Detecção de Intrusões por Monitorização do Comportamento através de Mudanças de *Clusters*

Tiago Fernandes[1,2] Luis Dias[1,2] Miguel Correia[2]
[1]CINAMIL, Academia Militar, Instituto Universitário Militar – Portugal
[2]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa – Portugal
fernandes.tfc@exercito.pt dias.lfxcm@exercito.pt miguel.p.correia@tecnico.ulisboa.pt

***Resumo*—O presente trabalho apresenta uma abordagem de detecção de intrusão que assinala a actividade maliciosa sem conhecimento prévio sobre ataques ou sem dados de treino. A abordagem *Cluster Change-Based Intrusion Detection* (`C2BID`) detecta intrusões através da monitorização do comportamento de máquinas (servidores, computares ou outros elementos de uma rede de computadores). Para esse efeito, o `C2BID` define e extrai características da rede, agrega máquinas com comportamento semelhante utilizando o *clustering*, e analisa depois como as máquinas se movem entre *clusters* ao longo de um período de tempo. Isto contrasta com os trabalhos anteriores na área que param na etapa de *clustering*. Avaliamos experimentalmente este sistema com um conjunto de dados de avaliação (artificiais) e um conjunto de dados do reais, obtendo melhor F-Score do que as soluções anteriores.**

***Palavras-Chave*—Detecção de intrusões, análise de *clusterings*, mudanças de comportamento, análise de segurança**

I. Introdução

No último século, as telecomunicações, o hardware e o software têm tido uma função vital da nossa sociedade, traduzindo-se numa enorme dependência da Internet e, consequentemente, na necessidade de a proteger. A *detecção de intrusões* tornou-se um tópico de investigação importante devido aos avanços nas tecnologias e ao número crescente de ataques em redes de computadores. Actualmente, os Sistemas de Detecção de Intrusão (SDI) estão concebidos para identificar ameaças e possíveis incidentes [1]. Embora estas e outras medidas de segurança permitam detectar e bloquear alguns ataques em tempo real, outros ataques são mais elusivos e podem causar danos mais graves. Especificamente, as Ameaças Persistentes Avançadas (APA) levam frequentemente semanas ou mesmo meses a ser detectadas, conforme relatórios de [2] uma APA leva uma média de 58 dias para ser detectada. Por conseguinte, os mecanismos de segurança tradicionais não fornecem protecção suficiente e as organizações devem investigar o tráfego e os registos para procurar padrões anómalos em janelas de tempo maiores.

Existem duas abordagens clássicas de detecção de intrusões: a detecção baseada em assinaturas e a detecção baseada em anomalias. A detecção baseada na assinatura identifica ataques conhecidos através da detecção de padrões observados no atacantes. A detecção de anomalias tenta encontrar padrões que não se coadunam com o comportamento normal esperado [3]. Ambas as abordagens requerem conhecimento prévio, respectivamente das assinaturas e do comportamento normal, algo que é inconveniente e que contornamos neste trabalho.

As técnicas de aprendizagem automática dividem-se em duas categorias: supervisionada e não supervisionada [4].

Métodos não supervisionados – técnicas por nós utilizadas – é interessante para a detecção de intrusão porque, por definição, não requer conhecimento prévio, ao contrário dos métodos clássicos baseados em assinaturas ou em anomalias. A aprendizagem não supervisionada e especificamente o *clustering* (agregação) têm recebido alguma atenção no contexto da detecção de intrusão. Um algoritmo de *clustering* é aplicado a vectores de dados, cada vector representando uma entidade (por exemplo, uma máquina ou um utilizador), para agrupar entidades com comportamento semelhante, ou seja, com valores semelhantes. Os grupos ou *clusters* resultantes podem ser analisados manualmente e assinalados como maliciosos ou pode ser aplicado um método automático de análise como: detecção outliers, limiares, ou considerando pequenos *clusters* suspeitos.

Muitos trabalhos assumem que apenas uma pequena parte do tráfego é malicioso, por exemplo, [5]–[7]. Isto leva a que ataques com várias máquinas, por exemplo, uma botnet, não seja facilmente detectado. Além disso, os atacantes podem frequentemente contornar os sistemas baseados em aprendizagem automática, executando ataques a baixo ritmo ou em múltiplas janelas de tempo, por exemplo, fazendo um *Port Scan* lento [5]. Alguns modelos [8], [9] dependem de *clustering* e definição de limiares, o que está associado a um período de treino, podendo ser evitado pelos atacantes. Outros dependem da classificação manual, pelo menos numa fase inicial [7], [10].

Neste artigo propomos uma nova abordagem para a detecção de intrusão em redes de computadores baseada na aprendizagem não supervisionada: *Cluster Change-Based Intrusion Detection* (`C2BID`). A nossa abordagem utiliza o *clustering*, tal como os trabalhos acima referidos, mas o *clustering* é apenas um primeiro passo para compreender se uma máquina é maliciosa ou não. A ideia principal da abordagem do `C2BID` é detectar intrusões através da monitorização das mudanças de comportamento de uma máquina. Para esse fim, o sistema define e extrai dados da rede – especificamente dados de fluxo de rede [11]–[13] –, agrega máquinas com comportamento semelhante utilizando o *clustering*, depois analisa como as máquinas se movem entre *clusters* (agrupamentos) ao longo

de um período de tempo e detecta movimentações anormais. Isto contrasta com os trabalhos anteriores na área que essencialmente param na etapa do *clustering*.

C2BID analisa o movimento de máquinas entre *clusters*, incluindo o aparecimento de novos *clusters*, através de *clustering* sequencial em sequências de janelas de tempo. Ao estudar o comportamento temporal dos *clusters* é possível identificar comportamentos anómalos e formação de *clusters* suspeitos. Isto resulta numa maior precisão do que a marcação de apenas *clusters* unitários e numa análise mais rápida do que a análise manual. C2BID utiliza a ideia de características dinâmicas – características definidas durante a execução, com base na actividade observada no porto TCP/UDP – inspirada no DynIDS [6]. Isto permite analisar o tráfego em muitos portos, evitando ao mesmo tempo a maldição da dimensionalidade [14], ou seja, perdendo a capacidade de detectar ataques devido a um número excessivo de características (por exemplo, mais de 1000 características, quando há 2^{16} portos). C2BID melhora os trabalhos anteriores ao correlacionar múltiplas janelas de tempo para detectar ataques a diferentes velocidades e lidar com limitações de janelas fixas...

A nossa avaliação mostra que este sistema foi capaz de detectar não só os ataques etiquetados, mas também encontrou ataques não etiquetados (não reportados) em ambos os conjuntos de dados, destacando as vantagens da sua abordagem não supervisionada. Além disso, este sistema obteve valores mais elevados para o F-score e reduziu a taxa de falsos positivos.

II. A ABORDAGEM DO C2BID

C2BID visa detectar automaticamente as máquinas suspeitas, analisando a forma como estas mudam de *cluster* para *cluster*, num determinado período de tempo. As máquinas que são suspeitas em termos destas mudanças, são assinalados como anómalas. O comportamento das máquinas é caracterizado utilizando características extraídas de dados da rede, por exemplo obtidos em routers com a característica Netflow [11]–[13], pelo que esta é uma abordagem de detecção de intrusão baseada em rede.

A figura 1 representa a abordagem que tem quatro etapas: extracção de características, *clustering*, criação de caminho históricos e detecção de *outliers*, cada uma delas é descrita pormenorizadamente nas secções seguintes.

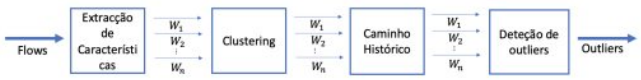

Figura 1. Fases da abordagem C2BID

A. Extracção de características

C2BID utiliza dois períodos de tempo distintos. Primeiro, a deteção é realizada no período de análise $\mathcal{T}_a$, e.g., 1 dia. Segundo, as características são extraídas da rede de acordo com janelas mais pequenas de várias durações, $\mathcal{W} = \{w_1, w_2, ..., w_n\}$. O tamanho da janela w_n tem de ser pelo menos quatro vezes menor que $\mathcal{T}_a$, e.g., alguns minutos. O sistema realiza *clustering* para $w_i \in \mathcal{W}$ durante $\mathcal{T}_a$ e analisa como os *clusters* mudam ao longo do tempo.

Para cada janela de duração $w_i \in W$, são extraídas características de todos os fluxos que aparecem nessa janela. Os fluxos são agregados por máquina, identificados por endereço IP, de modo a que as características sejam associadas a uma máquina e a caracterizem. Para simplificar, consideramos uma associação bijectiva entre máquina e endereços IP. Isto é uma simplificação porque as máquinas podem ter alguns endereços IP diferentes e os endereços IP podem mudar devido à utilização de DHCP. Contudo, conjecturamos que o mau comportamento pode ser observado através da inspecção das comunicações num dos endereços IP da máquina e assumimos que os endereços IP mudam suficientemente lentamente em comparação com o nosso tempo de análise, pelo que os resultados não são afectados.

Consideramos dois conjuntos de características: características fixas e características dinâmicas. As características fixas são sempre as mesmas. Consideramos 16 características fixas, que são normalmente utilizadas na literatura [5]–[7]. A primeira metade das 16 características fixas, com IP de origem como chave de agregação, são: número de diferentes IP contactados pela máquina, número de fluxos em que a máquina é a origem, número de diferentes portos de origem utilizadas pela máquina, número de diferentes portos de destino contactadas pela máquina, soma do comprimento total dos pacotes recebidos pelo anfitrião, soma do comprimento total dos pacotes enviados pela máquina, tamanho médio dos pacotes enviados e a relação entre o número de pacotes enviados e a sua duração. Estas características fixas descrevem a actividade geral da rede de uma máquina. Os outros 8 são semelhantes, mas para os IPs de destino.

Consideramos também um conjunto de características dinâmicas ou características baseadas em portos, seguindo uma ideia recentemente proposta por Dias *et al.* [6]. Alguns ataques são dirigidos a portos TCP/UDP específicos, por exemplo, SSH para o porto 22, pelo que é importante ter características para esses portos. Contudo, há 2^{16} portos vezes 2 (TCP e UDP), sendo que de 0-1023 são Portos de Sistema e de 1024-49151 são Portos de Utilizador [15], pelo que o número de características seria excessivo. Portanto, para cada período $\mathcal{T}_a$ a abordagem escolhe $\mathcal{N}_p$ portos de acordo com alguns critérios e utiliza 4 características para cada um destes portos: número de pacotes enviados do porto, número de pacotes enviados para o porto, número de pacotes recebidos no porto e número de pacotes recebidos do porto.

O resultado da etapa de extracção das características é um *vector de características* para cada máquina. Na avaliação consideramos apenas as máquinas que são internas à organização, uma vez que o número de máquinas externas tende a ser muito maior e menos interessante (apenas uma pequena fracção do tráfego dessas máquinas está presente nos fluxos).

B. Clustering

Após a extracção das características, os vectores das características são fornecidos como entrada para o algoritmo de

clustering. A ideia é agrupar máquinas com comportamento semelhante com base nas características fixas e nas características baseadas em $\mathcal{N}_p \times 4$ portos.

Como mencionado acima, o algoritmo de *clustering* seleccionado foi o K-means. Este algoritmo tem um hiperparâmetro: o número de *clusters*, k. Para definir o valor de k por cada janela de tempo $w_i \in \mathcal{W}$, utilizamos o método de cotovelo [16]. A ideia é testar vários números de k para atingir o número óptimo de *clusters*. O objectivo é atingir o valor de k onde a distância média desde a observação até ao centro do *cluster* tem uma descida mais acentuada. Este valor de k dá uma distribuição equilibrada das entidades sem cair em *overfitting*. A métrica euclidiana é utilizada para determinar a distância entre dois pontos.

O resultado do *clustering* é uma relação entre máquina-*cluster* para todas as janelas de tempo $w_i \in \mathcal{W}$. Cada máquina aparece num *cluster* em cada janela de tempo.

C. Caminho histórico

Dado um período de análise de duração $\mathcal{T}_a$ e janelas de duração $w_i \in \mathcal{W}$ dentro do período de análise, definimos o H de uma máquina h como a sequência de *clusters* em que h aparece. Da secção anterior é claro que H tem um *clusters* por período de duração w_i, mas é necessária uma clarificação: se h não tem fluxos num período, é atribuído a um *cluster* especial que contém hospedeiros inactivos. Para cada h e para cada período $\mathcal{T}_a$, há um caminho histórico H_i correspondendo a uma janela $w_i \in W$. Um caminho histórico representa o comportamento de uma máquina num período $\mathcal{T}_a$ olhando para janelas de tamanho w_i; diferentes ataques são mais fáceis de detectar em janelas de durações diferentes, como mostrado nas experiências.

Para construir o caminho histórico é necessário fazer a classificação dos *clusters*, ou seja, identificar quais os *clusters* que sobrevivem, desaparecem ou emergem entre duas janelas temporais. Isto não é trivial porque os *clusters* mudam, pelo que a abordagem tem de avaliar quais os *clusters* que são semelhantes em intervalos de tempo consecutivos.

A classificação dos *clusters* é feita de acordo com a proposta de Landauer *et al.* [17]. Dois *clusters* são considerados semelhantes se eles tiverem um certo limiar de sobreposição o_t (por exemplo, 50%), ou seja, o número elementos contidos em C' que teriam sido atribuídos aos *clusters* C se tivessem sido utilizados para a geração do mapa de cluster C. A sobreposição é utilizada para expressar matematicamente as relações de *clusters* (Equação (1)). Dividir a união destes dois conjuntos intersectados pela união de todos os conjuntos significa que o valor resultante está no intervalo $[0, 1]$, com 1 indicando uma correspondência perfeita e 0 indicando não haver qualquer correspondência. Dois conjuntos são considerados semelhantes se a sobreposição estiver entre $]0.5, 1]$. Na Equação (1), R_{curr} são as máquinas em C, R'_{prev} são as máquinas em C que também estariam em C', R'_{curr} são as máquinas de C', e R_{next} são as máquinas de C' que também estariam em C. Cada *clusters* recebe um ID único que é utilizado para o representar em todas as janelas de tempo em que aparece.

$$Overlap(C, C') = \frac{|\left(R_{curr} \bigcap R'_{prev}\right) \bigcup \left(R_{next} \bigcap R'_{curr}\right)|}{|R'_{curr} \bigcup R'_{prev} \bigcup R_{next} \bigcup R_{curr}|} \quad (1)$$

A secção seguinte explica a detecção do *outlier*. O resultado das etapas de criação do caminho histórico é um conjunto de caminhos históricos.

D. Deteção de outliers

A detecção de *outliers* é realizada com base no tipo de alterações de *cluster* que uma máquina fez entre janelas de tempo no período analisado. As alterações relevantes podem ser: de um *cluster* para outro, para o mesmo *cluster* ou de um *cluster* para a inactividade.

A detecção de *outliers* é feita em três fases (Figura 2):

- Calculando a probabilidade de cada máquina fazer um determinado conjunto de alterações, tendo em conta todas as alterações feitas no $\mathcal{T}_a$ analisado. Este cálculo é feito em paralelo para todas as diferentes janelas de tempo $\mathcal{W} = \{w_1, w_2, ..., w_n\}$;
- Aplicação do RRCF para detectar valores anormais de probabilidades;
- Filtrar os resultados do RRCF

Cada período de duração $\mathcal{T}_a$ é analisado individualmente. Estes períodos podem ser subdivididos em períodos de tempo $\mathcal{T}_s$, de uma forma em que o máximo W_n tenha pelo menos uma transacção no período $\mathcal{T}_s$, ou seja, $2T_s \times W_n < \mathcal{T}_s < \mathcal{T}_a$. Estas divisões são analisadas em paralelo. Entidades identificadas em pelo menos um período como *outliers* são consideradas como uma potencial ameaça. Estas divisões visam reduzir a possibilidade de uma máquina ser completamente excluído da análise porque, como detalhado mais tarde, uma máquina inactiva por mais de 60% do período $\mathcal{T}_s$ não é considerada.

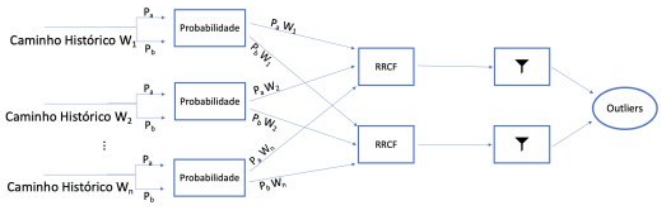

Figura 2. Modelo para deteção de *outliers*

1) Probabilidade: A probabilidade de cada entidade é calculada em paralelo pela divisão $\mathcal{T}_s$ e pela janela temporal ($\mathcal{W} = w_1, w_2, ..., w_n$). O produto final consiste num valor numérico que expressa a probabilidade de um IP alternar entre *clusters* no período estudado, tendo em conta todas as transições nesse período $\mathcal{T}_s$. As máquinas que não têm uma expressão considerável durante um período $\mathcal{T}_s$ têm de ser removidas. Consideramos um limiar de 60%, ou seja, todas as entidades que estejam inactivas durante pelo menos 60% do tempo são removidas. A partir das transições de todas as entidades é criada uma matriz de transição, esta matriz

expressa a frequência absoluta de cada transição entre dois *clusters*. Todas as transições de um *cluster* para a inactividade são excluídas, uma vez que estas transições não fornecem informações úteis relativamente a intrusões na rede.

Após a remoção, a soma de todas as linhas e colunas é calculada e a matriz é normalizada com o valor obtido, a matriz tornou-se uma matriz de frequências relativas.

Para obter o valor numérico para o conjunto de transições efectuadas por uma máquina, soma-se a probabilidade (para efeitos de cálculo, a transformação $f(x) = -\log(x)$ é aplicada) correspondente a uma transição da máquina. Qualquer valor em falta, que não exista na matriz de transição, é obtido através do percentil equivalente.Finalmente, os valores de cada probabilidade das diferentes janelas de tempo são combinados por hospedeiro numa forma vectorial, por exemplo $P_{entidade} = (P_{w_1}^{entidade}, ..., P_{w_n}^{entidade})$.

2) Robust Random Cut Forest: O RRCF [18] é um algoritmo não supervisionado que detecta *outliers*. Este algoritmo produz uma métrica para cada entidade, sendo capaz de lidar com dados em *streaming*, dimensões irrelevantes e valores duplicados que podem mascarar os *outliers*.

Os *outliers* são as entidades com o valor métrico mais elevado. Para a detecção de *outliers*, uma regra de decisão é definida com base no percentil mais baixo a partir do qual um hospedeiro é considerado um *outlier*.

3) Filtro: O RRCF não faz distinção entre valores de alta e baixa probabilidade. Tendo em conta que o objectivo é encontrar máquinas que tomem um caminho invulgar (alta probabilidade), é necessário remover aqueles que o RRFC identificou e não têm um caminho invulgar (baixa probabilidade). Este processo utiliza dois mecanismos: um filtro para valores mais baixos e uma lista branca. A lista branca é definida com base na arquitectura da rede. Pode incluir servidores DNS ou web, que têm padrões de tráfego diferentes de outras máquinas.

Por fim, o `C2BID` identifica os *outliers* como produto final.

III. Avaliação

O objectivo desta avaliação foi fazer a comparação com outras abordagens.

A. Métricas

Consideramos um *outlier* como sendo uma entidade, identificada por um endereço IP, assinalado pelo `C2BID`. Nas expressões seguintes, consideramos os Verdadeiros Positivos (TP) como hospedeiros correctamente classificados como *outliers*, os Verdadeiros Negativos (TN) como hospedeiros correctamente classificados como não *outliers*, os Falsos Positivos (FP) como hospedeiros erradamente classificados como *outliers* e os Falsos Negativos (FN) como hospedeiros erradamente classificados como não *outliers*. As métricas utilizadas na avaliação são Precisão (PREC, fracção *outliers* reais), Recall (REC, fracção *outliers* corretamente classificados) e o F-Score (medida de detecção global).

B. Caracterização do dataset

Utilizámos dois *dataset*s. O primeiro, *CIC-IDS2018* **Data˙Cic**, que designamos como o conjunto de dados artificiais, foi criada para testar e avaliar os SDIs da rede. Os seus autores desenvolveram uma abordagem sistemática para produzir um conjunto de dados de referência diversificado e abrangente. Na sua abordagem, criaram perfis de utilizadores com representações abstractas da actividade vista na rede. Os comportamentos benignos foram gerados utilizando perfis B. Tal perfil foi concebido para extrair o comportamento abstracto de um grupo de utilizadores humanos, encapsulando os comportamentos das máquinas dos utilizadores através da utilização de várias técnicas de aprendizagem automática e de análise estatística. O comportamento malicioso é gerado utilizando perfis M. Estes perfis visam descrever um cenário de ataques sem ambiguidades, de tal forma que os humanos possam interpretar estes perfis e, consequentemente, identificar os seus ataques. A topologia da rede representa uma empresa média típica, com 6 sub-redes, implantada numa plataforma de computação em nuvem.

Considerámos 6 cenários de ataque: Brute Force, ataques pela web, ataques de infiltração, DDoS e port scan . Em todos os dias, excepto no dia 4, os ataques ocorreram em dois períodos distintos, um ataque de cada vez. Os ataques foram realizados a partir de uma ou mais máquinas, utilizando o Kali Linux, foi criada uma rede específica (dentro da gama de IPs públicos) apenas para máquinas atacantes.

O segundo conjunto de dados, *dataset* militar ou *conjunto de dados reais*, foi obtido a partir do Sistema de Informação de Segurança e Gestão de Eventos em produção nessa rede, que recolhe eventos de Netflow a partir de routers internos [19]. A recolha destes fluxos pode dar-nos uma visão do mau comportamento das máquinas internas, não detectado pelos sistemas de segurança implantados. O conjunto de dados corresponde a um mês completo, com aproximadamente 5.500 computadores e 160 GB de dados. Os ataques foram ataques Brute Force (contra SSH e RDP) precedidos por um port scan a um ritmo lento (intervalo de 5 segundos). As principais razões para a escolha destes ataques foram: (1) ter ataques que passam despercebidos aos meios tradicionais de protecção; (2) para captar actividades de reconhecimento interno (por exemplo, port scan) e ataques lentos de dicionários utilizados por atacantes com informação privilegiada.

C. Resultados com o conjunto de dados artificiais

A contagem dos positivos foi feita considerando todos os IPs assinalados pelo `C2BID` em cada $\mathcal{T}_a$. Este método de contagem é susceptível aos FPs, uma vez que cada ataque conta apenas como um TP. Apenas os IPs internos foram analisados uma vez que o sistema necessita de uma fonte contínua de informação que é difícil de obter com os IPs externos. No caso particular deste conjunto de dados, todos os atacantes têm um IP externo e todas as vítimas têm um IP interno. Para cada dia, há uma vítima. A lista branca são todos os IPs internos contactada por mais de 90 % dos IP internos, tendo em conta todos os dias.

C2BID assinalou alguns IPs que não foram identificados como maliciosos pelos autores[1]. Inicialmente consideramo-los FPs, mas uma nova inspecção verificou-se que eram TPs.

Conseguimos identificar todas as vítimas durante todos os dias. Obtivemos, no total dos 9 dias, um PREC de $0,789$, REC de 1, e F-score de $0,882$. Se (erradamente) considerássemos como TP apenas os IPs marcados pelos autores e os outros como FPs, obteríamos um PREC de $0,473$, REC de 1, e F-score de $0,643$.

D. Resultados com o conjunto de dados reais

Os ataques identificados foram: (1) port scan lento com ritmo de 5s (atacante e vítima); (2) brute force RDP (atacante e vítima); e (3) brute force SSH (atacante). Também identificámos algumas máquinas não envolvidas em ataques emulados, mas marcados como *outliers* devido a uma má configuração confirmada pelo Centro de Operações de Segurança.

Em resumo, os alertas levantados pelo C2BID correspondiam a ameaças reais ou anomalias. Fomos capazes de identificar atacantes e vítimas num universo de 5000 hospedeiros. No total, este sistema provou ser útil num cenário prático sem esforço significativo de implantação, uma vez que apenas precisa de ser alimentado com eventos NetFlow.

E. Comparação com diferentes abordagens

Esta secção compara C2BID com três obras anteriores na área: OutGene [5], DynIDS [6], e FlowHacker [7]. Seleccionámos estes três porque são recentes. Não comparámos com mais soluções porque seriam mais antigas, não há implementações disponíveis, e/ou não seguiam as mesmas suposições relativamente à não necessidade de dados de treino.

Todas as três abordagens têm duas fases: extracção de características e *clustering*. Diferem uma da outra em termos de características e algoritmos de *clustering* utilizados. Um IP é considerado um *outlier* se estiver num *cluster* com apenas um elemento. Consideram-se apenas como *outliers* os IPs que estão isolados numa janela de tempo, específica, não analisando as mudanças de *cluster* ao longo do tempo.

Para a contagem de positivos utilizámos o mesmo método que antes: número de IPs diferentes marcados como *outliers* em cada $\mathcal{T}_a$ (1 dia). Este método de contagem difere dos utilizados pelos autores dos três trabalhos, o que leva a resultados diferentes dos fornecidos nos trabalhos originais. Esta alteração deve-se ao facto do C2BID não ser adequado para detecção em pequenas janelas de tempo, por exemplo, de minutos, como fazem estas obras; C2BID monitoriza as alterações de *clusters* ao longo de várias destas janelas de tempo.

Em todos os casos, foi possível obter melhores valores de PREC e F-score com C2BID, com ambos os conjuntos de dados. Para o conjunto de dados reais, o REC foi o mesmo em quase todos os algoritmos. A principal diferença entre estas abordagens e o C2BID é que o C2BID produziu muito mais FPs.

[1]dia 1: 172.31.65.123 e 172.31.67.109; dia 2: 172.31.66.112; dia 4: 172.31.65.56; dia 5: 172.31.69.19; dia 9: 172.31.66.100

IV. Trabalho relacionado

Existem vários artigos e livros sobre detecção de intrusão baseados em aprendizagem automática. Um estudo anterior, realizado por Leung e Leckie [20], utilizou *clusters* para detecção de intrusões na rede. Os autores apresentam o seu próprio algoritmo de *clustering*, (fpMAFIA), e testam-no com um conjunto de dados (KDD 99).

Alguns trabalhos recentes não utilizam o *clustering* e baseiam-se no conhecimento do que é um bom comportamento. Cinque *et al.* usou entropia para inferir desvios de uma linha de base [21]. DeepLog foi inspirado no processamento de linguagem natural e interpreta os registos como elementos de uma sequência que segue as regras gramaticais [22]. Kitsune utiliza um conjunto de redes neuronais chamadas *autoencoders* para distinguir entre padrões de tráfego normal e anormal colectivamente [23]. AI^2 era um sistema baseado em log onde a matriz de densidade, decomposição matricial e o replicador ANN (redes neuronais) eram usados para modelar o comportamento de união de diferentes máquinas dentro de um grande conjunto de dados brutos [24]. Este sistema foi capaz de aprender e defender-se contra ataques invisíveis. Marchetti *et al.* desenvolveu um sistema de detecção automática e precoce APA [25]. Extraiu e analisou registos de fluxo para construir uma análise comparativa do comportamento passado de cada hospedeiro através do cálculo das pontuações. A estrutura poderia identificar exfiltrações entre outros.

A monitorização de *clusters* permite-nos correlacionar alterações em *clusters* de duas janelas de tempo consecutivas. Ao longo do tempo os *clusters* podem sofrer algumas alterações. Estas alterações são detectáveis através da associação de *clusters* em múltiplas janelas de tempo. MClusT é uma estrutura que utiliza gráficos bipartidos e probabilidades condicionais para monitorizar transições definidas numa determinada taxonomia [26]. Landauer *et al.* baseou a sua abordagem em técnicas de avaliação de *clusters* onde os mesmos elementos são observados e agrupados ao longo do tempo [17]. Utilizam a evolução dos *clusters* para processar dados de registo em streaming, em lugar de se limitarem a conjuntos de dados de tamanho fixo. Para detectar anomalias, os autores verificaram se um valor futuro se encontra dentro do intervalo de previsão, utilizando a última entrada registada e assim criaram uma previsão para os próximos valores. Os autores identificam uma grande quantidade de FP nos resultados da avaliação do seu sistema. C2BID aplica um sistema de classificação semelhante ao apresentado em [17] para construir o caminho histórico. Utilizámos o binómio *clusters* e entidade em vez de log e entidade. C2BID diminui os FPs porque utiliza um método diferente de detecção *outliers* e estuda NetFlows em vez de logs. Por outro lado o MClusT, não se baseia na associação de *clusters* e probabilidade condicional, mas na associação directa entre elementos em diferentes janelas de tempo.