Detectores de Intrusão Baseados em Random Forest Aplicados a Smart Grids

Igor Ruys Cartucho
Programa de Engenharia Elétrica - COPPE
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil
igorruys@gmail.com

Markus Vinícius Santos Lima
Programa de Engenharia Elétrica - COPPE
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil
markus.lima@dee.ufrj.br

***Resumo*—O *smart grid* é um modelo de rede elétrica inteligente que, ao integrar tecnologias da informação e de comunicação com sistemas de energia, apresenta diversas melhorias em relação ao modelo convencional. Por depender fortemente de tecnologias da informação, esse novo modelo está sujeito a ameaças digitais, o que pode levar ao comprometimento de seus principais subsistemas. A *Advanced Metering Infrastructure* (AMI) é um desses subsistemas de vital importância para seu funcionamento, que é vulnerável a ciberataques e, por isso, precisa estar integrada a eficazes sistemas de segurança. Nesse sentido, este trabalho tem como objetivo apresentar uma solução de segurança para a AMI que emprega sistemas de detecção de intrusão distribuídos e estudar três modelos de detector de intrusão baseados em *random forest*. Para as etapas de treinamento e teste desses detectores, é utilizado o conjunto de dados NSL-KDD, amplamente utilizado para o estudo de detecção de ataques cibernéticos. Dentre os modelos propostos, o detector 3, que combina *random forests* com *K-means*, revelou-se um ótimo candidato a ser utilizado para a proteção da AMI, apresentando acurácia, taxa de falsos positivos e taxa de verdadeiros positivos de 94,6%, 9,6% e 97,8%, respectivamente.**

***Index Terms*—*smart grid*, cibersegurança, *machine learning*, *random forest*, IDS**

I. Introdução

A energia elétrica que abastece as diferentes regiões do país chega até os consumidores por meio de um complexo conjunto de estruturas e de sistemas de engenharia elétrica. Durante muitos anos, ela possuía uma estrutura sequencial muito bem definida e de fluxo de energia unidirecional, em que a energia elétrica era gerada nas usinas de geração, transportada ao longo das linhas de transmissão e entregue aos centros de consumo por meio de sistemas de distribuição [1].

Com o passar dos anos, percebeu-se que o modelo tradicional das redes de energia elétrica não era suficiente para atender a algumas exigências mais atuais, relacionadas a confiabilidade, eficiência energética, questões ambientais, entre outros [2]. Isso motivou a criação de um novo conceito de rede elétrica que pudesse integrar de maneira inteligente geradores e consumidores por meio da união de tecnologias de comunicação e informação mais recentes com os sistemas de energia. Isso resultou em uma rede com fluxo bidirecional de energia e de informação, que recebeu o nome de *smart grid* [3]. Dentre as vantagens trazidas por esse novo modelo estão a auto recuperação da rede elétrica, a possibilidade de acomodação de diversos tipos de geração e armazenamento de energia, a melhoria da qualidade da energia e a maior participação do consumidor, que passa a receber mais informações e a ter maior controle do seu consumo [3].

O funcionamento do *smart grid* é fortemente dependente de tecnologias de informação e comunicação, que formam uma rede de dispositivos conectados. Apesar de toda essa informatização possibilitar a criação de uma rede elétrica mais inteligente, isso também a torna mais vulnerável a ciberataques. Portanto, torna-se necessário haver um cuidado adicional com a cibersegurança dessa nova rede inteligente.

Nesse sentido, este trabalho tem como objetivo apresentar uma solução de segurança que é capaz de lidar com o problema das intrusões em um subsistema específico do *smart grid*, chamado AMI (*Advanced Metering Infrastructure*) [4] e estudar três modelos de detector de intrusão que podem ser utilizados na implementação desse sistema de segurança. Esses detectores são baseados em anomalia e utilizam um modelo de *machine learning* chamado *random forest*. A escolha da *random forest* foi feita com base em [5], que mostrou que esse modelo é um dos que apresenta os melhores resultados nesse problema. Os três detectores são treinados e testados utilizando o conjunto de dados NSL-KDD [6], amplamente utilizado no contexto de detecção de intrusão em redes.

O primeiro detector utiliza apenas uma *random forest* e já foi testado em [5], apresentando resultados promissores. Os outros dois detectores também utilizam *random forests*, mas possuem uma estrutura um pouco mais complexa. Eles foram apresentados em [7], mas não tiveram seus resultados documentados. Por apresentarem potencial de produzirem bons resultados, eles serão testados neste trabalho.

Este artigo está organizado da seguinte forma. Na **Seção II**, é feita uma introdução ao funcionamento e à estrutura da AMI e são apresentadas as principais questões de segurança relacionadas a ela. Em seguida, na **Seção III**, é feita uma revisão teórica do modelo *random forest*. Depois, na **Seção IV**, o conjunto de dados NSL-KDD é apresentado. Na **Seção V**, são explicados os procedimentos adotados durante a etapa de pré-processamento dos dados. Em seguida, na **Seção VI**, são apresentadas as métricas utilizadas para avaliar o desempenho dos detectores. Na **Seção VII**, é explicado o processo de construção dos três detectores de intrusão. Na **Seção VIII**,

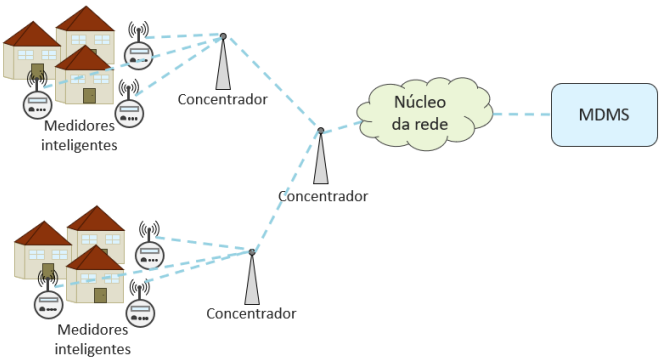


Figura 1. Representação da estrutura da AMI.

é feita uma comparação dos resultados obtidos ao aplicar os detectores na porção de teste do conjunto NSL-KDD. Finalmente, na **Seção IX** é apresentada a conclusão deste trabalho.

II. Infraestrutura de Medição Avançada (AMI)

Um dos mais importantes subsistemas do *smart grid* é a infraestrutura de medição avançada, também chamada de AMI (*Advanced Metering Infrastructure*) [4]. Ela tem como objetivo a constante coleta de dados e a garantia de que tanto os consumidores quanto os operadores tenham mais acesso a informações da rede elétrica. Isso é alcançado por meio do emprego dos seguintes elementos [4]:

- Medidores inteligentes: são dispositivos que registram o consumo elétrico do cliente e enviam os dados coletados para o fornecedor de energia.
- Rede de comunicação: garante a comunicação contínua entre a concessionária e os dispositivos de medição do lado do consumidor, empregando dispositivos como os concentradores para agregar os dados enviados de diferentes medidores inteligentes e encaminhá-los para outros concentradores.
- Sistema de recepção e gerenciamento de dados: também conhecido como MDMS (*Meter Data Management System*), tem a função de automatizar o processo de recebimento e organização dos dados, avaliar a qualidade dos mesmos e transformá-los em informação útil que possa ser utilizada pela concessionária.

Esses três componentes da AMI formam, portanto, uma infraestrutura conectada, em que os dados são coletados pelos medidores inteligentes, enviados à rede de comunicação, onde se encontram os concentradores, e encaminhados ao MDMS. Essa infraestrutura está resumida na **Figura 1**.

A. Ameaças à AMI

Como a AMI tem uma estrutura bastante dependente de tecnologias de informação e comunicação, ela está mais sucetível a ciberataques. Dentre as possíveis ameaças à AMI estão ataques de roubo de informação [8], ataques de inserção de dados falsos, também conhecidos como FDIA (*False Data Injection Attack*) [9] e DDoS (*Distributed Denial of Service*) [8]. Outros ataques de diferentes naturezas também podem ser conduzidos, aplicando diferentes princípios de redes de computadores. De forma geral, é possível separar esses ataques em quatro grandes classes:

- DoS (*Denial of Service*): essa ampla classe engloba diversos tipos de ataque que têm como objetivo tornar inutilizável qualquer porção da infraestrutura da rede (servidores, roteadores, dispositivos inteligentes) [10].
- *Probe*: é uma classe de ataques que têm como objetivo escanear o sistema e coletar diversas informações sobre ele (quais serviços são oferecidos, qual sistema operacional é utilizado...) [11].
- U2R (*User to Root*): nessa classe, encontram-se os ataques em que o atacante tem acesso a uma conta de usuário comum no sistema e, a partir disso, tenta explorar vulnerabilidades para ganhar acesso privilegiado [11].
- R2L (*Remote to Local*): engloba os ataques em que o atacante não tem acesso ao sistema, porém é capaz de enviar pacotes a ele remotamente por meio da rede de computadores, sem possuir nenhuma conta no sistema [11].

Com isso, fica clara a importância de integrar sistemas de segurança da informação à infraestrutura da AMI.

B. Solução de Segurança para a AMI

Uma solução de segurança operacional bastante popular é o sistema de detecção de intrusão ou IDS (*Intrusion Detection System*), que tem a função de observar o fluxo de dados de uma rede e ativar um alerta caso ele detecte a presença de uma atividade maliciosa. Os sistemas de detecção de intrusão podem ser divididos em duas grandes categorias: IDS baseado em assinatura e IDS baseado em anomalia [10].

O IDS baseado em assinatura funciona com base em uma lista de assinaturas de ataques, em que cada assinatura representa um conjunto de regras que caracteriza uma intrusão. À medida que pacotes passam pelo IDS, eles são comparados com cada uma das assinaturas da lista e, caso haja uma correspondência, um alarme é acionado. Uma desvantagem desse tipo de IDS é que é necessário que um ataque já tenha ocorrido anteriormente ou que suas características já sejam conhecidas para que seja possível registrá-lo na lista de assinaturas. Além disso, esse registro precisa ser feito manualmente por especialistas na área de segurança, o que pode ser trabalhoso e demorado, principalmente se considerarmos o fato de que esse banco de dados de ataques precisa ser atualizado com regularidade [10].

Uma alternativa é o IDS baseado em anomalia, que funciona a partir da criação de um perfil estatístico das atividades acontencendo na rede, o que pode ser feito com auxílio de modelos de *machine learning*. Quando acontece alguma atividade que fuja do comportamento padrão, como um aumento exponencial da chegada de determinados tipos de pacote [10], por exemplo, o IDS aciona um alarme. A principal vantagem desse método é que ele não depende do conhecimento prévio de ataques para funcionar e, por isso, tem um grande potencial para detectar novos ataques.

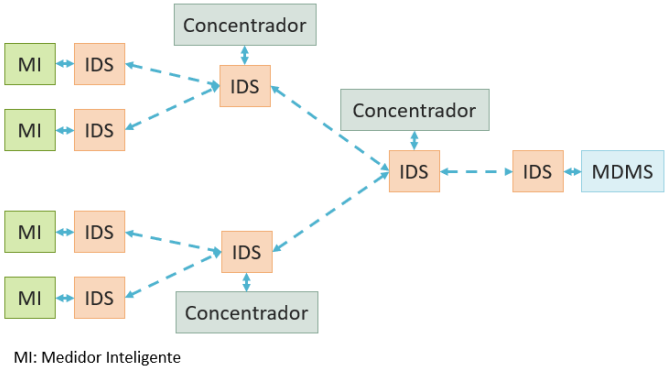


Figura 2. Arquitetura da solução de segurança para a AMI.

É possível integrar o detectores de intrusão à infraestrutura da AMI para ajudar a lidar com a questão da segurança da informação. Em [12], é proposta uma solução de segurança que faz uso de IDS's distribuídos em diferentes camadas da AMI. As principais conexões entre dispositivos dessa rede são: medidor-concentrador, concentrador-concentrador e concentrador-MDMS. Como essas conexões possuem fluxo bidirecional de dados, para garantir a segurança de todos os elementos, é necessário utilizar um IDS para cada componente, como mostra a **Figura 2**.

A partir dessa arquitetura de segurança proposta para AMI, serão desenvolvidos detectores de intrusão que possam ser integrados a essa solução, utilizando técnicas de *machine learning* (principalmente a *random forest*) e o conjunto de dados NSL-KDD.

III. Random Forest

A *random forest* é um modelo de *machine learning* que constrói uma coleção de árvores de decisão não correlacionadas e usa o resultado conjunto delas para a realização de predições. A técnica que está no núcleo desse método é chamada de *bagging*, que é usada para diminuir a variância de modelos que apresentam um viés reduzido, mas uma variância elevada, que é o caso das árvores de decisão [13]. Para o caso específico das árvores de classificação, o *bagging* consiste em criar um "comitê" de árvores que realiza uma votação para decidir qual classe deverá ser eleita como resultado da predição.

No *bagging*, esse "comitê" é criado aplicando-se ao *dataset* de treinamento uma técnica chamada de *bootstrapping*. Essa técnica consiste em criar um novo conjunto de dados a partir da seleção aleatória de amostras do conjunto de dados original, considerando que a reposição de amostras é permitida. Essa seleção de amostras é feita até que o novo conjunto de dados tenha o mesmo tamanho do conjunto de dados original. Assim, cada árvore criada usando o *bagging* é treinada com um conjunto de dados diferente.

Para realizar predições, cada uma das B árvores de um agregado criado com o *bagging* recebe um exemplo de teste em sua entrada e realiza uma predição. O resultado final desse agregado é a moda das predições das árvores. Para valores de B pequenos, a variância tende a ser maior e a acurácia do agregado tende a diminuir [13].

A *random forest* realiza algumas modificações na técnica original de *bagging* que permitem diminuir a correlação entre os pares de árvore do agregado e, consequentemente, diminuir ainda mais a variância. Durante o processo de construção de uma árvore de decisão, o algoritmo precisa, a cada etapa, escolher um atributo X_j de um total de p atributos para utilizar no critério da divisão de cada nó da árvore em nós filhos [13]. No processo de criação das árvores de uma *random forest*, no entanto, antes da divisão de um nó, um número $q < p$ de atributos são selecionados aleatoriamente como candidatos para X_j, de forma que os atributos não selecionados não podem ser escolhidos para X_j. Dessa forma, a *random forest* consegue reduziar a correlação entre suas árvores.

O algoritmo de treinamento de uma *random forest* [13] é mostrado abaixo:

1) Para $b = 1$ até B, onde B é o número de árvores na *random forest*:
   a) Construir um novo conjunto de dados usando *bootstrapping*.
   b) Treinar uma árvore de decisão T_b usando o conjunto de dados gerado no passo a). Repetir recursivamente os seguintes passos na construção de cada nó da árvore:
      i. Selecionar q atributos aleatoriamente do total de p atributos;
      ii. Selecionar o melhor X_j dentre os q atributos selecionados;
      iii. Dividir o nó em dois nós filhos.
2) Dar como saída o conjunto de árvores $\{T_B\}_1^B$.

Dada essa *random forest* treinada e composta pelo conjunto de árvores $\{T_B\}_1^B$, as predições podem ser realizadas da mesma forma como é feita para o método de *bagging*, em que o resultado final é dado pela classe mais "votada" entre as árvores. Caso se deseje o resultado na forma de probabilidade, basta realizar a média das probabilidades dadas como resultado por cada árvore individualmente [13].

IV. Conjunto de Dados NSL-KDD

O NSL-KDD [6] é uma versão melhorada do conjunto de dados KDD Cup'99 (KDD'99) [14], que foi utilizado durante uma competição internacional de mineração de dados. O KDD'99, por sua vez, é uma versão de outro conjunto de dados criado pela DARPA (*Defense Advanced Research Projects Agency*) em conjunto com o Lincoln Laboratory do Massachusetts Institute of Technology (MIT), que era composto originalmente por diferentes tipos de intrusões simuladas em um ambiente de rede militar. Ele contém dados de conexões TCP que foram coletados durante 9 semanas em uma rede local simulando uma rede típica da Força Aérea americana.

O NSL-KDD é uma modificação do KDD'99 [15], que foi criado com o intuito de corrigir alguns problemas muito criticados, como a grande quantidade de duplicatas (cerca de 78% de dados duplicados).

Tabela I
TOTAL DE AMOSTRAS DE TREINAMENTO E DE TESTE NO NSL-KDD.

Dados	Número de amostras
Treinamento	125 973
Teste	22 544

Tabela II
TIPOS DE ATAQUE E SUAS CLASSES NO NSL-KDD.

Tipo de Ataque	Classe do ataque
apache2, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm	DoS
ipsweep, mscan, nmap, portsweep, saint, satan	Probe
buffer_overflow, loadmodule, perl, rootkit, httptunnel,ps, sqlattack, xterm	U2R
ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster, sendmail, named, snmpgetattack, snmpguess, xlock, xsnoop, worm	R2L

O NSL-KDD já é previamente dividido em uma porção para treinamento e outra para testes. O número total de amostras em cada um é mostrado na **Tabela I**.

Esse conjunto de dados apresenta 41 atributos de entrada, dos quais 5 são binários, 33 são numéricos e 3 são categóricos (categorias na forma de texto).

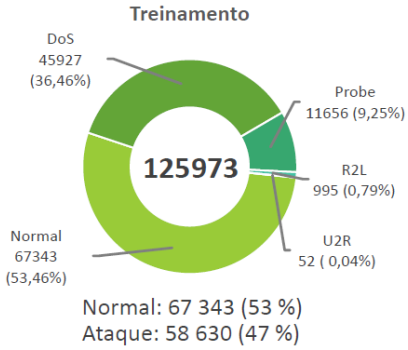
A cada amostra desse conjunto é associado um tipo de conexão: 'normal' ou um dos 39 tipos de ataque que estão listados na **Tabela II**. Os conjuntos de treinamento e de teste não contêm individualmente todos os 39 tipos de ataque. O conjunto de treinamento contém apenas 22 tipos e o conjunto de teste contém 37 tipos. Além disso, 2 tipos de ataque são exclusivos do conjunto de treinamento, enquanto 17 tipos são exclusivos do conjunto de teste. Isso faz com que o modelo treinado tenha mais dificuldade para acertar suas previsões no conjunto de teste. No entanto, a divisão desses dois conjuntos é feita propositalmente, de maneira a simular uma situação real em que novos tipos de ataque estão sempre sendo criados.

Os diferentes tipos de ataque podem ser agrupados em uma das quatro classes apresentadas anteriormente [16], nomeadamente DoS, *probe*, U2R e R2L, conforme mostra a **Tabela II**.

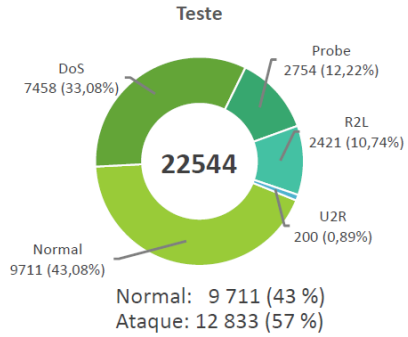
Na **Figura 3**, é mostrada a proporção das classes de ataque nos conjuntos de treinamento e de teste. Dessa forma, podemos perceber que existe um grande desbalanço nos dados, já que existem classes que apresentam muito mais amostras do que outras, como é o caso das classes U2R e R2L, que são minoritárias. Isso dificulta o aprendizado desses ataques menos frequentes e faz com que o modelo crie um viés para a detecção dos ataques mais frequentes.

V. PRÉ-PROCESSAMENTO DOS DADOS

O conjunto de dados passou por uma etapa de pré-processamento antes de ser utilizado. Primeiramente, analisando os dados mais detalhadamente, é possível perceber uma inconsistência em algumas amostras nos valores do atributo 'su_attempted', que indica se o comando 'su_root' foi utilizado durante uma conexão. O esperado é que esse atributo



(a) Conjunto de treinamento.



(b) Conjunto de teste.

Figura 3. Proporção das classes de ataque no NSL-KDD.

seja binário, porém, em algumas amostras, ele assume o valor 2. Portanto, em todas a amostras em que isso acontece, o valor 2 é substituído por 1 [7].

Além disso, todas as amostras apresentam o valor 0 para o atributo 'num_outbounds_cmds', ou seja, ele não pode ser utilizado para diferenciar um exemplo de outro. Portanto, ele foi desconsiderado.

Em seguida, os 3 atributos categóricos foram transformados em binários por meio da codificação *one-hot*. Além disso, os valores dos atributos de entrada foram padronizados.

Finalmente, os dados de treinamento foram rearranjados em uma nova porção de treinamento (80% do *dataset* de treinamento original) e em uma porção de validação (20% do *dataset* de treinamento original).

VI. MÉTRICAS

Para avaliar o desempenho dos detectores de ataques testados, são utilizadas as seguintes métricas:

$$\text{Acurácia} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FP}} \tag{1}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2}$$

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4}$$

$$F_1 = \frac{2 \times P \times \text{TPR}}{\text{P} + \text{TPR}} \tag{5}$$

em que

TP: *True Positive* (em português, Verdadeiro Positivo)
TN: *True Negative* (em português, Verdadeiro Negativo)
FP: *False Positive* (em português, Falso Positivo)
FN: *False Negative* (em português, Falso Negativo).

Nesse tipo de aplicação, é necessário dar atenção especial aos ataques não detectados pelo IDS (falsos negativos), pois a não detecção de uma intrusão impede que o sistema invadido tome medidas para amenizar os efeitos do ataque, o que pode levar a sérias consequências. Por esse motivo, na construção dos detectores deste trabalho, a taxa de verdadeiros positivos (TPR) receberá maior importância.

Apesar do foco na TPR, é importante que o detector também tenha uma taxa de acertos razoável e, por isso, a acurácia também deve ser levada em consideração. Além disso, é interessante observar como a taxa de falsos positivos (FPR) se comporta, pois também é desejável que essa taxa seja pequena. Por fim, o F1-score (F_1) também é usado, pois ele ajuda a avaliar o equilíbrio entre falsos positivos e falsos negativos.

VII. Arquitetura dos Detectores

Nos três detectores implementados neste trabalho, foi empregado o modelo *random forest*. Além disso, todos os experimentos foram realizados na linguagem de programação Python com auxílio da biblioteca *scikit-learn*.

Para cada detector, foi decidido que a saída dos modelos seria a probabilidade de o dado de entrada ser um ataque. Como regra de decisão, foi estabelecido que a entrada é considerada um ataque quando a probabilidade entregue pelo modelo for maior que um certo limiar de detecção. Nesse caso, escolhemos limiares menores do que 0,5, pois, para essa aplicação, o custo de um falso negativo é maior do que o de um falso positivo.

A. Detector 1

Esse detector foi construído utilizando-se uma única *random forest*, que foi treinada no conjunto de dados resultante da combinação do *dataset* de treinamento e de validação utilizando os parâmetros indicados na **Tabela III**. O parâmetro *n_estimators* indica o número de árvores de decisão que compõem a *random forest*, *max_depth* indica a profundidade máxima que as árvores de decisão podem ter e *random_state* é a *seed* para geração de números pseudo-aleatórios do algoritmo.

Os valores dos parâmetros da *random forest* foram obtidos a partir de testes realizados utilizando o *dataset* de validação. Primeiramente, a *random forest* foi treinada com o *dataset* de treinamento. Em seguida, para diferentes combinações dos parâmetros *n_estimators*, *max_depth* e *random_state* foram traçadas curvas das métricas de desempenho em função do valor do limiar de decisão. Quando os valores de *n_estimators*, *max_depth* e *random_state* que fornecem as melhores curvas são encontrados, o limiar de decisão é escolhido de forma a obter valores elevados de TPR e acurácia e valores reduzidos de FPR. A **Figura 4** mostra essas curvas para os parâmetros mostrados na **Tabela III** e o limiar de decisão de 0,27 escolhido segundo esse critério.

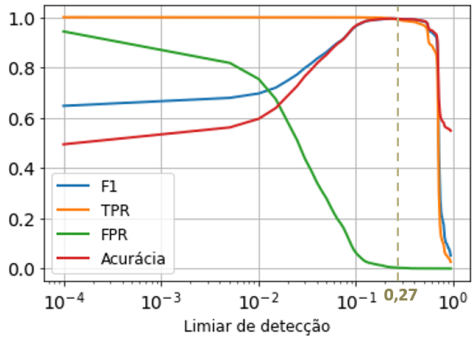


Figura 4. Curvas das métricas de desempenho para o detector 1 (única *random forest*) dentro do conjunto de validação. A linha tracejada indica o valor escolhido para o limiar de decisão.

As **figuras 5** e **6** mostram diagramas que resumem os processos de treinamento e teste desse detector.

Tabela III
Parâmetros do detector 1 (random forest).

Detector 1	
Parâmetro	**Valor**
n_estimators	400
max_depth	20
random_state	10000
Limiar de detecção	0,27

Figura 5. Diagrama ilustrando cada etapa do processo de treinamento do detector 1 (única *random forest*).

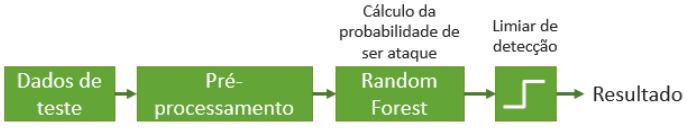


Figura 6. Diagrama ilustrando cada etapa do processo de realização de predições para o detector 1 (única *random forest*).

B. Detector 2

Esse detector agrupa os dados de treinamento em função da classe do ataque a qual eles pertencem e treina um modelo de *random forest* para cada grupo formado.

Durante a etapa de treinamento, os dados são divididos em três grupos: um para os ataques DoS, um para os ataques *probe* e um para os ataques U2R e R2L. Os ataques das classes U2R e R2L foram colocados no mesmo grupo por apresentarem características similares [5]. Além disso, se esses ataques fossem divididos em grupos diferentes, cada um desses grupos teria um número muito reduzido de exemplos, visto que as classes U2R e R2L representam uma proporção muito pequena do total de ataques. Por último, para cada um desses grupos é treinado um modelo de *random forest* diferente. A **Figura 7** mostra o diagrama que resume o processo de treinamento desse detector.

Durante a etapa de realização de predições, um conjunto de dados de teste é fornecido na entrada do detector. Consideremos, por exemplo, que apenas um exemplo seja fornecido na entrada. Esse exemplo é dado como entrada para cada uma das três *random forests* criadas na etapa de treinamento. A saída de cada uma dessas *random forests* é a probabilidade que o exemplo seja um ataque. Em seguida, é calculada a média das saídas das *random forests*. Se o resultado dessa média for superior ao limiar de detecção, o exemplo é considerado um ataque. A **Figura 8** mostra o diagrama que resume o processo de realização de predições com esse detector.

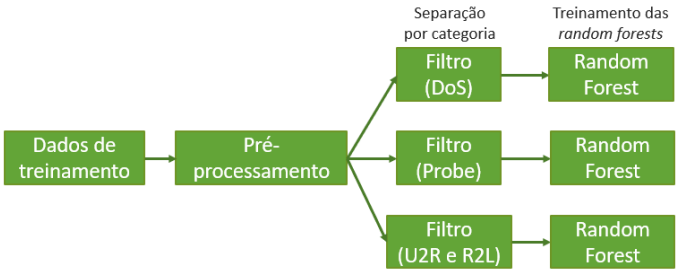


Figura 7. Diagrama ilustrando cada etapa do processo de treinamento do detector 2 (que divide os dados em função da classe do ataque).

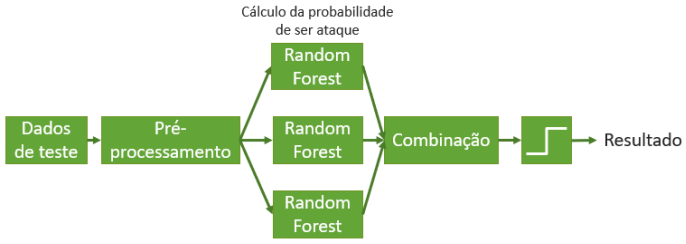


Figura 8. Diagrama ilustrando cada etapa do processo de realização de predições para o detector 2.

Os parâmetros utilizados no treinamento das *random forests* desse detector são mostrado na **Tabela IV**, que foram obtidos utilzando o mesmo procedimento experimental do detector 1. A **Figura 9** ilustra as curvas para as diferentes métricas e o valor escolhido para o limiar de detecção.

Ao analisar a estrutura desse detector, vemos que, assim como o detector 1, ele se baseia no modelo *random forest* para decidir se uma entrada é um ataque ou não. No entanto,

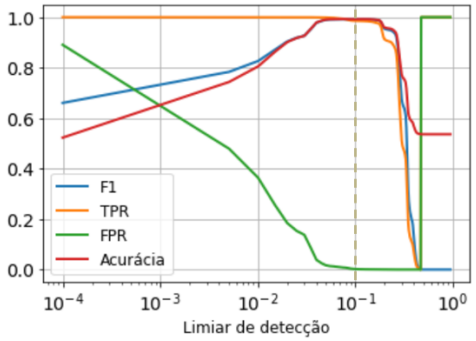


Figura 9. Curvas das métricas de desempenho para o detector 2 dentro do conjunto de validação. A linha tracejada indica o valor escolhido para o limiar de decisão.

Tabela IV
PARÂMETROS UTILIZADOS NO TREINAMENTO DAS RANDOM FORESTS DO DETECTOR 2.

Detector 2	
Parâmetro	**Valor**
n_estimators	400
max_depth	20
random_state	1
Limiar de detecção	0,1

o detector 2 conta com três *random forests* especializadas em uma classe de ataque diferente (com exceção da *random forest* treinada com ataques U2R e R2L). Dessa forma, ao receber um ataque como entrada, é esperado que a árvore especializada nesse ataque apresente em sua saída uma probabilidade de ataque suficientemente elevada para que, junto com a predição das duas outras árvores, faça com que a saída final do detector seja o rótulo 'ataque'.

C. Detector 3

Esse detector separa os dados de treinamento em 8 *clusters* utilizando *K-means* e atribui um modelo de detecção a cada um deles, que varia de acordo com a composição desses *clusters*.

Para o treinamento do modelo *K-means*, foi dada atenção aos parâmetros *max_iter*, que estabelece o número máximo de iterações do algoritmo, *n_init*, que determina quantas vezes os centroides são inicializados a cada iteração, *n_clusters*, que determina quantos *clusters* deverão ser formados e *random_state*, que representa a *seed* para geração dos números pseudo-aleatórios do algoritmo.

Durante o treinamento do modelo *K-means*, o algoritmo calcula as posições dos 8 centroides e faz a correspondência de cada exemplo com um dos 8 *clusters*.

Em seguida, a cada *cluster* é associado um modelo de detecção que depende do número n_A de exemplos que são ataques e do número n_N de exemplos que são normais no *cluster*, como mostrado abaixo:

- Se $n_A > 25$ e $n_N > 25$, atribui-se uma *random forest* ao *cluster*, que será treinada com os dados desse *cluster*;
- Se $n_A > 25$ e $n_N = 0$, atribui-se a função f_1 ao *cluster*, sendo f_1 a função que sempre tem 1 como saída, ou seja,

100% de probabilidade de ser um ataque, independente do que for dado como entrada;
- Se $n_A = 0$ e $n_N > 25$, atribui-se a função f_0 ao *cluster*, sendo f_0 a função que sempre tem 0 como saída, ou seja, 0% de probabilidade de ser um ataque, independente do que for dado como entrada;
- Se $n_A + n_B \leq 25$, considera-se que o caso é um *outlier* e atribui-se a função f_1 ao *cluster*.

A **Figura 10** mostra um diagrama que resume o processo de treinamento do detector 3.

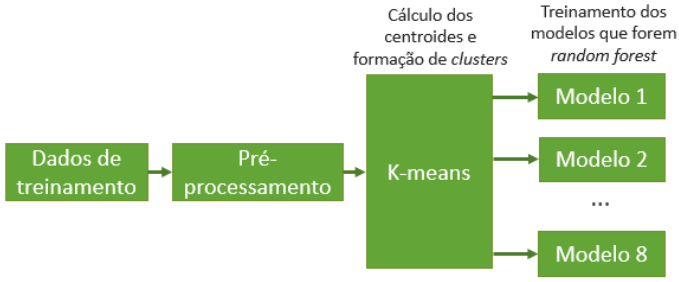


Figura 10. Diagrama ilustrando cada etapa do processo de treinamento do detector 3 (combinação de *K-means* com *random forests*).

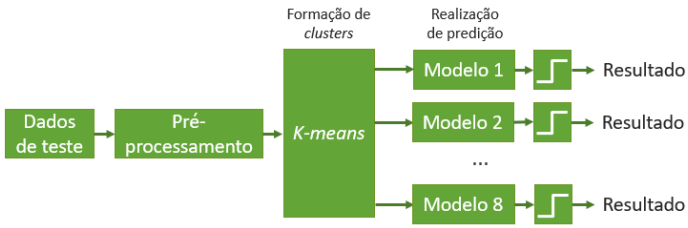


Figura 11. Diagrama ilustrando cada etapa do processo de realização de predições do detector 3 (combinação de *K-means* com *random forests*).

Quando esse classificador, depois de ter passado pela etapa de treinamento, é usado para fazer predições, um conjunto de dados de teste precisa ser fornecido na entrada. Considere que apenas um exemplo é fornecido na entrada. Esse exemplo é inicialmente direcionado para um dos *clusters*, cujos centroides foram definidos na etapa de treinamento. Em seguida, esse exemplo é dado como entrada para o modelo de detecção correspondente ao *cluster* ao qual ele pertence. Se a saída do modelo for maior que o limiar de detecção, o exemplo é considerado um ataque. A **Figura 11** mostra um diagrama que resume o processo de realização de predições com o detector 3.

Os valores que foram atribuídos aos parâmetros do *K-means* e das *random forests* são mostrados na **Tabela V**, que foram obtidos utilizando o mesmo procedimento experimental dos detectores 1 e 2. A **Figura 12** ilustra as curvas para as diferentes métricas e o valor escolhido para o limiar de detecção.

Ao analisarmos a estrutura desse detector, vemos que ele cria 8 modelos de detecção especializados em um determinado tipo de padrão. Quando uma entrada é dada para esse detector,

Tabela V
PARÂMETROS UTILIZADOS NO TREINAMENTO DOS MODELOS DO DETECTOR 3.

Detector 3	
Random forest	
Parâmetro	**Valor**
n_estimators	50
max_depth	20
random_state	1
Limiar de detecção	0,3
K-means	
Parâmetro	**Valor**
n_clusters	8
max_iter	100
n_init	25
random_state	10

o *K-means* identifica esse padrão e associa essa entrada a um dos 8 *clusters*. Assim, o modelo de detecção correspondente a esse *cluster*, que é especializado em dados com esse mesmo padrão, realiza a sua predição. Com isso, espera-se que essa arquitetura de detector forneça predições mais especializadas e possivelmente atinja melhores performances.

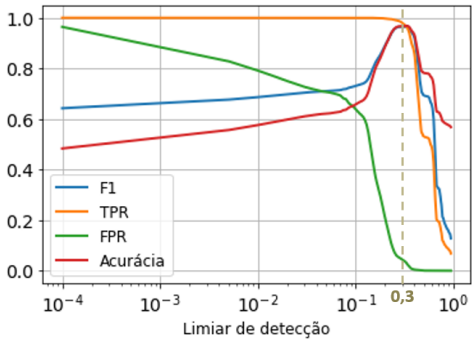


Figura 12. Curvas das métricas de desempenho para detector 3 (combinação de *K-means* com *random forest*) dentro do conjunto de validação. A linha tracejada indica o valor escolhido para o limiar de decisão.

É interessante mencionar que, diferentemente do detector 2, em que toda entrada passa obrigatoriamente pelas três *random forests* e a média das suas saídas é calculada, no detector 3, cada entrada é direcionada apenas para um dos 8 modelos de detecção, de forma que não é necessário calcular uma média na saída do detector.

VIII. RESULTADOS

Os detectores apresentados na **Seção VII** foram testados no conjunto de teste e os resultados são exibidos na **Tabela VI**.

Tabela VI
RESULTADOS DO TESTE PARA OS TRÊS DETECTORES.

	Detector 1	**Detector 2**	**Detector 3**
TPR	89,5 %	87,9 %	**97,8 %**
Acurácia	85,8 %	84,0 %	**94,6 %**
FPR	19,1 %	21,1 %	**9,6 %**
F1-score	87,8 %	86,2 %	**95,4 %**

A partir da análise dessa tabela, conclui-se que o detector 3 apresenta a melhor performance, conseguindo atingir valores de TPR e acurácia bastante elevados sem comprometer a FPR, que se manteve abaixo de 10%. Isso pode ser explicado pelo fato de esse detector dividir os ataques em grupos menores, utilizando 8 *clusters* que ficam responsáveis por exemplos com características semelhantes. Isso permite que o modelo de *random forest* associado a cada *cluster* aprenda melhor as características de cada grupo de exemplo e obtenha melhores resultados durante os testes.

Os detectores 1 e 2 apresentaram resultados bastante similares (com uma ligeira vantagem do detector 1). Para o detector 1, já era esperado um bom desempenho, como foi mostrado pelos autores em [5], que testaram a performance da *random forest* para o mesmo *dataset*. Para o detector 2, por outro lado, era esperado um desempenho superior ao do detector 1, pelo fato de o detector 2 utilizar três *random forests* especializadas em classes de ataques diferentes. No entanto, esse aumento de performance parece não ter acontecido pelo fato de as três *random forest* estarem influenciando a resposta final do detector, por meio da média de suas predições. Assim, para um dado ataque fornecido na entrada, por mais que a *random forest* especializada nesse ataque realize uma predição melhor, o resultado final do detector é impactado também pela predição das outras duas *random forests* não especializadas nesse ataque, o que faz com que a performance desse detector seja semelhante a do detector 1.

IX. Conclusão

O *smart grid*, por apresentar subsistemas altamente dependentes de tecnologias de informação e comunicação, estão muito suscetíveis a ciberataques. Com isso, fica evidente a necessidade de se investir em soluções de segurança para evitar danos ao *smart grid*. Foi apresentada uma possível solução que emprega IDS's baseados em anomalia distribuídos ao longo da rede de comunicação do *smart grid* e foram estudados três modelos de IDS para serem utilizados nessa solução de segurança, que foram construídos utilizando como base o modelo *random forest* associado ao conjunto de dados NSL-KDD.

A partir dos testes conduzidos e analisando os resultados do detector 1, foi possível confirmar que a *random forest* é um modelo que apresenta bom desempenho no problema de detecção de intrusão, conforme o que foi apresentado em [5]. Além disso, analisando o resultado dos outros dois detectores foi possível verificar que a estratégia do detector 3 de criar diferentes modelos de detecção especializados com o *K-means*, em conjunto com *random forests*, resultou em um detector com um melhor desempenho que os detectores 1 e 2. Dessa forma, o detector 3 se mostrou um forte candidato para ser utilizado no IDS da solução de segurança da AMI.

Em uma continuação das pesquisas realizadas nesse trabalho, pode ser interessante estudar a possibilidade de se aplicar técnicas de extração de atributos com o objetivo de reduzir o número de atributos com pouca relevância para o problema. Outro ponto de foco para futuras pesquisas seria a construção de modelos que fossem capazes de realizar não só a detecção de ataques, mas também a classificação deles em uma das quatro classes apresentadas (DoS, *probe*, U2R e R2L).

Agradecimentos

O presente trabalho foi financiado em parte pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Referências

[1] R. D. Fuchs, *Transmissão de energia elétrica: linhas aéreas; teoria das linhas em regime permanente*, vol. 1. Rio de Janeiro: Livros Técnicos e Científicos, 1977.

[2] Ministério de Minas e Energia. "Relatório Smart Grid," Grupo de Trabalho de Redes Elétricas Inteligentes [Online]. Disponível em: www.gov.br/mme/pt-br/assuntos/secretarias/energia-eletrica/relatorio-smart-grid-1/documentos. Acesso em: 02 de fev. 2022.

[3] Z. Buchholz, B.; Styczynski, *Smart Grids: Fundamentals and Technologies in Electric Power Systems of the future*. Berlin: Springer-Verlag GmbH, 2nd ed., 2020.

[4] S. K. Salman, *Introduction to the Smart Grid: Concepts, Technologies and Evolution*. London: The Institution of Engineering and Technology, 2017.

[5] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[6] University of New Brunswick. *NSL-KDD dataset* [Online]. Disponível em: www.unb.ca/cic/datasets/nsl.html. Acesso em: 24 de fev. de 2022.

[7] S. Kalbachou. (2020, Set. 8). *Intrusion Detection on NSL-KDD* [Online]. Disponível em: https://github.com/thinline72/nsl-kdd. Acesso em: 17 de jul. de 2022.

[8] D. Grochocki, J. H. Huh, R. Berthier, R. Bobba, W. H. Sanders, A. A. Cárdenas, and J. G. Jetcheva, "Ami threats, intrusion detection requirements and deployment recommendations," in *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pp. 395–400, 2012.

[9] E. Hossain, I. Khan, F. Un-Noor, S. S. Sikander, and S. H. Sunny, "Application of big data and machine learning in smart grid, and associated security concerns: A review," *IEEE Access*, vol. 7, pp. 13960–13988, 2019.

[10] J. F. Kurose and K. W. Ross, *Computer networking: a top-down approach*. New York: Pearson Education, 6ª ed., 2013.

[11] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[12] Y. Zhang, L. Wang, W. Sun, R. C. Green II, and M. Alam, "Distributed intrusion detection system in a multi-layer network architecture of smart grids," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 796–808, 2011.

[13] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2ª ed.* New York: Springer, 2009.

[14] University of California, Irvine. (1999, Out. 28). *KDD Cup 1999 Data* [Online]. Disponível em: kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Acesso em: 24 de fev. de 2022.

[15] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, 2009.

[16] A. Ingre, B.; Yadav, "Performance analysis of nsl-kdd dataset using ann," in *2015 International Conference on Signal Processing and Communication Engineering Systems*, pp. 92–96, 2015.