1 Apresentação

Este trabalho de graduação tem como intuito apresentar um estudo acerca da usabilidade e viabilidade de algoritmos de *machine learning* em sistemas de detecção de intrusão em rede. Sua utilização teria grande impacto na segurança tanto de servidores quanto de clientes, pois promete evitar ataques ao sistema detectando anomalias na rede.

Vamos abordar:

- conceitos de sistema de detecção de intrusão e de *machine learning*
- a arquitetura geral de um sistema baseado em anomalia
- o processo de extração e processamento de dados;
- e três testes efetuados, referentes a
    1. impacto de uma base de dados representativa
    2. impacto de tipos de ataques desconhecidos
    3. impacto da seleção de atributos.
- conclusão

2 Intro

2.1 Motivação

Pode ser que eu venha a me referir pela sigla NIDS, que significa *Network Intrusion Detection System*, mas no geral, apenas Sistema de Detecção ou até Sistema, mesmo.

Bom, roteadores e sistemas operacionais atuais vêm equipados com ferramentas básicas para previnir acesso não autorizado ao computador. Ainda assim, há pessoas dedicadas a burlar tais proteções. Uma máquina conectada à rede nunca está 100% segura. Recentemente veio à tona o problema do Shellshock, que com certeza já era conhecido em alguns meios. Uma coisa é um usuário baixar um vírus no windows que fica abrindo janelas, mas e o quando alguém acessa seu servidor, ganha permissão root, e você nem fica sabendo.

2.2 NIDS

A ideia dos sistemas de detecção de intrusão é cobrir esse ponto cego, detectando automaticamente comportamentos nocivos na rede, analisando o tráfego e pacotes. Atualmente, os sistemas em uso comercial são os chamados baseados em assinatura. Eles tentam casar padrões com um banco de dados de assinaturas de ataques conhecidos. O problema é que algumas alterações podem gerar assinaturas diferentes. Daí surgem os baseados em anomalia.

3 A-NIDS

3.1 Machine Learning

A ideia é usar algoritmos de machine learning para que o sistema aprenda o que é o comportamento normal da rede e bloqueie quaisquer conexões que sejam divergentes. Machine Learning tem aplicação em diversar áreas, como processamento de imagens e robótica. A ideia geral é o seu sistema aprender algo e usar isso para executar tarefas futuras. No nosso caso, é o de classificar as conexões como sendo ou "**normal**"ou "**anômala**".

Enquanto alguns detalhes de implementação podem variar, o padrão da esquematização básica de um A-NIDS é o seguinte:

- Aquisição de dados de tráfego – coleta informação sobre os quadros da rede para processamento futuro.
- Gerador de características do tráfego – extrai características do tráfego capturado.
- Gerador de modelo de tráfego (Treino) – contém informação base usada para guiar o detector de incidente.
- Detector de incidente (Classificação) – identifica atividades intrusivas.
- Gerenciamento de resposta – inicia manobras para superar uma intrusão em potencial. É inicializado pelo detector de incidente.

3.2 Dados

Dados extraídos têm grande influência no desempenho do sistema. Nessa área, podemos ter dados extraídos do cabeçalho de pacotes ou do corpo de pacotes. E as características derivadas podem ser tanto de uma única conexão, como geradas a partir de um conjunto de conexões.

Dados extraídos de cabeçalho são rápidos de serem obtidos e contém informações de origem, destino, portas, e flags TCP. Só que se é um ataque destinado a uma aplicação específica, devemos analisar o conteúdo dos pacotes. Isso já é bem mais custoso, pois lida com compressão de dados.
Atributos derivados de uma única conexão podem apontar um protocolo inesperado, tamanhos incomuns de dados, *timestamp* não condizente. Portanto, características SCD permitem detecção de uso anômalo da rede por *backdoors*, túnel HTTP e afins.
Já os derivados de múltiplas conexões são extraídos a partir de conexões dentro de um intervalo de tempo. Esses atributos são geralmente derivados usando média, desvio padrão, e porcentagem de fluxo de pacotes. Podem ser usados para detectar volumes excessivos de tráfego relacionados a *DoS* e *probing*.

3.3 Algoritmos

A característica principal dessa técnica é a habilidade de adaptar suas regras de classificação à medida que novos dados são recebidos. Isso garante pouca necessidade por intervenção humana, uma vez funcionando. O lado ruim do aprendizado de máquina é o alto custo computacional. Os algoritmos mais populares usados em A-NIDS são:

- Rede Bayesiana – cria uma rede de relações de probabilidade entre características.
- Modelos de Markov – compara a probabilidade dos dados observados com um *threshold* definido.
- Redes neurais – cria uma rede de *perceptrons* com habilidade de se adaptar.
- Support Vector Machines – encontra um hiperplano que eficientemente separa ambas as opções de classificação.
- Árvores de Decisão – cria uma árvore onde cada folha representa uma classe e arestas correspondem a diferentes valores de atributos.

4 Testes

4.1 Representabilidade

O primeiro teste consistiu de avaliar a performance da detecção mudando a relação entre a base de treino e de testes. Foi usada uma base de dados

livre NSL-KDD, com 1.140.000 instâncias. Ela foi dividida em 5 particionamentos, variando a porcentagem para treino e a porcentagem para teste. O algoritmo usado foi o SVM. Analisando a tabela, podemos ver uma correlação entre o tamanho da base de treino com a porcentagem de instâncias categorizadas corretamente. Com metade do tráfego sendo usado para treino, atingiu 99.64% de acerto. Isso pode parecer muito bom, mas em um mundo real, não há como usar 50% de todas as conexões para treinar o sistema. Além disso, o KDD já tem 16 anos, e de lá pra cá, a realidade da comunicação mudou.

4.2 Imprevisibilidade

Aí vem a teste a real capacidade dos algoritmos de machine learning se adaptarem a novos ataques. Para testar o impacto da imprevisibilidade, foi usada uma base gerada por mestrando da PUC, Eduardo Viegas. Através de simulações de rede, ele gerou três grupos de dados, contendo conexões normais e anômalas:

1. Inicial - 500.000 instâncias de conexões usando protocolo HTTP e SMTP.

2. Análogo - 148.343 instâncias contendo ataques parecidos ou derivados dos anteriores.

3. Desconhecido - 276.548 instâncias com novos ataques e novos protocolos, como SNMP e SSH.

O processo consistiu de treinar algoritmos de SVM e DT usando metade da base Inicial e avaliar o desempenho ao se classificar cada um dos grupos restantes. As taxas de acerto evidenciam uma grande perda de confiabilidade conforme o surgimento de novos ataques, sendo a classificação da base *Desconhecido* praticamente tão eficiente quanto uma adivinhação aleatória.

4.3 Redução

Tendo em vista o alto custo computacional para processar as 51 características iniciais, foi ponderada a viabilidade de se utilizar uma seleção de atributos prévia para agilizar o processamento na tomada de decisões. Utilizando o PCA, a quantidade de características foi reduzida para 39. o resultado dos algoritmos nesse cenário teve nenhum ganho significativo. Enquanto os atributos selecionados podem ser mais descritivos para os ataques vistos na base *Inicial*, talvez não sejam interessantes para identificar os ataques que residem além daquele horizonte.

5 Conclusão

O primeiro teste, usando apenas SVM, mostrou a necessidade de se gerar uma base de treino grande o suficiente, a fim de se evitar amostras pouco representativas. Os resultados do segundo teste sugerem que, possivelmente, os A-NIDS não tenham tanta escalabilidade com o surgimento de ameaças completamente desconhecidas e que, portanto, seriam necessárias bases específicas para aplicações particulares ao invés de sistemas genéricos. O último teste tinha a intenção de melhorar os resultados anteriores através da seleção de atributos, mas o seu ganho de performance não foi significativo.

O campo ainda está aberto para novos estudos e análises quanto a viabilidade do uso de machine learning em tempo real para a detecção de novas ameaças. Existem outros pontos não abordados nesse trabalho que também têm espaço para serem otimizados, tais quais a atual falta de métricas unificadas; o processamento do alto, rápido e contínuo fluxo de dados na rede; análise de dados cifrados e a taxa aceitável de Falso Positivo. Por fim, pode-se considerar que o maior desafio nos testes de detecção de intrusão em rede reside em adquirir uma base de treino e de testes que represente precisamente o atual comportamento da rede.

6 Observações

6.1 Existe algum software comercial?

Snort é open source, mas funciona baseado em assinatura, com plugin baseado em anomalia.
AlienVault possui um IDS baseado em assinatura e anomalia, com análise de protocolos.

6.2 Por que o baixo desempenho do SVM? Por que usá-lo, então?

Segundo o teorema chamado *no free lunch*, não existe garantia prévia de que um algoritmos de aprendizado de máquina será melhor que outro. Além disso o SVM depende de uma série de configurações, principalmente a função núcleo. Nesse caso eu usei uma função quadrática, pode ser que uma linear conseguisse resolver até mais rápido.

6.3 Cite outros tipos de segurança de rede

O primeiro passo é Autenticação, que está incluso nas ferramentas básicas que eu citei no início. Pode-se estender para autenticação two-factor ou three-factor, requerindo uma confirmação por celular ou por digital.
Um firewall é comumente usado para controlar os serviços e portas abertas para os usuários do sistema.
Existem os comuns anti-virus para analisar o binário de arquivos recebidos.
Pode-se usar alguma forma de criptografia para assegurar a confidencialidade da informação.
Honeypots para "distrair"os invasores e analisar seu comportamento.

6.4 Quais tipos de ataque podem ser detectados?

IP Spoofing, DNS Spoofing, Probing, Sechole, HTTP Tunnel, Imap, etc.

6.5 Como funciona o SVM?

O SVM é um algoritmo iterativo que tenta encontrar um hiperplano que melhor separe as classes observadas. Dado um gráfico em R2 com características x1 e x2 e uma distribuição de circulo e triangulo, poderíamos usar apenas um eixo para classificar as instâncias, fazendo uma linha vertical no meio. Entretanto, essa não é necessariamente a melhor solução, pois dá pouca margem

para erro. O SVM, tenta, então, maximizar as distâncias das ocorrências para o hiperplano, o que é uma tarefa não-linear.
Teremos, então um hiperplano dado por:

$$g(\vec{x}) = \vec{w} \cdot \vec{x}_i + w_0, \forall 1 \leq i \leq n$$

Inserindo novos dados na função, teremos resultados ≥ 1 para a primeira classe e ≤ -1 para a segunda.

6.6 Como funciona Cadeia de Markov?

É feita uma estimativa da probabilidade de uma instância pertencer a uma ou outra classe baseado em observações da base de treino. Leva em consideração a quantidade de vezes que cada atributo se sobressai para cada classe.