Mestrado em Engenharia Informática
Dissertação/Estágio
Relatório Final

Plataforma de detecção de intrusão com base na análise de sequências de chamadas de sistema

Fábio Falcão de França

ffranca@student.dei.uc.pt

Orientador:
Prof. Doutor Joel Perdiz Arrais
Engenheiro Mário Ulisses Costa
Data: 03 de julho de 2013

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Resumo

Este relatório foi concebido no âmbito da análise forense computacional, tomando como domínio principal a análise das sequências de chamadas de sistema (System Calls) durante a execução de um serviço de internet no sistema operativo Linux. O objetivo geral deste projeto é o desenvolvimento de um módulo de detecção de intrusão, através da análise destas sequências de System Calls, Processos e quaisquer informaçoes que sejam relevantes acerca da execução do serviço escolhido para realização dos testes. Os objetivos específicos deste projeto estendem-se desde a criação de scripts para captura destas informações e preenchimento das mesmas na base de dados do Delusion, à especificação de uma API de comunicação com a base de dados para que seja possível a fácil recuperação dos dados e a implementação de um módulo de detecção. Estes são posteriormente combinados com técnicas de machine learning de forma a realizar inferências dos aspectos de segurança do serviço que está a ser monitorado.

Palavras-Chave

System Call Sequences, Honeypot, Computer Forensics, Security, Machine Learning, Delusion, Vulnerability.

Dedicatória

Dedico este trabalho a Deus e ao meu pai, grande incentivador desde o início da minha caminhada. Sem ele, com certeza, nada disso seria possível. Mesmo a distância, ele continou sendo o grande representante da palavra "força" em minha vida. Ao Prof. e amigo Luiz Maurício Martins, o meu muito obrigado por toda o apoio e amizade que o mesmo demonstrou durante todo este tempo.

Agradecimentos

Gostaria de agradecer este trabalho inicialmente a Deus, por me permitir realizá-lo até o final. Também gostaria de agradecer a algumas pessoas que foram de fundamental importância para a realização do mesmo, nomeadamente o Mario Costa, da Visionspace Technologies, que desde o início do projeto acreditou que o mesmo pudesse ser realizado e que dedicou boa parte do seu tempo e paciência para explanações acerca de vários assuntos que foram de vital importância para a conclusão deste trabalho. O Prof. Joel Arrais, que adotou esta idéia, teve visão ao inserir outras áreas de conhecimento a este projeto e que sempre esteve disposto a me ajudar quando eu mais precisei. Aos dois, meu grande obrigado. Também queria agradecer outras pessoas que passaram pela minha vida durante este tempo e que de certa forma deixaram contribuições positivas, ou num momento de dificuldade, ou em um momento de empolgação sobre o projeto. Deixo meu muito obrigado a amiga Viviane Felintro, pois, quando precisei, esteve lá para dar força e apoio moral durante o desenvolvimento deste projeto. Agradeço também a outras pessoas que mantiveram apoio mesmo que a distância, estando no Brasil, como também em Coimbra, meus amigos e colegas da Coimbra MMA, que mesmo sem saber ajudaram bastante com palavras de apoio, mesmo sem perceber. Oss. Ao meu grande amigo mestre Célio Medeiros, grande incentivador desde o meu início do mestrado, meu muito obrigado por sempre estar disposto a ajudar. Ao mestre Douglas Ono, a certeza de que seus ensinamentos também deixaram bastante aspectos positivos em minha vida. Ao amigo Vasco, meu irmão português, não tenho palavras para descrever o quanto nossa amizade significa para mim.

Índice

Lista de Tabelas

Lista de Imagens

Lista de Acrónimos

SGBD – Sistema Gerenciador de Bases de Dados

BD – Base de Dados

API – Application Programming Interface

VMI – Virtual Machine Introspection

VST – VisionSpace Technologies

SQL – Structured Query Language

VMM – Virtual Machine Monitor

IDS – Intrusion Detection System

CPU – Central Processing Unit

ISO – International Organization for Standardization

ROC - Receiver Operating Characteristic

TPR – True Positive Rate

FPR – False Positive Rate

ODBC – Open Database Connectivity

DAO - Data Access Object

TLS – Transport Layer Security

SSL – Secure Sockets Layer

Capítulo 1
Introdução

Este relatório tem por objetivo apresentar o trabalho desenvolvido na empresa VisionSpace Technologies no contexo do estágio do Mestrado em Engenharia Informática. É proposto o desenvolvimento de um módulo de detecção de intrusão em servidores baseados em sistemas operacionais GNU/Linux, com base na análise das sequências de execução de system calls. É esperado que este módulo venha a integrar a ferramenta de produção Delusion, que tem por objetivo atual a captura de dados relativos à utilização do sistema operativo, utilizando os mesmos para realizar análise forense em estações de uma rede computacional. O desenvolvimento deste módulo de detecção de intrusão é baseado na utilização de técnicas de machine learning, técnicas estas, capazes de realizar inferências acerca do conjunto de system calls invocadas no momento da execução do software. A validação destas técnicas é realizada com base num dataset obtido a partir da execução do sistema de gestão de bases de dados MySQL, enquanto executava operações relativas ao seu funcionamento normal em um ambiente de produção. Através da aplicação destas técnicas, foi possível realizar a definição do comportamento observado nos dados de execução do MySQL poderia ser considerado como uma tentativa de ataque ao sistema monitorizado ou se tratava apenas de uma execução normal do mesmo. Deste modo, os objetivos deste relatório serão concebidos através da análise do conjunto de sequências de chamadas do sistema (System Calls), onde serão observadas quais as funções que foram executadas em sequência, informação esta, capaz de identificar uma utilização anómala dos softwares de serviço web que estão a ser analisados, utilizando para isto, a combinação dos dados com técnicas de machine learning.

Para comprovar a eficiência das técnicas de machine learning utilizadas e da técnica de análise de comportamentos anómalos usando sequências de System Calls utilizada neste relatório, também será utilizado um estudo de caso, onde um servidor gerenciador de bases de dados (SGBD), nomeadamente o MySQL, que teve seu comportamento observado a fim de obter dados que possam ser úteis ao processo de detecção de intrusão.

Esta dissertação está dividida em cinco capítulos, onde o primeiro está ligado à breve introdução supracitada. O segundo ao planeamento das atividades exercidas ao longo do estágio, do primeiro ao segundo semestre do ano letivo. O terceiro capítulo está ligado ao estado da arte das tecnologias e ferramentas associadas ao desenvolvimento do módulo de detecção de intrusão, da infraestrutura utilizada pelo módulo e dos métodos de machine learning utilizados no projeto. O quarto capítulo descreve todas as fases do desenvolvimento do módulo de detecção de intrusão, assim como a interface de integração com o Delusion e demais documentos arquiteturais que dizem respeito ao mesmo. Também consiste na avaliação do módulo de detecção de instrusão e que os resultados são condizentes com a expectativa inicial do projeto. O quinto e último capítulo éinteiramente dedicado as considerações finais, aos trabalhos futuros e aos resultados obtidos pelo módulo de detecção de intrusão, como também a avaliação de crescimento intelectual do aluno perante a realização deste projeto e dissertação.

Capítulo 2
Planeamento

Neste capítulo é apresentado o planeamento das atividades correspondentes as tarefas realizadas para este projeto.

Estas tarefas serão apresentadas de maneira a contemplar uma divisão referente aos dois semestres da disciplina de Estágio/Dissertação.

2.1 Planeamento do Primeiro Semestre

A **Imagem 1** representa a calendarização para o primeiro semestre da disciplina de Dissertação/Estágio, exibindo as tarefas executadas no período:

Nome	Data inicial	Data final
Planeamento do Estágio	17/09/2012	26/10/2012
Estado da Arte	29/10/2012	09/11/2012
Estudo sobre Processos	29/10/2012	29/10/2012
Análise da Base de dados Delusion	30/10/2012	30/10/2012
Estudo sobre o Delusion	30/10/2012	30/10/2012
Principais técnicas de intrusão	31/10/2012	31/10/2012
Soluções existentes	31/10/2012	31/10/2012
Estudo sobre System Calls	01/11/2012	09/11/2012
Especificação	12/11/2012	17/01/2013
Especificação da API de acesso a BD	12/11/2012	14/01/2013

Imagem 1 - Calendarização do planeamento do 1º semestre
Fonte: Documentação Interna Visionspace

O planeamento foi dividido nos seguintes módulos:

- Planeamento do Estágio, no qual foi feito um levantamento das atividades que seriam desenvolvidas durante este primeiro momento do estágio.
- Estado da Arte, contempla um estudo sobre as soluções existentes no mercado, tecnologias, ferramentas utilizadas, técnicas e demais mecanismos que foram exploradas no âmbito desta dissertação.
- Especificação, no qual é descrito o desenvolvimento da API de acesso à base de dados do Delusion, conforme indicação da empresa, assim como de novas funções de retorno de dados que foram sendo desenvolvidas conforme necessidade.

2.2 Planeamento do Segundo Semestre

A **Imagem 2** representa a calendarização para o segundo semestre da disciplina de Dissertação/Estágio, exibindo as tarefas executadas no período:

Nome	Data inicial	Data final
Planeamento do 2º semestre	15/02/2013	07/03/2013
Estudo sobre VMI	19/02/2013	29/03/2013
Instalação LibVMI	19/02/2013	04/03/2013
Testes LibVMI	04/03/2013	29/03/2013
Análise Técnicas de Machine Learning	02/04/2013	17/04/2013
Estudo sobre Machine Learning	02/04/2013	05/04/2013
Detecção de Intrusão usando Syscalls	05/04/2013	10/04/2013
Literatura sobre Naive Bayes Classifiers	09/04/2013	12/04/2013
Estudo sobre Orange API	15/04/2013	17/04/2013
Ambiente geração ficheiro de Features	10/04/2013	28/05/2013
Instalar Kali Linux no servidor VST	10/04/2013	10/04/2013
Acoplagem do sensor do Delusion	16/04/2013	17/04/2013
Criação do script gerador de queries SQL	16/04/2013	19/04/2013
Criação do script para o sensor Delusion	18/04/2013	19/04/2013
Instalação da Orange API	19/04/2013	19/04/2013
Geração do ficheiro do Orange	22/04/2013	23/04/2013
Criação da microaplicação Python de testes	22/04/2013	23/04/2013
Criação do script interm. Sensor/Orange	22/04/2013	28/05/2013
Criação do script Metasploit	03/05/2013	03/05/2013
Criação do "modelo normal" de SysCalls	06/05/2013	09/05/2013
Criação do "modelo anormal" de SysCalls	10/05/2013	15/05/2013
Upgrade da microaplicação Python(Matriz de confusão)	23/05/2013	27/05/2013
Geração do classificador Naive Bayes	27/05/2013	28/05/2013
Dissertação da 2ª parte do relatório	20/05/2013	28/06/2013

Imagem 2 - Calendarização do planeamento do 2º semestre
Fonte: Documentação Interna Visionspace

Capítulo 3
Estado da arte

Este capítulo tem por objetivo definir os fundamentos relacionados ao estado da arte das soluções, tecnologias e técnicas já existentes no mercado para o âmbito do projeto aqui explorado. O módulo de detecção de intrusão utilizou-se de diversas ferramentas, soluções, mecanismos e técnicas de machine learning para as fases de seu desenvolvimento. Este capítulo será dividido em cinco partes, onde o primeiro subcapítulo descreverá as ferramentas utilizadas no mercado que utilizam tecnologias semelhantes as utilizadas neste módulo de detecção de intrusão, ou seja, soluções de cunho semelhante ao aqui desenvolvido. O segundo subcapítulo, descreverá a tecnologia Virtual Machine Introspection(VMI), a qual embora não tenha sido utilizada no desenvolvimento deste módulo, foi bastante estudada no início do projeto e é uma tecnologia que vem ascendendo nos campos de pesquisa em segurança, entretanto, ainda não apresenta resultados que possam ser utilizadas em ferramentas comerciais sem pôr em risco o futuro do projeto. No terceiro subcapítulo serão descritas as ferramentas de segurança utilizadas neste projeto e a motivação para a sua utilização, em detrimento de outras soluções existentes no mercado. Também serão descritas as técnicas de ataque comumente utilizadas, vulnerabilidades, ferramentas de análise forense, e afins. No quarto subcapítulo, serão descritas as técnicas de machine learning utilizadas na pesquisa e no desenvolvimento do módulo de detecção de intrusão. Também serão justificadas as técnicas utilizadas em detrimento as outras técnicas de machine learning possíveis para resolução do problema posto em questão. O quinto e último capítulo, é um apanhado de tecnologias, ferramentas e técnicas adjacentes, que foram incluídas ou utilizadas no projeto de alguma forma.

3.1 Virtual Machine Introspection(VMI)

Segundo Bishop[1], VMI, Virtual Machine Introspection é uma técnica de análise forense, baseada em máquinas virtuais, onde o Monitor de Máquina Vrtual (VMM) tem total controle sobre as suas máquinas virtuais. Essa abordagem permite que um especialista em segurança possa analisar todo o conteúdo de suas máquinas virtuais, usando a ótica "olhar para dentro". Esta técnica foi utilizada inicialmente por Tal Garfinkel e Mendel Rosenblum, onde eles propunham uma abordagem VMI-based para a detecção de intrusão. Segundo eles, esta tecnologia permite que se faça uma análise "out-of-box", ou seja, é possível realizar a análise de uma máquina possivelmente atacada, sem realizar nenhum tipo de acesso ao sistema operacional presente na mesma. Isto se torna possível pelo fato do Monitor de Máquina Virtual (VMM) prover acesso total a máquina atacada, permitindo um domínio de proteção de hardware completamente diferente, não permitindo a interação entre o atacante e o monitor de máquina virtual. O VMM também provê total acesso ao hardware da máquina virtual, característica esta que aumenta a gama de benefícios para uma abordagem host-based. Eles também elaboraram uma proposta de Sistema Detector de Intrusão(IDS), baseado em três propriedades do VMM:

Isolamento: Onde um software que está sendo executado em uma máquina virtual não pode alterar ou até mesmo acessar informações que estão presentes no VMM, ou em uma outra máquina virtual qualquer.

[1] Matt Bishop, University of California Davis Paper: VMI Matt Bishop

Inspeção: Onde o VMM tem acesso aos registradores, toda a memória e todos os dispositivos de entrada e saída, inclusive o conteúdo dos dispositivos de disco rígido.

Interposição: Onde o VMM tem de forma prioritária, a permissão de executar código de forma arbitrária, interpondo quaisquer outras operações, tanto da própria máquina analisada, como de um possível atacante. VMM's como Disco[2] e Denali[3] tem ambos uma arquitetura muito complexa, na ordem de 30 mil linhas de código, para que seja possível garantir essa característica.

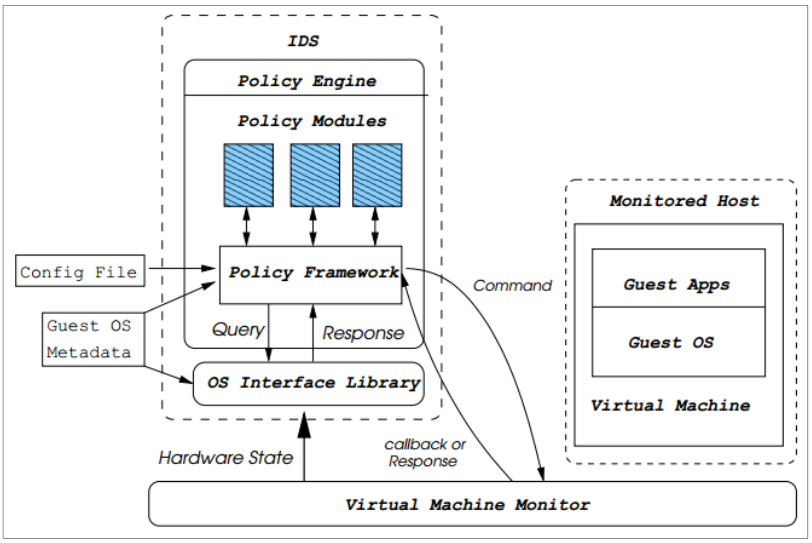


Imagem 3 - Exemplo de um IDS utilizando VMI
Fonte: Nance, K. ; Dept. of Comput. Sci., Univ. of Alaska at Fairbanks, Fairbanks, AK ; Bishop, M. ; Hay, Brian, 2009

A **Imagem 3**, exibe uma proposta de arquitetura de IDS utilizando VMI. Como podemos observar o host a ser monitorado é totalmente isolado da VMM, assim como da estrutura do IDS. Desta forma o VMM pode permitir ao IDS o monitoramento de muitos tipos de eventos da máquina monitorada, como por exemplo, eventos de CPU, de memória, entre outros. Entretanto, em alguns casos, a recuperação dessas informações pode ser bastante custosa, provendo assim uma perda significante de performance da máquina monitorada.

3.2 Segurança da Informação

O conceito de segurança da informação está diretamente ligado à preservação do bem mais importante e fundamental de qualquer organização: informação. Os pilares aqui descritos foram utilizados como base para o desenvolvimento do módulo de detecção de intrusão. Partindo deste ponto de vista, tal conceito visa de maneira prioritária a proteção à confidencialidade, autenticidade e disponibilidade das informações.

[2] E. Bugnion, S. Devine, and M. Rosenblum. Disco: running commodity operating systems on scalable multiprocessors.
[3] A. Whitaker, M. shaw, and S. D. Gribble. Scale and performance in the denali isolation kernel.

Segurança da Informação, segundo a própria norma ISO/IEC 17799:2005, é caracterizada como um conjunto de princípios considerados críticos, com o intuito de preservar a proteção da informação. Princípios estes, essenciais e fundamentais para o estabelecimento de padrões de segurança. A segurança da informação tem por base três pilares fundamentais:

Confidencialidade

Este é um princípio que descreve a garantia de que, uma informação será acessível apenas para os indivíduos autorizados. Em outras palavras, descreve que apenas as pessoas que devem acessar determinadas informações irão, de fato, acessá-las.

Integridade

Este princípio preza pela garantia da exatidão das informações a serem acessadas, assim como o acesso da mesma de forma íntegra e completa. Este conceito está diretamente ligado aos métodos de processamento e garantias que a mesma irá submeter-se. Em linhas gerais, trata-se de ter a certeza que a informação que fora gerada, será a mesma a ser acessada e utilizada.

Disponibilidade

Este princípio trata a garantia que o indivíduo terá de acessar as informações no momento em que necessitar, ou seja, garantirá que os mesmos estejam disponíveis sempre que necessário. Tal princípio também é largamente utilizado como indicador de nível de serviços de tecnologia em geral, uma vez que a sua percentagem permite a medida do tempo disponível do serviço.

A **Imagem 4** exibe uma representação gráfica dos pilares da segurança da informação:

Imagem 4 - Pilares da segurança da informação
Fonte: Adaptado de ALVES, Gustavo Alberto

3.3 Técnicas de Machine Learning

A organização do conhecimento em classes relacionadas é quase tão antigo quanto o conhecimento humano. Durante boa parte da história, documentos foram classificados em classes relacionadas de forma manual, até mesmo mais recentemente.

A obtenção de grandes volumes de dados rotulados e o impulso gerado pela crescente capacidade computacional de processamento ao desenvolvimento de métodos

de aprendizagem estatísticos vêm trazendo mais visibilidade às técnicas de aprendizagem de máquina.

A criação de máquinas que são capazes de aprender com a experiência sempre foi bastante discutida pela comunidade científica. Embora os limites dessa aprendizagem ainda não sejam conhecidos, no atual momento, essas máquinas podem receber um nível bastante significativo de capacidade de aprendizagem. Algumas dessas técnicas serão descritas aqui, servindo como embasamento teórico para as tecnologias que foram utilizadas neste projeto.

A possibilidade de realizar as classificações que até então eram feitas de forma manual Classificadores podem ser descritos como funções que são atribuídas a uma classe de objetos que são descritos por um conjunto de atributos. (MITCHELL, 2007)

3.3.1 Aprendizagem Supervisada

Este tipo de aprendizagem pode ser definida como toda aquela capaz de induzir um classificador, conhecendo suas entradas e suas saídas, a fim de predizer com alta confiabilidade as classes de exemplos futuros sem a necessidade de utilização de dados além do conjunto de exemplos de treino previamente utilizado. Neste tipo de aprendizagem, os dados de exemplos utilizados para a fase de treino já são previamente classificados, ou seja, conhece-se as características de um dado exemplo e a sua classificação final. Para este tipo de abordagem, podemos descrever três variáveis fundamentais:

Tarefa T: treinar um classificador que possa ser usado futuramente para classificar exemplos posteriores;

Performance P: que podemos descrever como a taxa de acerto(%) dos exemplos que foram classificados de maneira correta;

Experiência E: que podemos definir como sendo o dataset que será utilizado para realizar o treinamento do classificador, ou seja, um conjunto de dados previamente rotulados.

O processo de aprendizagem supervisada pode ser dividido em duas partes:

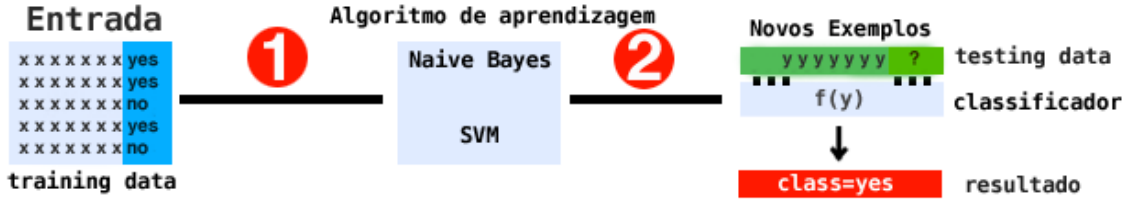


Imagem 5 - Exemplo de aprendizagem supervisada
Fonte: Ficheiro Pessoal

Como podemos verificar na **Imagem 5,** temos duas fases nesse tipo de aprendizagem, onde a fase 1 está ligada a fase de treino, ou fase de aprendizagem. Nesta fase é utilizado um dataset de entrada para treinar o classificador com o fim de utilizar as classificações previamente definidas como um ponto de partida. Neste ponto, cada exemplo representa um objeto definido pelas suas características e pela sua classe. Na fase 2, ou fase de teste, utilizaremos a função definida para induzir um classificador após a fase de treinamento para realizar novas classificações, em um novo conjunto de dados que possa ser definido

como entrada. Após a aplicação dessa função, podemos inferir a classe para cada conjunto de dados inseridos no classificador, ou seja, este classificador poderá ser utilizado para classificação de exemplos futuros.

Podemos definir dois esquemas de aprendizagem, online e offline.

No esquema de aprendizagem offline, a fase de treino é utilizada para induzir o classificador e após esta tarefa utilizaremos um novo conjunto de dados para execução da fase de testes. Apenas após a execução de ambas e validação dos resultados da fase de testes é que se deve prosseguir com a implementação de um classificador, que por sua vez, não permitirá o aprendizado com base em novos dados de treino.

No esquema de aprendizagem online, a fase de "treino" é compreendida tanto antes da execução da fase de testes como também após a implementação de um classificador previamente induzido. Em outras palavras, em tempo real o classificador poderá ser capaz de aprender novos conceitos e aprimorar de acordo com a quantidade de exemplos que são inseridos ao longo da execução do mesmo. Desta forma, podemos obter classificações mais aprimoradas, contudo, corre-se o risco de tornar o classificador mais genérico, ou seja, sua função de classificação pode variar positivamente ou negativamente ao longo do tempo.

Para definir o critério de sucesso para a aprendizagem, tomamos por base a porcentagem de acerto na classificação dos exemplos na fase de testes utilizando o classificador. (RUSSEL e NORVIG, 1995) Este critério é baseado na taxa de assertibilidade dos itens classificados. Comumente, é utilizada a curva ROC para ilustrar essa taxa de assertibilidade.

Segundo BRAGA[4], a curva ROC(Receiver Operating Characteristic) é um gráfico que relaciona a sensibilidade (ou taxa de verdadeiros positivos (TPR) com a taxa de falsos positivos (FPR)) que demonstra de forma eficiente a relação antagónica entre a sensibilidade e especificidade dos exemplos classificados.

As curvas ROC foram desenvolvidas no âmbito da teoria de detecção de sinal, por engenheiros de radares durante a Segunda Guerra Mundial para detecção de itens não-reconhecidos em campos de batalha.

[4] Braga ACS. Curvas ROC: Aspectos funcionais e aplicações. Universidade do Minho, dezembro de 2003, Portugal

A **Imagem 6** ilusta a utilização da curva ROC:

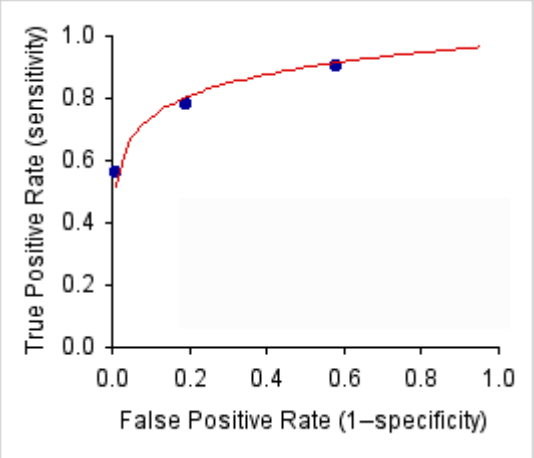


Imagem 6 - Exemplo de utilização da curva ROC
Fonte: Vassar College, NY, 1998
Disponível em: http://www.vassarstats.net/

3.3.2 Aprendizagem Não-Supervisada

Neste tipo de aprendizagem os dados utilizados para a fase de treino não estão previamente classificados, ou seja, não é conhecida as suas entradas e saídas. Esta abordagem é bastante utilizada em mineração de dados, tendo em vista a grande quantidade de dados extraídas de grandes bases de dados, onde a classificação para um determinado exemplo não é previamente conhecida. Esta se dá apenas por descoberta de relações, padrões, regularidades ou categorias de dados.O principal interesse desta abordagem é a descoberta destes padrões e o agrupamento dos mesmos em *clusters* de forma consistente. Os critérios de *clusterização* não são tão bem definidos, tendo em vista a dificuldade em classificar a similaridade dos exemplos que estão a ser trabalhados. Esta abordagem apresenta alguns paradigmas, como por exemplo, a representação da entrada de dados, de modo a garantir toda a estrutura da coleção de dados; as representação das saídas desejadas ou avaliação externa das saídas produzidas em oposição a abordagem de aprendizagem supervisada, entre outras. (OGURI, 2006)

A **Imagem 7**, exibe um exemplo de uma entrada de dados comum à utilização de aprendizagem não supervisada, onde não são conhecidos previamente os atributos de classificação:

Temp. Max.	Humidade	Nuvens	Ondulação
31º	10%	Altas	2m
10º	80%	Carregadas	3m
35º	9%	Sem nuvens	1.5m
36º	8%	Sem nuvens	3m
2º	90%	Carregadas	3m
27º	15%	Altas	1m

Imagem 7 - Exemplo de dados de entrada para aprendizagem não-supervisada
Fonte: Material Prof. Luís Nunes, ISCTE,2010

3.3.3 Naive Bayes

O classificador Naive Bayes é o mais utilizado em machine learning por sua versatilidade e poder adaptativo a diversas situações. "Naive" ou ingênuo, é assim definido por assumir que todas as suas características são condicionalmente independentes, permitindo assim a sua decomposição num produto de n-termos, onde cada termo representa uma característica do conjunto de dados. (CHAKRABARTI, 2002)

Mesmo assumindo esta abordagem simplista, este classificador apresenta a melhor perfomance em várias tarefas de classificação, como é amplamente discutido em (CHAKRABARTI, 2002) e (MCCALLUM e NIGAM, 1998).

Se um espaço de características apresenta um número elevado de dimensões, ou seja, possui muitas características e muitos valores assumíveis para cada característica, este algoritmo é extremamente indicado para utilização. (LANGLEY, 1992)

Nas primeiras décadas do século XVIII, vários problemas relacionados a probabilidade estavam resolvidos, tendo em vista certas condições, os ditos "forward probability". Um exemplo deste problema é: ao lançar um dado, qual a probabilidade de obter um determinado número, ou, dado um número conhecido de bolas brancas e pretas em uma urna, qual a probabilidade de ser sorteada uma bola branca. (BERNSTEIN, 1996) Entretanto, o problema inverso tornou-se difícil para resolução. Ao sortear n bolas da urna, o que poderia ser inferido acerca das bolas restantes na urna? Nesta abordagem, Thomas Bayes formalizou de forma pioneira uma teoria acerca de problemas desta natureza. (BAYES, 1763)

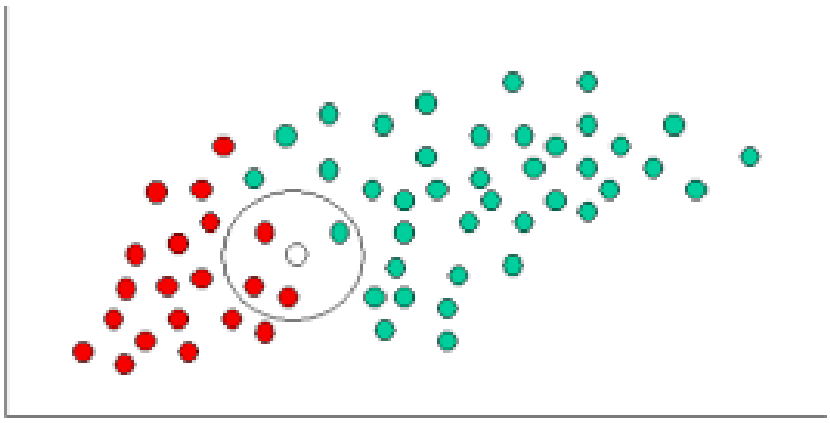


Imagem 8 - Representação da abordagem encontrada no classificador Naive Bayes
Fonte: Statsoft Electronic StatisticsTextbook

A **Imagem 8** exibe uma representação da situação encontrada no problema resolvido por Thomas Bayes, onde encontramos uma quantidade x exemplos positivos e y de exemplos negativos. Ao utilizar um novo exemplo, qual a probabilidade do mesmo pertencer aos dois grupos, dos exemplos positivos e negativos.

Este problema é justamente o mesmo que encontramos ao treinar um classificador, onde na fase de classificação calculamos uma distribuição geradora Pr(d|c) para cada classe c.

A **Imagem 9** exibe um conjunto de dados de treino, como exemplo para os passos a serem seguidos na **Tabela 1**, para classificação de um exemplo x:

x =(age<=30 , income =medium, student=yes, credit_rating=fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Imagem 9 - Conjunto de dados para treino utilizando Naive Bayes
Fonte: Data Mining: Concepts and Techniques, Han & Kamber, 2006

Para classificação de um determinado conjunto de dados utilizando *Naive Bayes*, devemos seguir alguns passos, conforme descrito na **Tabela 1**:

Tabela 1 - Classificação utilizando Naive Bayes

Passo	Exemplo
Estimar as probabilidades apriori P(cj) de cada classe	P(buys_computer = “yes”) = 9/14 = 0.643 P(buys_computer = “no”) = 5/14 = 0.357
Estimar as condicionais P(xi|cj) para cada atributo	P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222 P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6 P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444 P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4 P(student = “yes” | buys_computer = “yes”) = 6/9 = 0.667 P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2 P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667 P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

Calcular P(x|cj) para cada classe	P(x|buys_computer =“yes”)=0.222 * 0.444 * 0.667 * 0.667= 0.044 P(x|buys_computer = “no”) = 0.6 * 0.4 * 0.2 * 0.4 = 0.019
Usar Teorema de Bayes para calcular P(cj |x) para cada classe	P(buys_computer = “yes” |x) = P(x|buys_computer = “yes”) * P(buys_computer = “yes”)= **0.028** P(buys_computer= “no” |x) = P(x|buys_computer = “no”) * P(buys_computer = “no”) = 0.007

Tendo em vista que o produto das probabilidades calculadas utilizando o Teorema de Bayes foi maior para a classe “**yes**”, o exemplo x =(age<=30 , income =medium, student=yes, credit_rating=fair), deverá ser classificado como “**yes**”.

3.3.4 Abordagem de representação de System Calls em n-grams

Podemos descrever um n-gram como um conjunto de elementos de ordem N consecutivos, usado comumente quando se pretende representar dados que necessitem respeitar a sequência dos elementos. Neste modelo, as características representam a ocorrência de N elementos consecutivos. (OGURI, 2006)

Utilizamos esta abordagem para representar as características do conjunto de dados utilizados nesta dissertação.

Na **Imagem 10**, podemos verificar um exemplo desta abordagem:

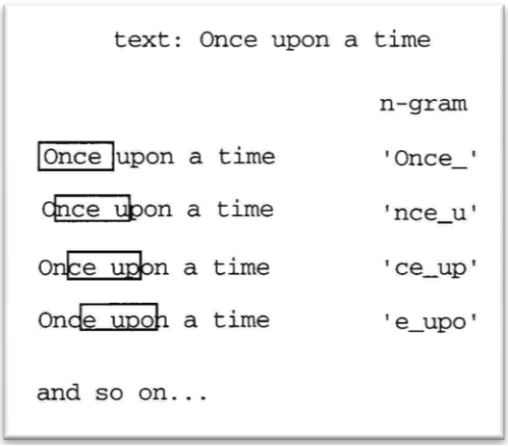


Imagem 10 - Exemplo da abordagem em n-grams
Fonte: GUSTAVSSON, Jonas, 1996

3.3.5 Matriz de confusão

Podemos definir uma matriz de confusão como sendo uma tabela que contém informações acerca das classificações reais e previstas das classes presentes no dataset utilizado pelo classificador e que serve para medir a performance de sistemas de classificação. (KOHAVI e PROVOST, 1998)

A **Imagem 11** exibe um exemplo de uma matriz de confusão, exibindo as colunas de dados reais e previstos:

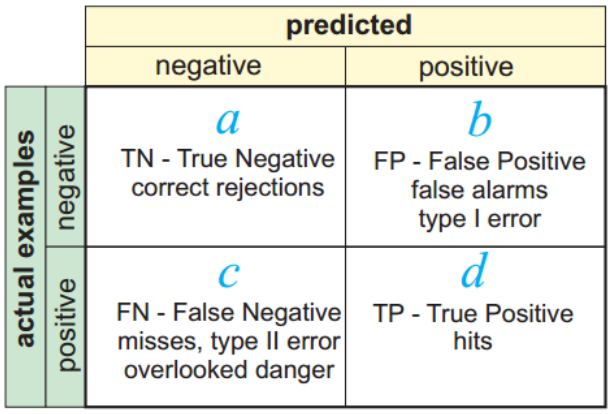


Imagem 11 - Exemplo de uma matriz de confusão
Fonte: KOHAVI e PROVOST, 1998

As métricas utilizadas para obter os dados de desempenho através da matriz de confusão são definidas conforme podemos observar na **Tabela 2,** com base na **Imagem 11**:

Tabela 2 - Métricas utilizadas para medição de desempenho com base na matriz de confusão

Métrica	Função
Precisão	(TN + TP)/total
Sensibilidade (True Positive Rate(TPR))	TP/actual examples positive
Especificidade	TN/actual examples negative
Repetibilidade	TP/predicted positive
False Positive Rate (FPR)	FP/actual examples negative
False Negative Rate	FN/actual examples positive

Na **Tabela 2**, podemos observar as duas taxas mais importantes para medição do desempenho de um classificador, TPR (**True Positive Rate**) e o FPR (**False Positive Rate**).

3.4. Tecnologias de desenvolvimento

3.4.1 DAO

Pode ser descrito como um objeto que fornece interface abstrata para algum tipo de objeto de uma base de dados. Um DAO fornecerá propriedades inerentes ao objeto como também, métodos de acesso as informações, tanto de recuperação como de alteração dos dados. Este isolamento fornece uma estrutura que separa as preocupações de acesso a bases de dados, com outras preocupações que uma aplicação venha a ter. Embora ela possa ser aplicada à maioria das linguagens de programação é comumente associada a tecnologias de aplicações web. Neste relatório os mesmos são utilizados para mapear os objetos da base de dados do software Delusion, para criação da API que fora implementada e descrita neste relatório. Da mesma forma ela é tradicionalmente associada à tecnologia ODBC.

3.4.2 Project jc-tree

Este projeto visa à implementação de árvores com um número indeterminado de nós, de tamanho também variável em *Java*, de forma simples e eficiente. Para isso, são utilizadas coleções Java, dentre elas, a ArrayList é utilizada para implementação da ArrayListTree, que será utilizada neste relatório para representar uma árvore de processos, com seus processos filhos associados. Outras soluções deste projeto, também visam a implementação de auto-balanceamento de árvores, subárvores na interface da ávore e SortedChildrenTree, que estende a ArrayListTree, mas mantém os nós filhos ordenados.

3.5. Tecnologias de sistemas e segurança da informação

3.5.1 SSL (Secure Socket Layer)

É um protocolo da camada de transporte capaz de prover privacidade e integridade de informações para serviços disponibilizados na internet, como SMTP, HTTP, SSH, entre outros. Para realizar tal função as partes envolvidas realizam autenticação na conexão e na transmissão dos dados transmitidos entre as mesmas. Entre outros tipos de hacking, este protocolo previne o famoso ataque man-in-the-middle, onde um intermediário intercepta uma conexão e se faz passar por uma das partes que originalmente realizaram a conexão, a fim de capturar dados. Este protocolo foi revisto algumas vezes, até se chamar TLS (Transport Layer Security), que nada mais é do que uma "versão 3.1" do antigo protocolo SSL. Este protocolo se baseia no conceito do método para troca de chaves Diffie-Hellman, que foi criado por Whitfield Diffie e Martin Hellman com o intuito de realizar uma troca rápida de chaves de criptografia, entre os envolvidos na conexão. O usuário A gera uma chave a partir da chave privada dele e a pública do usuário B. O usuário B gera uma chave a partir da chave privada dele e a pública do usuário A.

3.5.2 System Calls

Uma System Call (chamada de sistema) pode ser descrita como um programa que solicita um serviço a partir de chamadas feitas ao kernel do sistema operacional. Tais serviços incluem chamadas a recursos de hardware ou de software, como acesso a periféricos ou criação de novos processos, por exemplo. Estas foram criadas tendo em vista que os sistemas operacionais modernos necessitam de uma diferenciação a ser utilizada no sistema de prioridades do mesmo, para que programas invocados pelo sistema operacional e outros programas invocados diretamente pelo usuário sejam diferenciados. Nos sistemas baseados no Unix, tais chamadas são programas escritos em C, que são parte de uma API escrita essencialmente para disponibilizar funções que são comumente utilizadas pelo sistema operacional. Alguns exemplos de System Calls (Unix-like): open(), close(), write(), wait(), fork(), etc. A maioria dos sistemas operacionais possuem centenas de chamadas de sistema implementadas, como o Linux que tem em cerca de 300 chamadas de sistema. Outros, como o sistema operacional Plan9, possuem apenas 51 chamadas de sistema. Quando um programa escrito por um usuário necessita realizar uma operação que requer maior privilégio de execução, o sistema operacional realiza uma interrupção e o controle de execução é revertido para o sistema operacional. O mesmo invoca uma chamada de sistema que vai realizar o serviço requisitado pelo programa do usuário, onde após a execução e retorno do resultado, o controle de execução volta para a aplicação que originalmente estava a ser executada pelo usuário. Um exemplo de serviço que é executado por uma System Call é a abertura de um ficheiro no disco, para leitura dos dados. As System Calls são comumente associadas à funções para a realização de:

- Controle de Processos: call(), read(), execute(), kill(), wait(), etc. ;
- Gerenciamento de ficheiros: create(), delete(), open(), read(), etc. ;
- Gerenciamento de dispositivos: request, release, read, attach/detach, etc. ;
- Manutenção da informação: get/set system data, process, file ou device, etc. ;
- Comunicação: send(), receive(), create(), etc. ..

3.5.3 Processo

Um processo pode ser definido como um software em execução no sistema operacional (SO). Comumente, ele possui o código do programa executado e possui metadados acerca da sua atual situação. Um SO multitarefa comum, executa vários processos ao mesmo tempo, compartilhando utilização de CPU, memória e estando sob a gestão do compartilhamento de hardware, realizada pelo SO. Os SO's modernos também proveêm a comunicação inter-processos(IPC), proporcionando que os mesmos compartilhem acessos a dispositivos e recursos.

Comumente, o processo possui os seguintes recursos:

- Cópia do executável do programa;
- Área de memória, que contém o executável do programa, uma pilha de chamadas de sistema e informações acerca dos dados de entrada e saída;
- Descritores de ficheiros(File Descriptors);
- Atributos de segurança relacionados as permissões do processo;
- Processadores de estado, que conteêm informações de registro, memória e endereçamento físico.

O sistema operacional mantém a maior parte destas informações em estruturas chamadas PCB (Process Control Blocks), ou blocos de controle de processo. O objeto principal de estudo deste relatório é a análise de chamadas de sistema feitas através de processos no sistema operacional Linux, onde os processos podem ser representados por uma estrutura em árvore, onde cada processo pai (nó raiz) pode conter vários processos filhos (nós filhos) e estes nós filhos, conter outros vários nós filhos e assim por diante.

3.5.4 Honeypot

Um honeypot (pote de mel) pode ser definido como um conjunto de armadilhas a fim de criar um ambiente abstraído de um ambiente de produção, que tem o objetivo principal de captar informações de um possível ataque a infraestrutura, para um possível rastreamento do atacante e posteriormente tomar as possíveis medidas legais cabíveis disponíveis no país onde foi executado. Os honeypot's podem ter várias classificações, entretanto, as mais simples delas os diferem pelo seu propósito ou pela relação de interação:

Honeypot de produção: são mais simples e fáceis de implantação. Normalmente rodam dentro da mesma rede onde os serviços de produção estão rodando. Em princípio, não retornam grandes informações sobre o ataque e/ou o atacante, o que pode ser visto como algo que agrega pouco valor à segurança disposta em serviços computacionais;

Honeypot de pesquisa: são honeypots criados para a obtenção de um grau elevado de informações sobre técnicas de ataque, visando a melhoria da segurança computacional dos serviços de produção. São mais difíceis de implementar e implantar, uma vez que são mais utilizados em organizações governamentais, militares e etc.

Honeypot de alta interação: estes tem por objetivo principal simular o comportamento de um sistema de produção de maneira idêntica, utilizando a mesma gama de serviços disponíveis nos sistemas de produção. Oferecem um grau elevado de segurança tendo em vista a sua difícil identificação, uma vez que o mesmo não possui quaisquer diferenças de um ambiente que o atacante espera encontrar. Uma Honeynet especifica uma rede de honeypot's de alta-interação, simulando a rede inteira de serviços disponibilizados. Entretanto, esta técnica é bastante dispendiosa o que dificulta a sua implantação;

- Honeypot de baixa interação: estes honeypot's tentam simular apenas os principais serviços comumente utilizado pelos atacantes em uma rede comum. Por se tratar de domínios bem menores, a sua implantação tende a ser interessante e consequentemente, uma solução mais barata.

3.5.5 Metasploit

O Metasploit é um sistema criado para ajudar especialistas em segurança computacional a identificarem e mitigarem vulnerabilidades, aplicando inteligência de segurança de riscos a serviços disponibilizados na internet. Esta ferramenta oferece suporte à equipes de segurança relatarem e organizarem a gestão de riscos, disponibilizando uma interface de monitoria de recursos computacionais gerando relatórios de análise dos problemas identificados. Neste relatório, o Metasploit foi utilizado para gerar as sequências de chamadas de sistema que poderiam corromper o serviço web em execução, servindo assim, uma sequência de System Calls que visa à corrupção e possível brecha de segurança do sistema operacional.

3.5.6 Vulnerabilidade

Uma vulnerabilidade pode ser descrita como uma brecha de segurança em um sistema computacional. A possível exploração desta brecha pode trazer complicações de funcionamento ao sistema em questão, ou até mesmo, prover acesso a um agente externo ao sistema, podendo comprometer outros serviços e sistemas disponibilizados numa organização. As vulnerabilidades mais comuns nos dias de hoje podem prover a utilização de algumas técnicas específicas de ataque a sistemas computacionais, como buffer overflow, DDos, entre outros. As ferramentas implementadas para explorar tais vulnerabilidades são chamadas de exploits e comumente são escritas em linguagens como C, Assembly e Python.

3.5.7 Delusion

É um *Virtual-Appliance* que tem como objetivo principal coletar informações de rede e de máquinas específicas onde serão instalados serviços com falhas conhecidas. Funcionando como um *honeypot*, a idéia é prover tais informações para o administrador da rede, através de inspetores espalhados pelos segmentos da rede, a fim de obter uma maior cobertura de análise do tráfego transmitido.

Capítulo 4
Implementação do módulo de detecção de intrusão

Neste capítulo, será abordada uma explanação geral acerca de toda a implementação do módulo de detecção de intrusão, parte esta, integrante do software Delusion. Serão apresentados os problemas, os detalhes, a motivação e as propriedades ligadas ao desenvolvimento das componentes deste módulo.

4.1 Detalhes gerais

Antes de descrever a estrutura e as componentes presentes neste módulo, iremos descrever a motivação por trás do desenvolvimento deste módulo, ou seja, qual a justificativa da implementação deste módulo num software IDS como assim é denominado o Delusion.

O Delusion, por si só, segundo a documentação oficial, "permite monitoramento, em tempo real, das atividades lançadas pelo sistema operacional, através de um sensor desenvolvido para resgatar informações das system calls e utilizá-las a fim de análise". Comercialmente falando, isto poderia representar um problema, pois o Delusion não traria nenhum benefício quanto à detecção da intrusão, com base nos dados das system calls capturadas, característica esta que define um software IDS.

Tendo em vista o background apresentado, percebemos a necessidade de desenvolvimento de um módulo capaz de realizar ações de reação, após análise do material coletado pelo sensor, uma vez que o mesmo apenas analisa tais informações, mas não infere nenhum tipo de resultado, decisão ou ação de acordo com os dados encontrados. Seguindo esta linha e tendo em vista que os dados coletados, são lançados pelo sistema operacional na forma de system calls, percebemos após pesquisa na vasta literatura presente acerca de detecção de intrusão, uma linha de pesquisa que tinha como foco a análise das sequências de execução de system calls, como forma de detectar uma situação de risco para o sistema operacional.

Entretanto, estas abordagens presentes na literatura tratavam apenas pequenas aplicações disponíveis em sistemas operacionais GNU/Linux, como por exemplo, sendmail(agente de transferência de correio), lpr(gerenciador de filas de impressão) e wu.ftpd(servidor de transferência de ficheiros), conforme literatura escrita em 1998, desenvolvida no departamento de ciências da computação da Universidade do Novo México, Estados Unidos. Este projeto de pesquisa foi liderado pela Dra. Stephanie Forrest e descrevia um método para detecção de intrusões em sistemas GNU/Linux, baseado na análise de sequência de system calls executadas. Neste artigo, foram descritas as formas de representação do material coletado, para que fosse possível inferenciar se tais comportamentos coletados seriam classificados como "normais" ou "anormais". Esta abordagem define um modelo normal, advindo de uma coleta dos dados de execução das system calls no sistema operacional quando o mesmo se encontra em uma situação normal de utilização. Para fins de classificação, esta abordagem também define um modelo dito "anormal", onde estes softwares são utilizados a fim de explorar alguma vulnerabilidade, gerando assim, uma utilização não-adequada do mesmo.

Esta abordagem é utilizada em vários trabalhos presentes na literatura, entretanto, a forma de detecção de intrusão é explanada de maneira ampla através deles. Entretanto,

uma abordagem que é bastante utilizada é a criação de um classificador Naive Bayes induzido, pela forma como o mesmo consegue se adaptar em várias situações. Entretanto, nessa pesquisa literária, não conseguimos identificar a utilização de um serviço que fosse utilizado de forma ampla na internet, neste tipo de abordagem. Essa foi a principal motivação da implementação deste módulo, utilizar um serviço gerenciador de bases de dados(SGBD), que pode ser utilizado em um ambiente de produção. Utilizamos um SGBD pelo fato de que a maior parte das empresas mantém um serviço como este sendo executado em seu ambiente e desta forma, conseguiríamos realizar a validação deste método de detecção de intrusão, para que o mesmo possa ser utilizado de forma comercial no software Delusion.

4.2 Estrutura do módulo

O módulo de detecção de intrusão baseado nas sequências de system calls foi estruturado tomando por base um importante requisito não-funcional definido no capítulo 3, a compatibilidade com a arquitetura já existente no Delusion. Desta forma, o módulo deveria ser de fácil acoplamento ao software Delusion, permitindo assim uma posterior facilidade para desenvolvimento de uma interface web capaz de gerir o mesmo.

Como podemos observar na **Imagem 12**, a componente do módulo de detecção vem a ser acoplada na arquitetura do Delusion, permitindo que esta tarefa não demandasse alterações na mesma:

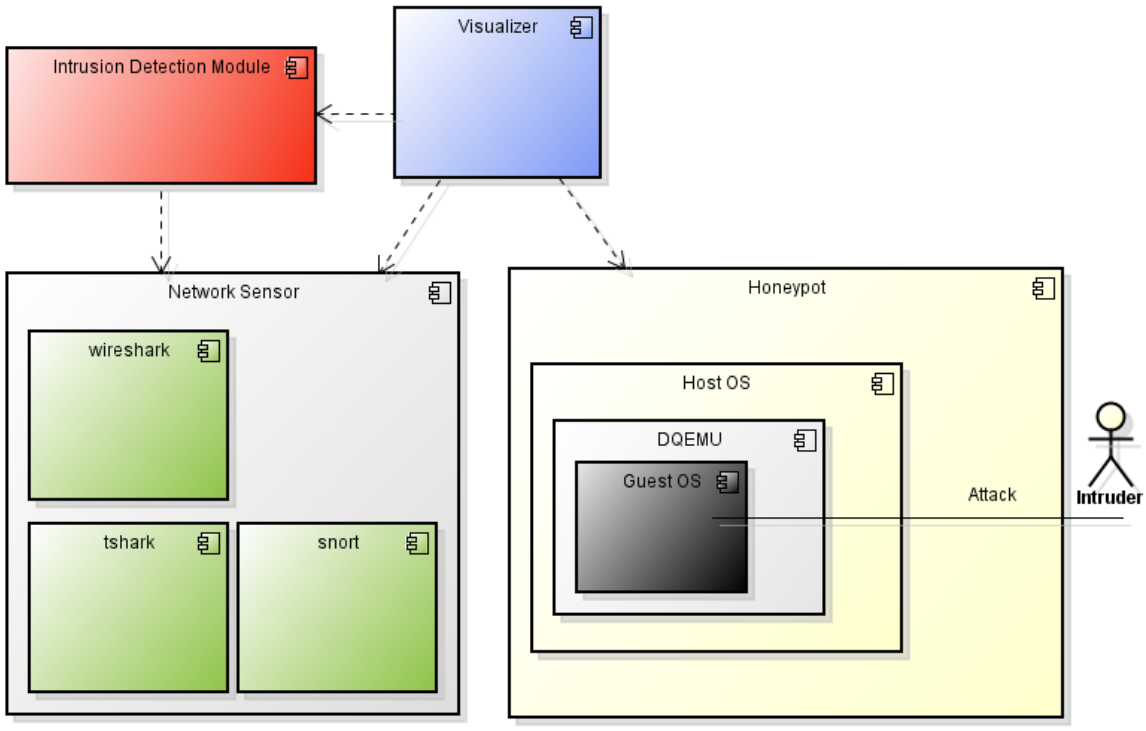


Imagem 12 - Acoplamento do módulo de detecção de intrusão na arquitetura do Delusion
Fonte: Adaptação da documentação oficial do software Delusion

Conforme podemos observar na **Imagem 12**, o módulo de detecção de intrusão (Intrusion Detection Module), foi acoplado a fim de obter os dados coletados pelo sensor (Network Sensor) do Delusion e ser capaz de dispor o resultado de suas atividades à interface web do Delusion (Visualizer). Desta forma, não modificamos a arquitetura do Delusion,

consequentemente criamos a possibilidade de acoplar diversas outras técnicas de detecção de intrusão que futuramente venham a se tornar interessante comercialmente a empresa, sem que haja a necessidade de modificação na estrutura já desenvolvida para o produto.

É importante salientar, que esta decisão foi tomada não só pela facilidade de desenvolvimento deste módulo de forma separada, mas também, pela dificuldade em vincular outras tecnologias ao produto, tendo em vista que são utilizadas outras linguagens de programação, tecnologias e que a maior parte dos seus criadores, já não fazem parte do desenvolvimento deste produto, o que torna esta abordagem a mais correta dentre as outras possíveis.

A estrutura do módulo de detecção foi concebida através de algumas fases, que serão descritas na **Tabela 2**:

Tabela 3 - Fases da estruturação do módulo detector de intrusão

Fase	Descrição
1. Captura dos dados	Nesta fase, os dados gerados pelo sensor do Delusion são carregados pelo módulo de detecção de intrusão.
2. Geração do modelo normal	Nesta fase, foram gerados os modelos de utilização pré-classificados como "normal", servindo como base para indução do classificador Naive Bayes.
3. Geração do modelo "anormal"	Nesta fase, foram gerados os modelos de utilização pré-classificados como "anormais, utilizando-se de um exploit para explorar uma vulnerabilidade já conhecida do sistema gestor de base de dados.
4. Geração do ficheiro de features	Nesta fase, foi gerado um ficheiro contendo o dataset que será utilizado para a fase de treino na indução do classificador Naive Bayes, com exemplos de utilizações normais e anómalas. É também nesta fase que as features são representadas como n-grams, onde cada n-gram é na verdade, uma sequência curta e pré-definida de system calls.
5. Fase de treino	Nesta fase, o ficheiro que foi gerado na **Fase 4**, serve como entrada para que o classificador Naive Bayes "aprenda" quais são as sequências que foram classificadas como normais e anómalas, e consequentemente definida a função do classificador para futuros exemplos de sequências de system calls.
6. Fase de teste	Nesta fase, o classificador Naive Bayes já tem a função probabilística definida a partir da **Fase 5** e

	consequentemente já possui a capacidade de calcular a probabilidade de uma sequência de system calls que foi executada no sistema operacional ser "normal" ou "anómala".
7. Execução do módulo detector de intrusão	Nesta fase, o módulo detector de intrusão recebe dados da utilização real do sistema gestor de base de dados, diretamente do sensor do Delusion e classifica os mesmos como "normais" ou "anômalos", tendo em vista a função definida na **Fase 5**.

A **Fase 3** da **Tabela 2** merece uma atenção especial, pelo fato de ter sido gerada utilizando uma ferramenta chamada Metasploit. A utilização desta ferramenta foi crucial para a obtenção do modelo "anormal" de utilização, pois, a mesma permitiu que fossem geradas tentativas de intrusão ao SGBD. Estas tentativas foram capturadas pelo sensor do Delusion para posterior utilização na geração do ficheiro descrito na **Fase 4**. Este ficheiro possui exemplos positivos e negativos de utilização, separados em micro-sequências de System Calls durante a execução do SGBD.

A **Imagem 13** exibe um exemplo sucinto do ficheiro gerado na **Fase 4**, com a respectiva coluna de classificação em destaque:

	label	read	write	accept	attack
13669	rt_sigprocmask-fcntl-alarm-read	1	0	0	0
13670	fcntl-alarm-read-rt_sigprocmask	1	0	0	0
13671	alarm-read-rt_sigprocmask-rt_sigtimedwait	1	0	0	0
13672	read-rt_sigprocmask-rt_sigtimedwait-read	2	0	0	0
13673	rt_sigprocmask-rt_sigtimedwait-read-rt_sigprocmask	1	0	0	0
13674	rt_sigtimedwait-read-rt_sigprocmask-rt_sigprocmask	1	0	0	0
13675	read-rt_sigprocmask-rt_sigprocmask-fcntl	1	0	0	1
13676	rt_sigprocmask-rt_sigprocmask-fcntl-write	0	1	0	1
13677	rt_sigprocmask-fcntl-write-shutdown	0	1	0	1
13678	fcntl-write-shutdown-close	0	1	0	1
13679	write-shutdown-close-madvise	0	1	0	1
13680	shutdown-close-madvise-_exit	0	0	0	1
13681	close-madvise-_exit-select	0	0	0	0
13682	madvise-_exit-select-select	0	0	0	0
13683	_exit-select-select-select	0	0	0	0
13684	select-select-select-select	0	0	0	0
13685	select-select-select-read	1	0	0	0
13686	select-select-read-rt_sigprocmask	1	0	0	0
13687	select-read-rt_sigprocmask-rt_sigprocmask	1	0	0	0
13688	read-rt_sigprocmask-rt_sigprocmask-fcntl	1	0	0	1

Imagem 13 - Exemplo sucinto do ficheiro de features gerado com a coluna de classificação
Fonte: Ficheiro pessoal

4.3 Pré-processamento para recolha de dados

Foram utilizadas algumas ferramentas criadas durante o projeto para que fosse possível a geração do ficheiro de features base para indução do classificador Naive Bayes. Inicialmente a recolha de dados do sensor do Delusion gera um ficheiro com as System Calls executadas em sequência, uma por linha. Este ficheiro necessita ser tratado e várias transformações precisam ser feitas até o ficheiro de features seja montado.

A primeira dessas transformações se dá na definição do tamanho das sequências que serão analisadas, ou seja, são geradas sequências de ordem n, utilizando uma janela deslizante para que possam ser representadas todas as sequências possíveis do ficheiro inicial. Para execução desta etapa, foram testadas sequências de vários tamanhos, baseadas em experiências prévias presentes na literatura, para situações semelhantes no âmbito da detecção de intrusão. Segundo WESPI, o tamanho da sequência a ser analisada se refere ao nível de agregação N, onde N = 1,2,3 significa um número N ou maior de System Calls que são agregadas para geração dos exemplos (ex: write-read-write) extraídos do dataset capturado pelo sensor do Delusion.

Outro ponto importante a ser explicado é a forma como as features são representadas. Segundo HUBALLI, a modelagem das features utilizando a abordagem de pequenas sequências de System Calls na forma de n-grams permite representar as características comuns a comportamentos "normais" e "anômalos" para cada exemplo presente no dataset gerado a partir dos dados obtidos pelo sensor do Delusion.

4.4 Metodologia aplicada

A metodologia aplicada para obtenção destas sequências foi baseada na comparação da capacidade de classificação utilizando um número fixo de System Calls (N), incrementado à medida em que a mesma apresenta resultados mais consistentes. Este número foi incrementado partindo de uma base de duas System Calls (N=2) e chegamos a conclusão que com um N=10, seria possível induzir o classificador de Bayes de modo a nos permitir à obtenção de um resultado mais preciso. A **Imagem 14** exibe o panorama geral aplicado para obtenção dos dados, tradução, agregação e extração de padrões para indução do classificador de Bayes, para sua posterior utilização em um ambiente de produção:

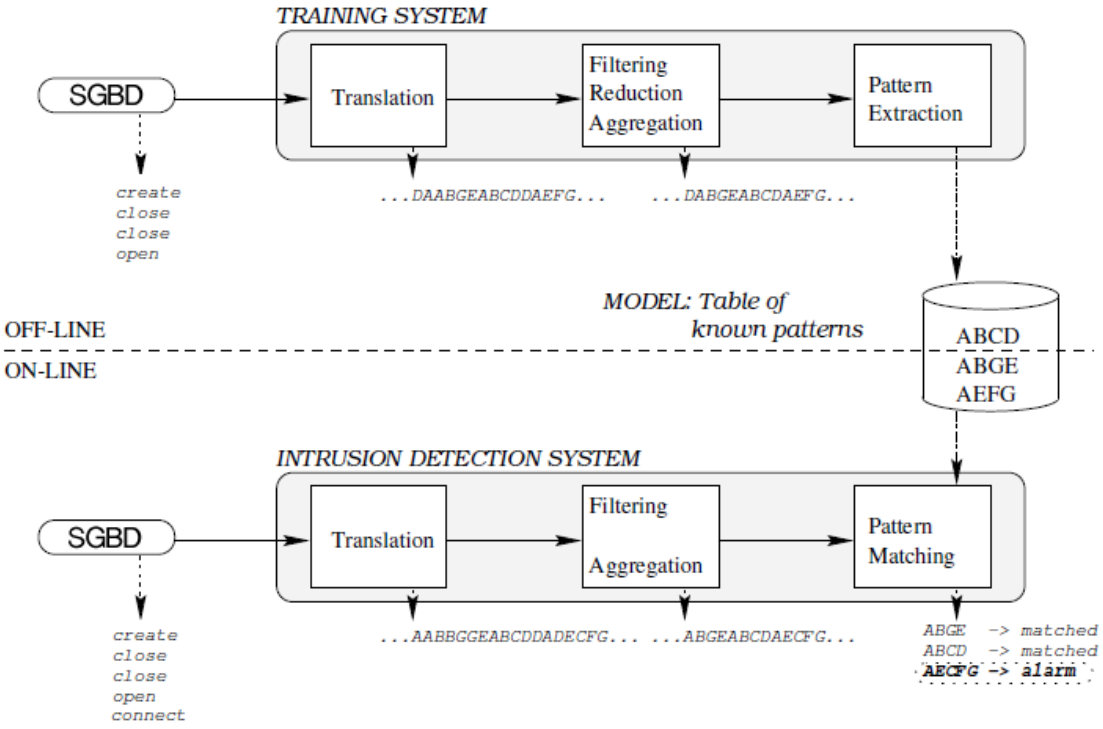


Imagem 14 - Sequência utilizada para modelagem do módulo detector de intrusão
Fonte: Adaptada de [29]

4.5 Ambiente para geração do ficheiro de features

Para definir o ambiente utilizado para obtenção dos dados presentes no ficheiro de features (responsáveis pela fase de indução do classificador Naive Bayes), foram utilizadas várias ferramentas aninhadas que executaram funções deste o tratamento de possíveis dados desnecessários para a classificação (como por exemplo, os argumentos das System Calls), agregação de System Calls em micro sequências de tamanho N, geração de n-grams e geração de colunas de classificação para cada exemplo de sequência dada. A **Imagem 15** exibe um esboço do ambiente utilizado para geração do ficheiro de features:

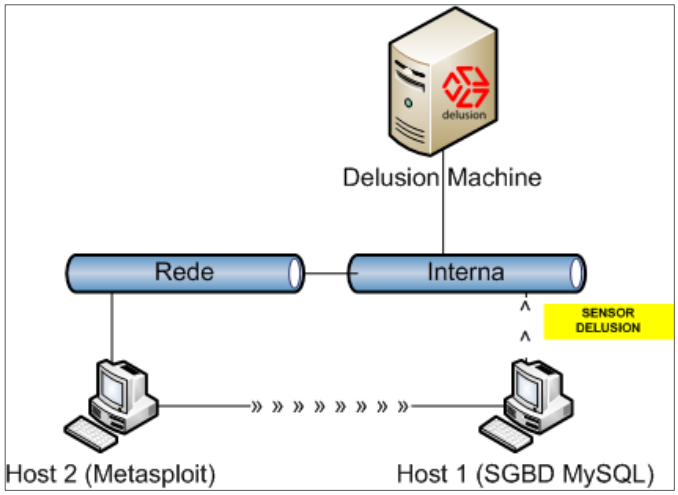


Imagem 15 - Esboço do ambiente para geração dos dados presentes no ficheiro de features
Fonte: Arquivo Pessoal

4.6 Tecnologias associadas

Para utilização do classificador Naive Bayes foi utilizada uma biblioteca de programação escrita em Python, chamada Orange. Esta biblioteca foi criada no âmbito da Faculdade de Computação e Ciência da Informação da Universidade de Ljubljana, Slovenia. Esta biblioteca é open-source e inclui, além de outras funções, um suporte a técnicas e componentes para machine learning. A utilização desta biblioteca foi crucial para a criação do classificador de Bayes, uma vez que a mesma possui vários componentes capazes de testar e validar os resultados provenientes da fase de treino e gerando os dados necessários para uma análise concisa dos métodos utilizados.

Podemos observar com maiores detalhes a utilização desta biblioteca no subcapítulo **4.6**, onde foram utilizados diversos métodos da mesma, capazes de realizar a validação dos resultados encontrados pelo classificador e o grau de assertibilidade da solução aqui desenvolvida. Para os testes iniciais, realizados no início da montagem deste módulo de detecção, utilizamos a interface gráfica presente nesta biblioteca, entretanto, não conseguimos uma boa performance da biblioteca nos testes mais severos (com grandes datasets), pelo fato de não possuir um suporte melhor desta interface para o gerenciamento de memória. A solução neste caso, foi utilizar as bibliotecas diretamente, no formato de uma microaplicação escrita em Python, capaz de utilizar os mesmos recursos presentes na interface gráfica, entretanto, com um suporte bem mais expressivo ao gerenciamento de memória e permitindo assim a utilização de grandes datasets, para uma classificação mais detalhada e precisa.

4.7 Resultados e testes

Para realizarmos os testes deste módulo de detecção de intrusão, foi necessário a escrita de uma microaplicação escrita em Python, conforme citada no subcapítulo anterior. A biblioteca Orange foi crucial para a realização dos testes, uma vez que a mesma possui os métodos necessários para que possamos realizar a classificação induzida, na fase de testes e que com posse dos resultados desta fase, pudéssemos, de fato, guardar o classificador Naive Bayes para ser utilizado na fase de testes.

A microaplicação gerada é simples e foi construída com poucas linhas de código, o que dispensou a utilização de alguma metodologia de desenvolvimento e/ou aplicação de técnicas de engenharia de software. O algoritmo utilizado nesta microaplicação está descrito na **Tabela 4**:

Tabela 4 - Sequência de tarefas realizadas na microaplicação de validação do módulo de detecção de intrusão

Tarefa	Ação
1. Carregar o ficheiro de features	Esta tarefa é responsável pelo carregamento do ficheiro de features gerado após as transformações dos dados obtidos pelo sensor do Delusion. Este ficheiro foi o responsável pela decisão de utilizar a biblioteca do Orange e ignorar a possibilidade de utilização da interface gráfica do mesmo. A justificativa para isso é que o ficheiro usado para a realização da fase de treino possui dados relativos a 2h de utilização do SGBD e após gerado o arquivo de features (sequências de system calls x 2-grams) o tamanho obtido foi 8.5GB aproximadamente, o que inviabilizava a utilização da interface por motivos referentes ao mau gerenciamento de memória do mesmo. Esta informação foi certificada por alguns utilizadores da ferramenta, no fórum oficial do produto.
2. Instanciar o classificador Naive Bayes	Esta tarefa é responsável pela instanciação do classificador Naive Bayes que vai ser utilizado para realização da fase de treino, juntamente com o ficheiro de features e outras configurações relativas à utilização do mesmo.
3. Iniciar a fase de treino	Esta tarefa é responsável por uma das fases mais importantes de todo o processo da implementação do módulo de detecção de intrusão. Isso é justificado pelo fato de ser nesta fase, que de fato vamos utilizar a teoria de Bayes para treinar o classificador, induzindo-o aos dados presentes no arquivo de features, ou seja, com base nas caracterísicas presentes neste arquivo e na coluna de classificação previamente informados. Nesta fase também serão utilizadas algumas propriedades relativas a configuração do classificador a ser gerado e sua interação com o arquivo de features, conforme foram definidas na **Tarefa 2** desta tabela. Para avaliar os resultados desta fase, foi utilizada uma técnica estatística chamda Cross-validation. Esta técnica é responsável pela estimativa de performance da fase de treino do classificador Naive Bayes, assim como na divisão do dataset em dois, para que sejam utilizados nas fases de treino e teste. O resultado desta tarefa servirá como parâmetro para a fase posterior desta tabela. Ela pode ser utilizada de várias formas, entretanto, utilizamos a 2-fold cross validation,

	por ser mais utilizada com grandes datasets, como é o caso do nosso classificador. Este tipo embaralha os exemplos presentes no dataset e os divide ao meio, utilizando uma parte para treino e outra para testes e depois inverte as partes e realiza novamente as fases de treino e testes.
4. Validação da fase de testes	Para realizar esta tarefa utilizamos dois componentes presentes na biblioteca do Orange. O primeiro deles é capaz de gerar uma matriz de confusão (Confusion Matrix). Esta matriz gera uma tabela que serve para calcular as métricas de performance da técnica de machine learning aplicada (no caso, Naive Bayes) no âmbito da aprendizagem supervisada. Em termos gerais, ela apresenta a quantidade de verdadeiros positivos e negativos e a quantidade de falsos positivos e negativos. Com base nestes dados, podemos ter uma análise completa da quantidade de exemplos classificados corretamente. O outro componente utilizado, foi a curva ROC. Conforme também descrito no **Capítulo 3**, esta curva é utilizada para ilustrar a taxa de assertibilidade dos itens classificados, tendo em vista que a mesma é montada com base na taxa de verdadeiros positivos por falsos positivos encontrados após a fase de testes do classificador de Bayes.

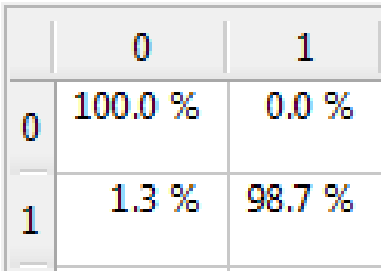
Descrevemos na **Tabela 4** as tarefas referentes a microaplicação para validação da utilização do classificador Naive Bayes. Estas tarefas são realizadas visando a criação do ficheiro de features, criado a partir da extração dos dados presentes no dataset de System Calls capturado pelo sensor do Delusion. Para os nossos testes, conforme foi descrito anteriormente, utilizamos várias métricas de tempo de captura de dados utilizando o sensor do Delusion, entretanto, baseado na quantidade de System Calls capturadas e para garantir que pudéssemos ter recursos de hardware suficientes para processar o arquivo de features, fixamos o tempo de captura em 60 mins.

Durante os 60 mins, utilizamos uma aplicação que realizava tarefas comuns de acesso à uma base de dados em ambiente de produção, entretanto, isolando e descartando a hipótese desta aplicação não sofrer quaisquer tipos de ataques. Desta forma, poderíamos considerar que os exemplos gerados, durante a execução desta ferramenta poderiam ser classificados posteriormente como "normais".

Ao mesmo tempo, é executada uma outra aplicação, em paralelo, que utiliza o Metasploit para realizar tentativas de ataque explorando uma vulnerabilidade presente na versão 5.0.51a-24+lenny2 do MySQL para o Linux. Esta versão apresenta um módulo SSL embutido no seu código fonte, yaSSL 1.9.8. O exploit "MySQL yaSSL CertDecoder::GetName" existente no Metasploit explora uma vulnerabilidade do tipo Buffer Overflow, ao enviar um certificado de cliente criado especialmente para um atacante executar um código arbitrário no sistema gestor de base de dados.

A execução do Metasploit também possui uma duração de 60 mins e gera código que posteriormente será classificado como "anômalo" para a criação do arquivo de features. Esta tarefa é importante porque a identificação das sequências que possam estar presentes durante um ataque é crucial para poder analisar de forma correta novos exemplos que estão a usar o classificador.

Após a execução das fases de treino e de testes em nosso projeto, obtivemos resultados convincentes, dada a dimensão do dataset utilizado, conforme podemos verificar na matriz de confusão exibida na **Imagem 16**:



	0	1
0	100.0 %	0.0 %
1	1.3 %	98.7 %

Imagem 16 - Exibição da proporção de verdadeiros encontrados
Fonte: Arquivo pessoal

Como podemos verificar, foram encontrados 100% de falsos negativos e uma taxa de 98.7% de verdadeiros positivos.

Analisando estes dados, podemos inferir que os dados de desempenho encontrados com base na matriz de confusão são os exibidos na **Tabela 5:**

Tabela 5 - Exibição das métricas encontradas nos resultados

Métrica	Valor
Especificidade(Spec)	0.9406
Classification Accuracy(CA)	0.9335
Area Under ROC Curve(AUC)	0.9757

Para ilustrar a taxa de assertibilidade da abordagem realizada no ambiente proposto, utilizamos uma Curva ROC, que contém a taxa de verdadeiros positivos (TPR) e a taxa de falsos positivos (FPR), que nada mais é, do que a sensibilidade ilustrando o eixo Y e o cálculo da diferença entre 1 e o valor da especificidade.

A **Imagem 17** exibe a curva ROC gerada a partir dos resultados obtidos:

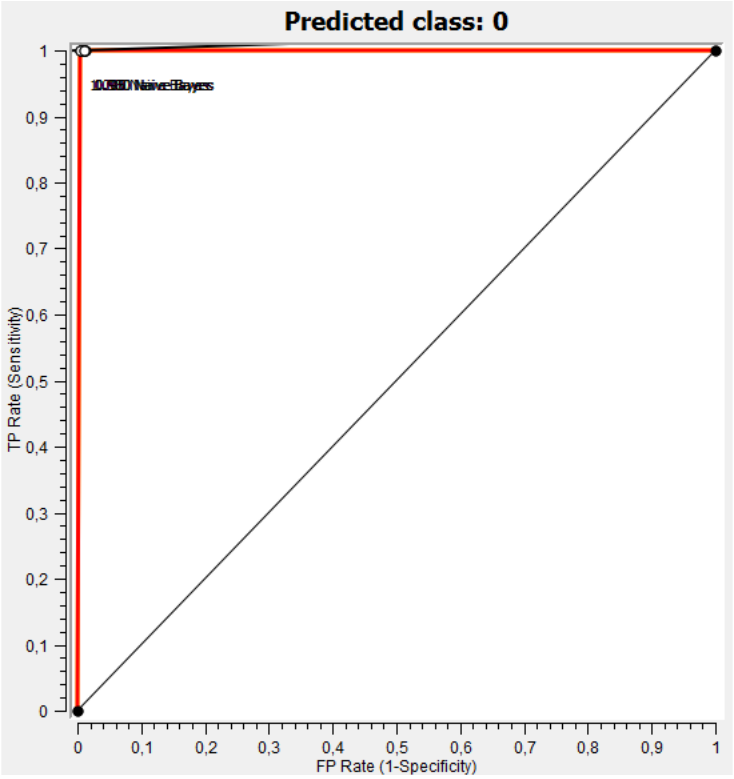


Imagem 17 - Curva ROC gerada pelo projeto
Fonte: Arquivo pessoal

Com isto, podemos inferir que o classificador terá sucesso ao classificar novos exemplos, tendo em vista a alta taxa de assertibilidade dos dados que foram classificados. Tanto a matriz de confusão como a curva ROC atestam isso através dos dados exibidos. Uma maior taxa de assertibilidade poderia ser conseguida ao utilizar uma maior quantidade de exemplos para dados de treino, uma vez que os mesmos só servirão para afinar ainda mais a função gerada pelo mesmo.

Capítulo 5
Considerações finais

Neste relatório foi definido uma API implementada em Java, para acesso as informações presentes na base de dados do software Delusion, de propriedade da VisionSpace Technologies. Além da descrição da API de acesso aos dados, este relatório também descreveu os scripts escritos em bash script, responsáveis pela captura e registro de System Call's e demais informações como argumentos e árvores de processos. Estes scripts serviram para realizar o preenchimento da base de dados através de uma instância do Delusion em execução em um sistema operacional Linux específico.

Este relatório também incluiu um estudo de caso sobre o algoritmo que foi implementado para realizar a análise das sequências de System Calls e ajudar o Delusion a definir o que de fato é um comportamento anômalo, como o comportamento de uma tentativa de invasão, tendo em vista que, sequências de System Calls podem definir também perfis normais de utilização de um serviço. Este relatório serviu para o desenvolvimento de uma importante vertente de utilização do software Delusion, assim como também serviu para o meu crescimento intelectual em diversas áreas computacionais, como escrita de scripts, utilização de ferramentas de gestão de informações de sistemas operacionais, modelação e criação de scripts SQL em bases de dados, implementação de ferramentas em Java, interoperabilidade entre aplicações e bases de dados utilizando ODBC e por mim, mas não menos importante, o desenvolvimento de técnicas e utilização de conhecimentos ligados à Segurança da Informação, voltados para a segurança de sistemas operacionais e serviços ligados à Internet.

No âmbito do módulo de detecção de intrusão, os resultados obtidos pelo classificador Naive Bayes foram satisfatórios, uma vez que 98% dos exemplos utilizados na fase de testes foram classificados de maneira correta, tendo em vista os dados obtidos pela matriz de confusão e exibidos na curva ROC, ambas estruturas descritas no subcapítulo 4.6. Uma possível melhoria e/ou extensão deste trabalho, diz respeito à criação de um módulo de resposta a ataques, com base nas informações obtidas no módulo desenvolvido neste relatório, a fim de criar um honeypot que responda as tentativas de ataque identificadas pelo módulo aqui desenvolvido.

Referências

MySQL:

[1] http://www.infowester.com/postgremysql.php

[2] http://dev.mysql.com/doc/refman/5.0/en/subqueries.html

[3] http://dev.mysql.com/doc/refman/5.0/en/charset-unicode.html

[4] http://dev.mysql.com/doc/refman/5.0/en/information-schema.html

[5] http://msdn.microsoft.com/pt-br/library/ms191179.aspx

[6] http://dev.mysql.com/doc/refman/5.0/en/ssl-connections.html

Project jc-tree:

[7] http://code.google.com/p/jc-tree/

ODBC:

[8] http://foldoc.org/ODBC

SQL:

[9] http://www.sigmod.org/publications/sigmod-record/0403/E.JimAndrew-standard.pdf

TLS/SSL:

[10] http://tools.ietf.org/html/rfc5246

[11] http://www3.rad.com/networks/applications/secure/tls.htm

DAO:

[12] http://www.oracle.com/technetwork/java/dataaccessobject-138824.html

System Calls e processos:

[13] http://docs.cs.up.ac.za/programming/asm/derick_tut/syscalls.html

[14] http://www.tldp.org/LDP/khg/HyperNews/get/syscall/syscall86.html

[15] William Shay , "Sistemas Operacionais"(1996), São Paulo:Makron Books

[16] Gary D. Knott (1974) A proposal for certain process management and intercommunication primitives ACM SIGOPS Operating Systems Review. Volume 8, Issue 4 (October 1974). pp. 7 – 44

Honeypot:

[17] http://www.projecthoneypot.org/

Distributed Web Honeypots

[18] http://projects.webappsec.org/w/page/29606603/Distributed%20Web%20Honeypots

[19] http://www.csirt.pop-mg.rnp.br/eventos/palestras/honeypots.pdf

Apache WebServer:

[20] http://httpd.apache.org/

Metasploit:

[21] http://www.metasploit.com/about/what-is-it/

Delusion:

[22] https://github.com/ulisses/delusion-requesitos/blob/master/informacao.tex

[23] T. Bayes, An Essay Towards Solving a Problem in the Doctrine of Chances, Philos. Trans. R. Soc. London, 1763

[24] P. Bernstein, Against the Gods: The Remarkable Story of Risk, John Wiley & Sons, New York, US, 1996.

[25] S. Chakrabarti, Mining the Web: Discovering Knowledge from Hyper-text Data, Morgan Kaufmann, 2002.

[26] A. McCallum and K. Nigam, A comparison of event models for Naive bayes text classication, In Proc. of the AAAI-98 Workshop on learning for text categorization, pp. 41-48.

[27] Jiawei Han and Micheline Kamber - Data Mining: Concepts and Techniques, 2nd ed. - The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor Morgan Kaufmann Publishers, March 2006. ISBN 1-55860-901-6

[28] Pedro Oguri, Aprendizado de Máquina para o Problema de Sentiment Classification. PUC-RIO, 2006

[29] A. Wespi, H. Debar, M. Dacier, and M. Nassehi. Fixed- vs. variablelength patterns for detecting suspicious process behavior. Journal of Computer Security, 8(2,3):159–181, 2000.

[30] R. Kohavi, F. Provost: Glossary of terms, Machine Learning, Vol. 30, No. 2/3, 1998, pp. 271-274.

[31] Braga ACS. Curvas ROC: Aspectos funcionais e aplicações. Universidade do Minho, dezembro de 2003, Portugal

[32] Nance, K. ; Dept. of Computer Science, Univ. of Alaska at Fairbanks, Fairbanks, AK ; Bishop,M. ; Hay, Brian, 2009

[33] Data Mining: Concepts and Techniques, Han & Kamber, 2006

[34] S. Forrest, S. A. Hofmeyr, A. somayaji, and T. A. Logstaff, "A Sense of Self for Unix Process,"Proceedings of 1996 IEEE Symposium on Computer Security and Privacy, 1996, pp. 120-128

[35] S. A. Hofmeyr, S.Forrest, and A.Somayaji, "Intrusion Detection Using Sequences of System Calls," Journal of Computer Security, Vol. 6, 1998, pp. 151-180

ANEXO I

Especificação da API de acesso à base de dados

Esta API foi criada com a finalidade de gerenciar o acesso aos dados contidos na base de dados do software Delusion de propriedade da empresa VisionSpace Technologies, de forma a obter informações acerca da utilização de serviços em um ambiente monitorado pelo software Delusion. Inicialmente serão especificados as tabelas pertencentes a base de dados, assim como os campos pertencentes as mesmas.

Especificação da base de dados

Tabelas

- Appliance
- Appliance_service
- Argument_value
- City
- Country
- CountryLanguage
- Honeypot
- Honeypot_session
- Host
- Interface
- Process
- Process_systemcall
- Service
- Systemcall
- Systemcall_argument

Principais entidades

As principais entidades descritas na base de dados em estudo neste relatório são as seguintes:

- Process

  Esta tabela provê informações referentes aos processos que estão a ser monitorados pelo Delusion. Através desta tabela é possível recuperar informações como o PID(Process Identifier), o número do processo pai (caso seja um processo filho), o nome do processo, além de recuperar o tempo de execução do processo e sua localização temporal de execução.

- Systemcall

Esta tabela provê informações referentes as chamadas de sistemas que estão disponíveis para serem executadas e que foram capturadas através do Delusion. Através desta tabela, é possível chegar à vários dados adjacentes, entretanto, nesta tabela é possível recuperar diretamente informações como o nome da System Call; os argumentos que elas possuem, a descrição de utilização delas e o link com a referência acerca da mesma.

- Interface

Esta tabela provê informações referentes às interfaces de rede anexas ao sistema que está sendo monitorado pelo Delusion. Nesta tabela são encontradas informações como o nome da interface; o endereço IP associado a esta interface no sistema operacional; a máscara de rede associada ao endereço IP; o endereço de broadcast associado ao endereço IP e o gateway padrão descrito nas configurações de rede desta interface no sistema operacional monitorado.

- Argument_value

Esta tabela provê informações referentes aos argumentos que uma System Call pode possuir. Por se tratar de dados por muitas vezes muito grandes, seus valores são representados em dados do tipo BLOB. Nesta tabela são encontradas informações como o valor do argumento, o tamanho do mesmo e o hash associado a esta entrada de argumento na base de dados.

- Process_systemcall

Esta tabela provê os identificadores responsáveis pelas ligações das System Call's de um determinado processo em questão. Por se tratar de uma tabela de ligação, que serve apenas para ligar os dados providos entre a tabela Process e a tabela Systemcall, esta não disponibiliza maiores informações. Apenas o tempo de execução de cada System Call executada em um determinado processo é fornecido, além dos identificadores de ligação entre as tabelas envolvidas.

- Systemcall_argument

Esta tabela provê os identificadores responsáveis pelas ligações dos argumentos e suas respectivas System Call's, de um processo específico. Por se tratar de uma tabela de ligação, nenhuma outra informação é fornecida além dos identificadores de ligação entre as tabelas envolvidas.

Especificação dos principais objetos DAO

Nesta seção serão descritos os principais objetos DAO criados para acesso à base de dados do software Delusion. Estes objetos mapeiam as entidades presentes na base de dados e possuem além das propriedades inerentes aos campos das tabelas da bases de dados como também métodos acessores, de cada propriedade, que servem para alterar ou recuperar a informação presente nestes objetos.

- Process

```
public class Process {
	long id;
	int pid;
	long pprid;
	int uid;
	String name;
	long startts;
	long endts;
	int honeypot;
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public int getPid() {
		return pid;
	}
	public void setPid(int pid) {
		this.pid = pid;
	}
	public long getPprid() {
		return pprid;
	}
	public void setPprid(long pprid) {
		this.pprid = pprid;
	}
	public int getUid() {
		return uid;
	}
	public void setUid(int uid) {
		this.uid = uid;
	}
	public String getName() {
		return name;
```

```
		}
		public void setName(String name) {
			this.name = name;
		}
		public long getStartts() {
			return startts;
		}
		public void setStartts(long startts) {
			this.startts = startts;
		}

	public long getEndts() {
		return endts;
	}
	public void setEndts(long endts) {
		this.endts = endts;
	}
	public int getHoneypot() {
		return honeypot;
	}
	public void setHoneypot(int honeypot) {
		this.honeypot = honeypot;
	}
}
```

- SystemCall

```
	public class SystemCall {
		int id;
		String name;
		String args;
		String description;
		String link;

		public int getId() {
			return id;
		}
```

```
        public void setId(int id) {
                this.id = id;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public String getArgs() {
                return args;
        }
        public void setArgs(String args) {
                this.args = args;
        }
        public String getDescription() {
                return description;
        }
        public void setDescription(String description) {
                this.description = description;
        }
        public String getLink() {
                return link;
        }
        public void setLink(String link) {
                this.link = link;
        }
}
```

- SystemCallArgument

```
public class SystemCallArgument {

long id;
long pcscid;
long argid;
long ind;
```

```
public long getId() {
        return id;
}
public void setId(long id) {
        this.id = id;
}
public long getPcscid() {
        return pcscid;
}
public void setPcscid(long pcscid) {
        this.pcscid = pcscid;
}
public long getArgid() {
        return argid;
}
public void setArgid(long argid) {
        this.argid = argid;
}
public long getInd() {
        return ind;
}
 public void setInd(long ind) {
        this.ind = ind;
 }
 }
```

- ArgumentValue

```
public class ArgumentValue {

long id;
String value;
int size;
String hash;

public long getId() {
```

```
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }
    public String getValue() {
        return value;
    }
    public void setValue(String value) {
        this.value = value;
    }
    public int getSize() {
        return size;
    }
    public void setSize(int size) {
        this.size = size;
    }
    public String getHash() {
        return hash;
    }
    public void setHash(String hash) {
        this.hash = hash;
    }
    }
```

Especificação dos métodos de recuperação dos dados

Nesta seção serão descritos os métodos criados para recuperação dos dados existentes na base de dados do Delusion. A necessidade de implementação de alguns dos métodos criados foram pré-definidas e de outros foram criados a partir da verificação da necessidade e da utilidade do mesmo para uma gestão mais rápida e simples de acesso aos dados armazenados na base de dados.

- getAllProcesses()

  Este método como o nome já induz, recupera todos os processos existentes na base de dados do Delusion. Este método não possui parâmetros e retorna uma lista de objetos do tipo Process:

```
public List<Process> getAllProcesses()
```

- getAllSystemCalls()

Este método como o nome já induz, recupera todos as System Calls existentes na base de dados do Delusion. Este método também não possui parâmetros e retorna uma lista de objetos do tipo SystemCall:

```
public List<SystemCall> getAllSystemcalls()
```

- getProcessByID(processID)

Este método recupera um processo específico na base de dados do Delusion e mapeia o mesmo para um objeto do tipo Process. Este método possui um parâmetro *processID*, que possui o código do processo na base de dados do Delusion e retorna um objeto do tipo Process:

```
public Process getProcessByID(long processID)
```

- getProcessByPID(processPID)

Este método recupera um processo específico na base de dados do Delusion e mapeia o mesmo para um objeto do tipo Process. Este método possui um parâmetro *processPID*, que possui o PID (*Process Identifier*) na base de dados do Delusion e retorna um objeto do tipo Process:

```
public Process getProcessByPID(long processPID)
```

- getChildProcesses(process)

Este método recupera uma lista de processos filhos de um processo pai específico, que é passado como parâmetro. Este método retorna uma lista de objetos do tipo Process, processos estes filhos diretos do processo passado por parâmetro:

```
public List<Process> getChildProcesses(Process p)
```

- getProcessTree(process)

Este método recupera uma árvore de processos a partir do processo pai que é passado por parâmetro. Para a implementação deste método foi utilizado uma estrutura de representação de árvores *n-ary* em Java, a estrutura ArrayListTree. Esta estrutura foi desenvolvida por Gaurav Saxena e é disponibilizado através do projeto *jc-tree*, disponível em http://code.google.com/p/jc-tree/. Esta implementação representa as árvores de n-nodes através de estruturas ArrayList como o exemplo abaixo:

```
Tree<String> tree = new ArrayListTree<String>
tree.add("Level-1");
tree.add("Level-1", "Level-11");
tree.add("Level-1", "Level-12");
tree.add("Level-2", "Level-21");
```

Segue abaixo a assinatura do método getProcessTree:

```
public ArrayListTree<Process> getProcessTree(Process p)
```

- getSystemCallByID(systemCallID)

Este método recupera uma System Call específica na base de dados do Delusion e mapeia a mesma para um objeto do tipo SystemCall. Este método possui um parâmetro *systemCallID*, que possui o código da System Call na base de dados do Delusion e retorna um objeto do tipo SystemCall:

```
public SystemCall getSystemCallByID(int systemcallID)
```

- getSysCalls(process)

Este método recupera uma lista de System Call's da base de dados, a partir de um processo que é passado por parâmetro. Este método possui um parâmetro *process* e retorna uma lista de objetos do tipo SystemCall:

```
public List<SystemCall> getSysCalls(Process p)
```

- getSysCallParams(systemcall, limit)

Este método recupera uma lista de argumentos de System Call's da base de dados, a partir de uma System Call específica que é passada por parâmetro. Este método possui um parâmetro systemcall, do tipo SystemCall e o parâmetro *limit* que é responsável pelo limite de objetos que devem ser retornados da base de dados. Tal parâmetro foi adicionado posteriormente neste método, tendo em vista a grande quantidade de informações que o mesmo pode vir a retornar, ultrapassando 1 milhão de resultados. Este método retorna uma lista de objetos do tipo SystemCallArgument:

```
public List<SystemCallArgument> getSysCallParams(SystemCall sc, int limit)
```