Análise de Alertas de Sistemas de Detecção de Intrusão: Uso de Aprendizado Supervisionado na Redução de Alertas Falsos Positivos

Eduardo Alves Moraes[1 2], **Carlos Alexandre Carvalho Tojeiro**[2], **Rodrigo Sanches Miani**[3], **Bruno Bogaz Zarpelão**[1]

[1]Departamento de Computação - Universidade Estadual de Londrina (UEL)
Londrina, PR - Brasil

[2]Faculdade de Tecnologia de Ourinhos - (FATEC - OU) - Ourinhos, SP - Brasil

[3]Faculdade de Computação - Universidade Federal de Uberlândia (UFU)
Uberlândia, MG - Brasil

{eduardo.moraes, carlos.tojeiro}@fatecourinhos.edu.br

miani@ufu.br, brunozarpelao@uel.br

Abstract. *Intrusion Detection Systems (IDS) detect various types of malicious behavior in computer systems, which can compromise their security and reliability. Although IDSs improve system protection, there is a problem: the generation of alerts that do not represent the real situation of the computational environment, referred to as false positive alerts. This paper presents an approach for reducing false positive alerts, using priority alert filtering and machine learning methods. A separation of alerts is performed based on their priorities as well as the insertion of new information based on other data sources. Then machine learning algorithms (kNN and Random Forest) are applied based on a supervised classifier to identify false positives. The approach has achieved its goal by presenting a significant reduction of false positive alerts in a case study conducted on a real corporate network.*

Resumo. *Sistemas de Detecção de Intrusão (IDS - Intrusion Detection System) detectam vários tipos de comportamentos maliciosos em sistemas computacionais, que podem comprometer sua segurança e confiabilidade. Embora os IDS melhorem a proteção dos sistemas, existe um problema: a geração de alertas que não representam a real situação do ambiente computacional, chamados de alertas falsos positivos. Este artigo apresenta uma abordagem de redução de alertas falsos positivos, utilizando filtragem de alertas por prioridade e métodos de aprendizado de máquina. É realizada uma separação de alertas com base nas suas prioridades e a inserção de novos atributos com base em outras fontes de dados. Em seguida, são aplicados os algoritmos de aprendizado de máquina (kNN e Random Forest) com base em um classificador supervisionado para identificar os falsos positivos. A abordagem atingiu o seu objetivo, apresentando uma redução significativa dos alertas falsos positivos em um estudo de caso realizado em uma rede corporativa real.*

 © 2017 Sociedade Brasileira de Computação

1. Introdução

Nas décadas de 80 e 90, o computador pessoal se popularizou, assumindo um papel fundamental no dia a dia das pessoas e organizações. Atualmente, as redes de computadores são os principais meios que as empresas, organizações e governos utilizam na disponibilização de diversos serviços de rede como *Web Services*, Bancos de Dados, Serviços em Nuvem, dentre outros. O maior desafio não é apenas manter a disponibilidade destas redes, mas também detectar a presença de indivíduos conectados, que por razões diversas, tentam paralisar o fornecimento dos serviços, ou interceptar as informações trocadas entre os dispositivos. Grande parte dos administradores de sistemas computacionais, em um dado momento, irão se deparar com um evento de intrusão de segurança durante suas carreiras. Ter um plano de detecção de intrusão resultará em uma notificação mais ágil, minimizando as consequências e permitindo uma rápida recuperação, em caso de algum sinistro [Kurose et al. 2010] [Julisch and Dacier 2002].

A quantidade de incidentes de segurança reportados no Brasil tem mostrado uma tendência de crescimento nos últimos anos, segundo as estatísticas da CERT.br (2015). Dos 722.205 incidentes reportados, 54,17% são ataques do tipo *scan*, 23,37% são ataques do tipo fraude ou páginas falsas, 9,09% são ataques que comprometem servidores Web, 6,61% são ataques do tipo *worm*, 3,51% são ataques de negação de serviços ditribuídos, 0,34% são ataques do tipo invasão à sistemas e os 2,91% são as demais categorias de ataques menos expressivas. Tais dados reforçam a demanda por ferramentas dedicadas à defesa das redes de computadores. Os Sistemas de Detecção de Intrusão (IDS - *Intrusion Detection System*) englobam o processo de monitoramento de redes e dos *hosts*, identificando e notificando a ocorrência de atividades maliciosas ou não autorizadas [Granadillo et al. 2016]. Entende-se como intrusão toda atividade que pode resultar em alteração de privilégios de usuários e permissões em arquivos, instalação de *malware*, acesso não autorizado a arquivos e sistemas, ataques de negação de serviços, presença de *worms* e vírus, estouros de *buffers*, entre outros [Ebrahimi et al. 2011]. Os IDS apresentam seus alertas no formato de *logs*. Por meio da análise destes *logs*, os administradores de sistemas conseguem efetivar um plano de detecção e resposta às intrusões.

No entanto, existe um problema relacionado à efetividade dos IDS: a geração de um grande volume de alertas, dos quais muitos são considerados falsos, dificultando a análise feita pelos administradores de sistemas. Alvarenga et al. (2015) e Granadillo et al. (2016) abordam em seus trabalhos o problema de grandes volumes de alertas que um IDS pode gerar, dificultando o trabalho dos analistas de segurança. Ambos apostam em técnicas de correlação de alertas para facilitar a análise destes grandes volumes de informação. Elshoush et al. (2014), como primeiro passo do processo de correlação de alertas que eles propõem, buscam descartar os alertas falsos, pois desta forma podem reduzir a quantidade de alertas a serem processados. Verma et al. (2015) afirmam que é necessário combinar técnicas de mineração de dados e aprendizado de máquina para compreender as grandes estruturas de dados formadas por alertas, suas correlações e seus comportamentos.

Neste trabalho, é proposto um processo de redução de alertas que se baseia na prioridade dos alertas e na remoção de falsos positivos. Para tanto, assumimos que é possível diferenciar alertas falsos de alertas verdadeiros a partir da análise de informações dos alertas como origem e destino do ataque, assinatura reportada no alerta, função que os *hosts*

 © 2017 Sociedade Brasileira de Computação

envolvidos desempenham na rede, e horário do evento. A partir desta hipótese, desenvolvemos um modelo baseado em aprendizado de máquina supervisionado para identificar os falsos positivos. Para avaliar a proposta, foi realizado um estudo de caso em uma rede corporativa real, utilizando dois algoritmos de classificação supervisionada: o *Random Forest* e o *kNN (k Nearest Neighbors)*. Os resultados mostraram altos índices de redução de alertas e capacidade de classificação de alertas falsos com taxas de acerto próximas a 100%.

Este artigo está organizado na seguinte forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 apresenta a proposta do trabalho, detalhando o processo de redução de alertas. Na Seção 4, é discutido o estudo de caso realizado em uma rede corporativa real e a análise dos resultados obtidos. A Seção 5 contém a conclusão deste artigo.

2. Trabalhos Relacionados

Pesquisadores têm proposto diferentes soluções para lidar com os grandes volumes de alertas gerados pelos IDS, incluindo, entre os pontos abordados, a identificação e tratamento dos alertas falsos positivos. Do ponto de vista da redução de falsos positivos, podemos organizar estes trabalhos em dois grupos.

O primeiro grupo inclui trabalhos que têm a redução de falsos positivos apenas como um passo intermediário de um processo maior. No trabalho de Shittu et al. (2012), é proposta uma abordagem de visualização de alertas de intrusão, utilizando um agente de correlação de alertas que melhora a compreensão de cenários de ataques. Dado um determinado conjunto de eventos gerados por um IDS, a abordagem dos autores seleciona o método mais eficaz para a correlação de eventos de segurança. A eliminação de falsos positivos é apenas um passo presente na etapa inicial do processo, a agregação dos alertas.

Elshoush (2014) propôs um processo no qual a filtragem de alertas falsos é um passo inicial, que visa diminuir o volume de dados a ser analisado durante a correlação, aumentando a sua eficiência. Primeiramente, o autor removeu uma serie de alertas considerados de baixo risco. Depois, os alertas restantes foram agrupados de acordo com os seus *hosts* de origem e destino. Os alertas que não foram agrupados nesta fase foram classificados como falsos. Os autores consideraram que alertas falsos têm um comportamento aleatório e por isso acabam não sendo agrupados de acordo com os critérios utilizados no trabalho. Em nosso trabalho, observamos que alertas falsos não estão ligados obrigatoriamente a situações incomuns. Pelo contrário, alertas falsos têm algumas características em comum, que costumam diferenciá-los de alertas verdadeiros.

No trabalho de Kawakani et al. (2016), os autores propõem uma abordagem baseada no agrupamento de alertas. Duas etapas se destacam: a primeira etapa é a correlação dos alertas e a observação do histórico destes alertas para a identificação das estratégias de ataque utilizadas. A segunda etapa é a associação dos alertas em tempo real às estratégias de ataque descobertas na primeira etapa. Nesta proposta, de maneira similar ao trabalho de Elshoush (2014), alertas que não atenderam aos critérios de agrupamento foram removidos do processo de correlação. No entanto, não houve uma análise mais cuidadosa para certificar se eles eram mesmo falsos ou não.

O segundo grupo de trabalhos inclui aqueles que têm a priorização dos alertas, e a consequente identificação de falsos positivos, como o objetivo final do processo proposto.

© 2017 Sociedade Brasileira de Computação

Em um trabalho mais recente de Shittu et al. (2014), os autores propõem uma métrica para priorização de alertas de intrusão usando a correlação dos alertas. Primeiramente, os alertas são correlacionados em grafos. Então, os alertas pertencentes a grafos que correspondem aos comportamentos menos frequentes recebem prioridades mais altas. Os alertas com prioridades mais baixas são identificados como alertas falsos, sendo filtrados.

Vidal et al. (2015) propuseram a utilização de algoritmos de clusterização na priorização de alertas. No trabalho, primeiramente, há uma fase de treinamento na qual o IDS analisa qual é o comportamento normal da rede. Logo após, o IDS gera alertas para pacotes que se distanciam deste comportamento normal. De acordo com a proposta, para cada novo alerta gerado, é calculada a distância entre o pacote que causou a geração do alerta e os pacotes normais do treinamento. Quanto maior a distância entre o pacote responsável pelo alerta e os pacotes não maliciosos do treinamento, menor será a prioridade do alerta. Alertas com prioridade baixa são considerados falsos.

O trabalho proposto neste artigo pertence ao grupo dos trabalhos que tem a redução de falsos positivos como objetivo final. No entanto, ao invés de correlacionar os alertas para estabelecer a prioridade deles e definir se são falsos ou não, propomos uma abordagem que analisa atributos do alerta que podem indicar se ele é falso ou não, como a função dos *hosts* de origem e destino na rede e o horário do alerta. Desta forma, cada alerta é analisado individualmente, sem necessitar de correlação com outros alertas.

3. Proposta

A proposta deste trabalho é reduzir a quantidade de alertas, filtrando aqueles que possuem prioridade baixa e utilizando algoritmos de aprendizado de máquina para identificar os falsos positivos. A Figura 1 apresenta uma visão geral do modelo proposto.

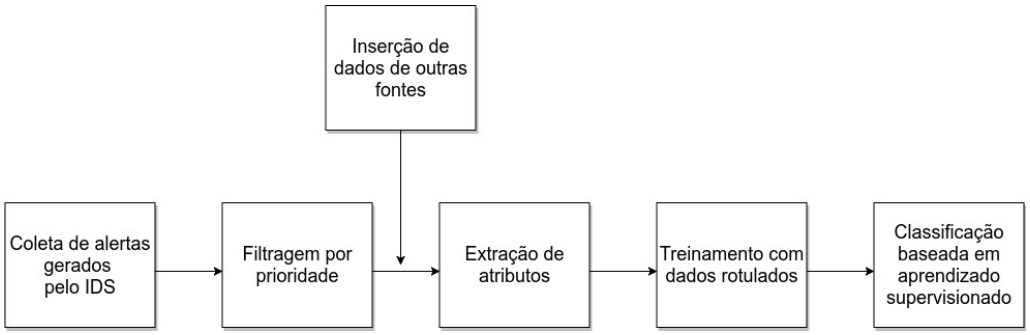

Figura 1. Modelo proposto para redução do número de alertas.

O primeiro passo do modelo proposto é a coleta dos alertas gerados pelo IDS. Logo após coletá-los, os alertas serão filtrados de acordo com a prioridade deles. Existe uma classificação dos alertas que permite aos administradores de sistemas um entendimento rápido e uma priorização mais efetiva de cada evento. Esta classificação normalmente é feita pelos IDS com base na assinatura do alerta. No IDS Snort[1], por exemplo, cada alerta tem uma prioridade determinada por um valor: 1 para prioridade alta, 2 para prioridade média, e 3 para prioridade baixa.

[1]https://www.snort.org/

 © 2017 Sociedade Brasileira de Computação

Na prioridade 1, os alertas representam uma ameaça muito grande ao sistema, precisando de uma análise imediata. A prioridade 2 é considerada importante, podendo se tornar uma ameaça caso não seja analisada e tratada. Na maioria dos casos, pode-se desconsiderar os alertas de prioridade 3 em uma primeira análise, pois são aqueles que não representam uma ameaça, mas sim uma característica mais informativa sobre o estado do sistema. A Figura 2 mostra um exemplo de alertas com prioridade baixa, que reportam uma tentativa de comunicação com a porta 80 do servidor de autenticação do *wireless access point*, gerando alertas com assinatura *(portscan) TCP Portsweep* com prioridade igual a 3. Estes alertas foram gerados para a rede utilizada no estudo de caso apresentado na Seção 4. Este tipo de evento é muito típico e normalmente está relacionado à solicitação de um *host* para que suas credenciais sejam verificadas durante a conexão com o *access point*. Mesmo que haja uma intenção maliciosa, esta tentativa de conexão tem um baixo grau de risco.

```
04/17/2016-16:52:23.705315 [**] [122:3:0] (portscan) TCP Portsweep [**]
[Classification: Attempted Information Leak] [Priority: 3] {TCP}
10.0.4.172:3556 -> 10.0.0.1:80

04/17/2016-16:54:23.913522 [**] [122:3:0] (portscan) TCP Portsweep [**]
[Classification: Attempted Information Leak] [Priority: 3] {TCP}
10.0.5.100:3447 -> 10.0.0.1:80
```

Figura 2. Exemplos de alertas de prioridade 3.

Na filtragem por prioridade, o nosso modelo prevê que todos os alertas de uma determinada prioridade sejam separados do conjunto de alertas que será submetido ao processo de busca por falsos positivos a ser realizado no próximo passo.

Depois de filtrar os alertas de acordo com a prioridade definida pelo IDS, se inicia o processo de identificação de falsos positivos utilizando aprendizado de máquina supervisionado. O primeiro passo neste processo é inserir novas informações de outras fontes à base de alertas que serão analisados. Esta tarefa é realizada pelo administrador da rede. Para demonstrar a importância de inserir informações de outras fontes à base de alertas, tomaremos como exemplo um alerta gerado pelo IDS utilizado no estudo de caso apresentado na Seção 4. O alerta é apresentado na Figura 3.

```
04/17/2016-13:52:23.053985 [**] [1:2001219:20] ET SCAN Potential SSH
Scan [**] [Classification: Attempted Information Leak] [Priority: 2]
{TCP} 218.93.17.146:38454 -> [redacted]:22
```

Figura 3. Exemplo de alerta gerado por um IDS Snort.

A Tabela 1 detalha quais são as informações providas no alerta. São informações básicas, que indicam algumas características do evento, mas não conseguem mostrar sozinhas, por exemplo, qual é a localização geográfica dos *hosts* envolvidos no ataque.

Buscando informações em outras fontes, é possível descobrir mais alguns detalhes sobre o evento que podem ajudar a identificar se o alerta é falso ou não. Utilizando um recurso de geolocalização de endereços IP como o GeoIP2 [2], pode-se extrair informações

[2]https://www.maxmind.com/pt/geoip-demo

© 2017 Sociedade Brasileira de Computação

Tabela 1. Descrição do alerta.

Atributo	Valor
Mês / Dia / Ano	04/17/2016
Hora / Minuto / Segundo	13:52:23.053985
Código da Assinatura	[1:2001219:20]
Assinatura	ET SCAN Potential SSH Scan
Classificação	[Classification: Attempted Information Leak]
Prioridade	[Priority: 2]
Protocolo	TCP
Endereço de IP de Origem	218.93.17.146
Porta de Origem	38454
Endereço de IP de Destino	(Endereço IP do Servidor Web da faculdade)
Porta de Destino	22

mais detalhadas sobre o endereço IP do atacante, como cidade, país, latitude, longitude, raio de ação, provedor de Internet e organização. Como estas informações estão relacionadas ao endereço IP, é importante considerar que existe a possibilidade do endereço IP ser falsificado pelo atacante. A Tabela 2 apresenta estas informações para o endereço IP 218.93.17.146. Este endereço IP é apontado como endereço do atacante no alerta da Figura 3.

Tabela 2. Geolocalização de endereço IP.

Atributo	Valor
Código do País	CN
Localização	Changzhou, Jiangsu, China, Ásia
Coordenadas aproximadas (Latitude e Longitude)	31.7833, 119.9667
Raio de ação (em Km)	50
Internet Service Provide (ISP)	China Telecom
Organização	Liyang Hongkou Primary School
Fonte: https://www.maxmind.com/pt/geoip-demo	

Os atributos destacados na Tabela 2 podem melhorar a qualidade das informações contidas no alerta. Neste caso, o endereço IP 218.93.17.146 está localizado na China, província de Jiangsu. A princípio, muitos analistas de segurança pensariam que, pelo endereço IP, o ataque teria sido realizado por algum invasor tentando conexão SSH na porta 22, utilizando um serviço de *Proxy* externo ou a Rede Tor, dificultando sua localização exata. Porém, o provedor do serviço de Internet é a China Telecom, e a organização é uma escola primária denominada Liyang Hongkou. De acordo com consulta no Google Maps[3], a escola realmente existe e as coordenadas do endereço IP coincidem com a localização física. Assim, levanta-se também a possibilidade de que o invasor tenha comprometido um sistema desta escola, passando a utilizar este endereço IP. De qualquer forma, todas as características ajudam a mostrar que o alerta é verdadeiro, já que a rede alvo pertence a uma faculdade brasileira, cujos serviços não são normalmente acessados por *hosts* na China, de acordo com seus administradores. Em nossa proposta, bus-

[3]https://www.google.com.br/maps

 © 2017 Sociedade Brasileira de Computação

camos, além de informações sobre a localização geográfica dos endereços IP, informações sobre a topologia da rede, determinando quais dos *hosts* envolvidos fazem parte da rede alvo, e caso façam parte, se eles são servidores ou não.

Considerando os dados dos próprios alertas e as informações adicionais, extraímos os seguintes atributos para cada alerta.

- Atributos referentes ao período - contribuem na identificação de alertas falsos conforme o horário da ocorrência do alerta. É atribuído um valor lógico para estes atributos (0 ou 1):
  - *manhã*: indica se o alerta foi gerado entre 6h00 e 11h59;
  - *tarde*: indica se o alerta foi gerado entre 12h00 e 17h59;
  - *noite*: indica se o alerta foi gerado entre 18h00 e 23h59;
  - *madrugada*: indica se o alerta foi gerado entre 0h00 e 5h59;
- Atributos referentes ao tipo de ataque - auxiliam no entendimento da natureza do ataque. Determinadas assinaturas e prioridades podem apresentar uma possibilidade maior de ter alertas falsos:
  - *assinatura*: código da assinatura presente no alerta;
  - *prioridade*: prioridade atribuída pelo IDS ao alerta;
- Atributos referentes à localização física do *host* e aos serviços atacados- a ideia é identificar *hosts* e serviços de rede que têm maior propensão a gerar alertas falsos e usar esta informação para classificar os alertas:
  - *porta_origem*: porta de origem do ataque presente no alerta;
  - *porta_destino*: porta de destino do ataque presente no alerta;
  - *parte1_end_origem*: 1º octeto do endereço IP de origem;
  - *parte2_end_origem*: 2º octeto do endereço IP de origem;
  - *parte3_end_origem*: 3º octeto do endereço IP de origem (o quarto octeto não foi necessário, pois na maioria dos casos os três primeiros já permitem a identificação da sub-rede);
  - *parte1_end_dst*: 1º octeto do endereço IP de destino;
  - *parte2_end_dst*: 2º octeto do endereço IP de destino;
  - *parte3_end_dst*: 3º octeto do endereço IP de destino (o quarto octeto não foi necessário, pois na maioria dos casos os três primeiros já permitem a identificação da sub-rede);
  - *país_origem*: indica se o endereço IP de origem está localizado no mesmo país da rede monitorada pelo IDS;
  - *país_dst*: indica se o endereço IP de destino está localizado no mesmo país da rede monitorada pelo IDS;
- Atributos referentes à função do *host* - auxilia na identificação de alertas verdadeiros e falsos com base na funcionalidade do *host* de origem ou destino. Por meio da análise do endereço IP, identifica se o *host* de origem ou destino pertence à rede monitorada. Em caso positivo, é possível verificar se ele é um servidor da rede interna ou não. O campo foi preenchido a partir da análise dos endereços IP e do conhecimento sobre a alocação destes endereços na rede:
  - *origem_servidor*: a partir da análise do endereço IP de origem, indica se a origem é um servidor da rede monitorada;
  - *dst_servidor*: a partir da análise do endereço IP de destino, indica se o destino é um servidor da rede interna;

© 2017 Sociedade Brasileira de Computação

– *origem_host_externo*: a partir da análise do endereço IP de origem, indica se a origem é um equipamento externo à rede monitorada;
– *dst_host_externo*: a partir da análise do endereço IP de destino, indica se o destino é um equipamento externo à rede monitorada.

Na etapa de treinamento dos dados, os algoritmos supervisionados construirão um modelo de classificação com base nos alertas rotulados pelo administrador de rede como falsos ou verdadeiros. Nos experimentos que serão apresentados na Seção 4, os algoritmos *Random Forest* e *kNN* serão utilizados no modelo para que os seus desempenhos sejam comparados. O último passo do modelo é a classificação feita pelos algoritmos sobre o conjunto de dados definido no passo anterior. Estes algoritmos, com base na fase de treinamento, classificarão cada alerta como falso ou verdadeiro.

4. Resultados e Discussões

Nesta seção, será apresentado um estudo de caso realizado em uma rede corporativa real para avaliar o modelo de redução de alertas proposto neste trabalho. A topologia de redes de computadores onde foi aplicado este estudo é representada na Figura 4.

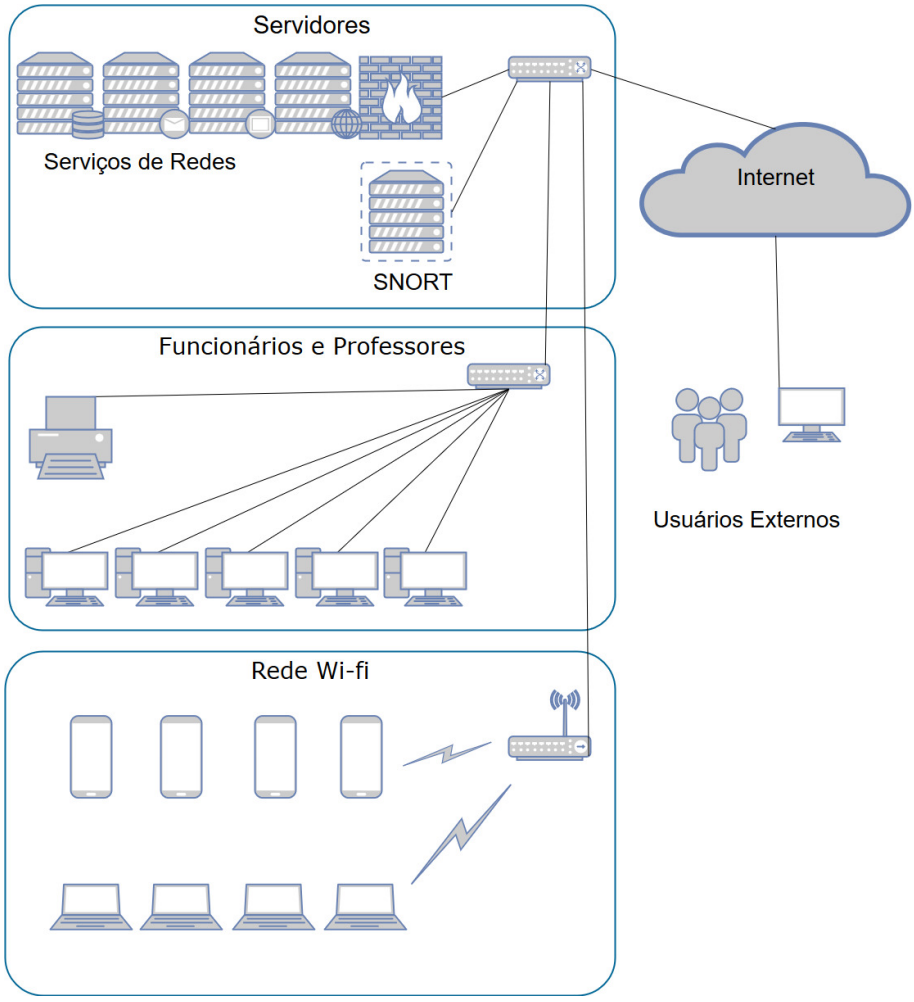

Figura 4. Topologia da rede de computadores.

© 2017 Sociedade Brasileira de Computação

A rede de computadores visualizada na Figura 4 pertence a Faculdade de Tecnologia de Ourinhos (FATEC-OU). Ela é acessada por cerca de 1750 usuários, entre alunos, professores e funcionários, tanto internamente por uma rede sem fios e pela rede cabeada dos departamentos, como externamente. Os servidores disponibilizam principalmente serviços Web, como a página principal da faculdade, o *blog*, serviços acadêmicos, materiais de ensino e bancos de dados acadêmicos.

Com base nas estatísticas mensais, nestes servidores são estabelecidas cerca de 1.000 conexões diárias. Na rede monitorada, há um IDS Snort instalado, cujas assinaturas são atualizadas pelo próprio sistema, que gera em média 230.000 alertas diários, ou seja, mais de 2 alertas por segundo.

Para o estudo de caso realizado neste trabalho, foram utilizados os alertas referentes ao mês de março de 2016 gerados pelo IDS Snort, pois neste mês ocorreram dois fatos importantes que geraram certas mudanças no comportamento dos alertas. O primeiro fato foi o início das atividades práticas nos laboratórios de informática da faculdade para o primeiro semestre de 2016, ocorrido no início da terceira semana deste mês, o que costuma propiciar ações inadequadas na rede e um grande aumento do número de alertas. O segundo fato foi a ocorrência de um feriado prolongado na quarta semana, causando a diminuição das atividades da rede e alterando o volume e as características dos alertas. A divisão por semanas ocorreu para se obter resultados mais precisos, pois se a análise fosse feita no mês como um todo, não se conseguiria observar se fatos como início das aulas práticas e feriados teriam algum impacto nos resultados. A Tabela 3 mostra a quantidade de alertas por prioridade gerados neste mês.

Tabela 3. Total de alertas por prioridade referente ao mês de Março de 2016

	Prioridade 1	Prioridade 2	Prioridade 3	Total geral
Semana 1	239	930	26.704	27.873
Semana 2	601	4.124	275.168	279.893
Semana 3	9.878	2.466	1.302.703	1.315.047
Semana 4	445	4.524	284.177	289.146
Semana 5	479	3.095	296.459	300.033
TOTAL	11.642	15.139	2.185.211	2.211.992

O primeiro passo do modelo proposto é a filtragem de todos os alertas com uma determinada prioridade. Em nosso estudo de caso, filtramos todos os alertas com prioridade 3, pois são aqueles que representam normalmente uma característica informativa sobre o estado do sistema. Cerca de 91% destes alertas são do tipo *Portsweep*, ou seja, tratam de uma varredura por portas abertas na rede. Normalmente, na rede monitorada, este alerta é gerado quando um dispositivo móvel de um aluno, funcionário ou professor solicita uma autenticação ao *access point* para poder se conectar à Internet. Os outros 9% de alertas são majoritariamente dos tipos:

- *ICMP test detected*: evento de segurança gerado quando um teste é realizado para verificar se o servidor está respondendo;
- *Protocol mismatch*: evento de segurança gerado quando um teste é realizado para verificar se um serviço está respondendo em uma determinada porta;

© 2017 Sociedade Brasileira de Computação

Seguindo o modelo proposto, foram adicionadas aos alertas informações de outras fontes de dados sobre os eventos de segurança como a localização geográfica do endereço IP de origem e de destino, e a função do endereço IP na rede analisada. Então, foi realizada a extração dos atributos elencados na Seção 3, formando o conjunto de dados que será analisado pelos algoritmos de classificação supervisionada. Um algoritmo supervisionado trabalha em duas etapas. Primeiro, há um treinamento no qual o algoritmo recebe como entrada um conjunto de dados rotulados. Neste treinamento, o algoritmo constrói um modelo de classificação com base nos dados rotulados. No estudo de caso, os alertas foram rotulados como verdadeiros ou falsos pelos profissionais da administração de redes da faculdade. Na segunda etapa, o classificador usa o modelo para classificar novas instâncias de dados que são apresentadas a ele. Utilizamos dois algoritmos de classificação supervisionada: *kNN* e *Random Forest*.

O *kNN* classifica uma instância com base em seus k vizinhos mais próximos. Para definir a distância entre uma instância e seus vizinhos, é necessário usar uma métrica como a distância Euclidiana. Uma vez que o algoritmo encontra os k vizinhos mais próximos, ele realiza um processo de votação. A classe que tiver mais instâncias entre os k vizinhos mais próximos é eleita como a classe da instância em análise. O algoritmo *Random Forest*, por sua vez, combina diversas árvores de decisão, sendo então um algoritmo de *ensemble learning*. Cada árvore é formada a partir de um conjunto de atributos obtidos aleatoriamente e instâncias amostradas de acordo com a técnica *bootstraping*. Durante a classificação, a instância a ser analisada é apresentada a todas as árvores e cada uma delas vota em uma classe como a mais adequada. A classe que obtiver mais votos será escolhida como a classe para a instância em questão.

Para medir o desempenho do classificador, foram utilizadas as seguintes métricas:

- Precisão: no conjunto de alertas classificados como verdadeiros, esta métrica mostra quantos realmente eram verdadeiros;

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

- Sensibilidade: considerando o conjunto de alertas que realmente eram verdadeiros, esta métrica mostra quantos destes alertas foram classificados corretamente;

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

- Especificidade: considerando o conjunto de alertas que realmente eram falsos, esta métrica mostra quantos destes alertas foram classificados corretamente;

$$Specificity = \frac{TN}{TN + FP} \tag{3}$$

- F-score: avalia a cobertura e a precisão da classificação, considerando a média ponderada da precisão e da sensibilidade. Quanto mais próxima de 1, melhores são a cobertura e a precisão da classificação;

$$F - Score = \frac{2 * (Precision * Recall)}{(Precision + Recall)} \tag{4}$$

 © 2017 Sociedade Brasileira de Computação

TP, FP, TN, e FN correspondem respectivamente a *true positive* (positivo verdadeiro), *false positive* (falso positivo), *true negative* (negativo verdadeiro), e *false negative* (falso negativo).

Para avaliar os dois classificadores escolhidos, utilizamos o método de validação cruzada. Este método tem como proposta particionar o conjunto de dados em n subconjuntos de mesmo tamanho. A partir deste ponto, um dos subconjuntos é utilizado para teste e o restante para treinamento. As rodadas de avaliação são repetidas n vezes e em cada uma destas rodadas um sub-conjunto diferente é usado para teste e os outros são usados para treinamento. Aplicamos o método de validação cruzada em cada uma das semanas do mês de março de 2016. Em cada semana, dividimos os alertas em 7 subconjuntos, isto é, $n = 7$. A Figura 5 mostra os resultado obtidos com a aplicação do algoritmo *kNN*.

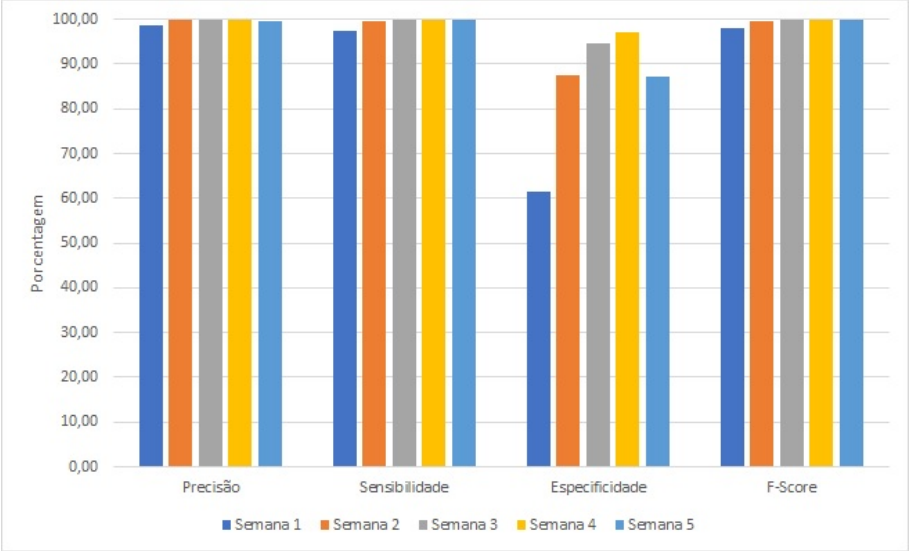

Figura 5. Resultados para o algoritmo kNN.

Na Figura 5, podemos observar que os resultados obtidos com o algoritmo de aprendizado de máquina *kNN* foram próximos para todas as semanas, com exceção da semana 1. Nesta semana, a métrica Especificidade (equação 3) teve o resultado de 61,54%, diferente das outras semanas que variaram entre 87,34% e 97,19%. Este resultado indica que na semana 1 o *kNN* encontrou dificuldades em classificar corretamente os alertas falsos, apontando erroneamente que alguns seriam verdadeiros. Este fato pode ter ocorrido em decorrência da quantidade de alertas falsos positivos ser muito pequena (53 alertas falsos positivos) quando comparada com a quantidade de alertas verdadeiros (1.116 alertas verdadeiros). O desbalanceamento entre as classes pode ser um obstáculo para o *kNN*. Apesar deste resultado ruim para a semana 1, nota-se que os resultados para todas as métricas nas outras semanas foram bons, ficando sempre próximos a 100%.

A Figura 6 mostra os resultados obtidos com a aplicação do algoritmo *Random Forest*.

Na Figura 6, os resultados obtidos com o *Random Forest* foram praticamente iguais para todas as semanas, sem exceção. Todas as métricas, para todas as semanas se situam entre 95,08% e 100%. O *Random Forest* conseguiu superar o *kNN* na semana

© 2017 Sociedade Brasileira de Computação

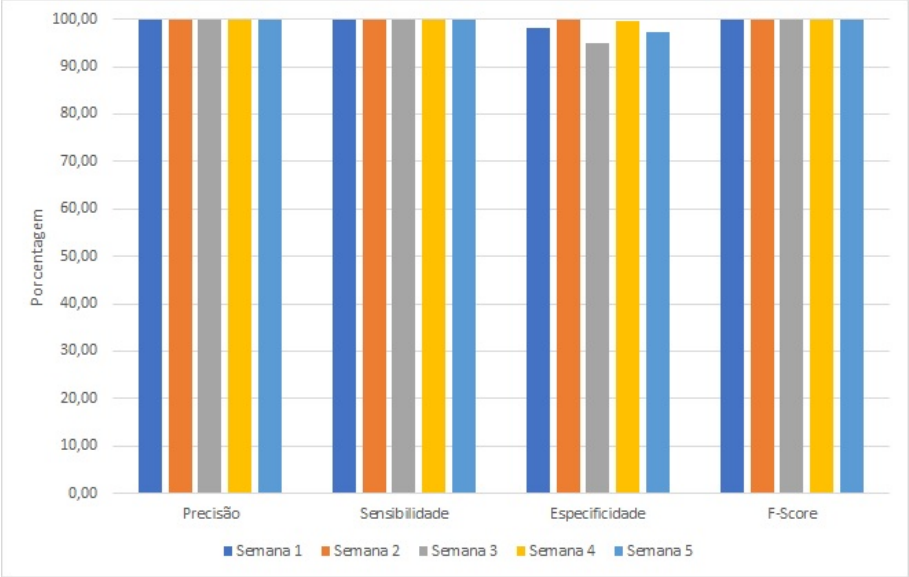

Figura 6. Resultados para o algoritmo Random Forest.

1, quando este último teve um resultado ruim para a Especificidade. No caso do *Random Forest*, mesmo com o desbalanceamento das classes para a semana 1, a Especificidade alcançada foi bastante alta. Isto ocorre porque o *Random Forest*, ao ser um algoritmo de *ensemble learning*, é mais robusto para lidar com o desbalanceamento de classes. Em geral, os resultados do *Random Forest* foram levemente melhores que os obtidos com o *kNN*.

À seguir, serão apresentados um caso de alerta classificado acertadamente como alerta falso e também um caso de alerta classificado corretamente como alerta verdadeiro. O objetivo é mostrar dois casos típicos que ilustram o funcionamento da proposta. A Figura 7 exibe um alerta classificado corretamente como falso positivo.

```
03/09/2016-16:24:49.373444  [**]  [1:1201:7]  ATTACK-RESPONSES  403
Forbidden [**] [Classification: Attempted Information Leak] [Priority:
2] {TCP} [redacted]:80 -> [redacted]:53207
```

Figura 7. Alerta falso positivo.

O alerta exibido na Figura 7 se refere a uma resposta de um servidor Web interno da faculdade para outro servidor interno, responsável por hospedar o *blog* da faculdade. Esta resposta indica que uma página está proibida de ser acessada, por estar passando por alguma atualização ou inserindo um novo aviso interno para os alunos ou professores. Apesar de ser apenas uma resposta informativa, ela foi classificada pelo IDS como ataque. É importante observar que os atributos que extraímos para cada alerta ajudam a mostrar que o alerta é falso. Como exemplo, podemos destacar os atributos *origem_servidor* e *dst_servidor*, que neste caso mostram que o alerta retrata um suposto ataque entre dois servidores da rede da faculdade. Este tipo de ocorrência, apesar de possível, é bastante incomum no ambiente em questão, como pode ser verificado durante o treinamento dos modelos supervisionados. Desta forma, as informações ajudam o classificador a determinar corretamente que o alerta é falso.

 © 2017 Sociedade Brasileira de Computação

A Figura 8 exibe uma alerta classificado corretamente como verdadeiro.

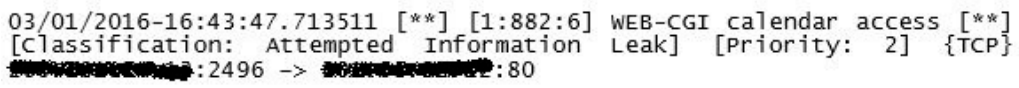

Figura 8. Alerta verdadeiro.

O alerta exibido na Figura 8 é referente a um ataque realizado por um *host* externo, localizado na cidade de Jacarezinho, no estado do Paraná. O atacante tenta executar um *script* de manipulação de calendário escrito em linguagem *Perl*, que permite a execução de comandos sem verificação de entrada de dados. O modelo que propomos neste trabalho classificou esta ocorrência como ataque ao analisar características como o *host* de origem, sua localização física e a assinatura do alerta.

Finalmente, podemos concluir que o modelo proposto para a redução de alertas obteve sucesso. Ambos os algoritmos de classificação supervisionada, destacando o *Random Forest* por sua robustez frente ao desbalanceamento de classes, mostraram ótimos resultados na identificação de falsos positivos. Antes, os dados brutos somavam 2.211.992 alertas para o mês de março de 2016. Com a filtragem dos alertas de baixa prioridade (prioridade 3), este número caiu para 26.781 alertas. Após a inserção de dados de outras fontes e a aplicação do algoritmo de aprendizado supervisionado *Random Forest*, foram identificados 784 alertas falsos, cerca de 3% dos alertas de prioridades 1 e 2. Ao final da aplicação do modelo proposto, ao invés de analisar 2.211.992 alertas, os administradores poderiam analisar 25.997 alertas em um mês, sem se preocupar com alertas falsos.

5. Conclusão

Este trabalho apresentou um modelo de redução de alertas falsos positivos por meio de filtragem de prioridades de alertas e técnicas de aprendizado de máquina utilizando um classificador supervisionado. Dois algoritmos foram utilizados e observou-se que ambos atingiram bons resultados: *kNN* e *Random Forest*. Dos algoritmos, o *Random Forest* foi o algoritmo que apresentou, de modo discreto, os melhores resultados quando comparado com o *kNN*, devido à sua capacidade de lidar melhor com desbalanceamento entre as classes. A filtragem de alertas por prioridade e a inserção de dados de outras fontes foram importantes no desempenho dos classificadores. Com o conhecimento sobre a topologia e as funções dos *hosts* internos da rede de computadores, pode-se destacar características mais específicas de cada evento de segurança gerado pelo IDS Snort como: função do *host* de origem e de destino, localização física do *host* e comportamento de *hosts* de dentro e fora do país. Selecionando os alertas relevantes para o classificador e filtrando os alertas que apresentaram baixa prioridade, os algoritmos, de modo geral, apresentaram um bom desempenho na identificação de alertas falsos positivos e, consequentemente, alcançaram os resultados esperados.

Para os trabalhos futuros, pretende-se propor métodos para correlacionar os alertas resultantes do processo de redução de forma a mostrar os cenários de ataque aos administradores de rede de forma mais amigável.

© 2017 Sociedade Brasileira de Computação

Referências

Alvarenga, S. C. d., Zarpelão, B. B., and Miani, R. S. (2015). Discovering attack strategies using process mining. In *The Eleventh Advanced International Conference on Telecommunications*, pages 119–125.

CERT.br − Centro de Estudos, R. e. T. d. I. d. S. n. B. (2015). *Estatísticas dos Incidentes Reportados ao CERT.br.* Disponível em: https://www.cert.br/stats/incidentes/ . Acessado em Junho de 2017. CERT.br.

Ebrahimi, A., Navin, A. H. Z., Mirnia, M. K., Bahrbegi, H., and Ahrabi, A. A. A. (2011). Automatic attack scenario discovering based on a new alert correlation method. In *Systems Conference (SysCon), 2011 IEEE International*, pages 52–58. IEEE.

Elshoush, H. T. I. (2014). An innovative framework for collaborative intrusion alert correlation. In *Science and Information Conference (SAI), 2014*, pages 607–614. IEEE.

Granadillo, G. G., El-Barbori, M., and Debar, H. (2016). New types of alert correlation for security information and event management systems. In *New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on*, pages 1–7. IEEE.

Julisch, K. and Dacier, M. (2002). Mining intrusion detection alarms for actionable knowledge. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 366–375. ACM.

Kawakani, C. T., Junior, S. B., Miani, R. S., Cukier, M., and Zarpelão, B. B. (2016). Intrusion alert correlation to support security management. In *Proceedings of the XII Brazilian Symposium on Information Systems on Brazilian Symposium on Information Systems: Information Systems in the Cloud Computing Era-Volume 1*, page 42. Brazilian Computer Society.

Kurose, J. F., Ross, K. W., Marques, A. S., and Zucchi, W. L. (2010). *Redes de Computadores ea Internet: uma abordagem top-down*. Pearson.

Shittu, R., Healing, A., Bloomfield, R., and Muttukrishnan, R. (2012). Visual analytic agent-based framework for intrusion alert analysis. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2012 International Conference on*, pages 201–207. IEEE.

Shittu, R., Healing, A., Ghanea-Hercock, R., Bloomfield, R., and Muttukrishnan, R. (2014). Outmet: A new metric for prioritising intrusion alerts using correlation and outlier analysis. In *Local Computer Networks (LCN), 2014 IEEE 39th Conference on*, pages 322–330. IEEE.

Verma, R., Kantarcioglu, M., Marchette, D., Leiss, E., and Solorio, T. (2015). Security analytics: essential data analytics knowledge for cybersecurity professionals and students. *IEEE Security & Privacy*, 13(6):60–65.

Vidal, J. M., Orozco, A. L. S., and Villalba, L. J. G. (2015). Quantitative criteria for alert correlation of anomalies-based NIDS. *IEEE Latin America Transactions*, 13(10):3461–3466.

 © 2017 Sociedade Brasileira de Computação