UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DANIEL BAYERL VIEIRA
LUCAS GUIMARÃES MIRANDA

MÉTODOS DE IMPUTAÇÃO ESTATÍSTICA PARA DADOS CATEGÓRICOS COM APLICAÇÕES EM SEGURANÇA CIBERNÉTICA

RIO DE JANEIRO
2022

DANIEL BAYERL VIEIRA
LUCAS GUIMARÃES MIRANDA

MÉTODOS DE IMPUTAÇÃO ESTATÍSTICA PARA DADOS CATEGÓRICOS COM APLICAÇÕES EM SEGURANÇA CIBERNÉTICA

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Daniel Sadoc Menasche

RIO DE JANEIRO
2022

Vieira, Daniel Bayerl
V658m Métodos de imputação estatística para dados categóricos com aplicações em segurança cibernética / Daniel Bayerl Vieira, Lucas Guimarães Miranda. – 2022.

50 f.

Orientador: Daniel Sadoc Menasche.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2022.

1. Imputação estatística. 2. Cadeia de Markov. 3. Mineração de regras. 4. Segurança cibernética. I. Miranda, Lucas Guimarães. II. Silva, Daniel Sadoc (Orient.). III. Universidade Federal do Rio de Janeiro, Instituto de Computação. IV. Título.

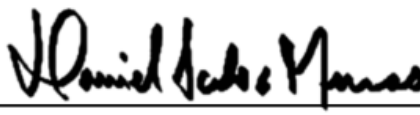
DANIEL BAYERL VIEIRA
LUCAS GUIMARÃES MIRANDA

MÉTODOS DE IMPUTAÇÃO ESTATÍSTICA PARA DADOS CATEGÓRICOS COM APLICAÇÕES EM SEGURANÇA CIBERNÉTICA

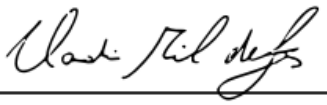
Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 08 de Abril de 2022

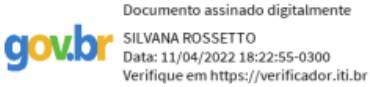
BANCA EXAMINADORA:



Daniel Sadoc Menasché
UFRJ



Claudio Miceli de Farias
UFRJ



Silvana Rossetto
UFRJ

AGRADECIMENTOS

Nós gostaríamos de agradecer primeiramente a Siemens que como financiadora do projeto, permitiu por meio da infraestrutura e interesse no tema, a criação deste trabalho. Também gostaríamos de agradecer ao nosso professor e orientador Daniel Sadoc Menasche que nos auxiliou na entrada do projeto, e durante todo o processo, onde compartilhou conhecimento e experiência necessária para realização deste trabalho. Nossos colegas de equipe, Leonardo Ventura, Matheus Nogueira, Miguel Angelo Bicudo e Lucas Senos que nos auxiliaram principalmente com a criação do banco de dados e coleta de informações do NVD, de onde partimos para coleta de dados nas demais plataformas, além dos *feedbacks* e ideias durante a realização do trabalho.

RESUMO

Neste trabalho buscamos entender e realizar medições em um fluxo de avisos de vulnerabilidades de software que são distribuídos através de plataformas de divulgação e as *tags* referentes a características destes avisos, que estão relacionadas com essas vulnerabilidades, com isso nosso objetivo é conseguir compreender o comportamento das plataformas nesses fluxos e o comportamento das *tags* relacionadas a ela, desta maneira realizando imputações estatísticas utilizando os modelos de cadeias de markov e *Rule Mining* onde criamos uma modelagem para interpretar estas séries temporais e tabelas de *tags* permitindo assim a realização de uma imputação estatística de dados que nos retornou resultados muito satisfatórios com acuracias variando de 60% a 75%.

Palavras-chave: Imputação estatística; Cadeia de Markov; Mineração de regras; Segurança cibernética.

ABSTRACT

In this work we seek to understand and perform measurements on a flow of software vulnerability advisories that are distributed through dissemination platforms, and we call that the flow of advisories, and we also work on the references about the characteristics of thar advisories, that are related to these vulnerabilities, with this, our objective is to be able to understand the behavior of the platforms in these flows and the behavior of the tags related to it, however, as these flows are unattributed in small time series with very irregular gaps, we chose to use alternative methods and models to solve this problem, in this way we perform statistical imputations using the Markov chains and Rule Mining methods where we created a model to interpret these time series and tags, thus allowing the performance of a statistical imputation that gave us very unsatisfactory results.

Keywords: statistical imputation; markov chain; rule mining; cyber security.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

NVD	National Vulnerability Database
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
EPSS	Exploit Prediction Scoring System

SUMÁRIO

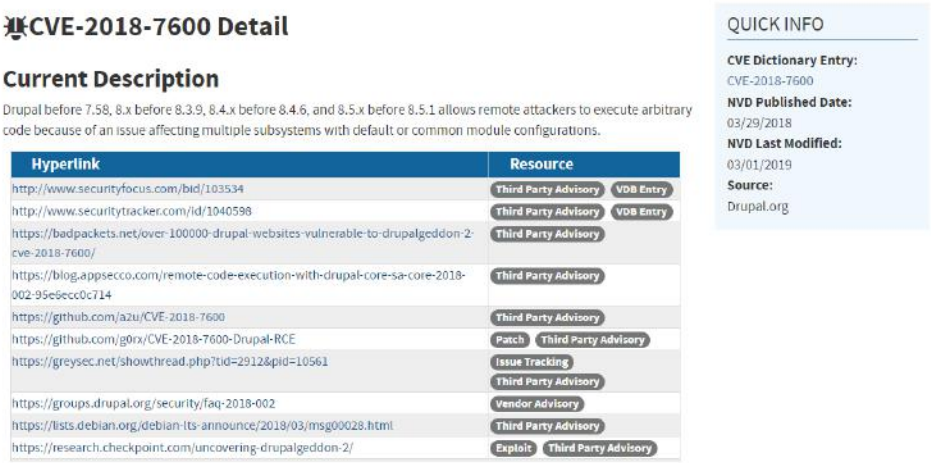
1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Vulnerabilidades de software que ameaçam os sistemas de computadores modernos são divulgadas diariamente. Plataformas como o National Vulnerability Database (NVD) ou Github, e sites de fornecedores são responsáveis por informar os usuários sobre tais vulnerabilidades, por meio de alertas de segurança. Cada vulnerabilidade, identificada por meio de seu *Common Vulnerabilities and Exposures identifier* (CVE id), é normalmente associada a muitos avisos de segurança, que refletem, por exemplo, a disponibilidade de *exploits* e *patches*. (MITRE, 2020).

Informações oportunas sobre a divulgação antecipada de vulnerabilidades são fundamentais. Informações precisas sobre o impacto das vulnerabilidades e sobre quais sistemas estão expostos ajudam a orientar as decisões, por exemplo, relacionadas ao gerenciamento de *patch* ciente de risco (WANG et al., 2017). *Neste TCC, nosso objetivo é tentar identificar padrões nesse fluxo de vulnerabilidades, para buscar prever e inserir dados faltantes.*

As plataformas foram escolhidas baseado nas referências realizadas pelo NVD em cada vulnerabilidade, onde podemos coletar não só os links, como as *tags* associadas aquela divulgação.



CVE-2018-7600 Detail

Current Description

Drupal before 7.58, 8.x before 8.3.9, 8.4.x before 8.4.6, and 8.5.x before 8.5.1 allows remote attackers to execute arbitrary code because of an issue affecting multiple subsystems with default or common module configurations.

Hyperlink	Resource
http://www.securityfocus.com/bid/103534	Third Party Advisory, VDB Entry
http://www.securitytracker.com/id/1040598	Third Party Advisory, VDB Entry
https://badpackets.net/over-100000-drupal-websites-vulnerable-to-drupalgeddon-2-cve-2018-7600/	Third Party Advisory
https://blog.appsecco.com/remote-code-execution-with-drupal-core-sa-core-2018-002-95e9ecc0c714	Third Party Advisory
https://github.com/a2u/CVE-2018-7600	Third Party Advisory
https://github.com/g0rx/CVE-2018-7600-Drupal-RCE	Patch, Third Party Advisory
https://greysec.net/showthread.php?tid=2912&pid=10561	Issue Tracking, Third Party Advisory
https://groups.drupal.org/security/faq-2018-002	Vendor Advisory
https://lists.debian.org/debian-lts-announce/2018/03/msg00028.html	Third Party Advisory
https://research.checkpoint.com/uncovering-drupalgeddon-2/	Exploit, Third Party Advisory

QUICK INFO

CVE Dictionary Entry:
CVE-2018-7600
NVD Published Date:
03/29/2018
NVD Last Modified:
03/01/2019
Source:
Drupal.org

Figura 1 – Tabela de hiperlinks referenciados pelo NVD

Na Figura 1 podemos ver como a estrutura da pagina do NVD se parece, onde temos a vulnerabilidade em questão e os hiperlinks com suas respectivas *tags* relacionadas à vulnerabilidade em questão. Também podemos ver algumas informações sobre a vulnerabilidade, assim como sua data de divulgação e a sua última modificação.

1.2 DESAFIOS

Existem vários desafios relativos à caracterização do fluxo de informações nas plataformas de segurança. Primeiro, cada plataforma de segurança usa seu próprio formato para disseminar dados sobre vulnerabilidades. As informações recuperadas dessas plataformas devem ser curadas, por exemplo, para extrair datas de publicação pelas fontes. Em segundo lugar, não existem modelos para capturar a dinâmica de como as informações fluem no ecossistema. Tais modelos podem ser instrumentais para avaliar o que ocorreria com o cenário de segurança em determinados eventos, como plataformas importantes sendo atacadas, por exemplo, por um ataque DDoS ou duas plataformas se fundindo. Por exemplo, plataformas relevantes, como Security Focus e Security Tracker, não foram atualizados recentemente e o impacto ainda não está claro. Terceiro, alguns avisos de segurança podem ser compartilhados em plataformas privadas ou em fóruns de blackhat, tornando-os inacessíveis ao público em geral. Quarto, as *tags* que são avaliadas nem sempre são completas, logo muitos elementos que deveriam ter algumas *tags*, não estão apresentando, por isso precisamos mudar a maneira de interpretar os dados para extrair a informação da melhor maneira possível.

1.3 MÉTODOS

Para enfrentar os desafios acima, relatamos os resultados de uma campanha de medição em grande escala para coletar informações temporais sobre eventos associados a vulnerabilidades de software. Os dados são selecionados de forma a extrair datas de cada um dos avisos de segurança analisados. A série temporal e as *tags* resultante de avisos de segurança associados a vulnerabilidades são nossos objetos de estudo. Em seguida, propomos um modelo analítico para expressar o fluxo de informações por meio de alertas de segurança em várias plataformas. O modelo é baseado em uma rede de filas, onde cada plataforma corresponde a uma fila que acrescenta um atraso na propagação da informação, e a partir deste modelos pretendemos realizar os processos de imputação estatística.

1.4 CONTRIBUIÇÕES

De maneira geral, mesmo nosso foco sendo imputação de dados para varias séries temporais categóricas curtas e *tags*, ambos sobre vulnerabilidades, nossa contribuição vai além disso. Foi encontrado pouquíssimo conteúdo para o preenchimento de dados faltantes em séries temporais categóricas curtas, com isso a estratégia adotada pode ser replicada para outras áreas que possam ser estruturada de maneira semelhante.

1.5 PUBLICAÇÕES

O trabalho apresentado neste TCC está relacionado com duas publicações nossas:

1. **Jornal – TNSM:** Miranda, L., Vieira, D., Nogueira, M., Ventura, L., Bicudo, M., Martins, M., de Aguiar, L. P., Menasché, D. S., Bicudo, M. A., Nogueira, M. S., Lovat, E. (2021). On the flow of software security advisories. IEEE Transactions on Network and Service Management, 18(2), 1305-1320.

2. **Conferência – The 16th International Conference on Risks and Security of Internet and Systems:** Miranda, L., Vieira, D., Nogueira, M., Ventura, L., Bicudo, M., Martins, M., de Aguiar, L. P., Menasche, D. (2020, November). Measuring and modeling software vulnerability security advisory platforms. In International Conference on Risks and Security of Internet and Systems (pp. 31-48). Springer, Cham.

Parte do trabalho também foi apresentado na JIC/SIAC e conta com um vídeo na Internet: https://www.youtube.com/watch?v=7QvcM3MEoSg

O material apresentado da Seção 5.2.2 em diante deste trabalho de conclusão de curso não constou em nenhuma de nossas publicações anteriores. Ou seja, todo o material relacionado a mineração de regras é original, incluindo o Capítulo 6.

1.6 MODELO

Nós propomos parametrizar uma rede de filas composta por filas para expressar o fluxo de informações por meio de alertas de segurança em várias plataformas. O modelo é inspirado nos diagramas de Sankey, que são derivados dos dados coletados, as *tags* são componentes vinculados às vulnerabilidades porém independentes do estudo de fluxo.

1.7 TERMINOLOGIAS IMPORTANTES

A seguir vamos introduzir conceitos básicos terminológicos que vão ser utilizados no trabalho.

CVE id é o identificador comum de vulnerabilidade e exposição, que é um id exclusivo usado para se referir a vulnerabilidades. **Data de divulgação do NVD** é a data em que o CVE foi divulgado no National Vulnerability Database (NVD). Tanto o CVE quanto o NVD são mantidos pelo *U.S. Department of Homeland Security* (DHS). Todas as vulnerabilidades que recebem um CVE são eventualmente publicadas pelo NVD (RUOHONEN, 2019).

Aviso de segurança é qualquer informação publicada sobre um determinado CVE, incluindo patches, exploits e notas. *Englobamos as divulgações do NVD como avisos de segurança, cujas datas de publicação são iguais às datas de publicação do NVD CVE.*

Aviso de segurança da plataforma é uma plataforma que emite avisos de segurança. Exemplos incluem NVD e Security Focus.

hiperlink de aviso de segurança da plataforma é um hiperlink para um aviso de segurança publicado em uma determinada plataforma.

Data de aviso de segurança é a data em que o material contido no aviso de segurança foi publicado por seu autor.

Data de divulgação do CVE da plataforma é a data de aviso de segurança mais antiga para um determinado CVE entre os avisos coletados de uma determinada plataforma.

2 ACHADOS EMPÍRICOS SOBRE DIVULGAÇÃO DE AVISOS POR PLATAFORMAS DE SEGURANÇA

Neste capítulo apresentamos achados empíricos sobre divulgação de avisos por plataformas de segurança. Na Seção 2.1 apresentamos estatísticas básicas sobre número de vulnerabilidades (CVEs) sobre as quais cada plataforma gerou avisos, e quantos deles que foram realizados como primeira divulgação dentre todas as plataforma que divulgaram aquela vulnerabilidade. Em seguida na seção 2.2 mostramos a relação entre as demais plataformas e o NVD que é a plataforma com maior quantidade de divulgações, e as relações entre elas, como o tempo entre a divulgação entre uma plataforma e a próxima que divulgou informações sobre uma mesma vulnerabilidade. E por fim mostramos na seção 2.3 o diagrama utilizado como base para criação das séries temporais que foram utilizadas em todo trabalho para realização das imputações estatísticas em séries temporais.

2.1 ESTATÍSTICAS BÁSICAS SOBRE VULNERABILIDADES

As plataformas divulgam informações diariamente, e com isso podemos extrair algumas informações relevantes no intuito de obter informações sobre seus comportamentos.

Podemos ver na Figura 2 a quantidade de informação que foi capturada de cada plataforma.

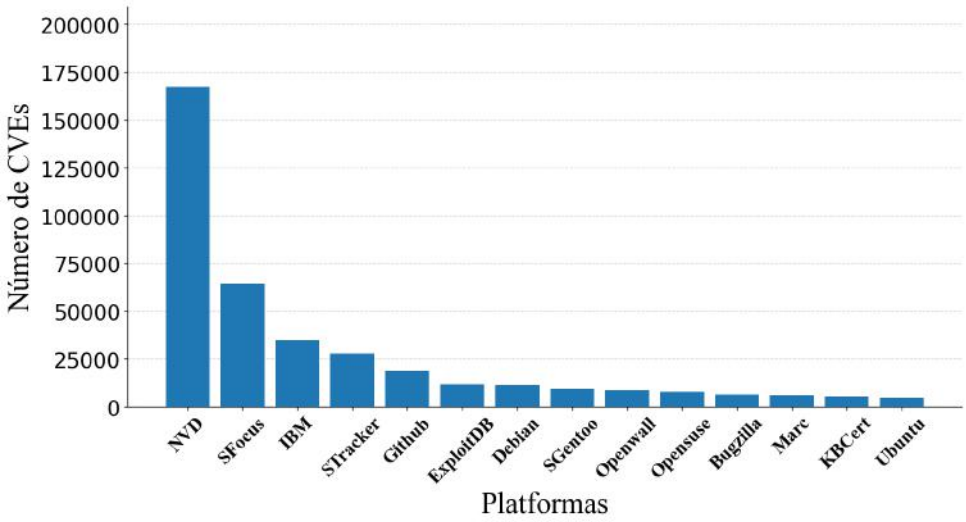


Figura 2 – Quantidade de divulgações por plataforma

Podemos ver que a concentração de divulgações esta muito situada em uma pequena quantidade de plataformas, que por sua vez, pela maior quantia de informação, são bem mais fáceis de se extrair algum comportamento delas nas filas.

Outra informação relevante que podemos considerar é quantas dessas divulgações foram realizadas como primeira divulgação por cada plataforma, isso pode ser visto na Figura 3.

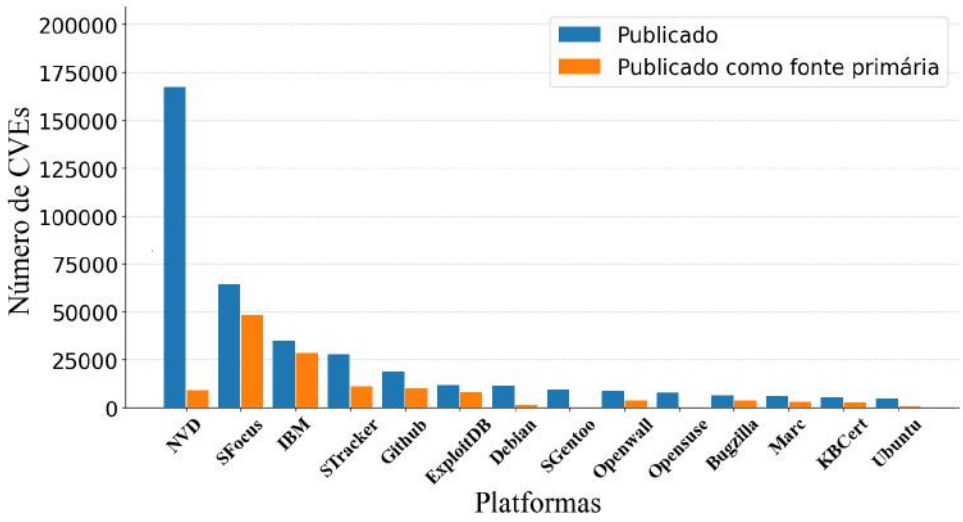


Figura 3 – Porção de primeiras divulgações por plataforma

Podemos ver que algumas plataformas claramente tem um padrão de divulgar uma informação como primeira fonte, como por exemplo o Security Focus e a IBM, e outras tendem mais a replicar informação, como por exemplo o NVD.

2.2 RELAÇÕES ENTRE AS PLATAFORMAS

Outra informação que é muito relevante é como essas plataformas se relacionam entre si, em especial com o NVD que é o que possui a maior quantidade de informação dentre todas, sendo o centro de divulgações com maior quantidade de vulnerabilidades divulgadas.

Conseguimos observar na Figura 4 dada a quantidade de divulgações que cada plataforma possui, quantas delas foram divulgadas antes do NVD.

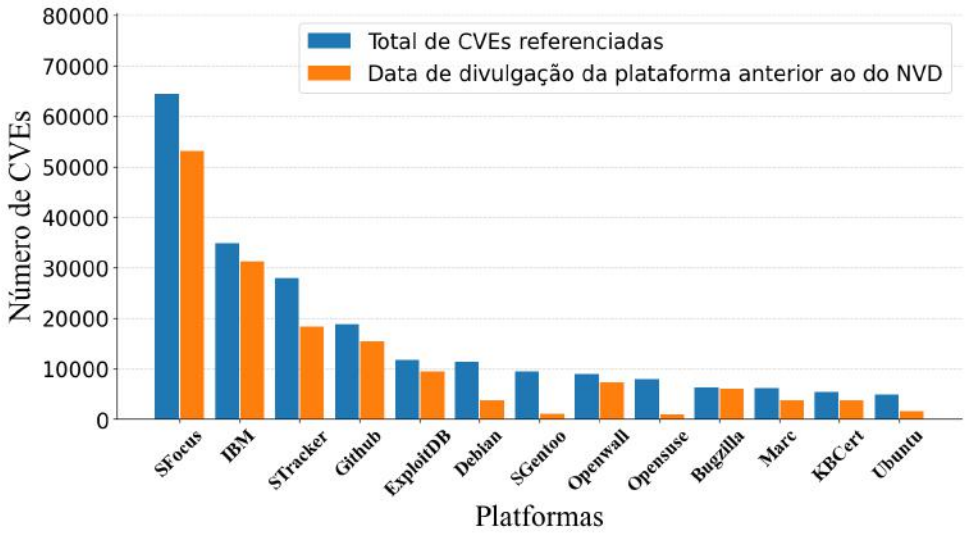


Figura 4 – Quantidade de avisos com divulgação anterior ao NVD por plataforma

Podemos ver que o NVD geralmente divulga informações de forma mais tardia, não sendo o primeiro a divulgar as informações.

Na Figura 5 podemos partir para uma visão quantitativa buscando encontrar algum padrão no tempo médio de diferença em dias, entre a divulgação no NVD e em uma das demais 13 plataformas para uma mesma vulnerabilidade.

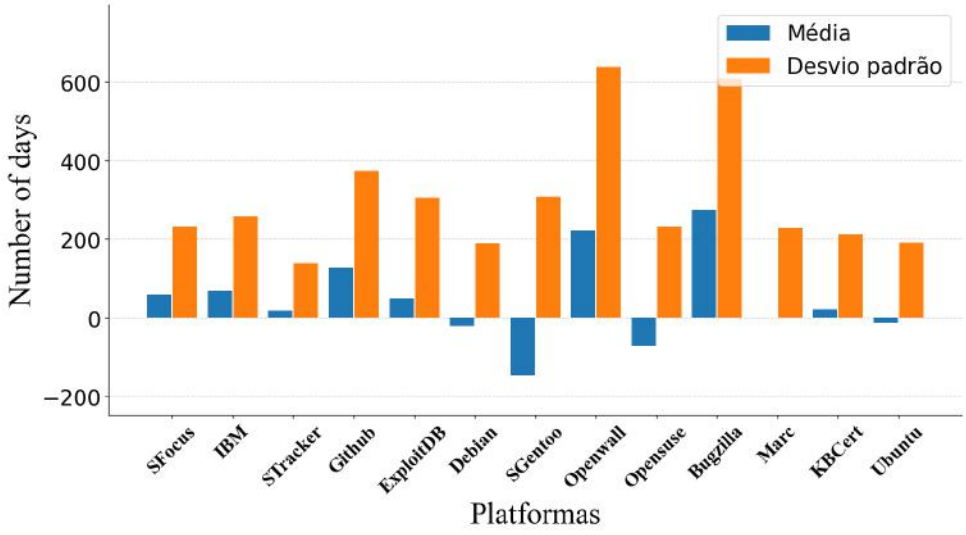


Figura 5 – Diferença entre as datas de divulgação das plataformas e o NVD

Podemos observar que pelo alto desvio padrão fica muito difícil de definir um período específico nessas diferenças, o que nos encaminha para uma abordagem menos quantitativa

e sim qualitativa, buscando inferir a posição da plataforma nas divulgações, e não sua data exata.

Uma métrica importante a ser analisada é quanto tempo a informação fica em uma plataforma antes de ser passada a diante, e para isso podemos observar a Figura 6.

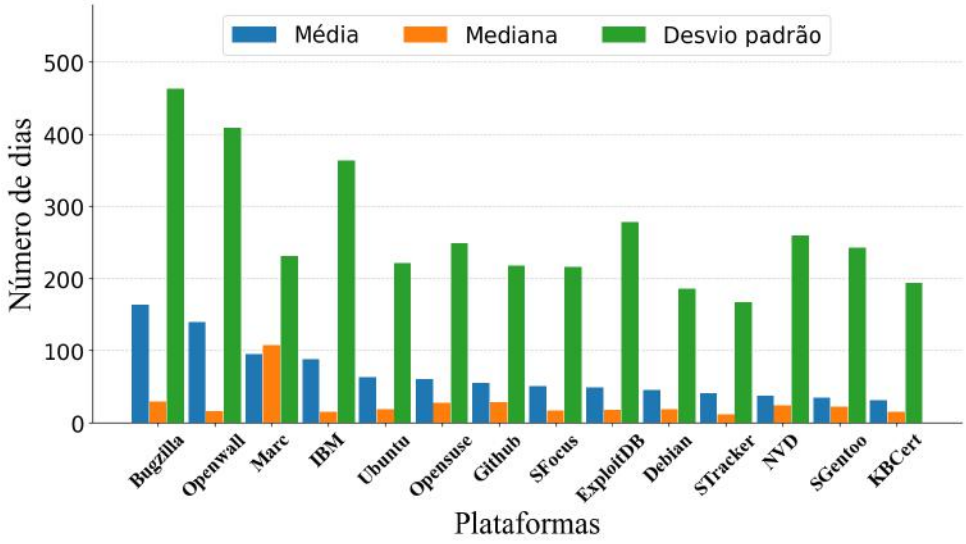


Figura 6 – Tempo que um aviso permanece em cada plataforma

Podemos ver que por possuir um desvio padrão bem alto, voltamos a averiguar que trabalhar com quantidade de dias não se mostra uma abordagem mais adequada, e definindo por fim o porque de optarmos por uma abordagem de camadas, ou seja classificando as plataformas por ordem de divulgação, não levando em consideração a quantidade de dias entre um aviso e outro.

2.3 DIAGRAMA DE FLUXO DAS VULNERABILIDADES

O fluxos de avisos através das platoformas pode ser demonstrado por um *sankey Diagram* (BLINDER et al., 2019), esta abordagem de camadas foi escolhida pela distribuição irregular das datas das plataformas,o *sankey Diagram* é um diagrama utilizado para analisar os fluxos entre valores categóricos, e com isso podemos identificar padrões na forma de divulgar das plataformas.

A Figura 7 foi construída definindo camadas, ou seja cada faixa representa uma camada de divulgação, essa camada determina qual foi a posição que a plataforma divulgou informações, e tem seu fluxo particionado, mostrando para onde essa informação foi depois, neste exemplo abordamos apenas as três primeiras camadas, pois elas apresentam boa parte dos dados, já que as vulnerabilidades não tendem a possuir muitas plataformas em comum divulgando informações sobre elas.

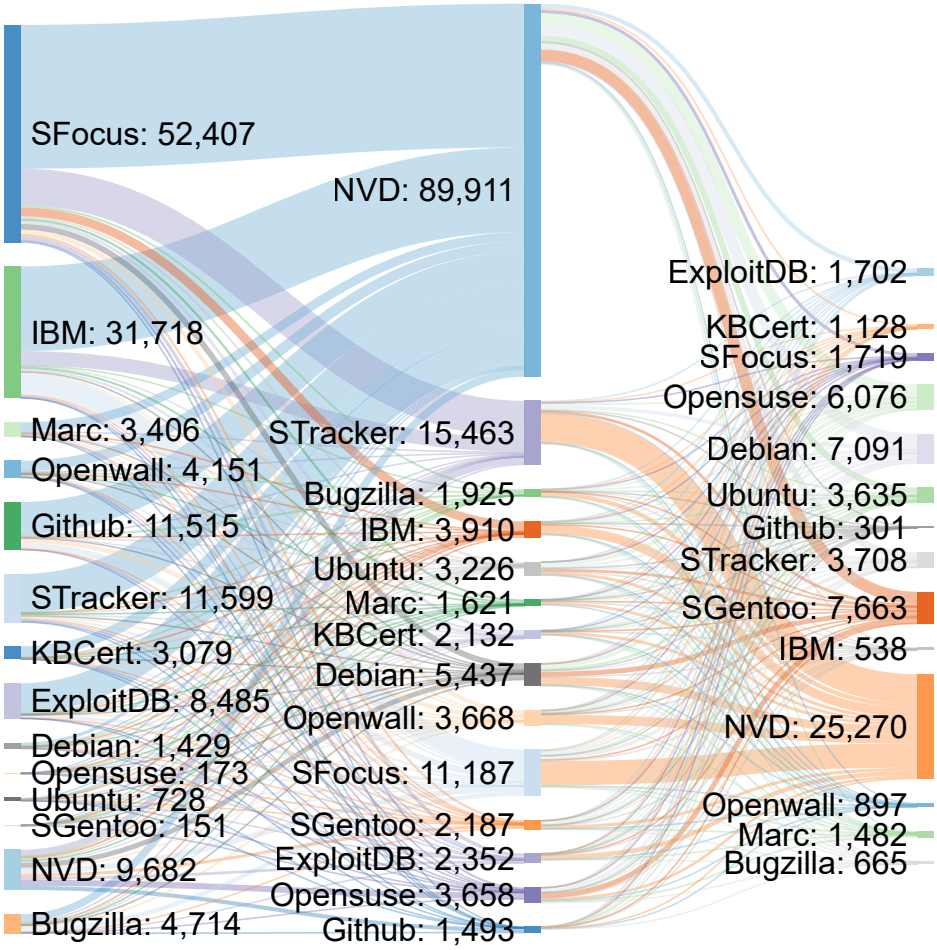


Figura 7 – Gráfico Sankey Diagram de fluxos

Podemos ver que as plataformas que mais ocupam a primeira camada podem ser classificadas como fontes, ou seja, elas tendem a divulgar a informação primeiro. As plataformas que se concentram nas camadas centrais, podem ser classificadas como retransmissoras, ou seja elas apenas passam informação, geralmente a partir das plataformas que divulgaram informações primariamente. E por último temos os sumidouros, que são as plataformas que divulgam informação sobre uma vulnerabilidade por último.

3 IMPUTAÇÕES

Neste capítulo vamos dar uma introdução sobre as imputações estatísticas explicando de maneira geral os métodos utilizados e a abordagem realizada, começando pela seção 3.1 que apresenta de maneira geral as imputações realizadas. E depois apresentamos nas seções 3.2 e 3.3, as imputações estatísticas em séries temporais e em tags respectivamente.

3.1 INTRODUÇÃO ÀS IMPUTAÇÕES ESTATÍSTICAS

O principal objetivo desse trabalho é buscar métodos de imputação estatística de dados para modelos convencionais e não convencionais, utilizando técnicas de maneiras pouco convencionais para o método de imputação estatística em séries temporais categóricas, que sejam organizadas temporalmente como filas, esse tipo de estrutura não é muito comum, logo o foco é desenvolver estratégias próprias para eles. Também é visado métodos para imputação estatísticas de *tags* faltantes, pois esses dados estão no mesmo escopo de trabalho sobre vulnerabilidades, e encontramos diversos problemas de falta de *tags*, onde elas se viam explicitamente necessárias, com isso tivemos que mudar o escopo para avaliar as *tags* como informação e sua ausência não.

3.2 IMPUTAÇÃO ESTATÍSTICA EM SÉRIES TEMPORAIS

Como mencionado acima nosso exemplo de séries temporais, se mostra um pouco específico, por isso não foram encontrados métodos já criados que fossem efetivos para os mesmos. Sua estrutura é uma rede de filas, onde cada fila representa uma vulnerabilidade, tendo uma ordem cronológica de divulgação dessa vulnerabilidade em cada plataforma que a divulgou, as filas tem de 2 a 12 elementos de tamanho, onde esses elementos são respectivos a 14 plataformas diferentes. Cada plataforma tem uma quantidade de dados diferentes, e um padrão de divulgação, nosso intuito é construir um modelo que consiga observar esse padrão e utilizar o mesmo, para preencher os dados faltantes. Com isso vamos apresentar dois métodos diferentes para realização dessa imputação de dados, um deles utilizando cadeia de Markov, que consiste em um método com aplicações matemáticas probabilísticas. O outro método consiste em aplicação de um método de aprendizado de máquina não utilizado para séries temporais, porém que foi adaptado.

3.3 IMPUTAÇÃO ESTATÍSTICA DE TAGS

Cada aviso de segurança em geral está associado a *tags*. Tais *tags* podem indicar, por exemplo, se o aviso se refere a um *patch*, *exploit* ou simplesmente um eventual risco

ainda não confirmado. A inclusão de *tags* nem sempre ocorre de forma uniforme, principalmente em plataformas do tipo *Github*. Assim, visando ilustrar mais uma aplicação de mineração de regras para fins de imputação de dados de segurança, mostramos resultados preliminares sobre o seu uso para fins de imputação automática de *tags* em eventos do *Github*.

4 FUNDAMENTOS

Neste capítulo vamos apresentar os fundamentos utilizados para a realização do trabalho, começando pela seção 4.1 que vai apresentar os conceitos de uma cadeia de Markov, e porque ela vai ser útil no nosso trabalho. Na seção 4.2 vamos apresentar o *Rule mining* sua proposta e utilização, assim como sua aplicação no nosso trabalho. Por último vamos apresentar na seção 4.3 o conceito que envolve a imputação estatística e qual é a sua utilidade dentro e fora do trabalho.

4.1 CADEIAS DE MARKOV

A cadeia de Markov é um processo estocástico onde o próximo estado de uma sequência depende apenas do estado anterior (NORRIS, 1998), que pode ser modelado como apenas a transição entre dois valores, ou os casos onde modelamos o estado anterior e o próximo como um conjunto de valores que ocorreu em sequência. Para o nosso trabalho consideramos a versão mais simples da cadeia de Markov, pois por não ter um volume muito grande de dados, precisamos de uma abordagem um pouco menos específica. Esta cadeia pode ser representada por uma matriz de transições, onde temos ocupando um dos eixos os estados fonte, e o outro os estados destino, e nas células temos a probabilidade de partir de uma fonte para um determinado destino. Estes valores são normalizados por linha, logo uma fonte vai ter uma distribuição de probabilidade entre seus destinos que totalizam 100%. Esta cadeia pode ser preenchida utilizando tanto fórmulas dependendo do modelo a ser utilizado, ou utilizando simulações, que é o caso que foi utilizado no nosso trabalho.

4.2 MINERAÇÃO DE REGRAS: ALGORITMO *APRIORI*

O *Apriori* é um algoritmo de mineração de regras que busca encontrar correlação entre diferentes *tags* de uma tabela, e para isso ele utilizada de um grupo de métricas das quais nos interessam para o foco de estudo desse trabalho apenas duas, o suporte e a confiança. Na coluna de antecedentes, temos os padrões que precisam ser satisfeitos pelo evento para ele receber a *tag* que aparece na coluna de consequentes. O suporte (indicado na fórmula 4.1) corresponde à evidência de quão frequentemente o antecedente e o consequente aparecem, concomitantemente, no conjunto de dados, e confiança (indicada na fórmula 4.2) se refere a frequência de vezes em que a sentença se-então (*if-then*) é de fato satisfeita dado que o antecedente ocorre. [1]

[1] https://en.wikipedia.org/wiki/Association_rule_learning

$$\text{suporte} = P(A \cap B) = \frac{\#(A \cap B)}{\#\text{ eventos}} \tag{4.1}$$

$$\text{confiança} = P(B|A) = \frac{\#(A \cap B)}{\#\text{ eventos em que } A \text{ ocorre}} \tag{4.2}$$

4.3 IMPUTAÇÃO ESTATÍSTICA

A imputação estatística é um método utilizado para preencher dados faltantes, ou para realizar previsões de valores dentro de um modelo. O nosso modelo consiste em plataformas que divulgam vulnerabilidades, em formatos de filas, logo nosso intuito é conseguir prever ou preencher dados faltantes dentro dessas filas, para isso é necessário estruturar o seu problema de uma maneira que facilite a realização da imputação e validação dos procedimentos realizados, para que esse processo possa ser realizada de uma maneira concreta e correta.

5 IMPUTAÇÃO ESTATÍSTICA EM SÉRIES TEMPORAIS

Neste capítulo vamos abordar as imputações estatísticas em séries temporais, começando pela seção 5.1 onde apresentamos os desafios na modelagem dos dados para realização das imputações, mostrando também como eles foram estruturados. Na seção 5.2 mostramos os dois métodos utilizados para a realização das imputações estatísticas, detalhando cada um deles e mostrando seus resultados. E por último na seção 5.3 realizamos uma comparação entre eles, com o intuito de mostrar em quais cenários cada um deles se destacam, apresentando suas vantagens e desvantagens.

5.1 MODELAGEM E DESAFIOS

A modelagem de dados foi realizada seguindo o diagrama da figura 8.

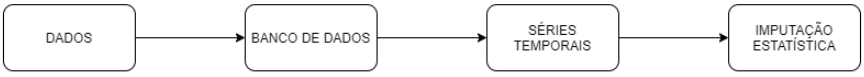


Figura 8 – Diagrama geral dos tratamentos ao ínicio das imputações

Primeiramente precisamos coletar os dados necessários. Para isso utilizamos uma base de dados previamente estabelecida que aloca os dados do NVD, onde temos os dados das vulnerabilidades, assim como links de referências que existem nas paginas de cada vulnerabilidade presente no NVD, mostrando outros locais onde possuem informações sobre a mesma,com isso conseguimos tirar as informações para a mesma vulnerabilidade presente em outras plataformas.

Com os links de referência para outras plataformas, nós selecionamos as melhores plataformas referenciadas pelo NVD, para terem seus dados individualmente coletados e comparados com os demais. As plataformas foram selecionadas usando como parâmetro o número de vulnerabilidades diferentes em que o NVD referencia aquela plataforma, buscando aquelas que abrangem a maior quantidade de vulnerabilidades diferentes.

Depois de ranquearmos e selecionarmos as melhores plataformas, tivemos que coletar os dados delas, para isso utilizamos de *crawlers*. Basicamente desenvolvemos um código que a partir do hiperlink coleta a informação desejada do site pelo xPath html, logo para realizar essa coleta os sites tem que ser individualmente analisados para descobrir como são estruturados e se é possível coletar a data para qualquer link que seja direcionado para

ele. Esse foi um critério para retirar algumas plataformas do ranqueamento, já que ou não apresentavam os dados exigidos, ou não apresentava o dado de uma maneira estruturada.

Tendo as datas de divulgação das plataformas para cada vulnerabilidade referenciada, podemos ir para o próximo passo que é a construção das séries temporais. Nossas séries temporais tem como o parâmetro temporal o que nós chamamos de camadas, que basicamente é a ordem na qual as plataformas divulgaram informações sobre a vulnerabilidade em questão, caso duas plataformas divulguem no mesmo dia, ambas ocupam a mesma camada, e os valores são categóricos pois se referem as plataformas. Desta maneira temos as séries estruturadas por uma chave, que é a vulnerabilidade, seguida de trios de informação, o primeiro nos diz a camada que a plataforma se encontra, o segundo a data que foi divulgado (Cada camada apresenta uma data única), e em terceiro a plataforma que realizou a divulgação, onde retiramos os resultados apresentados na Figura 9.

```
CVE-1999-0018,1,1997-11-24,SFocus,2,1997-12-05,NVD
CVE-1999-0021,1,1997-10-16,SFocus,2,1997-11-05,NVD
CVE-1999-0025,1,1997-05-24,SFocus,2,1997-07-01,IBM,3,1997-07-16,NVD,4,2000-12-15,KBCert
CVE-1999-0036,1,1997-05-26,NVD,2,1997-09-01,IBM
CVE-1999-0039,1,1997-05-06,NVD,1,1997-05-06,SFocus,2,1997-08-01,IBM
CVE-1999-0047,1,1997-01-20,SFocus,2,1997-01-28,NVD
CVE-1999-0095,1,1988-10-01,NVD,1,1988-10-01,SFocus,2,2019-06-05,Openwall
CVE-1999-0112,1,1997-05-01,NVD,2,1997-05-20,IBM
CVE-1999-0113,1,1994-05-23,NVD,2,1996-12-04,SFocus
CVE-1999-0115,1,1997-09-01,NVD,2,1997-09-08,SFocus
CVE-1999-0118,1,1998-11-01,NVD,2,1998-11-19,Marc
CVE-1999-0132,1,1993-06-01,IBM,2,1996-08-15,NVD
CVE-1999-0144,1,1997-06-01,NVD,2,1997-06-11,IBM,2,1997-06-11,SFocus,3,1997-06-12,Marc
```

Figura 9 – Séries temporais geradas

Com as séries temporais construídas, podemos partir para o próximo passo, que é a modelagem e imputação estatística dos dados, onde realizamos dois métodos com abordagem e resultados diferentes.

5.2 MÉTODOS REALIZADOS PARA AS IMPUTAÇÕES ESTATÍSTICAS

A seguir, descrevemos dois métodos para imputações estatísticas: cadeias de Markov (Seção 5.2.1) e mineração de regras (Seção 5.2.2).

5.2.1 Imputações utilizando as cadeias de Markov

A modelagem e imputação estatística pelo método da cadeia de Markov que é explicado mais detalhadamente na seção 4.1 foi realizado seguindo diagrama da figura 10.

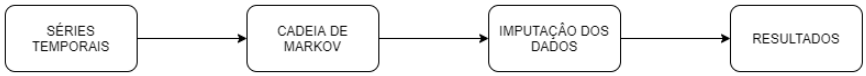


Figura 10 – Diagrama dos passos para aplicar o método das cadeias de markov

A partir das séries temporais explicadas na seção anterior, foi criada uma matriz de transições, representada na figura 11, onde os estados são as plataformas, e a transição é a probabilidade dele se deslocar para qualquer outra plataforma. Para preencher a matriz foi realizada uma varredura onde as plataformas em uma camada apontam para outras em outra camada, desta maneira é contabilizado uma transição entre as plataformas, e depois normalizamos esses valores, podemos preencher a matriz que nos informa dado um estado X qual a probabilidade de irmos para o estado Y, podemos ver também um estado simbolizado pelo infinito, esse estado seria um sumidouro no qual a ultima plataforma a divulgar uma determinada vulnerabilidade é conectada no final da série.

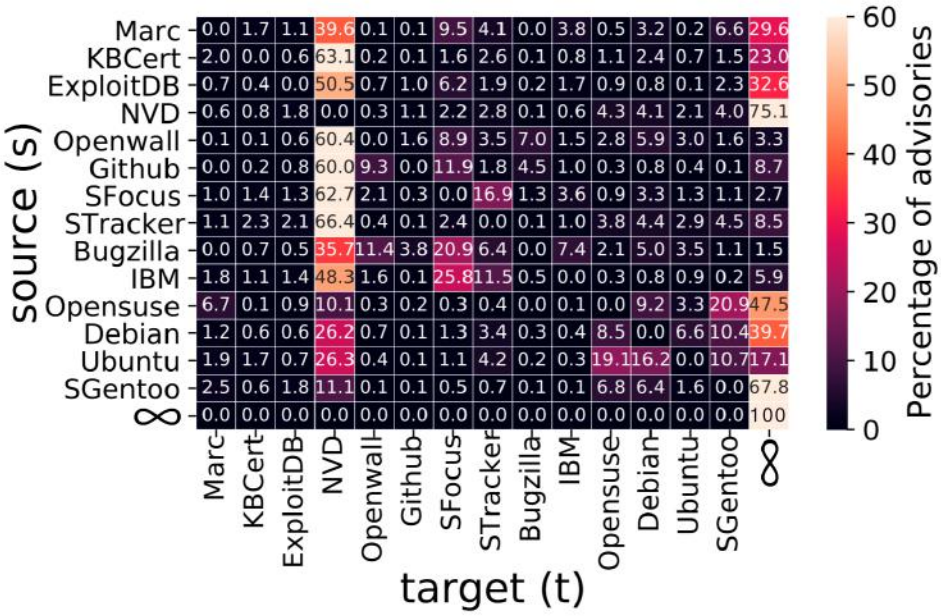


Figura 11 – Matriz de roteamento gerada das séries temporais

Com a cadeia de Markov criada, nós precisamos definir como será realizada essa imputação de dados, com isso nossa imputação assume que temos uma sequência e queremos definir em qual camada a plataforma X entraria nessa sequência. Para realizar a imputação precisamos definir um método de pontuação para as cadeias, e esse método foi o

de usar as probabilidades da matriz de transição, para dizer a probabilidade de ocorrer aquela cadeia, basicamente multiplicando os valores de cada transição da sequência.

Com o método de pontuação definido, podemos realizar a imputação, onde para testar o sistema de comparação criado, nós passamos em todas as plataformas de todas as sequencias, para cada uma nós retiramos ela e tentamos reinserir ela em todas as posições possíveis, calculamos a pontuação de cada sequência e comparamos para ver qual posição tem a maior pontuação, e com isso retiramos a posição dada pelo modelo.

Ao comparar a posição dada com o modelo com a posição real da plataforma na sequência, nós obtivemos os resultados presentes na figura 12.

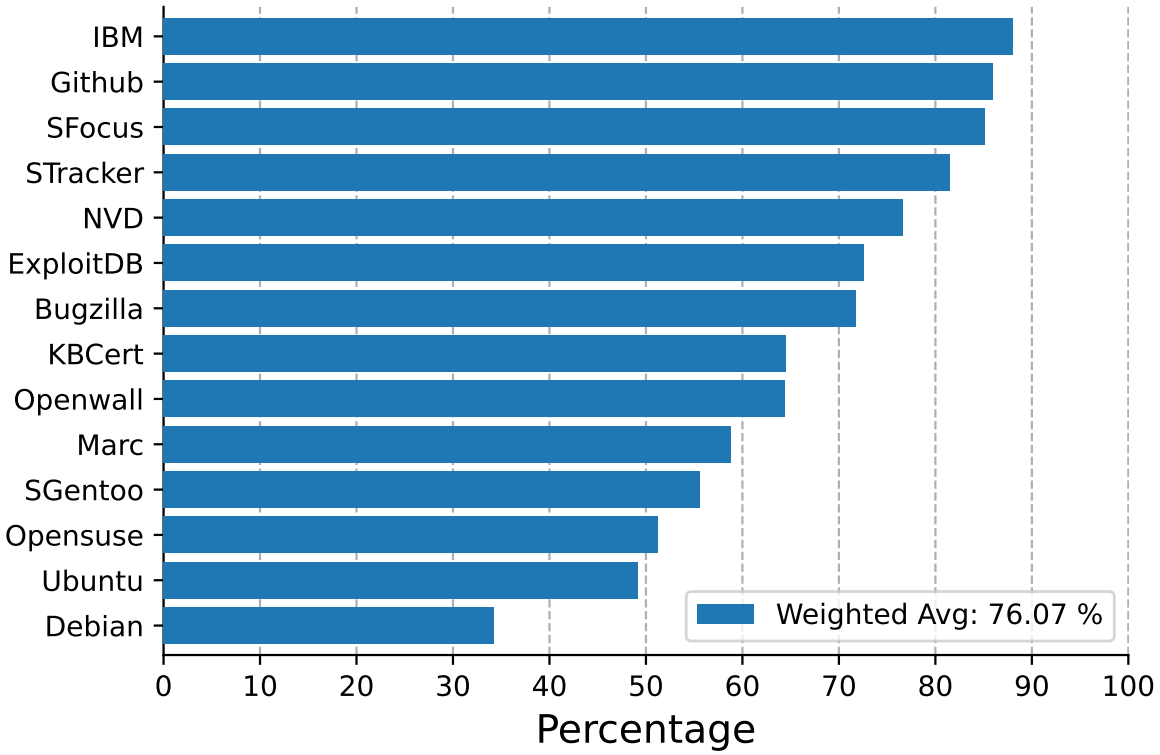


Figura 12 – Resultados do método da cadeia de Markov

Podemos observar que obtivemos um ótimo resultado, ainda mais sabendo que as assertividades nas plataformas com maior volume de dados foram bem altos. O comportamento identificado foi que plataformas com uma menor entropia, ou seja que tem sua posição média variando menos entre as vulnerabilidades que ela divulgou, apresentam resultados melhores já que seu comportamento é mais fácil de interpretar, essa comparação pode ser observada na figura 13.

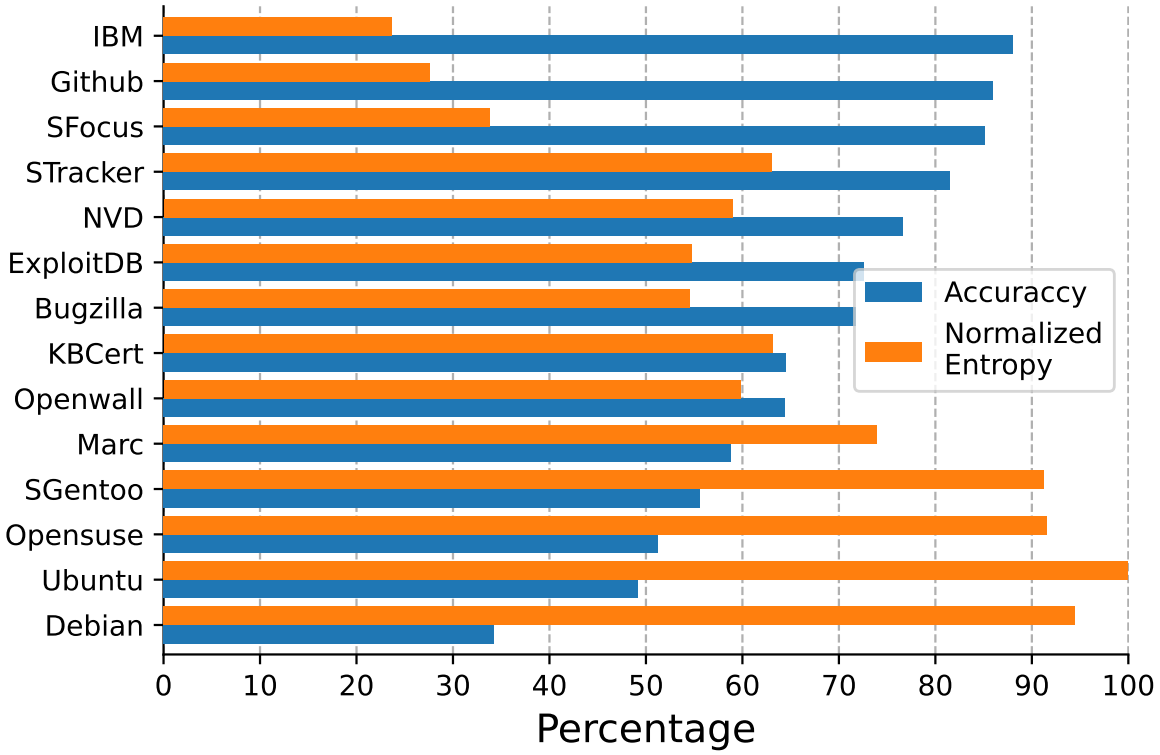


Figura 13 – Resultados do método da cadeia de Markov com entropia

5.2.2 Imputações utilizando Rule Mining

O segundo método utilizado foi o *Rule Mining* e toda sua aplicação foi conforme o diagrama da figura 14.

Figura 14 – Diagrama dos passos para aplicar o método do *Rule Mining*

Neste método utilizamos o *Rule Mining* explicado na seção 4.2, que é um método feito para encontrar regras e padrões entre *tags* de uma tabela. Como nossos dados estão estruturados como séries temporais, é preciso encontrar uma forma de converter esses dados para uma tabela de *tags*.

Para converter os dados das séries temporais para *tags*, foi utilizado do fator do nosso sistema ser modelado em camadas, e os valores serem categóricos, desta maneira conseguimos definir um universo finito de possibilidades de combinação entre plataforma e camada. A nossa maior sequência apresenta 12 camadas, e o número de plataforma que temos é 14, logo o número total de combinações entre camadas e plataformas que temos é 168. A nossa tabela presente na figura 15 foi estruturada com a chave sendo a vulnerabilidade, e as colunas foram estruturadas da seguinte maneira, temos 12 colunas para cada plataforma enumerada de 1 a 12 (PLATAFORMA-CAMADA), onde cada uma dessas colunas nos diz se para aquele sequência, existe uma determinada plataforma em uma determinada camada.

	CVE	SFocus_1	SFocus_2	SFocus_3	NVD_1	NVD_2	NVD_3
0	CVE-1999-0002	True	0	0	0	True	0
1	CVE-1999-0003	0	True	0	True	0	0
2	CVE-1999-0005	True	0	0	0	True	0
3	CVE-1999-0006	True	0	0	0	True	0
4	CVE-1999-0018	True	0	0	0	True	0
5	CVE-1999-0021	True	0	0	0	True	0
6	CVE-1999-0025	True	0	0	0	0	True
7	CVE-1999-0036	0	0	0	True	0	0
8	CVE-1999-0039	0	True	0	True	0	0
9	CVE-1999-0047	True	0	0	0	True	0
10	CVE-1999-0095	0	True	0	True	0	0

Figura 15 – Estrutura *tags*

Tendo a estruturação dos dados em formato de *tags* sendo concluída, podemos realizar o treinamento com o *Rulemining* que recebe como entrada a tabela onde serão explicitadas as *tags* que devem ser avaliadas, e tem como saída as regras encontradas pelo modelo, as regras são estruturadas da seguinte maneira, temos os antecedentes, que são um grupo de *tags* que implica em alguma regra, os consequentes que são um grupo de *tags* que é implicado nessa regra, e o *confidence*, que é a métrica que nos diz a confiabilidade dessa regra, essa tabela é apresentada na figura 16.

rule_num	antecedents	consequents	support	confidence
0	['ExploitDB_1', 'NVD_4', 'SFocus_3']	['IBM_2']	0,039	0,996
1	['ExploitDB_1', 'IBM_2', 'NVD_4']	['SFocus_3']	0,039	0,990
2	['IBM_1', 'NVD_4', 'STracker_3']	['SFocus_2']	0,029	0,981
3	['ExploitDB_1', 'SFocus_3']	['IBM_2']	0,041	0,977
4	['ExploitDB_1', 'NVD_4']	['IBM_2']	0,040	0,965
5	['ExploitDB_1', 'NVD_4']	['SFocus_3']	0,040	0,960
6	['ExploitDB_1', 'IBM_2', 'SFocus_3']	['NVD_4']	0,039	0,958
7	['ExploitDB_1', 'NVD_4']	['IBM_2', 'SFocus_3']	0,039	0,956
8	['SFocus_3', 'Marc_2']	['IBM_1']	0,011	0,954

Figura 16 – Regras criadas pelo *Rule Mining*

Com as regras coletadas precisamos criar uma metodologia para aplicar e testar o modelo de imputação de dados, para isso vamos utilizar como método de pontuação o *confidence*.

Com o método de pontuação definido, podemos realizar a imputação, onde para testar o sistema de comparação criado, nós passamos em todas as plataformas de todas as *tags* (que estão presentes naquela sequencia), para cada uma nós retiramos ela e averiguamos todas as regras que existem, e em quais delas, temos um antecedente nos conjuntos de *tags* presentes naquela cadeia, e onde nos consequentes temos nossa plataforma alvo em alguma posição podendo estar junta com as outras *tags* da regra que deve estar presente na sequencia, logo se minha plataforma alvo é a X, seria uma regra valida, uma que diz que as plataformas A e B implicam nas plataformas X e C, para ela ser levada em consideração, precisa tanto A,B e C existirem nessa cadeia. Após pegar todas as posições que as regras indicaram, basta comparar entre elas e retirar a com maior *confidence*.

Ao comparar a posição dada com o modelo com a posição real da plataforma na sequência, nós obtivemos os resultados mostrados na figura 17.

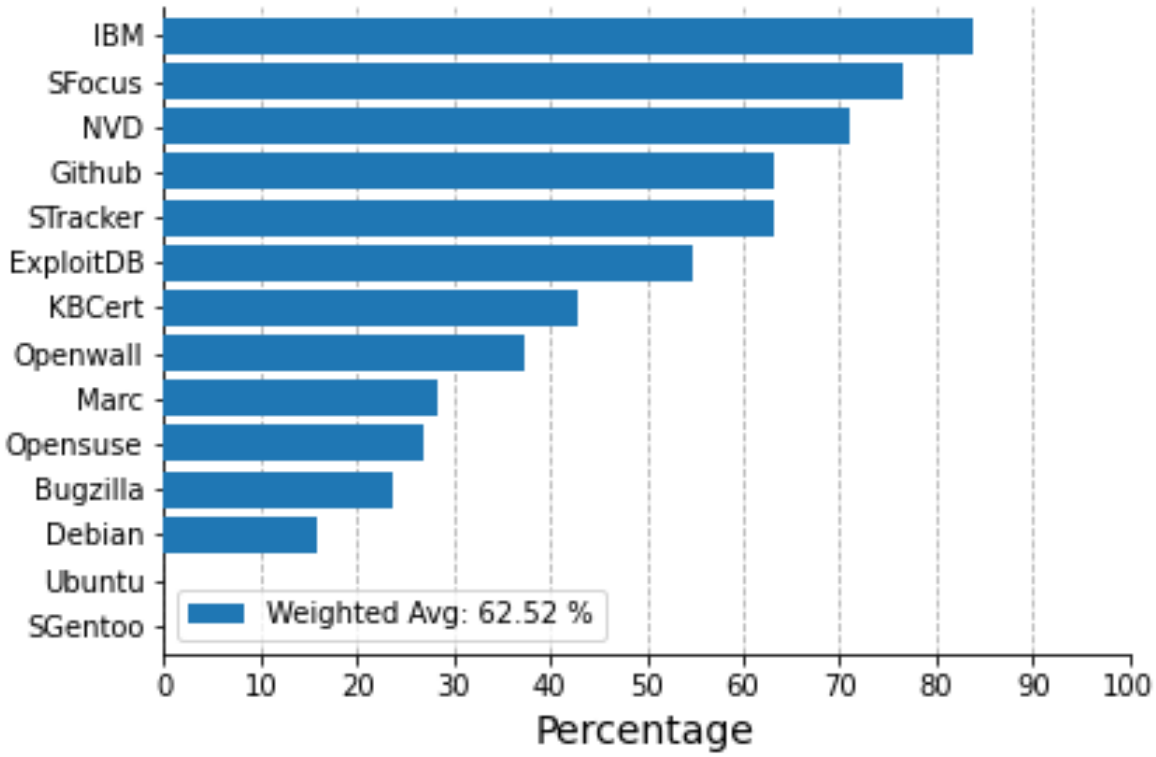


Figura 17 – Resultados do método *Rule Mining*

Podemos ver que o resultado foi inferior ao apresentado pelo método da cadeia de Markov, mas o principal fator observado é que as plataformas que apresentam maior número de vulnerabilidades relacionadas tiveram melhor resultado. Com isso podemos relacionar como um possível motivo da baixa performance de algumas plataformas,a falta de dados para encontrar regras que sejam úteis.

5.3 COMPARAÇÕES ENTRE OS MÉTODOS

Podemos ver que o método da cadeia de Markov, mostrou resultados bem melhores que o método utilizando *Rule Mining*, porém ambos tem seus pontos positivos e negativos.

As cadeias de Markov observam nossa cadeia como se fossem bigramas, ou seja avaliando a relações entre pares subsequentes, e como temos pouca variedade de plataforma e não temos um volume tão grande de dados se mostra o ideal com ótimos resultados como foi visto anteriormente. Pela quantidade pequena de dados de algumas plataformas, esse método também se mostra muito útil pois ele não se mostra tão afetado por falta de dados quanto o *Rule Mining*.

O *Rule Mining* como mencionado se mostra ineficiente para plataformas com pequeno volume de dados, porém ele nos mostra uma visão mais ampla que as demais visões, pois

nesse método nós não observamos apenas uma relação entre 2 elementos, mas sim entre X elementos, podendo extrair informações não só de elementos subsequentes, mas também informações de preenchimento como visto em algumas regras mostradas. Logo o *Rule Mining* por avaliar mais cenários acaba não se beneficiando tanto em pequenos volumes de dados, por não conseguir gerar regras pela pequena quantidade de ocorrências, mas por outro lado ele pode identificar padrões que não seriam observados utilizando a visão de bigrama utilizado na cadeia de Markov, desta maneira com um volume geral maior de dados, o *Rule Mining* tende a ter uma performance cada vez melhor, enquanto as cadeias de Markov tendem a ficar genéricas demais, podendo ter uma redução na sua assertividade.

6 IMPUTAÇÃO ESTATÍSTICA DE TAGS

Neste capítulo consideramos diferentes eventos associados a vulnerabilidades de software, como por exemplo, *patching* e *weaponization*. Cada vulnerabilidade em geral está associada a vários *links* (URLs) com informações sobre tal vulnerabilidade, e cada um desses links possui diferentes *tags* associados. Tanto o National Vulnerability Database (NVD) quanto o Github, em particular, proveem tags para alguns dos links associados às vulnerabilidades. Nosso propósito é então propagar tais tags, também chamadas de etiquetas (*label propagation*) dos links que já foram pre-etiquetados para links que ainda não contém tags.

Começamos o capítulo com uma descrição dos dados disponíveis no NVD e Github (Seção 6.1). Em seguida, respondemos as seguintes perguntas: 1) **por que** é relevante fazer o *tagging* automático? (Seção 6.3) 2) **como** fazer o *tagging* automático? (Seção 6.4). Em particular, indicamos que o *tagging* automático é relevante porque o *tagging* manual leva tempo, e que uma forma de automatizar o processo consiste no uso de mineração de regras.

6.1 DADOS DISPONÍVEIS: NVD E GITHUB

Para nosso estudo de caso, escolhemos trabalhar com dados do NVD e Github. O NVD é a plataforma de segurança de referência sobre vulnerabilidades de software. Ele possui várias referências para sites externos, incluindo o Github, que tornou-se um dos *top* 10 sites mais referenciados pelo NVD. Tradicionalmente, sites como ExploitDB continham informações valiosas e atualizadas, de forma estruturada, sobre *exploits* e *patches* para vulnerabilidades. Entretanto, tal papel parece ter sido assumido atualmente pelo Github, que tornou-se a plataforma padrão para divulgação de códigos fonte.

Cabe destacar que tanto o NVD quanto o Github incluem *tags*, associados a cada URL. Entretanto, os mecanismos usados para inclusão de *tags* no NVD e no Github são distintos, e não são claramente documentados. Um de nossos propósitos é explicar os mecanismos de inclusão de *tags* das duas plataformas, e avaliar a que nível pode-se aprender sobre uma plataforma a partir da outra. Finalmente, estamos também interessados em compreender a consistência entre as plataformas, e se uma plataforma pode ser usada para eventualmente corrigir ou completar a outra.

A fim de ilustrar as *tags* que constam no NVD, para links associados ao Github, fazemos uma análise longitudinal na Figura 19.

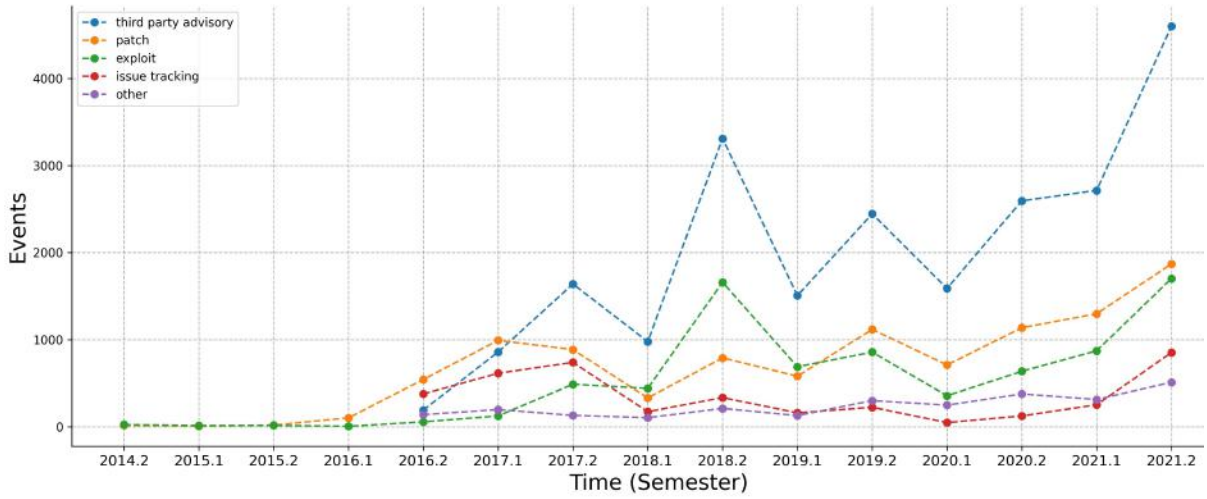


Figura 18 – Tags marcadas no NVD para eventos do Github (citados pelo NVD).

Na Figura 19 vemos que

- a quantidade de material do Github publicado no NVD vem aumentando, sugerindo o aumento da relevância do Github para fins de segurança (classicamente, o ExploitDB assumia tal papel, mas aos poucos o ExploitDB vem perdendo espaço para o Github)
- ao longo dos anos o Github vem publicado primordialmente *third party advisories*
- o Github publica em geral mais patches do que exploits
- entretanto, o número de patches e exploits vem crescendo proporcionalmente, sendo que em 2018 houve mais exploits do que patches, e em 2021 os números muito se aproximaram
- de 2014 a 2016, só existiam duas tags: exploit e patch. A partir de 2016, surgiram outras tags.

6.2 DATAS RELEVANTES

Consideramos três datas relevantes:

- t_C: data de publicação da CVE, pelo NVD
- t_T: para cada evento associado à CVE, no NVD, a data em que ocorreu a marcação de suas tags

- t_F: para cada evento associado à CVE, a data em que ele foi publicado por sua fonte (no nosso caso, Github).

O tempo $t_T - t_C$ caracteriza o atraso que leva, no NVD, para se marcar tags associadas aos eventos. Essa marcação é manual. A marcação automática de tags visa diminuir tais tempos (Figura 19).

O tempo $t_F - t_C$ caracteriza o atraso que leva-se para uma fonte divulgar dados sobre uma CVE, desde o instante em que a CVE é divulgada pelo NVD – a maioria dos link citados pelo NVD divulga informações sobre a CVE antes da CVE ser incluída no NVD, o que faz com que muitas dessas diferenças $t_F - t_C$ sejam negativas (Figura 21).

Note que quando dividimos os eventos em grupos, iremos calcular, para cada evento, a métrica $t_F - t_C$ e tal observação irá contribuir para o grupo (*cluster*) no qual a observação encontra-se. O conceito de *clusters* de *tags* é descrito na Seção 6.3.1.

6.3 MOTIVAÇÃO: TEMPO PARA INCLUSÃO MANUAL DE *TAGS* NO NVD E GITHUB É ALTO

A seguir, discutimos o tempo entre a descoberta de uma URL (evento) para uma certa vulnerabilidade e a inclusão de *tags* associadas a tal URL (evento). O processo de inclusão de *tags* é atualmente manual, o que motiva a automatização proposta neste capítulo. Além de diminuir o tempo de etiquetamento, o uso de regras automatizadas permite o entendimento das suas atribuições. Afinal, mecanismos automáticos são reprodutíveis, e facilitam a documentação. Abordagens baseadas em experiência humana são mais flexíveis, mas também sujeitas a subjetividade.

O processo de publicação de informações sobre vulnerabilidades passa por vários estágios: 1) divulgação da CVE, no NVD; 2) criação de um evento, sobre a CVE, pelo fabricante ou outra parte interessada; 3) inclusão do link no NVD para o evento; 4) inclusão de tag no link do NVD. Entre 1) e 4), conforme apontado na Figura 19, temos o tempo desde a publicação de uma CVE no NVD até surgir determinada *tag* para um evento associado a ela. É interessante notar que os links de patching são os que, na média, levam mais tempo para serem incluídos: eles levam em torno de 4 meses a mais do que o tempo para incluir tag de exploit.

A Figura 19 mostra o tempo para incluir *tags*, indicando que esse tempo varia de acordo com a categoria. Usamos aqui as *tags* originais conforme atribuídas pelo NVD. Na seção a seguir, consideramos *clusters* de tags para facilitar a análise.

É interessante notar que o tempo médio é sempre maior que 129 dias. Ou seja, leva-se, em média, no mínimo 4 meses para se incluir *tags* nos eventos apontados pelo NVD. Reduzir esse tempo é um dos propósitos da marcação automática de tags, proposta neste capítulo.

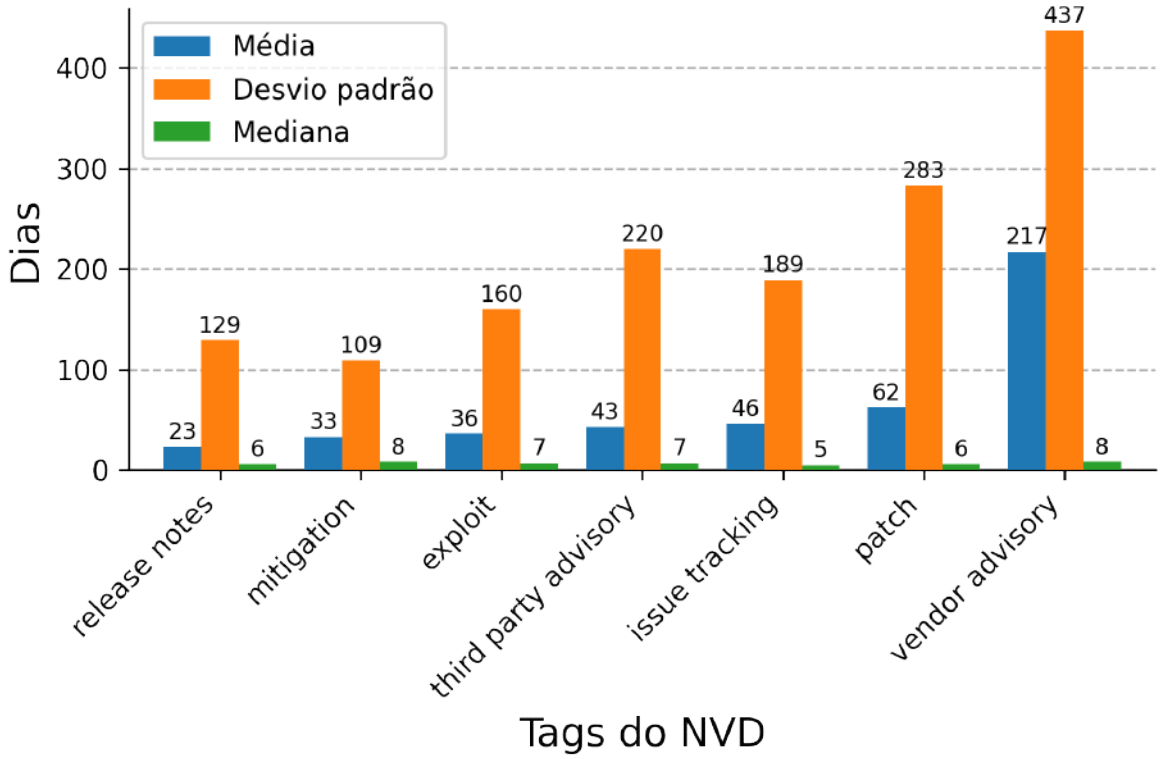


Figura 19 – Média, desvio padrão e mediana de dias desde a publicação de uma CVE no NVD até surgir determinada *tag* em eventos (URLs) associados a ela.

6.3.1 Por que *clusters?*

A Figura 20 mostra os grupos (*clusters*) de *tags*. Agrupamos as *tags* em grupos para facilitar a análise, e apresentação dos resultados. Além disso, depois de agruparmos as *tags* em grupos verificamos uma substancial melhora do desempenho dos algoritmos em consideração, tendo em vista que a tarefa de inferir o grupo é mais simples do que a de inferir cada uma das *tags* individualmente.

third_party_advisory_cluster	vendor_advisory_cluster	issue_tracking_cluster	patch_cluster	exploit_cluster
third_party_advisory us_government_resource permissions_required tool_signature vdb_entry	vendor_advisory	issue_tracking press_media_coverage mailing_list	patch release_notes mitigation	exploit technical_description

Figura 20 – Tabela com os clusters das tags do NVD.

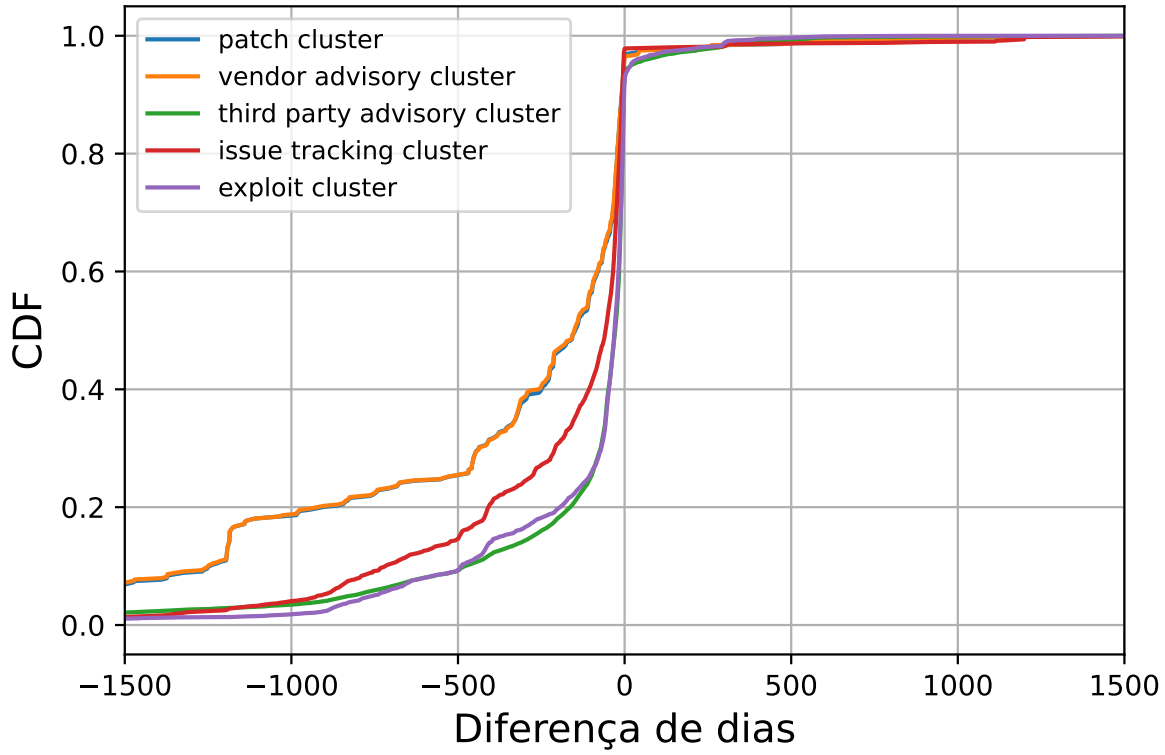


Figura 21 – CDF da diferença de dias desde a publicação de uma CVE no NVD até a publicação pelo Github: data de divulgação no Github menos data de publicação da CVE. A maioria dos eventos ocorre antes de o CVE ser publicado no NVD, o que mais uma vez motiva uma forma automática de marcar tags do NVD quando o evento foi publicado no Github. Note que para cada evento (URL) extraímos uma observação de $t_F - t_C$ e agrupamos tais observações por *clusters* associados às tags para gerar as cinco curvas, correspondentes aos cinco clusters, no gráfico.

A Figura 21, a ser contrastada com a Figura 19, mostra a CDF entre o tempo de publicação de um evento no Github e o tempo de publicação da correspondente CVE no NVD. Enquanto a Figura 19 leva em conta as tags originais do NVD, a Figura 21 leva em conta *clusters* de *tags*.

Algumas observações sobre a Figura 21:

- as linhas de *vendor advisory* e *patch* ficam muito próximas uma da outra: isso ocorre porque muito frequentemente quando surge um patch, surge um vendor advisory logo em seguida, ou um pouco antes. Existem ainda os casos em que o mesmo evento pertence ao mesmo tempo aos dois *clusters*, ou seja, é marcado como patch e vendor advisory

- as linhas de *third partry advisory* e *exploit* ficam também relativamente próximas:

conforme discutido no item acima, enquanto patches costumam vir dos *vendors*, *exploits* costumam vir de *third party advisories*

- entre os dois elementos discutidos nos itens acima, temos os *issue trackings*, que quando são lançados antes da divulgação dos CVE, são lançados muitas vezes um pouco antes dos exploits, mas não antes dos patches. Ou seja, o ciclo natural costuma ser: patch, issue tracking e exploit, quando esses eventos ocorrem antes da divulgação do CVE.

- quando a publicação da CVE pelo NVD ocorre antes da publicação do evento no Github (valores positivos no eixo x), fica mais difícil distinguir entre o comportamento associado aos diferentes clusters, no que se refere ao tempo entre a publicação da CVE e a ocorrência do evento no Github. Esse caso, entretanto, só ocorre para uma fração entre 40% e 20% dos dados, ou seja, em geral o NVD funciona como um *hub* de informações já sedimentadas sobre CVEs publicados, e não como uma fonte de novas notícias.

Na Figura 21 consideramos *clusters* de tags para facilitar a análise, enquanto que na Figura 19 consideramos tags originais do NVD. Enquanto na Figura 21 mostramos a CDF do tempo para surgimento de eventos no Github (notando que muitas vezes os eventos ocorrem antes da CVE ser publicada) na Figura 19 mostramos o tempo para inclusão de tags (que muitas vezes ocorre de forma tardia). Em ambos os casos, temos uma forte motivação para o *tagging* automático. Sem a CVE ser publicada no NVD, não há como o NVD marcar *tags* nos eventos. Mesmo quando a CVE é publicada antes dos eventos, leva-se um tempo para marcar as *tags*, e sem as *tags*, é difícil percorrer de forma sistemática os eventos, por exemplo, classificando patches e exploits, motivando assim formas automáticas de etiquetamento, conforme descrito a seguir.

6.4 METODOLOGIA PARA *TAGGING* AUTOMÁTICO

Nossa metodologia envolve os seguintes passos:

1. coleta dos dados do NVD e Github, usando o NVD como referência
2. montagem do *dataset* em que cada linha corresponde a uma URL (evento) e cada coluna corresponde a um atributo (por exemplo, um atributo do tipo verdadeiro/falso pode indicar se um determinado *tag* se aplica a um determinado evento). Cabe destacar que como um determinado link pode estar associado a diferentes CVEs, pode ocorrer de ele aparecer múltiplas vezes no *dataset*, e inclusive verificamos situações em que as aparições distintas correspondem a valores de atributos distintos
3. aumentar o *dataset* construído no item anterior com atributos adicionais (por exemplo, presença ou ausência de determinado termo chave, etc) – tais atributos podem conter informações temporais ou não
4. executar algoritmo a priori para aprender as regras
5. colher regras e aplicá-las (propagá-las) n vezes, onde n é estabelecido pelo usuário
6. interpretar e analisar os resultados

6.5 ENGENHARIA DE ATRIBUTOS

Procedemos com engenharia de atributos (*feature engineering*) para determinar quais atributos serão usados pelo algoritmo de mineração de regras, a fim de automatizar o processo de *tagging* de eventos (Figura 22). A principal ideia consiste em identificar atributos que tragam considerável informação sobre as classes de interesse (e.g., patching e exploitation). Verificamos, por exemplo, que a extensão do arquivo, a *label* do Github, bem como o tipo de URL do Github e palavras chaves contidas no HTML do Github são informações valiosas para se determinar *tags* no NVD. A maioria dessas informações, naturalmente, está intrinsicamente relacionada com as *tags* do NVD: por exemplo, é de se esperar que as etiquetas do Github estão correlacionadas com as *tags* do NVD. Entretanto, não é de nosso conhecimento nenhum trabalho anterior que tenha explorado tais relações.

Note que todos os atributos apresentados na tabela da Figura 22 são estáticos. Entretanto, em trabalhos futuros também vislumbramos usar atributos temporais (por exemplo, se já existir um evento com *tag* exploit), e nenhum com patch, é provável que o próximo evento seja um patch.

Dentre a escolha dos atributos, a seleção de palavras chaves foi uma das mais desafiadoras: quais palavras são relevantes para determinar a classe de um evento? Algumas palavras naturais, como “proof of concept”, “PoC” e “exploit” aparecem na tabela abaixo.

Outras palavras foram selecionadas manualmente, baseando-se em experimentação manual. Foi tentado usar a informação mútua, e outras técnicas alternativas, para seleção automática de palavras, mas sem os desejados resultados.

NVD (5)	Extensão de arquivo URL GitHub (38)	GitHub Issues Labels (8)	Tipo de URL GitHub (4)	Palavras-chave HTML GitHub (14)
third_party_advisory_cluster	MD_url_file_ext	cve_issue_label	commit_url_type	proof_of_concept_word
issue_tracking_cluster	TXT_url_file_ext	vulnerability_issue_label	issues_url_type	poc_word
patch_cluster	c_url_file_ext	exploit_issue_label	pull_url_type	exploit_word
exploit_cluster	cpp_url_file_ext	bug_issue_label	other_url_type	patch_word
...	...	...		...

Figura 22 – Tabela com as categorias, total e exemplos de *tags* que utilizamos.

Na primeira coluna da tabela na Figura 22 mostramos quatro dos cinco *clusters* apresentados na Figura 20. Na segunda coluna, indicamos que a extensão do arquivo, no Github, extraída direto a partir do final do URL do link, tem papel importante na classificação. Tal extensão pode ser extraída com facilidade a partir da URL, sem a necessidade de se fazer o download do documento do Github, ou seja, ela pode ser colhida apenas acessando-se o link no NVD. Na terceira coluna, temos a etiqueta do evento, conforme marcada pelo usuário do Github. Tal etiqueta é colocada manualmente pelos autores de uma *issue*, e está disponível apenas para *issues*. A etiqueta não precisa ser escolhida a partir de uma lista pre-selecionada, ou seja, o usuário tem plena liberdade para colocar a etiqueta que desejar. Na tabela, apresentamos algumas das etiquetas que julgamos relvantes para nosso trabalho. Lembramos que os elementos (URLs) do Github são classificados como de um dentre vários tipos: *issues*, *commit*, *pull*, etc. Estes tipos são listados na quarta coluna, e marcamos como *other_ url_ type* todos os diferentes de issues, commit e pull. Note que as etiquetas da terceira coluna estão disponíveis apenas para *issues*. Além disso, para saber o tipo da URL também não precisamos fazer o download do HTML do Github. Ou seja, precisamos fazer o download do HTML apenas para colher etiquetas do Github para *issues* (terceira coluna) e para colher palavras chaves (quinta-coluna). Os valores entre parêntesis na Figura 22 representam o número de possíveis *tags* dentro de cada categoria. Temos, por exemplo, 14 palavras-chave relevantes em nosso estudo, e apresentamos 4 delas na tabela, a saber, `proof of concept`, `poc`, `exploit` e `patch`.

6.6 REGRAS COLHIDAS: ALGORITMO *APRIORI*

Listamos na Figura 23 algumas regras colhidas com o algoritmo *Apriori* explicado na seção 4.2.

Antecedents	Consequents	Support	Confidence
['other_url_type', 'poc_word', 'exploit_cluster']	['third_party_advisory_cluster']	0.05	0.99
['other_url_type', 'exploit_word', 'cve_word', 'exploit_cluster']	['third_party_advisory_cluster']	0.03	0.99
['md_url_file_ext']	['other_url_type', 'third_party_advisory_cluster']	0.06	0.97
['commit_url_type', 'third_party_advisory_cluster', 'c_word', 'fix_word']	['patch_cluster']	0.03	0.97
['issue_tracking_cluster', 'commit_url_type']	['patch_cluster']	0.03	0.96
['third_party_advisory_cluster', 'issues_url_type', 'poc_word']	['exploit_cluster']	0.07	0.87
['exploit_word', 'fix_word']	['third_party_advisory_cluster']	0.04	0.87
['poc_word', 'third_party_advisory_cluster', 'c_word', 'exploit_cluster']	['issues_url_type']	0.03	0.86
['poc_word']	['exploit_cluster']	0.12	0.83
['patch_cluster', 'c_word']	['third_party_advisory_cluster']	0.11	0.83
['third_party_advisory_cluster', 'cve_word', 'poc_word']	['exploit_cluster']	0.05	0.83
['commit_url_type']	['patch_cluster']	0.24	0.8
['issue_tracking_cluster', 'cve_word']	['issues_url_type']	0.05	0.77
['third_party_advisory_cluster', 'pull_url_type', 'fix_word']	['patch_cluster']	0.03	0.77
['other_url_type', 'third_party_advisory_cluster', 'md_url_file_ext']	['exploit_cluster']	0.05	0.75

Figura 23 – Tabela com exemplos de regras geradas pela mineração de regras e aplicadas para fazer a propagação de etiquetas.

A Figura 23 mostra alguns exemplos de regras geradas pela mineração de regras e aplicadas para fazer a propagação de etiquetas. As regras são interpretáveis:

- a primeira e segunda regras sugerem que exploits em geral não são publicados pelos *vendors* mas sim por *third parties*
- a terceira regra sugere que o tipo MD para a URL implica em *third party advisory*
- a quarta regra sugere que se consta a palavra *fix* em geral temos um *patch*
- a sexta regra indica que a palavra *PoC* (proof-of-concept) sugere *exploit*

6.7 RESULTADO DA PROPAGAÇÃO DE ETIQUETAS

A seguir, mostramos o resultado do algoritmo de propagação de etiquetas, usando mineração de regras. O estado inicial dos eventos é apresentado na Figura 24(a). Temos um total de 27,297 eventos, que constam tanto no NVD quanto no Github. Deste eventos, 23,195 são marcados como *third party advisories*, 12480 como *patches*, e 8426 como *exploits*. Após uma iteração do algoritmo de propagação de etiquetas, o número de elementos nestas categorias cresce para 26970, 14711, e 10340 respectivamente. Após a segunda e terceira iterações, todos os elementos foram marcados como *third party advisories*. De fato, as empresas não costumam incluir oficialmente no Github informações sobre patches e exploits de seus produtos, então é coerente que todas as postagens sejam marcadas como *third party advisories*. A parte mais interessante se refere aos patches e exploits, que crescem para 16669 e 13009 após a segunda iteração, e 19650 e 15002 após a terceira iteração.

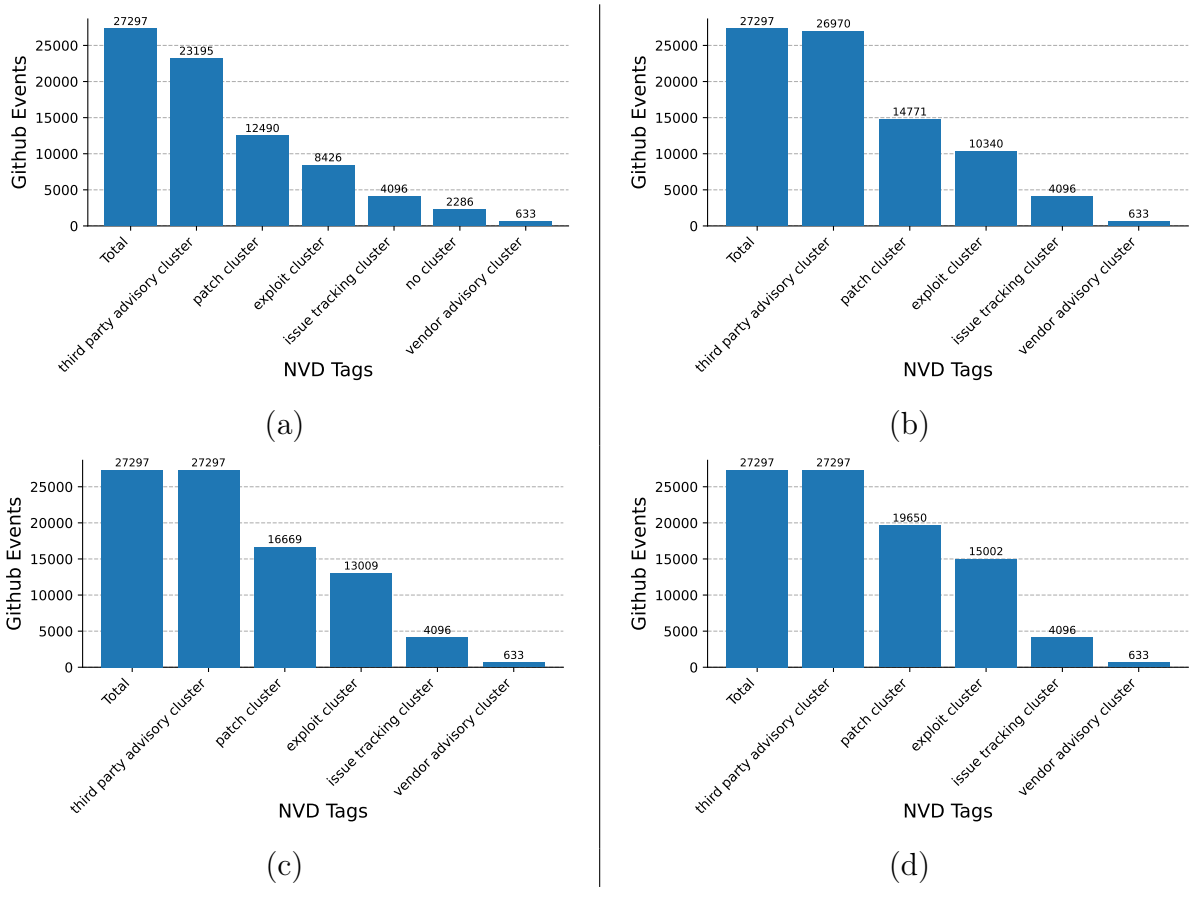


Figura 24 – Propagação de tags segundo o algoritmo de mineração de regras aplicado de forma iterativa

6.8 VALIDAÇÃO

Note que ao longo das três primeiras iterações do algoritmo de propagação de etiquetas, tivemos um aumento do número de eventos marcados como patches e exploits. Após uma inspeção manual, verificamos que de fato a propagação de etiquetas marcou adequadamente os eventos de acordo com sua natureza. Entretanto, execuções de iterações adicionais parecem levar a um *overfitting* dos dados, com propagações que não refletem a realidade. Ainda estamos investigando formas ótimas de determinar o número de iterações do algoritmo para obter o maior número possível de propagações, mas ainda assim garantindo a qualidade.

Deixamos também como objeto para trabalho futuro rodar o algoritmo com dados para os quais tenhamos *ground truth* e verificar até qual iteração o algoritmo consegue capturar o *ground truth* (calculando acurácia, *precision*, *recall*, etc).

6.9 ANÁLISE DOS RESULTADOS E IMPLICAÇÕES

Em resumo, neste capítulo verificamos que o algoritmo de propagação de etiquetas usando regras de associação oferece uma boa alternativa para aumentar o número de eventos etiquetados no NVD. A abordagem gera regras explicáveis, que podem produzir *insights* interessantes sobre segurança e permitindo o melhor uso dos eventos disponíveis na base do NVD para fins de gerenciamento de *patching* e análise de risco.

7 TRABALHOS RELACIONADOS

Neste capítulo vamos abordar os principais trabalhos relacionados com o tema de vulnerabilidades de software e o modelo de fila utilizado no trabalho, começando pela seção 7.1 que mostramos outros trabalhos que buscam analisar o comportamento de vulnerabilidades, em seguida vamos abordar trabalhos voltados para analise evolutiva ao longo do tempo na seção 7.2, seguido pela seção 7.3 que vai apresentar trabalhos voltados para quantificar a ameaça de cada vulnerabilidade, e por ultimo temos a seção 7.4 que mostra os trabalhos relacionados com a rede de filas, que foi o modelo escolhido para representar o fluxo de vulnerabilidades entre as plataformas.

7.1 TRABALHOS RELACIONADOS COM O USO DO NVD PARA ANALISE DE VULNERABILIDADES

Contamos com o NVD como nossa referência para selecionar as plataformas de segurança analisadas. Estudos anteriores se basearam no NVD onde foram criadas cerca de 45 clusters diferentes envolvendo os padrões entre as vulnerabilidades coletadas do NVD (HUANG; TANG et al., 2010) e prever vulnerabilidades de software e avaliar os riscos correspondentes principalmente voltados para vulnerabilidades *zero day* e vulnerabilidades ainda não mapeadas que acabam sendo muito impactantes por ainda não terem sido estudadas como as demais (ZHANG; OU; CARAGEA, 2015). Algo em comum em ambos os trabalhos utilizou a lista de hiperlinks fornecida pelo NVD com avisos sobre cada vulnerabilidade onde podem ser tiradas informações importantes como a data de publicação, que para alguns casos é um dos principais fatores para determinar o quão bem explorada a vulnerabilidade já pode ter sido, e neste trabalho de conclusão de curso foi um dos nossos principais objetos de estudo.

7.2 TRABALHOS RELACIONADOS COM VULNERABILIDADES E SUA EVOLUÇÃO AO LONGO DO TEMPO

O ciclo de vida das vulnerabilidades que consiste no processo de surgimento de uma vulnerabilidade, e nos processos de *exploitation* e *weaponization* onde é a utilização dessa vulnerabilidade de fato para atacar o produto, neste trabalhos (FREI et al., 2006; SHAHZAD; SHAFIQ; LIU, 2012) são estudados esses conceitos que são muito importantes para as empresas saberem como lidar e como se preparar para eventuais problemas, assim como mensurar o quão seguro é um produto. As praticas de *patching* por sua vez, que são os meios pelos quais a empresa consegue se proteger das vulnerabilidades aparecem neste trabalho (WANG et al., 2017) com o foco principal em *Industrial Control Sistems*

(ICS), que por sua vez precisam de uma atenção especial com a aplicação do *patching* pois isto significa que o sistema talvez precise ficar fora do ar, e isso gera custos e deve ser bem planejado. Nosso trabalho considera o ciclo de vida da vulnerabilidade de um novo ângulo, levando em conta os atrasos das plataformas e aproveitando os diagramas Sankey e as redes de filas. Enquanto a maioria dos trabalhos se concentra na previsão de *weaponization* e *exploitation*, nós nos concentramos no fluxo de avisos entre plataformas, que é fundamental para modelos que quantificam o risco de uma vulnerabilidade baseada na quantidade de referências que são encontradas a respeito da mesma. Também podemos atrelar as analises de *tags* que geram informações de *exploit* e *patch* que quando atreladas as séries temporais podem ser relacionadas com datas e assim criar um ciclo de vida para as vulnerabilidades estudadas.

7.3 TRABALHOS RELACIONADOS COM VULNERABILIDADES E A QUANTIFICAÇÃO DE AMEAÇA

E temos o papel de *Common Vulnerability Scoring System* (CVSS) que são exploradas em (RUOHONEN; HYRYNSALMI; LEPPÄNEN, 2017; JOH; MALAIYA, 2010; MELL; SCARFONE; ROMANOSKY, 2006) e fornecem uma maneira de capturar as principais características de uma vulnerabilidade e produzir uma pontuação numérica que reflete sua gravidade, levando em conta métricas a respeito de como esta vulnerabilidade atinge o sistema, o NVD possui uma calculadora para o mesmo (National Vulnerability Database, 2022).Fóruns de black hat e outras fontes de informação não estruturadas, como o Twitter (SABOTTKE; SUCIU; DUMITRAș, 2015), que possuem dificuldades em serem exploradas o twitter por não ser uma plataforma voltada para vulnerabilidades de software e possuir grandes limitações de escrita, como os fóruns black hat que não apresentam a estruturação de uma plataforma de divulgação de vulnerabilidades que fornecem informações bem mais acuradas. Neste TCC, também contribuímos para medições que podem ser instrumentais para determinar a probabilidade de uma vulnerabilidade ser explorada, por exemplo, sob a estrutura *Exploit Prediction Scoring System Calculator* (EPSS), em que o número de avisos que divulgaram informações sobre um determinado CVE é um parâmetro-chave (JACOBS et al., 2019), e um dos principais focos do nosso trabalho é a criação de fluxos que determinam por quantas plataformas uma vulnerabilidade passa ao longo do seu ciclo de vida.

A maior parte da literatura sobre divulgações de vulnerabilidades acompanha como as vulnerabilidades evoluem em um produto de software (JOHNSON et al., 2016), que é um tema muito valioso principalmente para conseguir entender quando vulnerabilidades vão surgir e ajudar a empresa a se preparar. Também existem trabalhos voltados para evolução de vulnerabilidades entre módulos de software (HU et al., 2019), onde busca trabalhar conjuntos de software ao invés deles individualmente, conseguindo assim men-

surar riscos levando as dependências daquele sistema em consideração e os riscos que elas submetem ao conjunto. Neste TCC, adotamos uma abordagem diferente em relação à evolução das vulnerabilidades de software, focando na evolução das vulnerabilidades entre as plataformas de segurança. Acreditamos que essas abordagens são complementares. Em particular, uma das plataformas consideradas em nosso estudo, o Github (DECAN; MENS; CONSTANTINOU, 2018; HORAWALAVITHANA; BHATTACHARJEE et al., 2019), possui em si um rico histórico de *patchs* e atualizações de software cujas linhas do tempo complementam as séries temporais consideradas neste TCC, sendo uma plataforma muito rica para coleta de dados principalmente temporais a respeito de vulnerabilidade.

7.4 TRABALHOS RELACIONADOS COM APLICAÇÃO DO MODELO DE FILAS

Modelamos um conjunto de plataformas de divulgação de avisos de vulnerabilidade como uma rede de filas. Existe uma vasta literatura sobre o uso de redes de filas, por exemplo, para acompanhar o fluxo de pacientes entre departamentos hospitalares (VAN-BERKEL; PETER, 2011), neste caso a rede de filas foi o método que atendeu melhor as necessidades do modelo para trabalhar com fluxo de pacientes. Também é possível encontrar aplicações dos modelos de fila para fluxo de dispositivos em redes celulares (BOUCHERIE; DIJK, 2000), onde sua aplicação pode auxiliar na otimização das frequências disponíveis que muitas vezes se mostram limitadas diante do crescimento do uso de celulares na época em que o trabalho foi referenciado foi realizado.E por último temos o fluxo de usuários nos canais de transmissão ao vivo (WU; LIU; ROSS, 2009). Neste TCC, uma rede de $M/G/\infty$ filas é usado para caracterizar o fluxo de avisos entre plataformas, esta fila é caracterizada pelas chegadas serem modeladas como um processo de Poisson, e os tempos de serviço são uma distribuição geral, possuindo infinitos servidores que trabalham simultaneamente. No nosso trabalho focamos no uso da rede de filas, porém não focamos nos tempos de serviço dado que a distribuição dos mesmos se mostrou muito instável e irregular como mostrado na seção 2.2

8 CONCLUSÃO

O que podemos observar com este trabalho no que diz respeito a séries temporais categóricas, é que por existir muito pouco conteúdo na internet sobre trabalhos com diversas séries temporais com poucos elementos, foi preciso adaptar e utilizar métodos inovadores, como foi o caso do *Rule Mining* que apesar de ser projetado para outro objetivo, foi possível adaptar o mesmo e extrair ótimos resultados como vistos anteriormente, mesmo trabalhando com *datasets* de tamanho pequeno, mas de maneira geral a maior contribuição para trabalhos futuros, é a utilização do *Rule Mining* que pode ser expandido e utilizado para outros diversos cenários que compartilham de características com o mostrado neste trabalho. Utilizando métodos estatísticos como a cadeia de Markov se obtiveram resultados ainda melhores, principalmente pela pouca quantidade de dado que não pareceu afetar esse método como afetou o anterior. O que podemos concluir é que ambos os métodos se mostraram muito promissores e os resultados muito satisfatórios, mostrando que ambos se usados de maneira correta podem ser muito eficientes e podem acrescentar muito quando tratamos de problemas de dados faltantes em séries temporais categóricas. Quanto a imputação estatística de tags com o foco no Github, verificamos que o algoritmo de propagação de etiquetas usando regras de associação oferece uma boa alternativa para aumentar o número de eventos etiquetados no NVD. Unindo estes eventos etiquetados com seus respectivos dados temporais, poderemos adquirir novas informações relevantes para o entendimento do ciclo de vida de vulnerabilidades.

REFERÊNCIAS

BLINDER, R. et al. Comparative evaluation of node-link and Sankey diagrams for the cyber security domain. In: SPRINGER. **IFIP Conference on Human-Computer Interaction**. [S.l.], 2019. p. 497–518.

BOUCHERIE, R. J.; DIJK, N. M. V. On a queueing network model for cellular mobile telecommunications networks. **Operations Research**, INFORMS, v. 48, n. 1, p. 38–49, 2000.

DECAN, A.; MENS, T.; CONSTANTINOU, E. On the impact of security vulnerabilities in the npm package dependency network. In: **Proceedings of the 15th International Conference on Mining Software Repositories**. [S.l.: s.n.], 2018. p. 181–191.

FREI, S. et al. Large-scale vulnerability analysis. In: **SIGCOMM workshop on Large-scale attack defense**. [S.l.: s.n.], 2006. p. 131–138.

HORAWALAVITHANA, S.; BHATTACHARJEE, A. et al. Mentions of security vulnerabilities on Reddit, Twitter and Github. In: **Intl. Conf. Web Intelligence**. [S.l.: s.n.], 2019. p. 200–207.

HU, W. et al. Open source software vulnerability propagation analysis algorithm based on knowledge graph. In: IEEE. **IEEE Intl. Conference on Smart Cloud**. [S.l.], 2019. p. 121–127.

HUANG, S.; TANG, H. et al. Text clustering on national vulnerability database. In: IEEE. **Computer engineering and applications**. [S.l.], 2010. v. 2, p. 295–299.

JACOBS, J. et al. Exploit prediction scoring system (EPSS). **arXiv preprint arXiv:1908.04856**, 2019.

JOH, H.; MALAIYA, Y. K. A framework for software security risk evaluation using the vulnerability lifecycle and cvss metrics. In: **Workshop on risk and trust in extended enterprises**. [S.l.: s.n.], 2010. p. 430–434.

JOHNSON, P. et al. Time between vulnerability disclosures. **Computers & Security**, Elsevier, v. 62, p. 278–295, 2016.

MELL, P.; SCARFONE, K.; ROMANOSKY, S. Common vulnerability scoring system. **IEEE Security & Privacy**, IEEE, v. 4, n. 6, p. 85–89, 2006.

MITRE. **Common Vulnerabilities and Exposures**. 2020. https://cve.mitre.org/.

National Vulnerability Database. **Calculadora CVSS do NVD**. 2022. https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator.

NORRIS, J. R. **Markov chains**. [S.l.]: Cambridge university press, 1998.

RUOHONEN, J. A look at the time delays in CVSS vulnerability scoring. **Applied Computing and Informatics**, Elsevier, v. 15, n. 2, p. 129–135, 2019.

RUOHONEN, J.; HYRYNSALMI, S.; LEPPÄNEN, V. Modeling the delivery of security advisories and CVEs. **Computer Science Information Systems**, v. 14, n. 2, p. 537–555, 2017.

SABOTTKE, C.; SUCIU, O.; DUMITRAș, T. Vulnerability disclosure in the age of social media: exploiting twitter for predicting real-world exploits. In: **24th {USENIX} Security Symposium ({USENIX} Security 15)**. [S.l.: s.n.], 2015. p. 1041–1056.

SHAHZAD, M.; SHAFIQ, M. Z.; LIU, A. X. A large scale exploratory analysis of software vulnerability life cycles. In: **Intl. Conf. Software Engineering**. [S.l.: s.n.], 2012. p. 771–781.

VANBERKEL, P. T.; PETER, T. Interacting hospital departments and uncertain patient flows: Theoretical models and applications. **Enschede: Ipskamp Drukkers BV**, Citeseer, 2011.

WANG, B. et al. Characterizing and modeling patching practices of industrial control systems. **Proceedings of the ACM on Measurement and Analysis of Computing Systems**, ACM New York, NY, USA, v. 1, n. 1, p. 1–23, 2017.

WU, D.; LIU, Y.; ROSS, K. Queuing network models for multi-channel p2p live streaming systems. In: IEEE. **IEEE INFOCOM 2009**. [S.l.], 2009. p. 73–81.

ZHANG, S.; OU, X.; CARAGEA, D. Predicting cyber risks through national vulnerability database. **Information Security Journal**, Taylor & Francis, v. 24, n. 4-6, p. 194–206, 2015.