.3.78 <<
          $dicionario_de_senhas]
2     ABF1 - [mysql -u root && mysql -u root $dicionario_de_senhas &&
           sudo -u postgres psql postgres && sudo -u postgres psql
          postgres -W $dicionario_de_senhas]
3     DSF3 - [server.info='' && server.number='']
4     ARF1 - [ssh admin@192.168.3.78]
5     DRF1 - [PermitRootLogin NO]
6     DSF3 - [server.info='' \&\& server.number='6.0.25.1']
```

```
1     ARF1 - [ ]
2     ABF1 - [ ]
3     DSF3 - [ ]
4     ARF1 - [ ]
5     DRF1 - [ ]
6     DSF3 - [ ]
```

DECLARAÇÃO

Eu, ANTÍDIO BARBOSA DE OLIVEIRA FILHO, brasileiro, solteiro, RG 140465587, SSP-CE, CPF 416.982.153-04, graduado em Letras e Mestre em Linguística pela Universidade Federal do Ceará, Registro MEC n.º 46254, declaro, para os devidos fins, haver feito a revisão gramatical da dissertação de autoria de Davi Teles França, intitulada UM SISTEMA INTELIGENTE PARA AMBIENTES SIMULADOS DE TREINAMENTO EM SEGURANÇA DA INFORMAÇÃO, a ser apresentada na UNIVERSIDADE ESTADUAL DO CEARÁ, como requisito parcial para a obtenção do grau de mestre em Ciência da Computação.

Fortaleza, 17 de agosto de 2016

Antídio Barbosa de Oliveira Filho

ANTÍDIO BARBOSA DE OLIVEIRA FILHO
RG. 14065587