UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Rodrigo Borges Machado

Análise de conteúdo sobre vulnerabilidades de segurança em redes sociais

Uberlândia, Brasil

2020

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Rodrigo Borges Machado

Análise de conteúdo sobre vulnerabilidades de segurança em redes sociais

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Rodrigo Sanches Miani

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informção

Uberlândia, Brasil

2020

Aos meus pais Hélio Antônio Machado e Maria Alice Borges da Silveira

Agradecimentos

Agradeço primeiramente a Deus, por possibilitar que eu caminhasse por todo esse caminho de luta com a cabeça sempre erguida, com força para passar pelos obstáculos e saúde para seguir lutando.

Aos meus pais por todo amor, carinho, incentivo e apoio que sempre me deram. Por sempre me incentivarem a buscar mais, a querer mais, e nunca parar de lutar. Por possibilitar uma educação boa e ensinar a ter a humildade que a vida pede.

Às minhas irmãs por todo carinho, apoio e compreensão pelo tempo que tive que abdicar para dedicar a minha formação.

Aos meus amigos pelo incentivo e compreensão durante toda essa jornada. Principalmente por saberem como me animar quando estava em momentos difíceis. Obrigado à Confraria por existir e me fazer tão bem.

À minha namorada Milena Arantes Bertolini que sempre esteve ao meu lado enquanto fiz esse trabalho. Obrigado pelo carinho e por todo amor que me deu. Desculpe pelos compromissos cancelados, teremos toda a vida para compensar. Obrigado por estar sempre ao meu lado.

A todos os professores que me incentivaram, desde os professores do meu ensino fundamental, médio e agora no ensino superior. Um obrigado especial aos professores do Instituto Federal do Triângulo Mineiro, que sempre me ajudaram mesmo depois de eu já ter formado no ensino médio.

Ao meu orientador, Professor Doutor Rodrigo Miani, pela paciência, prontidão e confiança. Obrigado por ser sempre um amigo desde as aulas na classe, até no stress desse trabalho. Obrigado por ter sempre um tempo para poder conversar, seja assunto pessoal ou profissional.

"Que todos os nossos esforços estejam sempre focados no desafio à impossibilidade. Todas as grandes conquistas humanas vieram daquilo que parecia impossível."

Charles Chaplin.

Resumo

Vulnerabilidades em sistema é um assunto que vem ganhando espaço no mundo da informática e da tecnologia. O objetivo desse trabalho é apresentar o quanto esse assunto tem evoluído em redes sociais, mais específico na rede do Twitter. Foi desenvolvido um sistema que tem o objetivo de buscar, filtrar, armazenar e analisar *tweets* que falam sobre vulnerabilidades da rede dentre os anos de 2015 e 2019. A conclusão leva a crer que o número vem crescendo de pessoas que comentam na rede sobre o assunto, assim como o número de vulnerabilidades em sistemas.

Palavras-chave: Vulnerabilidade de Segurança, Segurança da Informação, CVE, Twitter

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
CVE	Commom Vunerabilities and Exposures
CSV	Comma Separated Values
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
SQLite	Tecnologia de banco de dados desenvolvida em *C*.
XML	Extensible Markup Language

Sumário

1 Introdução

A Internet se tornou uma das principais evoluções da humanidade. A globalização é a prova de que todo o avanço funcionou e essa evolução foi necessária. Nesse contexto, a pesquisa sobre segurança da informação foi ampliada, principalmente devido à necessidade.

Quando se tem grandes sistemas dedicados a manter informações sensíveis, seja de pessoas físicas ou jurídicas, a segurança se vê necessária. O trabalho de assegurar que informações sejam mesmo sigilosas e íntegras é um dos desafios da informática. Há sempre a possibilidade de alguém sem direitos ter acesso a algum dado, ou mesmo que esse dado a ser salvo seja alterado, perdido ou modificado.

Riscos podem ser definidos como a probabilidade de alguma situação, seja ela física ou não, com certo potencial de causar danos, de ocorrer, em decorrência de exposição em um intervalo de tempo a alguma vulnerabilidade. Esta é definida como alguma fraqueza de um sistema, podendo envolver pessoas, processos ou tecnologias que podem ser exploradas para se ter algum ganho com o ato (PEOTTA; GONDIM, 2006).

A partir de uma vulnerabilidade há um risco, como já foi definido, que origina uma ameaça. Essa se define como qualquer circunstância ou evento com certo potencial de causar algum impacto sobre os princípios da segurança, a saber: confidencialidade, integridade ou disponibilidade da informação (PEOTTA; GONDIM, 2006).

As principais fontes de dados sobre incidentes de segurança do Brasil são o Centro de Atendimento a Incidentes de Segurança (CAIS), vinculado à Rede Nacional de Pesquisa (RNP) e o Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br). Os dados dessas instituições levam a acreditar que os incidentes de segurança apresentam um crescimento cada vez maior e de maior impacto. A revolução digital, desenvolvimento das tecnologias e o crescimento das vendas de dispositivos com acesso à rede são alguns dos fatores que explicam esse crescimento (MIANI, 2020).

A Figura 1 mostra em resumo o número de vulnerabilidades encontradas no dia 24/10/2019, na semana corrente à essa data, nos meses 10/2019 e 09/2019 e no ano de 2019. O objetivo é elencar a quantidade de vulnerabilidades que surgiram nesse período, seja diário, semanal, mensal e anual.

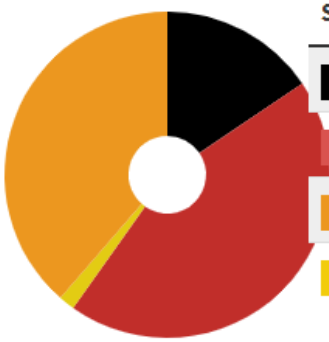
CVEs Received and Processed

Time Period	New CVEs Received by NVD	New CVEs Analyzed by NVD	Modified CVEs Received by NVD	Modified CVEs Re-analyzed by NVD
Today	20	23	19	1
This Week	162	298	239	13
This Month	1269	1423	7318	32
Last Month	1525	1457	878	38
This Year	13822	15008	12045	4978

Figura 1 – Vulnerabilidades anunciadas recentemente. Extraído de CVE (2018).

Ainda como exemplo, a Figura 2 apresenta a quantidade de vulnerabilidades já sumarizadas e interpretadas de acordo com sua complexidade e o quão críticas são. Logo, a Figura 1 quantifica as vulnerabilidades, enquanto a Figura 2 qualifica as mesmas.

CVSS V3 Score Distribution



Severity	Number of Vulns
CRITICAL	7956
HIGH	22658
MEDIUM	19753
LOW	820

Figura 2 – Número de vulnerabilidades já interpretadas no ano de 2019. Extraído de CVE (2018).

O número de vulnerabilidades que são encontradas em sistemas, nos últimos anos, cresceu de forma exponencial (NIST, 2018). Isso não significa que todas as vulnerabilidades descobertas foram exploradas. A grande quantidade de demanda para uma baixa quantidade de produção de reparos faz com que apenas vulnerabilidades julgadas como críticas sejam exploradas, uma vez que as restantes são deixadas de lado (UTO; MELO, 2009).

Os atacantes que encontram essas vulnerabilidades, em sua essência, mostram suas habilidades em redes sociais apresentando seus ataques e compartilhando alguns dados recolhidos. Muitas vezes disponibilizam até mesmo instruções de como fazer o ataque. Alguns pedem para outros usuários instalarem aplicativos em suas máquinas para ajudar nessas invasões, assim máquinas de usuários podem atuar como *botnets* auxiliando um ataque.

A discussão sobre as vulnerabilidades em redes sociais mostra como o assunto é tomado pelo público. Assim, pessoas passam a entender mais sobre sistemas e seus *bugs*, começam a identificar quais são as formas de se identificar vulnerabilidades e quais os passos para apresentá-las ao mundo para que sejam exploradas.

O número de pessoas que discutem temas como vulnerabilidades e *bugs* em sistemas têm crescido assim como o número de vulnerabilidades (CVE, 2018). O foco é conseguir reduzir o número de falhas em sistemas. A discussão do tema leva as pessoas a entenderem mais sobre o assunto, assim mais pessoas contribuem em encontrar falhas em sistemas. Além disso, essas discussões acabam apresentando vulnerabilidades que podem ser exploradas.

O objetivo principal desse trabalho é construir uma aplicação que coleta e organiza dados de vulnerabilidade de segurança que são discutidos no *Twitter*. Um objetivo secundário consiste em usar a aplicação desenvolvida para avaliar a forma com que as vulnerabilidades de segurança são mencionadas no *Twitter*. As seguintes questões serão investigadas:

- Qual o comportamento do número de *tweets* relacionados a vulnerabilidades de segurança nos últimos anos?
- Quais são os tipos de vulnerabilidades com maior incidência de discussão no *Twitter*?
- Qual é a razão entre vulnerabilidades de segurança presentes em redes sociais e vulnerabilidades descobertas?

Será criado, então, um sistema que fará uma busca na *API GetOldTweets* através de filtros pré selecionados afim de construir uma base de dados para verificação dos *tweets* a partir de tal busca. Assim, será possível responder essas questões

A monografia está organizada em cinco capítulos:

- Capítulo 1: descreve as motivações para esse trabalho, além da importância da pesquisa na área de segurança para o desenvolvimento da tecnologia.
- Capítulo 2: explica os conceitos e referências bibliográficas que serão utilizadas para o desenvolvimento do trabalho.
- Capítulo 3: descreve o sistema criado para a produção desse trabalho e apresenta quais tecnologias e técnicas foram utilizadas. Apresenta imagens do programa em execução e alguns resultados.
- Capítulo 4: apresenta os resultados do trabalho. Mostra a partir de tabelas e gráficos como ficou organizado os dados e quais seus padrões.
- Capítulo 5: apresenta a conclusão do trabalho apresentando o que pode ser de continuidade do mesmo.

2 Revisão Bibliográfica

Este capítulo apresenta um levantamento bibliográfico sobre segurança da informação e vulnerabilidades. Tem como objetivo definir conceitos e apresentar exemplos da utilização dos termos e sua usabilidade no mundo real. Serão listadas as definições e as técnicas utilizadas para efetuar o trabalho e como serão considerados os dados para sua utilização posterior. Também serão apresentadas as ferramentas a serem utilizadas para a busca da pesquisa.

2.1 Segurança da Informação

Segurança da Informação é o conjunto de orientações, normas, procedimentos, políticas e demais ações que têm por objetivo proteger o recurso da informação (FONTES, 2017). Assim, a segurança é o principal pilar de pesquisa desse trabalho. A segurança da informação envolve segurança de sistemas, redes, serviços, entre outros sistemas de informações.

Proteger a informação é responsabilidade de cada usuário na organização em que coopera e cabe à organização orientar seus colaboradores em relação à proteção da informação. Nesse contexto, então, é de responsabilidade da organizar traçar suas normas de segurança. Quando se utiliza sistemas com dados sensíveis pessoais, então, o próprio usuário deve definir suas regras em relação à proteção da informação.

De acordo com Spanceski (2004), uma das formas de proteger a informação é conhecer os pilares da área da segurança da informação. Esses, são considerados as divisões dessa ciência que juntos formam toda a segurança da informação. Os pilares são:

- Integridade: pilar que assegura que sistemas e informações serão modificados apenas nas condições previstas, assim impede que sejam manipuladas afetando os usuários. A integridade é a garantia da exatidão e completeza da informação e dos métodos de processamento (ABNT, 2005);
- Disponibilidade: garante que os dados, sistemas e serviços estejam disponíveis para aqueles que têm o direito de utilizá-los. A disponibilidade é a garantia de que os usuários autorizados obtenham acesso à informação e aos ativos correspondentes sempre que necessário (ABNT, 2005).
- Confidencialidade: esse pilar assegura que as informações não serão acessadas por agentes não autorizados. A confidencialidade é a garantia de que a informação é acessível somente por pessoas autorizadas a terem acesso (ABNT, 2005).

A integridade de uma mensagem A é o conceito de que não há nenhuma alteração em seu conteúdo mantendo sua estrutura original. Garantir sua integridade significa não permitir que a informação seja modificada, alterada ou mesmo destruída, que seja legítima e permaneça consistente (DANTAS, 2011).

Ocorre a não integridade de uma informação quando seu conteúdo é corrompido, falsificado, roubado ou destruído (DANTAS, 2011). Quando uma informação está íntegra, então seus dados são seguros e há a garantia que são originais.

A disponibilidade reporta a garantia de que uma informação está acessível. Deste modo, não se restringe apenas a algum sistema online estar apto a se conectar, mas uma informação que esteja ao alcance dos usuários destinatários. Quando não há disponibilidade significa que a informação não está disponível para ser utilizada, ou seja, está fora do alcance dos usuários destinatários (DANTAS, 2011). Para ter disponibilidade deve ser assegurando a garantia da leitura e acesso da informação no momento em que se necessita da mesma.

A confidencialidade é a certeza de que uma informação é acessível apenas para o remetente e para o destinatário, sendo que qualquer outro terceiro não consiga ter a informação encaminhada (DANTAS, 2011). A garantia da confidencialidade é deixar de que outro terceiro não tenha acesso à informação original e que não possa divulgá-la.

É natural que com a evolução e o desenvolvimento a segurança da informação tenha se focado muito na área de comunicação de dados e informação. Esse fato abre porta para outras pontas de atenção, como:

- Autenticidade: garante que a mensagem é do próprio remetente. É a garantia de que a informação é oriunda da fonte que lhe é atribuída e elaborada por quem tem autoridade para tal. Diz respeito à idoneidade da fonte, ou seja, digna de confiança (DANTAS, 2011);
- Confiabilidade: garante que a fonte emitente da mensagem é confiável e autêntica. É a garantia de que a informação é confiável, oriunda de uma fonte autêntica e que expressa uma mensagem verdadeira. Diz respeito ao conteúdo (DANTAS, 2011).
- Não repúdio: garante que a informação sempre chegará ao destino final e será sempre interpretada. É a garantida de que a informação chegará ao destino certo e não será repudiada. (DANTAS, 2011).
- Responsabilidade: é o agregado de responsabilidade de todos os envolvidos na criação, armazenamento e transporte da informação. É a coparticipação de responsabilidades por todos os que produzem, manuseiam, transportam e descartam a informação, seus sistemas e redes de trabalho. (DANTAS, 2011).

Com esse conjunto de propriedades presente na comunicação em rede (ABNT, 2005), o remetente tem a garantia de quem recebe suas informações é realmente o destinatário que se diz

ser. Garante ainda que a informação recebida está exatamente igual à enviada pelo remetente. Assegura também que nem o emissor nem o remetente neguem que houve comunicação.

2.2 Vulnerabilidades de Segurança

No contexto de segurança da informação, vulnerabilidade pode ser uma falha em um sistema ou serviço que permita realização e concretização de um ataque (MARTINELO; BELLEZI, 2014). Para que um atacante consiga explorar essa vulnerabilidade, ele precisa de um meio ou ferramentas que possibilitem seu ataque, assim ele conecta com a fraqueza do sistema ou serviço, essas ferramentas ou mesmo técnicas são chamadas de *exploits* (WHITEMAN; HERBERT, 2011).

A Figura 3 representa o ciclo de vida de uma vulnerabilidade. De modo geral, uma vulnerabilidade tem o seu descobrimento e, a partir disso, há os passos para solucioná-lo. Representando um sistema que esteja já em base de produção, a vulnerabilidade se destaca como um *bug*, mas não um simples *bug*, mas sim um que tenha potencial de afetar o sistema. No modo de produção do sistema, a equipe então desenvolverá uma versão que corrija aquele *bug* e liberará essa correção, seja como um instalador de sistema, para aplicativos *desktop* ou mesmo para mobile, ou então irá alterar os arquivos no servidor, caso seja um site ou serviço de API (*Application Programming Interface*).

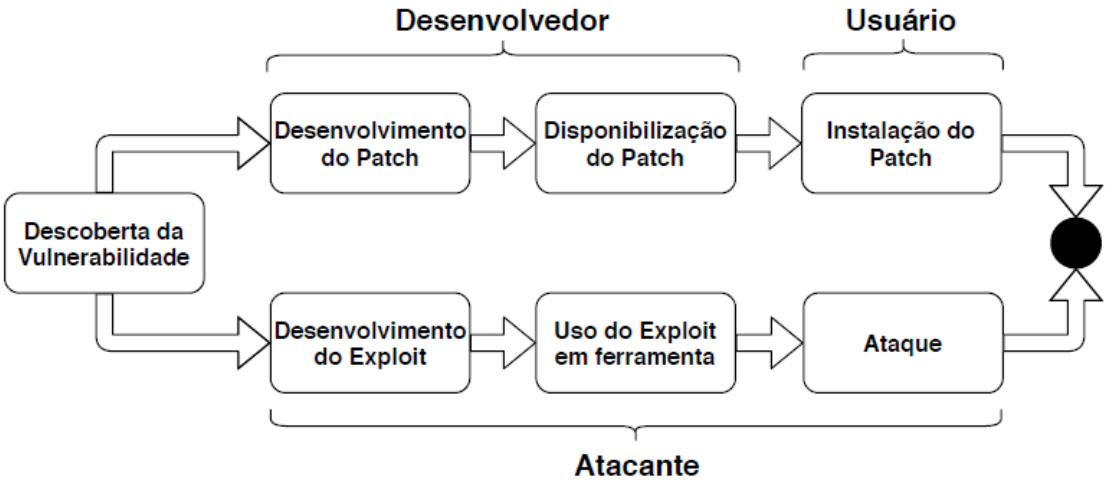


Figura 3 – Ciclo de vida de uma vulnerabilidade. Extraído de Xiao et al. (2018)

O ciclo de vida de um software pode ser vinculado com a parte de descoberta de vulnerabilidades, no qual às vezes é até previsto no escopo do projeto. Quando se encontra algum *bug* que leve a algum ponto crítico de atenção, a equipe de desenvolvimento do sistema, serviço ou software, tem a responsabilidade de disponibilizar uma nova versão, tal que a mesma

possua a correção dessa vulnerabilidade. Alguns exemplos de problemas que podem originar vulnerabilidades de segurança são (PFLEEGER; PFLEEGER; MARGULIES, 2015):

- Falhas no software ou no Hardware. Desde parâmetros externos, como fator ambiental; a fator interno como falta de memória.
- Erros de usabilidade ou falha de administração.
- Erros no projeto e de implementação.

Em sumo, não é possível a construção de uma lista que contenha todas as possíveis situações que podem se transformar em vulnerabilidades (PFLEEGER; PFLEEGER; MARGULIES, 2015). Logo, conclui-se que as situações não são finitas, mas há uma tentativa de mensuração.

Depois de lançado um software inicia-se a fase de descoberta de vulnerabilidades. Essa fase não é composta apenas pela equipe de desenvolvimento do sistema/serviço, mas também dos usuários. Atacantes entram nesse contexto até mesmo como aliados para descoberta de aberturas. O problema existe quando eles tentam tirar proveito desse cenário. Quando descoberto vulnerabilidades, essas podem ser divulgadas ao público através de fóruns públicos ou mesmo quando liberado correção de tal vulnerabilidade.

Redes sociais são um caminho também de divulgação de problemas em sistemas e serviços. O *Twitter*, base desse trabalho, tem um número de dados muito grande referente à discussão de problemas como esse, que será apresentado posteriormente.

Vários órgãos e departamentos ligados a segurança da informação como o NIST (*National Institute of Standards and Technology*), FIRST (*Forum of Incident Response and Security Teams*), CERT (*Computer Emergency Response Team*) entre outros, se juntaram para criar um padrão para pontuação/mensuração de vulnerabilidades de software chamado de CVSS [Schiffman (2005)] (PEOTTA; GONDIM, 2006).

Alguns dos dados sobre vulnerabilidades e falhas de segurança são apresentados ao público através de sites específicos a esse fim. Esses sites armazenam informações sobre tais incidentes e possibilitam buscas a partir desses dados. Sites como as organizações NVD (NIST, 2018), Secunia (SECUNIA, 2009), US-CERT (CERT, 2018) e *Open Source Vulnerability Database* (OSVDB, 2002). Para o presente trabalho será utilizado o NVD (NIST, 2018) e o US-CERT (CERT, 2018).

O NVD *(National Vulnerability Database)* (NIST, 2018) é uma referência na coleta de dados sobre vulnerabilidade e é sincronizado com o CVE *(Common Vulnerabilities and Exposures)* (CVE, 2018). Enquanto o CVE cadastra vulnerabilidades, o NVD categoriza e avalia os riscos delas. Ademais, o CVE apresenta de modo simples detalhes sobre a vulnerabilidade, *exploits* e se há correção. Ele é um banco de dados público, logo qualquer usuário tem acesso

à sua base de conhecimento. O principal gestor do portal é o MITRE *(Massachusetts Institute of Technology's Digital Computer Laboratory)*, que além de disponibilizar a informação tem a intenção de padronizar esses dados com a ajuda do NVD (PEOTTA; GONDIM, 2006).

Pelo site do NVD, assim como no CVE, é possível listar as vulnerabilidades. No site do CVE a organização dessa informação não tem uma boa visualização, enquanto no NVD é listado de forma fácil e interpretável. O site do CVE fornece possibilidade de baixar um arquivo contendo as vulnerabilidades filtrando por ano e possibilitando vários tipos de arquivos diferentes, como apresentado na Figura 4.

Download Formats

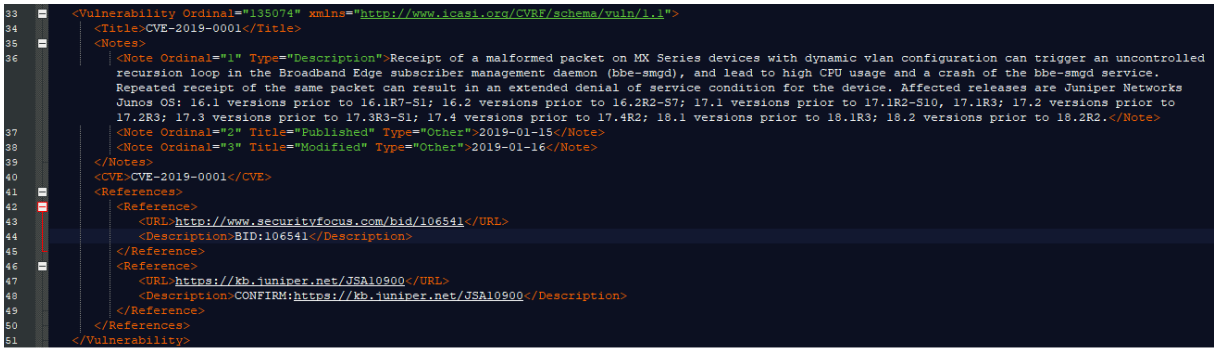
CVE downloads data last generated: 2019-10-22

Format	Unix compressed (.Z)	Gzipped (.gz)	Raw	Other
Comma Separated	allitems.csv.Z	allitems.csv.gz	allitems.csv	NOTE: suitable for import into spreadsheet programs
HTML	allitems.html.Z	allitems.html.gz	allitems.html	
Text	allitems.txt.Z	allitems.txt.gz	allitems.txt	
XML	allitems.xml.Z	allitems.xml.gz	allitems.xml	cve_1.0.xsd

CVRF	All Entries	By Year
(Learn more about CVE and CVRF)	All entries - Raw (cvrf.xml)	CVE-2019-xxxxxx entries CVE-2018-xxxxxx entries CVE-2017-xxxxxx entries CVE-2016-xxxxxx entries CVE-2015-xxxxxx entries CVE-2014-xxxx entries CVE-2013-xxxx entries CVE-2012-xxxx entries CVE-2011-xxxx entries CVE-2010-xxxx entries

Figura 4 – Lista de opções de *donwload* de arquivos contendo vulnerabilidades

Esses arquivos para download contém as vulnerabilidades do período selecionado. Caso selecionado o tipo, na primeira linha da tabela, é baixado todo um arquivo com aquela extensão selecionada e no formato desejado. Nesse arquivo será colocado todas as vulnerabilidades desde 1999. Quando baixado uma lista de vulnerabilidades de algum ano em específico, é carregado uma página web em XML com os dados daquele ano. A Figura 5 mostra a configuração de um arquivo XML de exemplo.

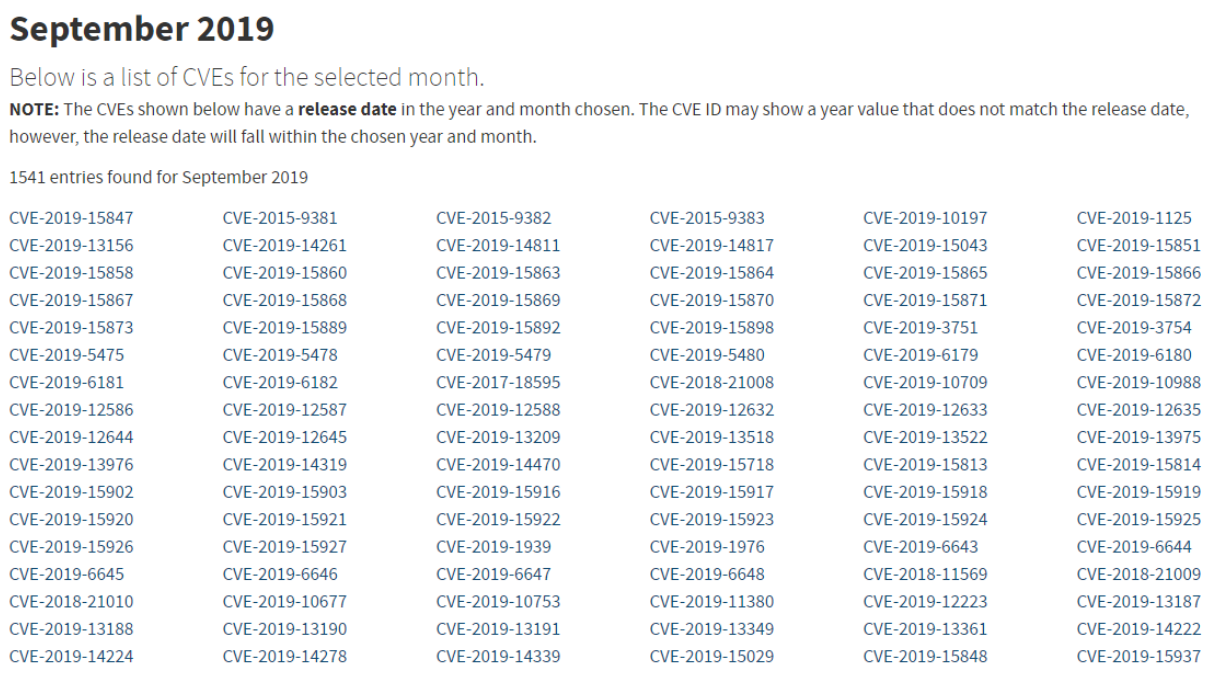


```
<Vulnerability Ordinal="135074" xmlns="http://www.icasi.org/CVRF/schema/vuln/1.1">
  <Title>CVE-2019-0001</Title>
  <Notes>
    <Note Ordinal="1" Type="Description">Receipt of a malformed packet on MX Series devices with dynamic vlan configuration can trigger an uncontrolled
    recursion loop in the Broadband Edge subscriber management daemon (bbe-smgd), and lead to high CPU usage and a crash of the bbe-smgd service.
    Repeated receipt of the same packet can result in an extended denial of service condition for the device. Affected releases are Juniper Networks
    Junos OS: 16.1 versions prior to 16.1R7-S1; 16.2 versions prior to 16.2R2-S7; 17.1 versions prior to 17.1R2-S10, 17.1R3; 17.2 versions prior to
    17.2R3; 17.3 versions prior to 17.3R3-S1; 17.4 versions prior to 17.4R2; 18.1 versions prior to 18.1R3; 18.2 versions prior to 18.2R2.</Note>
    <Note Ordinal="2" Title="Published" Type="Other">2019-01-15</Note>
    <Note Ordinal="3" Title="Modified" Type="Other">2019-01-16</Note>
  </Notes>
  <CVE>CVE-2019-0001</CVE>
  <References>
    <Reference>
      <URL>http://www.securityfocus.com/bid/106541</URL>
      <Description>BID:106541</Description>
    </Reference>
    <Reference>
      <URL>https://kb.juniper.net/JSA10900</URL>
      <Description>CONFIRM:https://kb.juniper.net/JSA10900</Description>
    </Reference>
  </References>
</Vulnerability>
```

Figura 5 – Arquivo XML exemplificando uma vulnerabilidade de janeiro de 2019

Como é demonstrado na Figura 5 a estrutura do XML permite visualizar os campos que detalham a vulnerabilidade. Assim, o arquivo XML com as vulnerabilidades possibilita uma busca por códigos e até mesmo textos nas ocorrências, o que será utilizado no trabalho a fim de buscar os resultados com as postagens no *Twitter*.

O site da NVD, entretanto, tem a possibilidade de listagem das vulnerabilidades através da seleção de um mês de algum ano, como apresentado na Figura 6, filtrando o mês de setembro de 2019. Assim, é possível ter um filtro melhor caso esteja procurando apenas informações para uma leitura em específico.



September 2019

Below is a list of CVEs for the selected month.

NOTE: The CVEs shown below have a **release date** in the year and month chosen. The CVE ID may show a year value that does not match the release date, however, the release date will fall within the chosen year and month.

1541 entries found for September 2019

CVE-2019-15847	CVE-2015-9381	CVE-2015-9382	CVE-2015-9383	CVE-2019-10197	CVE-2019-1125
CVE-2019-13156	CVE-2019-14261	CVE-2019-14811	CVE-2019-14817	CVE-2019-15043	CVE-2019-15851
CVE-2019-15858	CVE-2019-15860	CVE-2019-15863	CVE-2019-15864	CVE-2019-15865	CVE-2019-15866
CVE-2019-15867	CVE-2019-15868	CVE-2019-15869	CVE-2019-15870	CVE-2019-15871	CVE-2019-15872
CVE-2019-15873	CVE-2019-15889	CVE-2019-15892	CVE-2019-15898	CVE-2019-3751	CVE-2019-3754
CVE-2019-5475	CVE-2019-5478	CVE-2019-5479	CVE-2019-5480	CVE-2019-6179	CVE-2019-6180
CVE-2019-6181	CVE-2019-6182	CVE-2017-18595	CVE-2018-21008	CVE-2019-10709	CVE-2019-10988
CVE-2019-12586	CVE-2019-12587	CVE-2019-12588	CVE-2019-12632	CVE-2019-12633	CVE-2019-12635
CVE-2019-12644	CVE-2019-12645	CVE-2019-13209	CVE-2019-13518	CVE-2019-13522	CVE-2019-13975
CVE-2019-13976	CVE-2019-14319	CVE-2019-14470	CVE-2019-15718	CVE-2019-15813	CVE-2019-15814
CVE-2019-15902	CVE-2019-15903	CVE-2019-15916	CVE-2019-15917	CVE-2019-15918	CVE-2019-15919
CVE-2019-15920	CVE-2019-15921	CVE-2019-15922	CVE-2019-15923	CVE-2019-15924	CVE-2019-15925
CVE-2019-15926	CVE-2019-15927	CVE-2019-1939	CVE-2019-1976	CVE-2019-6643	CVE-2019-6644
CVE-2019-6645	CVE-2019-6646	CVE-2019-6647	CVE-2019-6648	CVE-2018-11569	CVE-2018-21009
CVE-2018-21010	CVE-2019-10677	CVE-2019-10753	CVE-2019-11380	CVE-2019-12223	CVE-2019-13187
CVE-2019-13188	CVE-2019-13190	CVE-2019-13191	CVE-2019-13349	CVE-2019-13361	CVE-2019-14222
CVE-2019-14224	CVE-2019-14278	CVE-2019-14339	CVE-2019-15029	CVE-2019-15848	CVE-2019-15937

Figura 6 – Vulnerabilidades reportadas pelo NVD em setembro de 2019

A Figura 6 mostra parte das 1541 vulnerabilidades divulgadas para o mês de setembro de 2019. Quando selecionado algum CVE uma página é apresentada mostrando os detalhes daquela vulnerabilidade. Como um exemplo, o CVE-2019-15847 é apresentado na Figura 7.

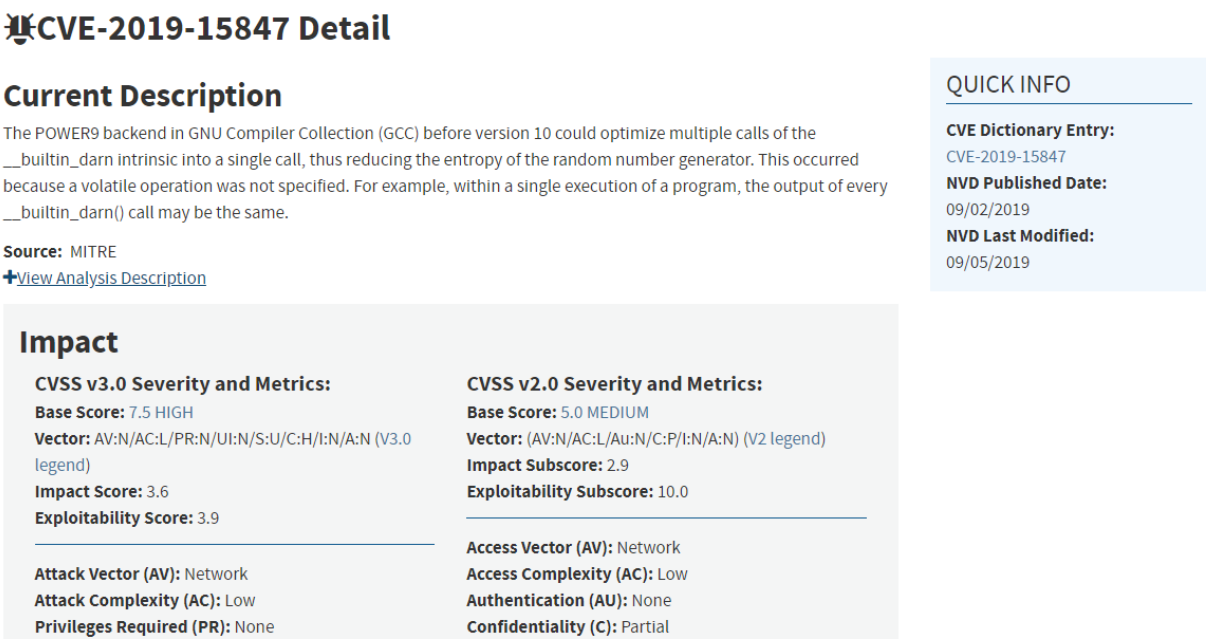


Figura 7 – Vulnerabilidade CVE-2011-4042

A Figura 7 apresenta os detalhes da vulnerabilidade CVE-2019-15847. Descrição *(Description)*, a tabela de impacto *(Impact)* e a data de publicação da vulnerabilidade *(NVD Published Date)*, são alguns dos campos que são informados. Esses dados são oriundos do site da CVE e os mesmos arquivos XML que são possíveis de serem baixados são a base utilizada para a apresentação da informação. Logo, o site tem o propósito de apresentar a informação de uma forma mais legível e atraente para o usuário.

O trabalho em questão utiliza esses dados do CVE como uma base de comparação de assuntos que estão postados no *Twitter*. Dá-se, então, a estima para a listagem XML que possibilita uma mineração de dados em cima dos resultados obtidos do *Twitter*.

2.3 Trabalhos Relacionados

Esta seção tem o objetivo de descrever alguns trabalhos que têm relação ao tema principal desse projeto: discussão sobre vulnerabilidades em redes sociais. Os trabalhos Sabottke, Suciu e Dumitraș (2015), Correia (2016), BATISTA (2007) e Kroth (2018) serão apresentados posteriormente com o objetivo de descrevê-los e vinculá-los ao tema central desse trabalho.

O número de vulnerabilidades tem crescido a cada ano, assim como é apresentado na Figura 6, onde apenas no mês de setembro de 2019 é apresentado muitas novas incidências de

bugs e *backdoors*. Esse crescimento é exponencial. Um outro exemplo que prova isso é que em 2014 foi o marco da primeira aparição de um CVE de 5 dígitos, pois o banco de dados do CVE, que atribui identificadores exclusivos às vulnerabilidades, adotou um novo formato que não limita mais o número de IDs do CVE a 10.000 por ano (SABOTTKE; SUCIU; DUMITRAș, 2015), assim é possível notar a quantidade crescente das vulnerabilidades, pois a nova métrica é incremental.

Um dos problemas é que esse crescimento de novas vulnerabilidades é muito maior que a capacidade de correções pela parte dos criadores (BATISTA, 2007). Nesse aspecto, tem-se a classificação qualitativa das vulnerabilidades encontradas. Essa classificação tem por objetivo identificar quais são as vulnerabilidades com um nível de prioridade maior (SABOTTKE; SUCIU; DUMITRAș, 2015).

Com a ascensão do número de vulnerabilidades é necessário que haja aumento também do número de pessoas a combatê-las. Nesse processo, o primeiro passo é identificá-las. As redes sociais estão nesse contexto, pois apresentam um meio que possibilita discussão sobre o assunto em questão. O *Twitter*, foco desse trabalho, é a rede social onde o assunto tem crescido, principalmente devido ao aumento de número de usuários, como apresentado na Figura 8. Em virtude das informações dos usuários obtidas nessa rede social é possível, de forma automática, listar quais são as principais vulnerabilidades discutidas no momento e o quão importante elas são de serem sanadas e exploradas (CORREIA, 2016).

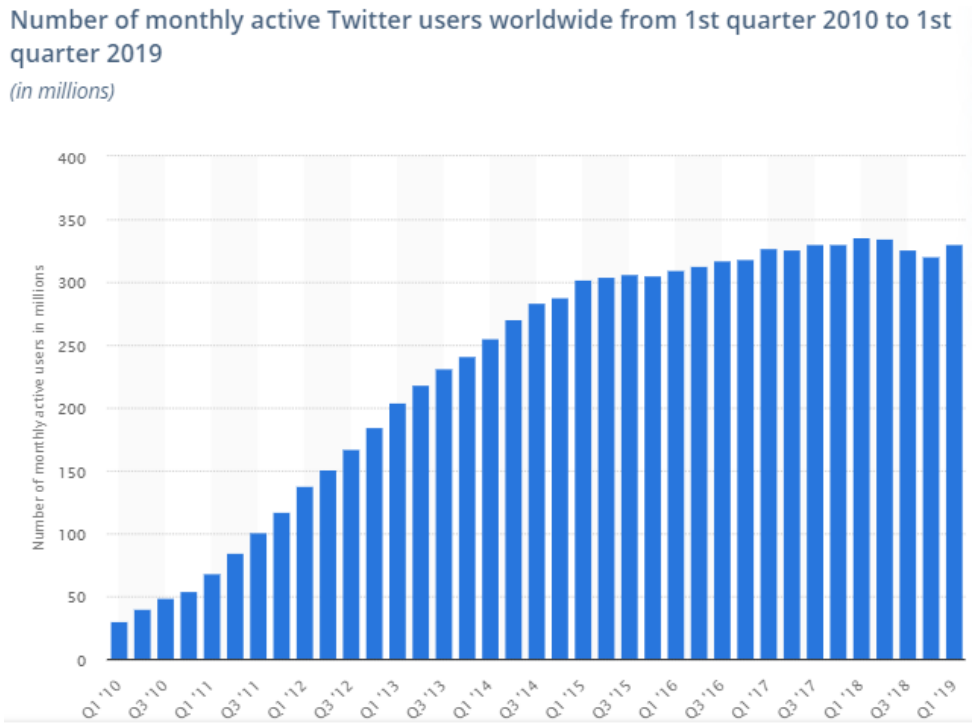


Figura 8 – Número de usuários na rede do *Twitter* entre os anos de 2010 e 2019. Extraído de Statista (2019)

Como é apresentado na Figura 8, conclui-se que o número de usuários na rede é crescente e começou a ter um certo ponto de estabilidade nos últimos anos. Outrossim, o número de usuários é muito grande, mostrando que essa rede social gera impacto na vida de muitas pessoas.

Sabottke, Suciu e Dumitraș (2015) é um trabalho que apresenta uma proposta de identificar, através de buscas no *Twitter*, vulnerabilidades discutidas nessa rede social. Tem o objetivo de explorar qualitativa e quantitativamente os dados de vulnerabilidades disseminadas na rede (SABOTTKE; SUCIU; DUMITRAș, 2015).

Muitas vulnerabilidades são encontradas e anunciadas diariamente. A capacidade de corrigir esses *bugs* é inversamente proporcional a esse crescimento de vulnerabilidades. Sabottke, Suciu e Dumitraș (2015) então tenta propor uma forma de mensura das vulnerabilidades discutidas no *Twitter*, assim consegue levantar quais os problemas devem ser focados primeiramente. Levanta também quais são os usuários que mantém essas discussões. Ademais, muitas das vulnerabilidades não têm grande impactos em sistemas, muitas vezes até sendo um falso-positivo, então o trabalho tem o objetivo de levantar apenas as vulnerabilidades que realmente devem ser corrigidas.

Sabottke, Suciu e Dumitraș (2015) buscam criar um sistema que filtre *tweets* no *Twitter* através de alguma palavra chave e, a partir do resultado da busca, listar quais desses *tweets* falam sobre vulnerabilidades existentes e quais são exploráveis no mundo real. Para isso, o objetivo é criar um sistema que faça essa busca e essa sumarização qualitativa e quantitativa dos *tweets*.

Utilizando outras bases de segurança, o sistema se conecta à bases públicas, como *ExplitBD*, *Microsoft Security Advisories e Symantec*, assim consegue filtrar as vulnerabilidades e buscar quais já foram exploradas. Essas bases de dados têm o objetivo de listar de forma organizada as vulnerabilidades com maior nível de atenção.

Assim, o sistema monta sua base de dados através do *Twitter*. Faz ainda a classificação através das bases públicas de listagens de vulnerabilidades, de quais desses *bugs* devem ser tratados e faz o levantamento do quanto eles aparecem em publicações na rede social.

Correia (2016) propõe um trabalho que utiliza de aprendizagem automática para conseguir identificar os *tweets* da rede social do *Twitter* que estão correlacionados a problemas de segurança de softwares e também relacionados a vulnerabilidades. Assim como Sabottke, Suciu e Dumitraș (2015), tem o objetivo de listar e filtrar de forma qualitativa e quantitativa as vulnerabilidades encontradas.

O autor tem como base o estudo de aprendizagem de máquina, uma vez que utiliza essa tecnologia com o objetivo de filtragem dos dados e, a partir dos resultados, melhorar a busca criando novos parâmetros para a pesquisa. O trabalho ainda lista uma série de restrições que a API utilizada para coleta dos dados possui, apresentando seus resultados apesar das limitações. Este sistema é composto por duas fases, a fase de treino, onde é necessário o filtro humano

para que o primeiro filtro seja feito; e a fase de operação, onde o sistema trabalha por sozinho identificando seus filtros.

Para a primeira parte a necessidade da interferência humana se dá devido à natureza das técnicas de aprendizagem automáticas supervisionadas (CORREIA, 2016). Posterior a isso, o sistema consegue, através de aprendizagem, filtrar automaticamente os *tweets* de interesse.

Essa fase inicial tem o objetivo de, depois de recolhidos os *tweets*, seja classificados por um analista quais são os dados relevantes em relação à ameaça de segurança à infraestrutura de TI. Após essa rotulação, o próprio sistema com os algoritmos de aprendizagem passa a conseguir um melhor resultado na obtenção de dados (CORREIA, 2016).

Na fase de operação o sistema funciona de forma automática. Após o treino, o sistema consegue identificar melhor os filtros dos *tweets*. Assim, apresenta uma lista com os *tweets* selecionados que serão analisados por um analista posteriormente.

BATISTA (2007) estuda e apresenta dados sobre segurança em softwares e o quão difícil é de se abordar o tema colocando métricas de segurança, uma vez que nos dias atuais não é possível apresentar uma métrica capaz de indicar o nível de segurança que o software em desenvolvimento terá. Esse trabalho tem o objetivo de comprovar que não é possível colocar uma métrica assertiva, colocando a responsabilidade de segurança antes de se distribuir o software e assim sempre lançar um software fidedigno.

Com o objetivo de apresentar alguns métodos de avaliação quantitativa, apontando suas falhas, o trabalho propõe áreas de pesquisa para o estudo de métrica de segurança de software, para assim melhorar a avaliação. Desse modo, consegue-se a perspectiva do quão rentável é o investimento em segurança de software, a capacidade e competência de combater vulnerabilidades e como os riscos de segurança estão sendo gerenciados.

O trabalho define segurança como algo interpretativo e relativo. Quando se tem um cenário em que se coloca o nível de segurança X para um software, é relativo que a segurança do sistema tem que atendê-lo no nível X. Como exemplo, um sistema de banco necessita de uma segurança alta em seu software, uma vez que um sistema de uma loja pequena de roupas de uma cidade de interior não precisa da mesma segurança. A métrica de segurança de um sistema, então, advém de sua necessidade de segurança e contexto de utilização.

Kroth (2018) é um trabalho que busca identificar vulnerabilidades em serviços na Internet. Esse trabalho filtra vulnerabilidades que já são conhecidas em outros sistemas, então busca por ela em outros serviços web que podem apresenta-las da mesma forma.

O autor define que as vulnerabilidades disponíveis na rede advém de displicência por parte das organizações criadoras dos sistemas. Por parte delas, o investimento em segurança não se baseia em resultado, sendo um custo sem retorno. Assim, os sistemas não têm um teste de invasão por uma equipe especializada, o que deixa ao público uma dúvida sobre o quão seguro o sistema é. O problema maior se encontra com o público que não conhece sobre segurança e

disponibilizam seus dados em sistemas desse porte, que não têm o *"selo de segurança"*.

O trabalho se baseia em delimitar a conexão da rede da Internet para uma rede LAN. Assim, o objetivo é identificar vulnerabilidade já conhecidas em sistemas em geral, contabilizando o número de sistemas que não se atualizaram, ou mesmo foram feitos, sob as medidas de segurança já definidas (KROTH, 2018).

Os trabalhos citados têm focos diferentes nas perspectivas de segurança e objetivos diferentes. Os trabalhos conduzidos por Sabottke, Suciu e Dumitraș (2015) e o Correia (2016) são os que mais se aproximam com o presente trabalho, uma vez que faz também uma busca em uma rede social em cima de palavras chaves e as explora para definição de vulnerabilidades em sistemas. Porém, o trabalho proposto por BATISTA (2007) se relaciona com o projeto sendo uma porta de caminho sobre o estudo de segurança de software, tendo sua base na métrica do quão um software pode ser mensurado como seguro, fazendo uma busca sobre o que seria seguro na definição de segurança de sistemas. Por fim, o trabalho apresentado por Kroth (2018) também se destaca ao buscar vulnerabilidades já conhecidas pela sociedade em sistemas públicos, sendo essas vulnerabilidades já discutidas em redes sociais também.

3 Desenvolvimento

Neste capítulo será apresentada a metodologia usada no decorrer do trabalho. Serão demonstrados e exemplificados os sistemas criados no projeto e como foram desenvolvidas as etapas para criação do mesmo. Será explicado também as tecnologias envolvidas nos sistemas.

3.1 Ferramentas Desenvolvidas

Para o desenvolvimento do projeto foi desenvolvido um sistema batizado de *Scanner*. Esse sistema é responsável por fazer um filtro no *Twitter* em suas publicações.

O *Scanner* é um sistema distribuído dividido em três módulos distintos:

- *TwitterSearch*: módulo que fornece um formulário para preenchimento dos filtros a serem utilizados na busca do *Twitter*;
- *Tweet*: módulo que faz conexão com a API de comunicação com o *Twitter* enviando os filtros da busca e interpretando seu retorno em uma base de dados;
- *APIConectorJSON*: módulo que organiza a base de dados de retorno afim de armazenar a informação em um banco de dados, um arquivo XML, um arquivo CSV e um arquivo JSON.

A Figura 9 apresenta o modelo de relacionamento entre os módulos do sistema.

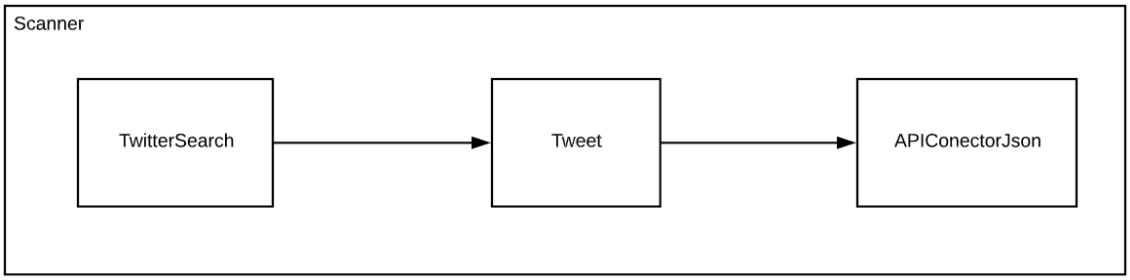


Figura 9 – Modelo de relacionamento dos módulos do sistema.

Como é expresso na imagem, o *Scanner* se comunica em três partes (módulos) gerando o resultado esperado. Essa comunicação é feita através da transição de dados entre cada uma das etapas, como será descrito posteriormente.

O *Scanner* funciona de modo que o usuário faça sua busca no *Twitter* e tenha seu resultado armazenado em alguma base de dados escolhido previamente pelo próprio usuário. Então,

utilizando esse sistema, o usuário tem o total acesso dos *tweets* de resultado da sua busca, seja em um arquivo XML, ou mesmo em um JSON. A busca no *Twitter* é feita a partir da API do *GetOldTweets* (COMMUNITY, 2019).

Para esse trabalho, o objetivo é listar as vulnerabilidades discutidas na rede social *Twitter*. A partir dos trabalhos similares a esse, foram utilizados filtros já pré selecionados para encontrar essas vulnerabilidades. Com esses resultados, foi feito um estudo dos dados desses *tweets*, assim como dos usuários que os postam.

Também foi criado um sistema de busca que tem o objetivo de, a partir dos resultados do *Scanner*, verificar quais *tweets* estão realmente relacionados com alguma vulnerabilidade presente no site do *NVD*. Esse sistema foi batizado de *FindWords*. Ele será descrito nas seções posteriores.

Será apresentado a seguir a especificação de cada módulo do sistema incluindo a descrição da API utilizada no trabalho.

3.1.1 *TwitterSearch*

Esse é um módulo cujo objetivo é acionar os demais organizando a informação dos filtros de entrada para a busca dos dados. Desenvolvido em C#, o sistema apresenta um formulário para preenchimento dos filtros desejados e, após preenchido o formulário pelo usuário, cria um arquivo JSON que servirá de entrada para os outros módulos como filtro. O próprio *TwitterSearch* já faz a chamada dos outros aplicativos necessários.

A Figura 10 apresenta a primeira parte da tela principal do sistema.

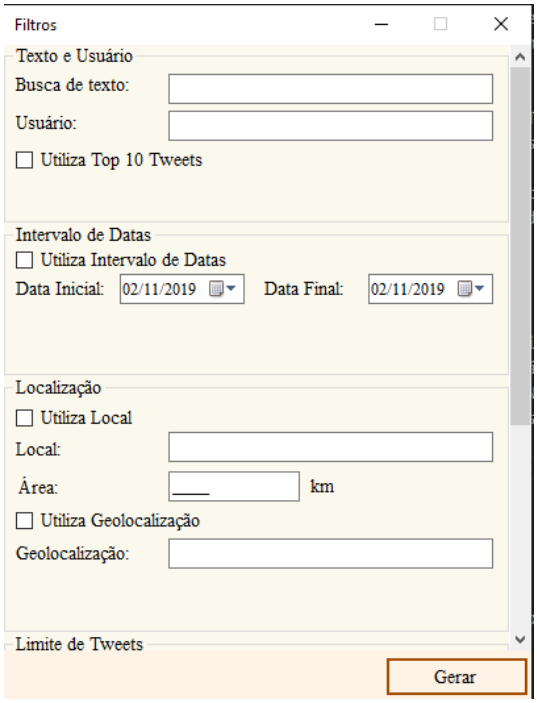


Figura 10 – Parte da tela principal do sistema

A seguir cada um dos campos da tela principal do *TwitterSearch* apresentado na Figura 10 será descrito:

- Texto e Usuário:
  - Busca de Texto: essa busca irá filtrar no *Twitter* publicações que, em seu conteúdo, tenham a sentença digitada. Vale ressaltar que essa palavra será comparada de forma limpa e completa, ou seja, o texto para ser considerado igual deve ter a mesma palavra em seu conteúdo, não apenas parte dela.
  - Usuário: irá pesquisar na rede social publicações do usuário digitado. Esse usuário é o nome que o mesmo utiliza na rede.
  - Utiliza Top 10 *Tweets*: essa opção selecionará os Top 10 *tweets* do usuário digitado na opção anterior. Esse top 10 significa os 10 *tweets* mais importante daquele usuário, ou seja, os que faram mais *retweetados* e curtidos.
- Intervalo de Datas
  - Utiliza Intervalo de datas: essa opção ativa o intervalo de datas. Se selecionada, o sistema irá filtrar a data da publicação do *tweet* no intervalo das datas selecionadas nos combos.
  - Data Inicial: data inicial a se considerar os *tweets* do filtro. A data inicial é incluída no filtro também, ou seja, *tweets* publicados no dia selecionado também serão retornados.

  - Data Final: data final a se considerar os *tweets* do filtro. A data final não é incluída no filtro, ou seja, *tweets* publicados no dia selecionado não serão retornados.

- Localização

  - Utiliza Local: essa opção ativa o filtro de localização onde o *tweet* foi publicado. Essa opção não considera a localização do usuário que postou o *tweet*, mas sim a localização que estava no momento que que foi publicado.
  - Local: esse filtro pode considerar cidade, estado, país ou mesmo região. Um exemplo de utilização seria: Berlin, Germany.
  - Área: perímetro a se considerar a partir do local selecionado. Uma opção de entrada seria 50km, então utilizando a opção anterior, de Berlin, esse filtro contataria até 50km em volta dessa cidade.
  - Utiliza Geolocalização: essa opção permite que o usuário utiliza geolocalização para filtrar os *tweets*. assim como o Local, a geolocalização irá identificar a partir do local em que o *tweet* foi publicado.
  - Geolocalização: entrada de latitude e longitude. Um de entrada exemplo seria: 55.75, 37.61. A Área, já mencionada, também funciona se preenchida assim como a geolocalização.

Essa é a primeira tela do sistema. Esses filtros colocados são os principais e comumente mais utilizados. O trabalho Sabottke, Suciu e Dumitraș (2015) utiliza filtro de texto e data para obtenção dos resultados. Em sua pesquisa, os dados têm por base o ano de 2015 filtrando a palavra CVE, que é a sigla utilizada para filtro de vulnerabilidades (CVE, 2018).

A Figura 11 apresenta a segunda parte da tela principal do sistema.

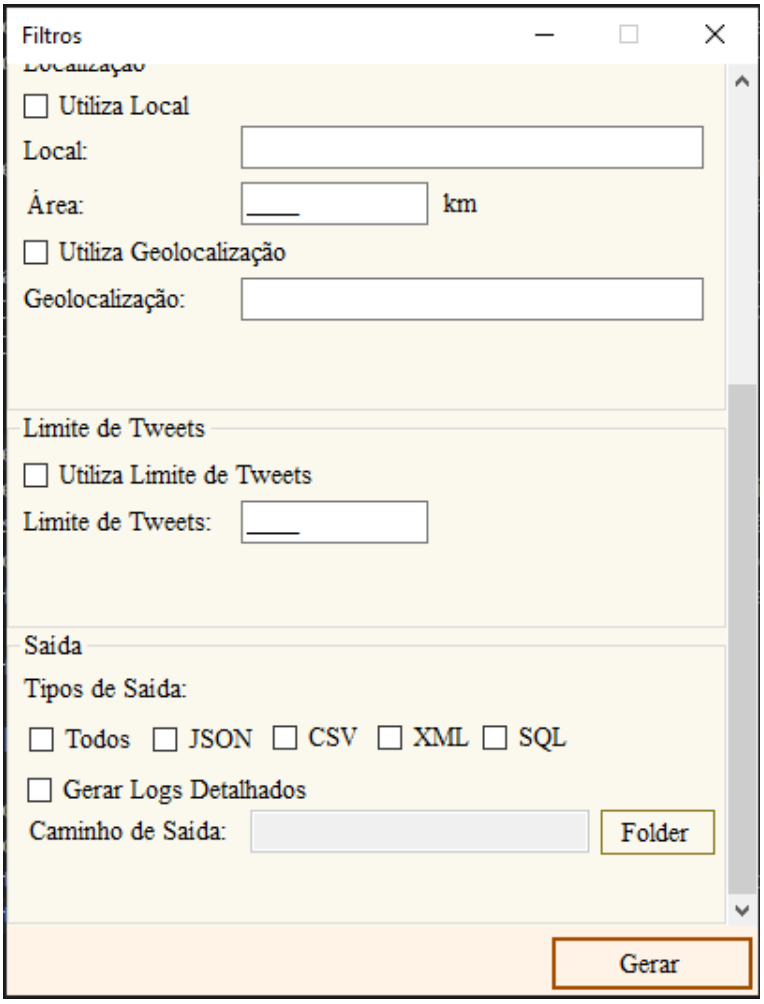


Figura 11 – Segunda Parte da tela principal do sistema

A seguir, cada um dos campos da segunda tela do *TwitterSearch* apresentado na Figura 11 será descrito:

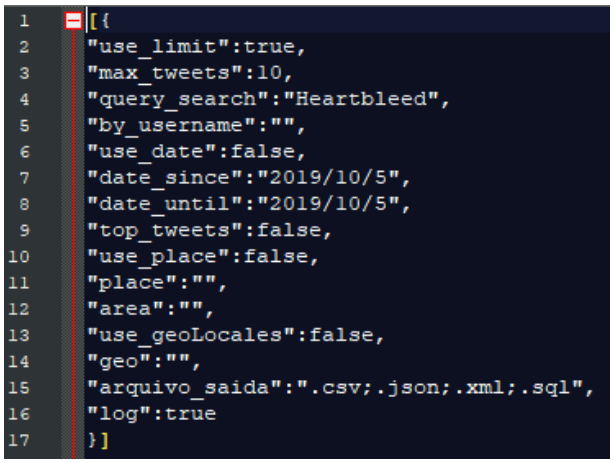
- Limite de Tweets
  - Utiliza Limite de *Tweets*: essa opção valida se vai limitar o número de *tweets* no retorno da busca.
  - Limite: número de *Tweets* máximo de retorno da consulta.
- Saída
  - Tipos de Saída: essa opção seleciona qual o tipo de saída do programa: JSON, CSV, XML, SQL.
  - Gerar Logs Detalhados: essa opção faz com que os sistemas gerem um log detalhado sobre suas atividades.

– Caminho de Saída: essa opção permite que o usuário selecione onde os arquivos de resultado devem ser copiados.

Esse módulo corresponde a primeira parte do trabalho, cujo objetivo é apenas apresentar o formulário e, a partir do preenchimento dele, gerar um arquivo JSON de configuração de filtro de entrada para os outros módulos do sistema. Assim, sendo o sistema final um sistema distribuído onde todas as pontas utilizam de informações compartilhadas, não dá para apontar um módulo como o principal.

3.1.2 *Tweet*

Esse módulo foi desenvolvido em *Python* e tem o objetivo de fazer a conexão com a API do *GetOldTweets* (COMMUNITY, 2019). Para sua execução, ele precisa de um arquivo de entrada que contenha as informações dos filtros que será colocado na busca dos *tweets*. A Figura 12 ilustra um exemplo de um arquivo de entrada desse módulo. O *Tweet*, citado anteriormente, é o módulo que tem o papel de gerá-lo.



```
[{
"use_limit":true,
"max_tweets":10,
"query_search":"Heartbleed",
"by_username":"",
"use_date":false,
"date_since":"2019/10/5",
"date_until":"2019/10/5",
"top_tweets":false,
"use_place":false,
"place":"",
"area":"",
"use_geoLocales":false,
"geo":"",
"arquivo_saida":".csv;.json;.xml;.sql",
"log":true
}]
```


Figura 12 – Exemplo de entrada do programa *Tweet*.

Esses filtros presentes na Figura 12 são os enviados ao *GetOldTweets*(COMMUNITY, 2019) para obter o retorno.

O sistema conta ainda com a opção de *log*. Na Figura 12 existe uma opção *"log"*. Essa opção faz com que cada passo do sistema seja relatado em um arquivo de *log* no caminho: *raiz da aplicação/log/"data do log".log*. Essa opção é importante, pois nem sempre a API do *GetOldTweets* está funcionando corretamente, o que faz com que o sistema apresente lentidão em sua execução e às vezes causa falha na busca dos resultados, retornando parte dos resultados totais.

Ademais, o sistema faz uma interação com o usuário afim de apresentar qual módulo está executando, conforme Figura 13. Assim, o usuário tem o controle de qual etapa o sistema está trabalhando naquele intervalo de tempo.

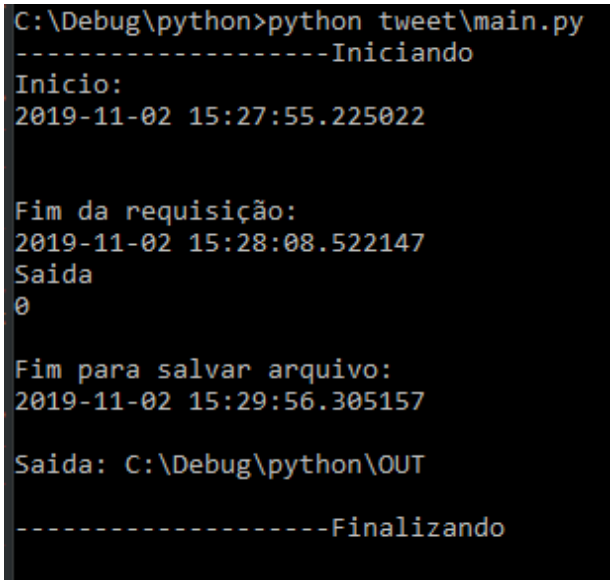


Figura 13 – Exemplo de execução do sistema *Tweet*

Como é apresentado na Figura 13, o sistema apresenta a data e hora de início da execução. Posteriormente começa a realizar a requisição para a API. Quando finalizado apresenta a data e hora de finalização requisição. Então o módulo *Tweet* chama o módulo *APIConectorJSON*, que faz o tratamento dos dados. Quando finalizado apresenta data e hora da finalização do tratamento dos dados. Apresenta ainda a saída da requisição: 0 para sucesso e qualquer outro valor para código de erro do sistema operacional. No final apresenta o caminho onde os arquivos foram salvos.

O módulo *Tweet* funciona então em duas fases: (1) seleciona os filtros no arquivo de entrada e faz a requisição com a API gerando um arquivo de saída X e (2) seleciona esse arquivo de saída e chama o módulo *APIConectorJSON* passando o arquivo X como entrada, passando ainda quais são os tipos de saída desejados.

3.1.3 *APIConectorJSON*

Esse módulo funciona com o propósito de organizar as informações obtidas da API *GetOldTweets* (COMMUNITY, 2019). Ele recebe de entrada um arquivo JSON com os dados da requisição e recebe também quais são as saídas esperadas conforme citado na seção anterior. Desse arquivo é retirado todos os objetos e para cada um dos tipos de saída é feito um tratamento diferente. Além disso, todo o processo também pode ser detalhado em arquivo de *log* caso haja algum erro. Esse *log* é o mesmo do sistema *Tweet*.

Os tipos de saída são configurados da seguinte forma:

- JSON: arquivo com estrutura pré estabelecida, é um modelo para armazenamento e transmissão de informações no formato texto e que é bastante utilizado por aplicações Web que trabalham com a tecnologia AJAX. É um objeto *JavaScript*. A própria API do *GetOldTweets* (COMMUNITY, 2019) retorna o arquivo JSON com o resultado, então o sistema apenas copia o arquivo para a pasta de saída.

- CSV: arquivo com a estrutura de informações separados por vírgulas. Comumente utilizado para apresentação de informações em planilhas e cópias de dados entre bases de informações, como banco de dados. Esse tipo de armazenamento agrupa as informações de texto em planilhas. O sistema o utiliza colocando os tipos na primeira linha do arquivo e as informações nas demais.

- XML: é um tipo de armazenamento que utiliza a linguagem de marcação recomendado pela W3C. É utilizado para representação de dados de forma estruturada mantendo as informações de acordo com suas especificidades. Utiliza de *tags* para caracterizar uma informação. O sistema utiliza as *tags* colocando os tipos dos dados nelas e a informação no espaço restante.

- SQL: é um armazenamento em banco de dados comum, tabelado de forma relacional. O sistema treabalha com o SQLite e insere a informação em uma tabela onde a coluna é o tipo de informação e as linhas são as tuplas de informações. O sistema utiliza esse tipo colocando os tipos dos *tweets* como as colunas da tabela RETORNO e as linhas são os *tweets* que estavam no arquivo JSON de entrada.

3.1.4 Acessando a API *GetOldTweets*

A API do *GetOldTweets* é uma biblioteca desenvolvida em *python* que tem o objetivo de fazer uma busca no *Twitter* a partir dos filtros possíveis, que serão descritos posteriormente. Seu funcionamento se baseia na leitura de um *browser* que acessa a rede social, então o resultado é obtido a partir dessa busca. Logo, o resultado sempre fica ordenado de forma cronológica, ou seja, pela data da publicação, mas de forma decrescente.

Essa API possui vários recursos para serem utilizados, não se limitando apenas à uma biblioteca. É possível acessá-la através de linha de comando passando os parâmetros de busca, assim como é possível chamá-la a partir de um programa externo, seja ele escrito em *python* ou não. Cada uma das formas de chamá-la gera um resultado diferente: se chamado por linha de comando é gerado um arquivo CSV com as informações, já chamando a partir de um programa externo a biblioteca retorna um arquivo JSON com os *tweets*. A estrutura dos objetos de retorno é dada com os seguintes atributos:

- *id* (*string*); Código referente ao tweet

- *permalink* (*string*); Link da publicação

- *username* (*string*); Nome do usuário que fez a publicação
- text (*string*); Texto do *tweet*
- *date* (*datetime*); Data da publicação
- *retweets* (*integer*); Número de vezes que o *tweet* foi *retweetado*
- *favorites* (*integer*); Número de vezes que o *tweet* foi favoritado
- *mentions* (*string*); Lista as menções da publicação
- *hashtags* (*string*); Apresenta as hashtags da publicação
- *geo* (*string*); Localização onde o *tweet* foi publicado

Para o presente trabalho foi escolhido utilizar a API como uma biblioteca de um sistema externo. Então, é conectado na API e a mesma retorna o arquivo JSON com os dados da busca para que esses dados sejam processados da forma que for escolhida.

A biblioteca funciona em duas partes: listar os filtros para a busca (classe *TwitterCriteria*) e realizar a busca propriamente dita (*TweetManager*). A classe de organização das buscas possui os seguintes métodos que possibilitam a realização dos filtros:

- *setUsername* (*str or iterable*): nome do usuário para a busca.
- *setSince* (*str. "yyyy-mm-dd"*): data de início da busca. Se não for dado um valor não há limite de busca. Se setado um valor esse é incluído no resultado, ou seja, se colocado a data 2015-01-01 no resultado haverá *tweets* dessa data.
- *setUntil* (*str. "yyyy-mm-dd"*): data final da busca. Se não colocado a data final será o dia em que a busca foi realizada, com resultados incluindo esse dia. Se setado o dia colocado não é considerado no resultado, ou seja, se colocado 2015-12-31 não haverá *tweets* dessa data.
- *setQuerySearch* (*str*): texto para verificação no *Twitter*. Esse texto será buscado na rede como um todo e deve haver um *"match"* completo da palavra para o *tweets* ser considerado resultado da busca. Ou seja, se filtrado a palavra "CVE", como será feito por esse trabalho, será retornado todos os *tweets* da rede que tenham essa palavra. A busca não é *case sensitive*, ou seja, não importa texto em maiúsculo e minúsculo.
- *setTopTweets* (*bool*): se setado como *"true"* é retornado apenas os *Top tweets*. Esse conceito de *Top tweet* se baseio no número de curtida e número de seguidores do usuário que publicou o *tweet*.
- *setNear*(*str*): uma referência de área localização onde o *tweet* foi publicado.

- *setWithin* (*str*): filtra a busca por uma área de localização a partir da opção colocada no *setNear*.
- *setMaxTweets* (*int*): o número máximo de *tweets* a ser suportado na busca dos dados. Se não setado ou se o valor for menor que 1 será retornado a busca total dos *tweets*, sem limite.

A biblioteca possui ainda algumas limitações baseadas nas condições em que a busca é efetuada. Basicamente, se a rede em que o usuário está conectado for oscilante no momento de leitura e requisição dos dados é possível que o retorno não seja composto por todos os *tweets* da busca, uma vez que a API faz uma busca na rede social do *Twitter* lendo um *browser*, ou seja, se não carregar os dados no navegador a biblioteca não haverá informações para leitura, ou se carregar apenas parte da busca, irá retornar apenas esse trecho de resultado.

Para facilitar a busca de ocorrência dos problema listados, basta conferir no resultado a data do último *tweet* retornado; se a data for uma não reconhecida, ou seja, não for a desejada, houve erro na busca. Esse erro também pode ocorrer devido ao estouro da memória principal, pois a API faz a busca utilizando dessa memória para armazenamento dos dados para apenas depois montar o arquivo de saída. Para o presente trabalho foram enfrentados ambos os problemas.

3.1.5 *Scanner*

O conjunto dos módulos citados e explicados anteriormente dão origem ao sistema *Scanner*. Esse possui:

1. a interface do *TwitterSearch*, o que possibilita preenchimento do formulário disponibilizando um arquivo JSON de configuração de filtragem e que chama o "cérebro"do programa: o *Tweet*.
2. O *Tweet*, que processa o arquivo de entrada com os filtros requisitando a API *GetOldTweets* (COMMUNITY, 2019) com esses dados de entrada, obtendo os *tweets* de retorno que satisfazem a busca, criando, assim, um novo arquivo JSON de saída que contém uma lista com os objetos dos *tweets* retornados pela API.
3. O *APIConectorJSON* que organiza os dados recebendo o JSON de resultado com os objetos dos *tweets* e recebendo também os tipos de saída selecionado no *TwitterSearch*. Depois de finalizado o próprio módulo *Tweet* copia as saídas para o caminho selecionado pelo usuário.

.

A Figura 3.1.5 apresenta um fluxograma de como o sistema atua.

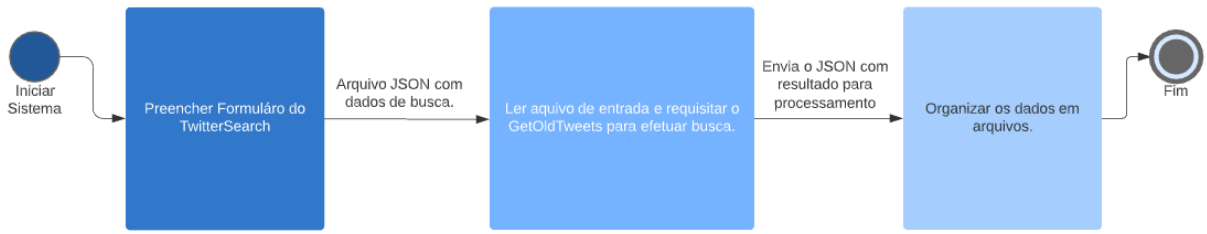


Figura 14 – Fluxo de como o sistema interliga as informações

Conforme apresentado na Figura , o fluxo do sistema é consistente e cada parte do mesmo tem sua função para o funcionamento do sistema como um todo. Sendo um sistema distribuído, é possível identificar ainda a necessidade de cada um pelo outro módulo.

Um exemplo de funcionalidade segue na Figura 15 que apresenta as seguintes entradas de dados:

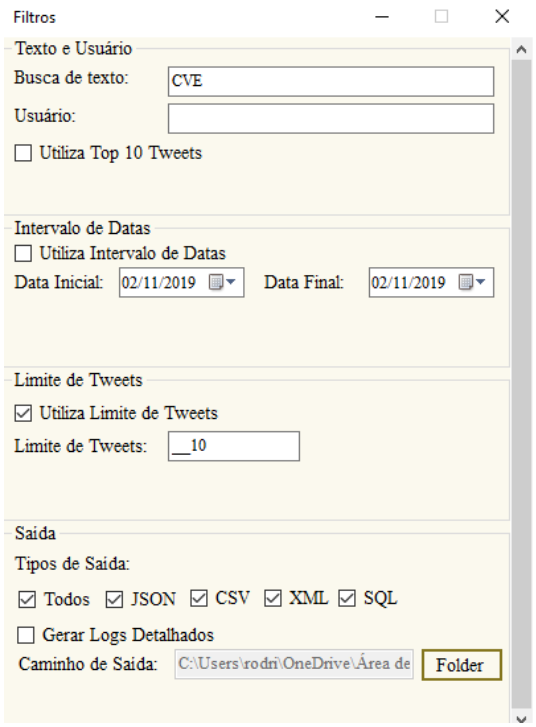


Figura 15 – Entrada de exemplo (imagem editada com o propósito de listar apenas os filtros utilizados)

Com esses dados de entrada, o sistema aciona os outros módulos e obtém as saídas nos arquivo em JSON, CSV, XML e SQL. Todos esses arquivos possuem os dados de retorno do *Twitter* com a busca feita. Tomando como exemplo um desses *tweets* do retorno da busca, o de ID 1190312935895158789, ele é listadas nas Figuras 16, 17, 18 e 19.

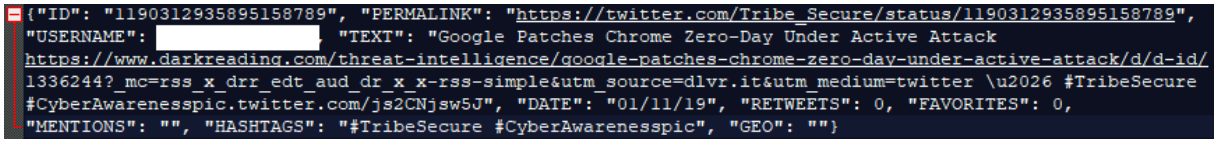


Figura 16 – Exemplo do *tweet* em saída JSON (o texto foi alinhado para uma melhor visualização)

Figura 17 – Exemplo do *tweet* em saída CSV

Figura 18 – Exemplo do *tweet* em saída XML

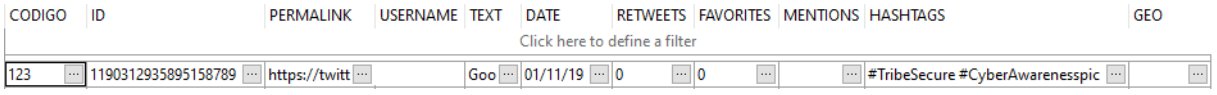


Figura 19 – Exemplo do *tweet* em saída SQL

As Figuras 16, 17, 18 e 19 apresentam as saídas possíveis do sistema, mostrando suas configurações. Essas saídas diferentes possibilitam que sejam processadas de maneiras distintas. Então um serviço *web* consegue interpretar esses dados, assim como um aplicativo para *Desktop* ou *Mobile*.

Como o retorno do *Scanner* monta uma base de dados consideravelmente grande, ele já mostra que o volume de dados de quando não se coloca limite de número de *tweets* pode crescer ainda mais. A fim de testar, foi realizada uma busca com um número ilimitado de *tweets* e sem definir nenhuma data. Como resultado, passaram-se mais de 96 horas sem o programa finalizar tais requisições.

Em suma, esse é o sistema no qual o trabalho se baseia. É com ele que a análise sobre as quantidade de vulnerabilidades discutidas no *Twitter* será possível, assim como a frequência da mesma nos últimos anos.

Esse é um programa genérico, não limita apenas em busca de vulnerabilidades, mas para qualquer palavra ou sentença que se queia. Ele possibilita buscas em toda a rede do *Twitter*, logo serão os filtros que farão com que a busca seja orientada aos objetivos do trabalho.

A seguir, o programa utilizado para melhorar a interpretação dos resultados será apresentado.

3.1.6 *FindWords*

O *FindWords* tem o objetivo de unir arquivos XML de um diretório e agrupá-los em apenas um arquivo XML, ou JSON, ou CSV ou SQL. Desse modo, é fácil a interpretação desses arquivos e a tradução para dados probabilísticos e de comparação.

Para o presente trabalho foi escolhido agrupá-los em um banco de dados *SQL*. A partir desse agrupamento, foi possível unir em apenas um banco de dados todos os registros dos *tweets* de busca, assim como os dados de vulnerabilidades presentes no site do NVD. Dessa forma, a interpretação dos dados se tornou mais fácil, pois os dados se encontravam em apenas uma base de dados. Com a ferramenta *FindWord* é possível identificar os *tweets* que mencionam uma vulnerabilidade que aparece também no site do NVD e qual vulnerabilidade é essa. Esse sistema também gera *logs* para identificação da etapa que está sendo processada.

A seção 3.2 irá listar como foi feita a coleta de dados do *Twitter* e quais filtros foram utilizados. Ela irá apresentar como esses filtros impactaram na buscas e quais os objetivos de coloca-los da maneira que foram colocados.

3.2 Coleta de dados do *Twitter*

O *Scanner* faz uma busca no *Twitter* a partir da API *GetOldTweets* (COMMUNITY, 2019) passando os parâmetros de filtro. Essa é a coleta de dados que o sistema faz da rede social. Para trabalhar com os dados, eles são organizados em outros arquivos que possibilitam uma melhor visualização dos dados, assim o trabalho de análise fica mais simples.

No site do NVD, conforme citado anteriormente, é apresentada uma lista de opções para download das vulnerabilidades. O site organiza essas informações colocando-as em um arquivo ZIP para cada ano. Assim, é possível baixar essas informações e utilizar delas para analisar os dados de retorno. O site do CVE também fornece a opção de baixar como um arquivo XML com os mesmos dados, mas o arquivo não é em formato ZIP e acaba sendo um pouco maior para download (The MITRE Corporation, 2018).

A coleta de dados então é feita a partir de duas bases para a análise desse trabalho: *Twitter*, através da API do *GetOldTweets* (COMMUNITY, 2019); e o NVD, que processa seus dados a partir da base do CVE. Através dessas bases de dados é possível chegar aos objetivos do trabalho.

Tomando como exemplo a Figura 20, tem-se a vulnerabilidade de código CVE-2019-2110. A Figura 21 apresenta suas informações no site do CVE.

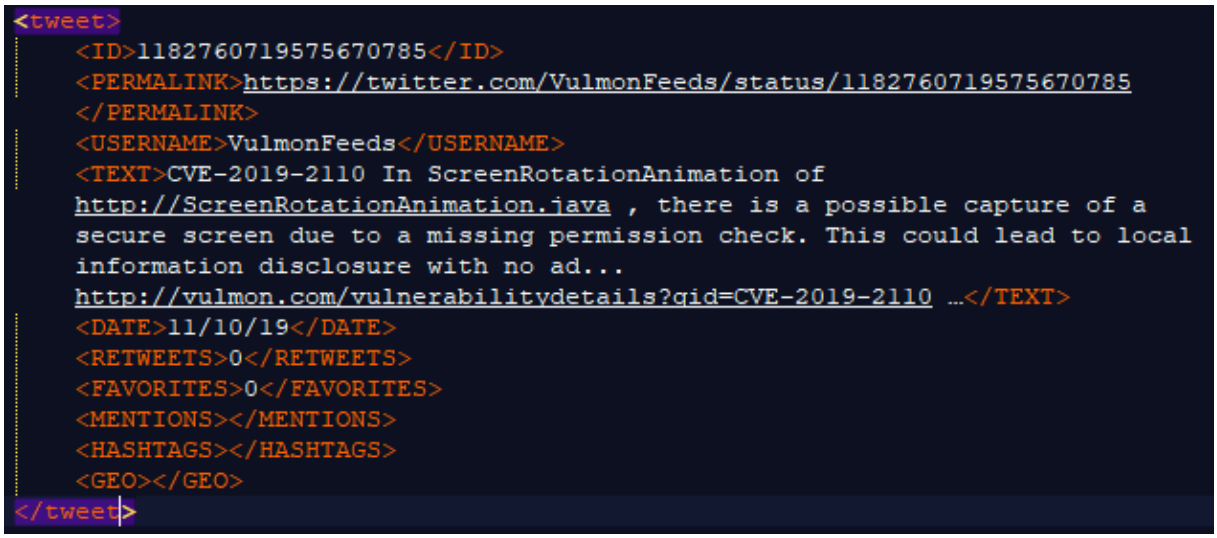


Figura 20 – Exemplo de publicação de uma vulnerabilidade no *Twitter* em formato XML

A Figura 20 apresenta os dados em XML da vulnerabilidade encontrada apresentando informações que estão disponíveis nessa publicação. O campo que será mais utilizado nesse trabalho é o *"TEXT"*, o qual armazena qual o texto do *tweet*.

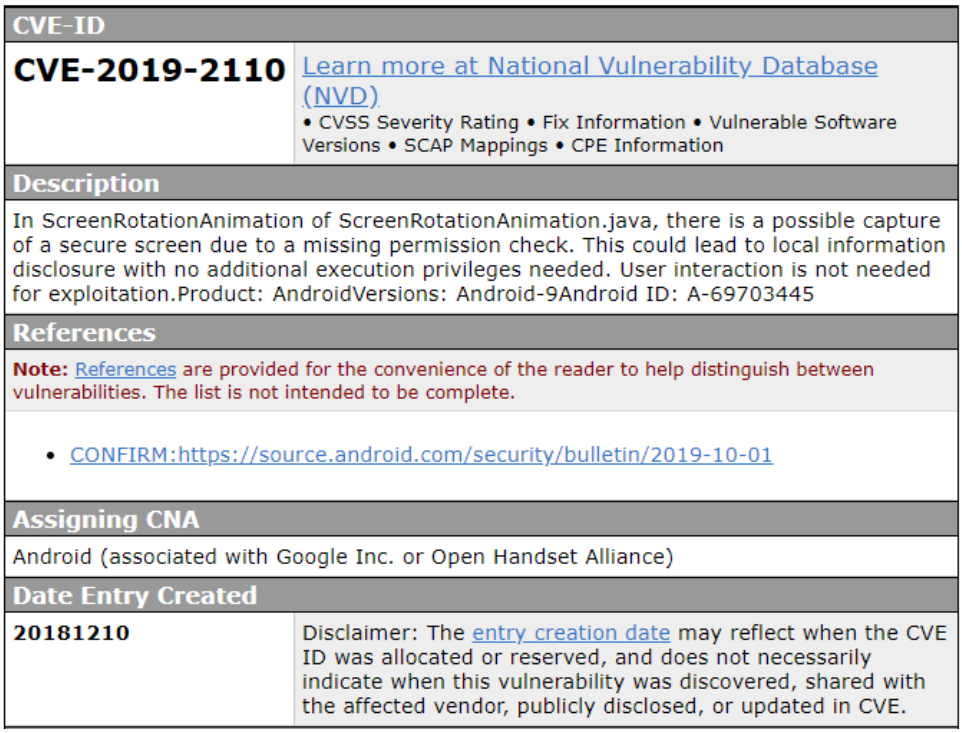


Figura 21 – Descrição da vulnerabilidade CVE-2019-2110 no site do CVE. Extraído de CVE (2018)

A Figura 21 descreve no NVD os dados sobre a vulnerabilidade diretamente do site. É apresentado os principais dados sobre a vulnerabilidade, como seu *ID* e sua descrição.

O arquivo disponibilizado no site do NVD referente ao ano de 2019 descreve a vulnerabilidade, como apresentado na Figura 22. Assim como o arquivo XML baixado no site do CVE, apresentado na Figura 23.

```
<entry name="CVE-2019-2110" published="2019-10-11" seq="2019-2110" type="CVE">
  <desc>
    <descript language="es" modified="2019-10-15" source="cve" title="el archivo
    ScreenRotationAnimation.java en la función ScreenRotationAnimation en
    Android" translator="translator-178">En la función ScreenRotationAnimation
    del archivo ScreenRotationAnimation.java, se presenta una posible captura de
    una pantalla segura debido a una falta de comprobación de permisos. Esto
    podría conllevar a la divulgación de información local sin ser necesarios
    privilegios de ejecución adicionales. No es requerida una interacción del
    usuario para su explotación. Producto: Android; Versiones: Android-9; ID de
    Android: A-69703445.</descript>
  </desc>
  <refs/>
</entry>
```

Figura 22 – Descrição da vulnerabilidade CVE-2019-2110 no arquivo disponibilizado no NVD

```
<Vulnerability Ordinal="139006" xmlns="
http://www.icasi.org/CVRF/schema/vuln/1.1">
    <Title>CVE-2019-2110</Title>
    <Notes>
        <Note Ordinal="1" Type="Description">In ScreenRotationAnimation of
        ScreenRotationAnimation.java, there is a possible capture of a secure
        screen due to a missing permission check. This could lead to local
        information disclosure with no additional execution privileges needed.
        User interaction is not needed for exploitation.Product:
        AndroidVersions: Android-9Android ID: A-69703445</Note>
        <Note Ordinal="2" Title="Published" Type="Other">2019-10-11</Note>
        <Note Ordinal="3" Title="Modified" Type="Other">2019-10-11</Note>
    </Notes>
    <CVE>CVE-2019-2110</CVE>
    <References>
        <Reference>
            <URL>https://source.android.com/security/bulletin/2019-10-01</URL>
            <Description>CONFIRM:
            https://source.android.com/security/bulletin/2019-10-01</Description>
        </Reference>
    </References>
</Vulnerability>
```

Figura 23 – Descrição da vulnerabilidade CVE-2019-2110 no arquivo disponibilizado no CVE

Ou seja, a coleta dos dados não se restringe somente ao *Twitter*, mesmo que o objetivo seja a verificação de discussão sobre vulnerabilidades na rede social. A coleta é feita nos sites que tratam de vulnerabilidades para a identificação correta do que é ou não vulnerabilidade.

Parte da análise está em filtrar na rede social, a outra está na identificação dessas vulnerabilidades.

Para o presente trabalho os filtros foram feitos para o ano 2018. Logo, tanto no site do NVD, quanto no CVE, foram baixados os arquivos referentes aos anos de buscas e os mesmos serão utilizados em conjunto com o filtro feito nas publicações do ano de 2018 por parte do *Scanner*. A máscara de busca no *Twitter* foi *CVE-AAAA-IIII*.

Então o *Scanner* é o responsável por buscar esses dados do *Twitter* e o *FindWords* será o responsável por relacionar os dados recolhidos da rede social com os dados presentes nos sites do CVE e do NVD. Assim faz-se a interseção entre os resultados da busca no *Twitter* com os dados recolhidos do CVE e do NVD.

Para a interpretação desses dados e afim de obter um resultado mais claro da pesquisa, o resultado obtido do *Scanner* ainda é submetido a algumas consultas SQL. Essas, possibilitam a obtenção de resultados que apresentam, de forma numérica, algumas informações importantes para o presente trabalho.

No próximo capítulo serão listados os resultados dessas buscas comparando outros anos. O número de vulnerabilidades têm crescido, isso é fato, o CVE prova isso a partir dos tamanhos dos arquivos de cada ano. Esse aumento também é esperado no número de pessoas discutindo o tema no *Twitter*.

4 Resultados

O objetivo desse capítulo é apresentar um estudo de caso para validar as ferramentas que foram construídas. Em particular, os seguintes pontos serão avaliados:

1. Qual o comportamento do número de *tweets* relacionados a vulnerabilidades de segurança nos últimos anos?
2. Quais são os tipos de vulnerabilidades com maior incidência de discussão no *Twitter*?
3. Qual é a razão entre vulnerabilidades de segurança presentes em redes sociais e vulnerabilidades descobertas?

A resposta para tais perguntas poderá esclarecer alguns pontos sobre a relação entre redes sociais e vulnerabilidades de segurança.

4.1 Análise dos dados

Para chegar ao objetivo do trabalho foi feito um levantamento da quantidade de vulnerabilidades que foram discutidas no ano de 2018 na rede social do *Twitter*. Utilizando a ferramenta do *Scanner* foi feito o filtro nesse período e, a partir dos resultados, foi vinculado à lista do site do CVE para identificação da existência da vulnerabilidade discutida através do *FindWords*.

Para a busca foi utilizado o filtro a partir da máscara *CVE-AAAA-IIII*, onde 'A' referencia o ano e 'I' o algorismo sequencial do CVE. Dos resultados, foram selecionados aqueles em que a vulnerabilidade está presente também no site do *NVD*.

Como base de comparação, foi feito também o levantamento para os anos de 2015, 2016 e 2017. O filtro utilizado foi o mesmo, máscara *CVE-AAAA-IIII* para busca, diferenciando-se apenas no intervalo de datas.

Além do *FindWords*, foi feito um aplicativo cuja principal função foi juntar todas as informações obtidas do *Scanner* em um único arquivo. Então, para todas as buscas feitas de cada ano, foi agrupado em um banco de dados *SQLite* o resultado completo. Desse resultado, para a busca do *FindWords*, foi criado ainda uma tabela de cada ano, além de uma única tabela com todos os resultados. Ademais, há uma tabela que contém todos os dados de vulnerabilidades, obtida a partir do site do NVD. O *FindWords* então buscou dessa tabela, para cada ano, quais *tweets* realmente tinham uma vulnerabilidade real descrita ali. Assim, foi possível identificar quais vulnerabilidade do site do NVD foram discutidas no *Twitter*, assim como quais *tweets* eram realmente relevantes.

Com o intuito de auxiliar a interpretação dos dados, um único repositório, com as seguintes tabelas, foi criado pelo *FindWords*:

- NVD: Tabela que armazena as vulnerabilidades no site do NVD. O *FindWords* gerou, a partir dos arquivos XML do site do NVD, a tabela com dados de 2015 a 2018.
- RETORNO: Tabela que armazena todos os *tweets* de 2015 à 2018.
- *NVDMATCH*: Essa tabela armazena a relação entre *Tweets* e vulnerabilidades do site do nvd. Então ela é o resultado da tabela RETORNO junto com a tabela NVD. Para cada *tweet* foi coletado a sentença que aparece a máscara *CVE-AAAA-IIII* e verificado tal corresponde com alguma vulnerabilidade listada no NVD.
- RETORNO2015: Tabela que armazena todos os *tweets* de 2015 que entraram na busca.
- RETORNO2016: Tabela que armazena todos os *tweets* de 2016 que entraram na busca.
- RETORNO2017: Tabela que armazena todos os *tweets* de 2017 que entraram na busca.
- RETORNO2018: Tabela que armazena todos os *tweets* de 2018 que entraram na busca.

4.2 Comportamento do número de *tweets* relacionados a vulnerabilidades de segurança

De acordo com os resultados obtidos com as buscas, foi possível identificar o aumento no número de *tweets* que falam sobre vulnerabilidades ao longo do tempo. Serão apresentados alguns gráficos e uma tabela que comprovam essa teoria nessa seção.

O gráfico apresentado na Figura 24 apresenta uma série temporal na qual é listada a quantidade de *tweets* que contêm vulnerabilidades presentes também na lista do site NVD. Foi filtrado por ano, então na linha representando o ano de 2015, por exemplo, a busca é feita por *tweets* que citam vulnerabilidades descobertas em 2015 apenas. Na linha que representa o ano de 2016, apresenta vulnerabilidades buscando apenas *tweets* publicadas em 2016 e assim sucessivamente.

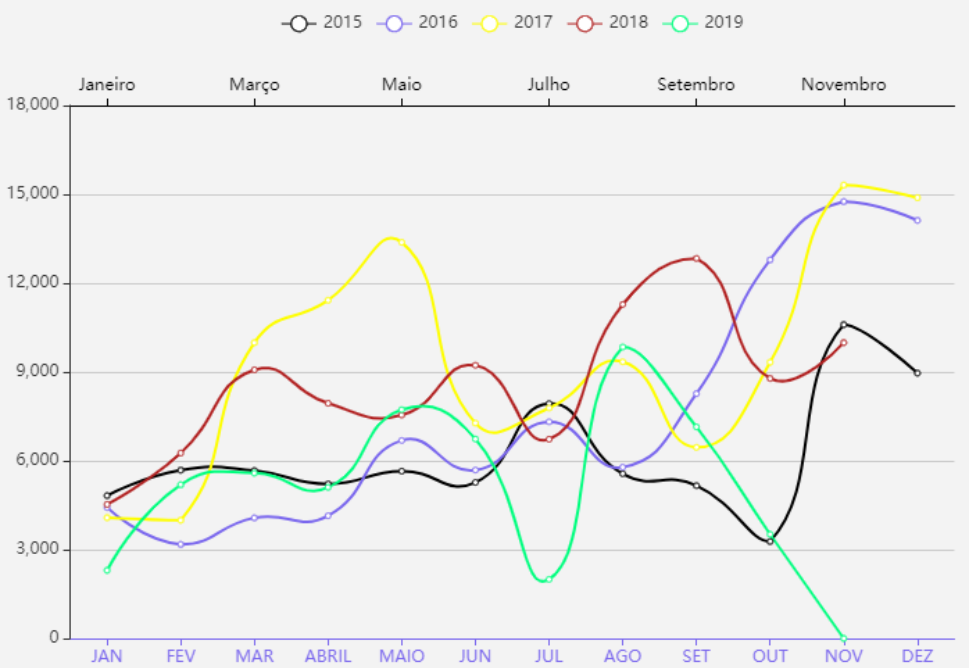


Figura 24 – Série temporal listando os *tweets* que citam vulnerabilidades em seu respectivo ano

Conforme apresentado na Figura 24, é possível identificar que não segue um padrão em relação ao crescimento ou decaimento de citações de vulnerabilidades em *tweets*. Outrossim, é possível identificar aumento do número de menções em relação a cada ano.

Já na Figura 25 apresenta uma série temporal na qual é listada a quantidade de *tweets* que contém vulnerabilidades presentes também na lista do site NVD, mas independentemente do ano. Ou seja, *tweets* que falam sobre vulnerabilidades independentemente de quando elas foram anuncidas no site do NVD.

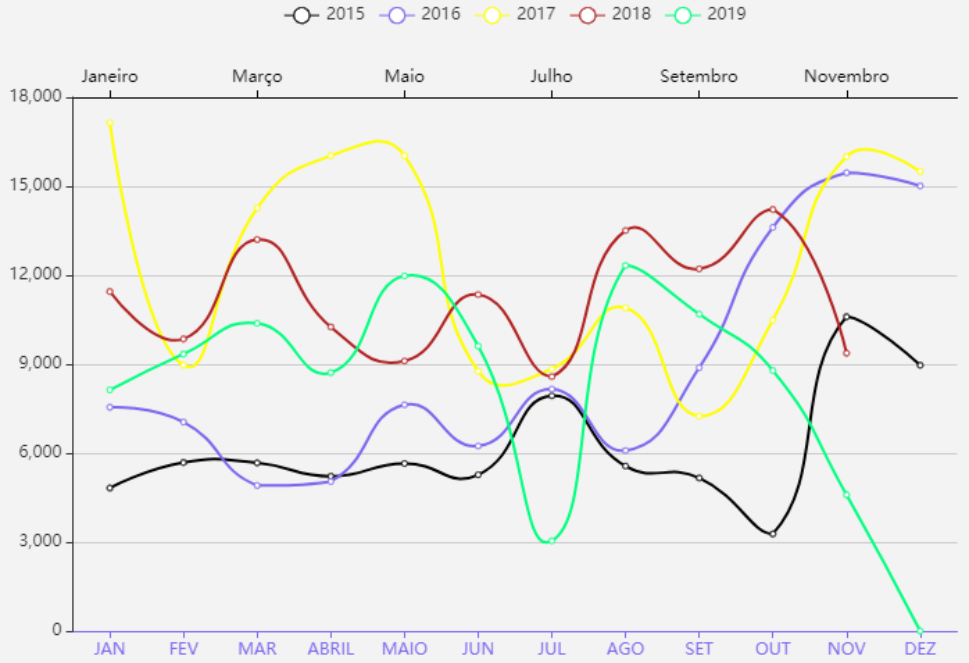


Figura 25 – Série temporal listando os *tweets* que citam vulnerabilidades em seu respectivo ano

Assim como na Figura 24, é possível identificar que o crescimento na Figura 25 é gradual, mas se comparar números por anos. Se na Figura 24 apresentava apenas os *tweets* que foram mencionados no *Twitter*, se Figura 25 trás a relação dessas vulnerabilidades que também existem no site do NVD, independentemente do ano que a mesma foi incluída no site.

A seguir serão listadas duas tabelas que apresentam a quantidade de vulnerabilidades de forma mensal do intervalo entre 2015 e 2019. Nelas está especificado a média e o desvio padrão das vulnerabilidades citadas no *Twitter*, além da contabilidade de quantos *tweets* foram publicados no intervalo de tempo.

A Tabela 1 apresenta a relação entre as vulnerabilidades publicadas em *tweets* em cada um dos anos da pesquisa desse trabalho. Nela é numerado os *tweets* que mencionam uma vulnerabilidade descoberta no mesmo ano em que o *tweet* foi publicado.

Tabela 1 – Tabela de vulnerabilidades que apresenta dados de ano após ano, mostrando quantos *tweets* apresentaram uma vulnerabilidade daquele mesmo ano corrente.

	2015	2016	2017	2018	2019	Total	Média	D.P.
Janeiro	4824	4424	4079	4525	2300	20152	4030,4	897,3290589
Fevereiro	5684	3177	3993	6261	5192	24307	4861,4	1125,714813
Março	5669	4072	9990	9072	5584	34387	6877,4	2258,66913
Abril	5219	4144	11423	7942	5103	33831	6766,2	2649,69005
Maio	5650	6687	13377	7545	7726	40985	8197	2692,458133
Junho	5269	5687	7273	9227	6736	34192	6838,4	1392,372881
Julho	7933	7317	7783	6728	1997	31758	6351,6	2217,473662
Agosto	5563	5784	9342	11281	9843	41813	8362,6	2287,129432
Setembro	5162	8267	6455	10936	8432	39252	7850,4	1959,968224
Outubro	3274	12785	9328	12834	7145	45366	9073,2	3614,821567
Novembro	10599	14754	15312	8782	3517	52964	10592,8	4311,204351
Dezembro	8961	14120	14885	9991	0	47957	9591,4	5311,919751
Total	73807	91218	113240	105124	63575	446964	89392,8	18595,52051

A Tabela 2 apresenta a relação entre as vulnerabilidades publicadas em *tweets* em cada um dos anos da pesquisa desse trabalho. Mas a busca é feita por todas as vulnerabilidades, independentemente do ano em que elas foram criadas e divulgadas no site do NVD.

Tabela 2 – Tabela de vulnerabilidades que apresenta dados de ano após ano, mostrando quantos *tweets* apresentaram uma vulnerabilidade de qualquer ano.

	2015	2016	2017	2018	2019	Total	Média	D.P.
Janeiro	4824	7547	17141	11450	8132	49094	9818,8	4224,448433
Fevereiro	5684	7049	8968	9851	9342	40894	8178,8	1566,690703
Março	5671	4907	14267	13203	10379	48427	9685,4	3815,603837
Abril	5220	5041	16030	10255	8712	45258	9051,6	4025,608953
Maio	5650	7629	16031	9108	11976	50394	10078,8	3621,775664
Junho	5269	6241	8762	11347	9608	41227	8245,4	2219,011185
Julho	7933	8158	8841	8580	3042	36554	7310,8	2157,772685
Agosto	5564	6087	10898	13507	12327	48383	9676,6	3255,368157
Setembro	5162	8881	7245	12214	10687	44189	8837,8	2484,699209
Outubro	3274	13613	10484	14216	8783	50370	10074	3943,487441
Novembro	10599	15454	15999	9372	4594	56018	11203,6	4206,396871
Dezembro	8964	15013	15499	10402	0	49878	9975,6	5596,446966
Total	73814	105620	150165	133505	97582	560686	112137,2	26928,39338

A partir das tabelas mencionadas é possível identificar que o número de vulnerabilidades citadas no *Twitter* é alto. Quando se compara a tabela 2 com a 1 fica mais notável que o número de vulnerabilidades lançadas em um ano x não é muito citado nesse mesmo ano x. Ou seja, em um ano a discussão sobre vulnerabilidades por parte dos usuários não se limita apenas às vulnerabilidades daquele ano, mas sim vulnerabilidades antigas.

4.3 Tipos de vulnerabilidades mencionadas no *Twitter*

Utilizando o banco de dados gerado pelo *FindWords*, cujo conteúdo é a junção de todas as informações obtidas a partir do *Scanner*, é possível listar quais são as vulnerabilidades mais discutidas. Então tomando como comparação todo o período de pesquisa desse trabalho, 2015 à 2019, temos a Figura 26 com as 5 vulnerabilidades mais discutidas.

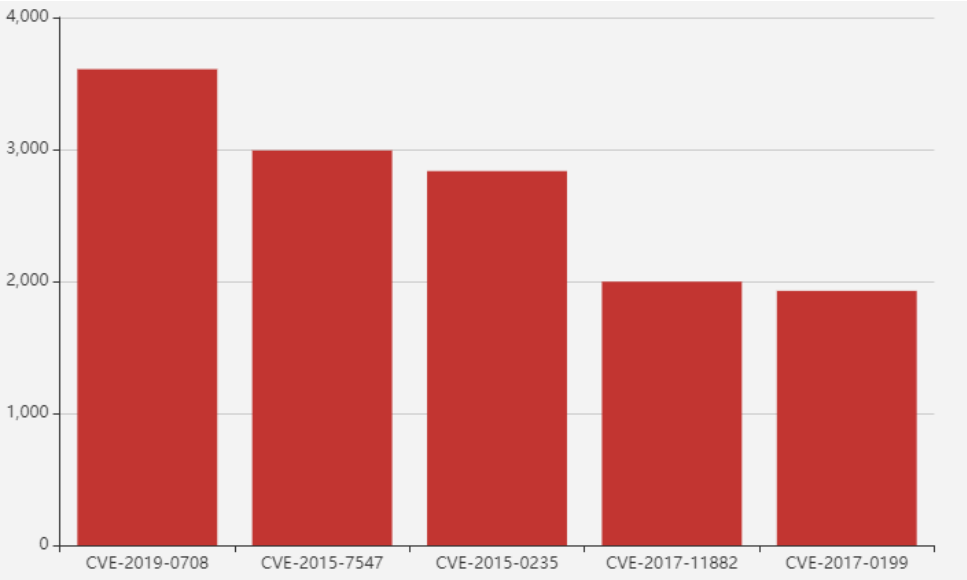


Figura 26 – Lista das vulnerabilidades mais discutidas entre 2015 e 2019

Conforme é possível identificar na Figura 26, o número de menções dessa vulnerabilidade é alto, assim como a diferença entre a primeira para a segunda colocada. Esse salto numérico não se vê entre a segunda e a terceira e entre a quarta e a quinta.

A Tabela 3 apresenta quais são os aspectos gerais dessas vulnerabilidades. A tabela contém o nome da vulnerabilidade, a qual sistema ela se refere, o ano de sua publicação, o número de *tweets* que falaram sobre ela e apresenta também o *base score* daquela vulnerabilidade. Essa última informação está associada a severidade da vulnerabilidade. Os valores desse campo variam em um intervalo de 0 a 10, sendo 0 para severidade mais baixa e 10 para severidade mais alta.

Tabela 3 – Tabela das vulnerabilidades mais discutidas entre 2015 e 2019

Vulnerabilidade	Sistema Afetado	Ano Publicação	Número	BS NVD
CVE-2019-0708	Remote Desktop Services	2019	3608	9,8
CVE-2015-7547	GNU C Library	2015	2992	8,1
CVE-2015-0235	GNU C Library	2015	2836	N/A
CVE-2017-11882	Microsoft Office 2007	2017	1998	7,8
CVE-2017-0199	Microsoft Office 2007 ao 2016	2017	1928	7,8
CVE-2016-5195	Linux Kernel	2016	1737	7,8
CVE-2017-5638	Apache	2017	1689	10
CVE-2016-1000	Adobe Flash Player	2016	1604	5,3
CVE-2017-10003	Oracle Sun Systems Products Suite	2017	1458	7
CVE-2018-1177	Foxit Reader	2018	1350	7,2

Outra informação importante a tocar dessas vulnerabilidades mais discutidas são os *exploits* das mesmas. Esses, são definidos como códigos capazes de explorar uma vulnerabilidade (NEWSOME et al., 2006). O site *Exploit Database* (OffSec Services Limited, 2020) cataloga alguns desses *exploits*. Para as vulnerabilidades citadas na Tabela 5, existem os *exploits* listados na Tabela 4.

Tabela 4 – Tabela que apresenta os *exploits* das vulnerabilidades mais discutidas

Vulnerabilidade	Exploit
CVE-2015-7547	39454
CVE-2015-7547	40339
CVE-2017-11882	43163
CVE-2017-0199	41894
CVE-2017-0199	41934
CVE-2017-0199	42995
CVE-2017-5638	41570
CVE-2017-5638	41614
CVE-2016-1000	39610

A coluna “Exploit” lista a código do Exploit de acordo com o *Exploit Database*. Eles podem ser buscados no site *exploit-db* (OffSec Services Limited, 2020). Como é apresentado nem todas as vulnerabilidades tem ao menos um *exploit*, enquanto outros têm mais de um. Isso não necessariamente significa que uma vulnerabilidade seja mais crítica que a outra. Pode ser que um *exploit* para uma certa vulnerabilidade exista mas não está catalogado no *exploit-db*. Contudo, esse indicador é interessante para entender o impacto de uma vulnerabilidade em sistemas reais.

Após mostrar as vulnerabilidades mais discutidas entre os anos de 2015 e 2019, será feita uma análise sobre menções de vulnerabilidade em um determinado ano do calendário. Escolheu-se o ano de 2018 para isso. A Figura 27 apresenta as vulnerabilidades mais discutidas nesse ano, e a Tabela 5 apresenta a descrição de cada uma delas.

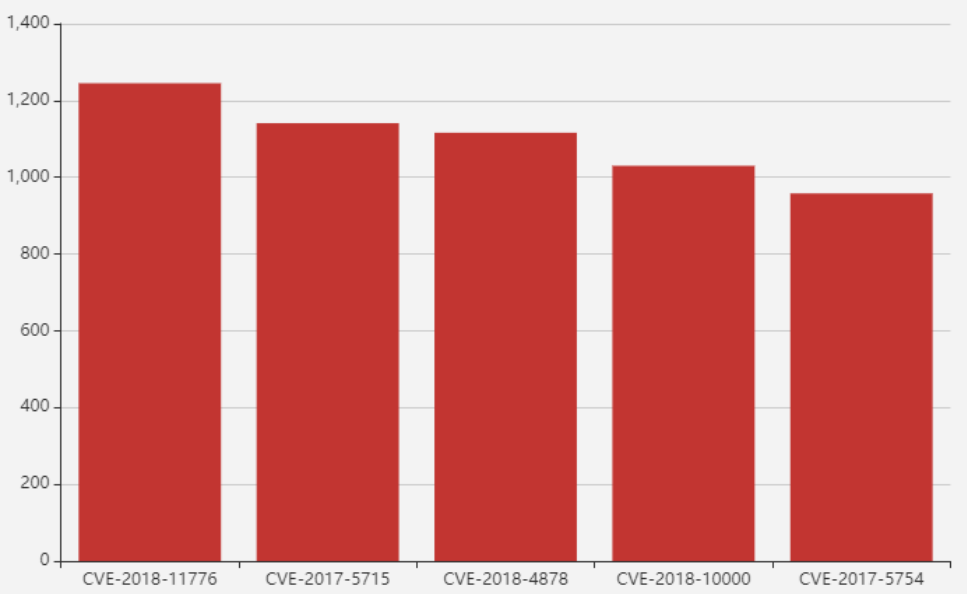


Figura 27 – Lista das vulnerabilidades mais discutidas no ano de 2018

Tabela 5 – Tabela das vulnerabilidades mais discutidas em 2018

Vulnerabilidade	Sistema Afetado	Ano Publicação	Número	BS NVD
CVE-2018-11776	Apache	2018	1269	8,1
CVE-2018-4878	Adobe Flash Player	2018	1164	9,8
CVE-2017-5715	Microprocessadores Intel Atom e Celeron e ARM Cortex	2017	1160	5,6
CVE-2018-10000	Video Downloader for Google	2018	1040	5,8
CVE-2017-5754	Microprocessadores Intel Atom e Celeron e ARM Cortex	2017	977	5,6

A partir dos dados da Tabela 5 é possível identificar quais são os tipos de assuntos mais discutidos, tendo *microprocessadores* como o assunto mais visado, com 2137 *tweets* ao todo. Dentre esses mais discutidos, é notável a presença dos *CVE-2017-5715* e o *CVE-2017-5754*, que foram vulnerabilidades amplamente discutidas na comunidade de segurança da informação, sendo conhecidos como *Spectre* e *Meltdown*, respectivamente. São vulnerabilidades que permitem que aplicativos tenham acesso a dados de outros aplicativos em processamento pelo processador, assim pode haver o vazamento de informações sensíveis, como senhas (Graz University of Technology, 2018).

Um outro ponto interessante é que a comunidade também discute vulnerabilidades de anos anteriores, como é visto na Tabela 5. Isso pode ser um reflexo de um interesse contínuo em tal vulnerabilidade devido à sua severidade ou ainda a descoberta de um novo *exploit*. No caso do ano de 2018, o impacto causado pelas vulnerabilidades já mencionadas *Spectre* e *Meltdown* continuou a ser notado no ano posterior.

Tabela 6 – Tabela que apresenta os *exploits* das vulnerabilidades mais discutidas em 2018

Vulnerabilidade	Exploit
CVE-2018-11776	45260
CVE-2018-11776	45262
CVE-2018-11776	45367
CVE-2018-4878	44412
CVE-2017-5715	43427

A Tabela 6 apresenta os códigos dos *exploits* referente às vulnerabilidades mencionadas na Tabela 5. Nota-se, mais uma vez, que nem todas as vulnerabilidades possuem seus *exploits* disponíveis, pois os mesmos não foram desenvolvidos ou publicados. Ainda estudando as vulnerabilidades mais discutidas em 2018, a vulnerabilidade mais discutida de 2016 nos *tweets* de 2018 vem apenas na 416ª posição. Esse fato mostra que a discussão está sobre as vulnerabilidades consideradas mais recentes. Essa vulnerabilidade citada é a CVE-2016-1238.

4.4 Relação entre vulnerabilidades existentes e discutidas no *Twitter*

Para o presente trabalho foi realizado também a análise de quantas vulnerabilidades surgiram em cada ano e quantas delas foram citadas no *Twitter*. Então foi feito o levantamento de quantas vulnerabilidades foram documentadas em cada ano, e dessas, quais foram citadas por algum *tweet*. A busca foi limitada até o ano de 2018, pois 2019 ainda foi concluído.

A Tabela 7 apresenta a quantidade de vulnerabilidades descobertas em cada ano, ou seja, vulnerabilidades documentadas e presentes no site do NVD; e quantas dessas vulnerabilidades foram citadas no *Twitter*. Apresenta também a porcentagem de quantas das vulnerabilidades documentadas foram citadas na rede social.

Tabela 7 – Tabela com a relação entre vulnerabilidades descobertas e discutidas no *Twitter*

Ano	Vulnerabilidades Descobertas	Quantidade citadas no Twitter	Porcentagem Citadas
2015	7880	7353	93,31218274
2016	9248	8671	93,76081315
2017	15580	13176	84,56996149
2018	15235	14873	97,62389235

Como é possível identificar na Tabela 7, em média 91.75% das vulnerabilidades foram citadas no *Twitter*. Esse número é um valor bem elevado, dado que o interesse por segurança da informação tem crescido nos últimos anos devido a todo avanço de tecnologias.

Em contra partida ainda há algumas vulnerabilidades que não foram citadas no *Twitter*. Tomando um apanhado delas, é possível concluir que muitas delas tem um *score* relativamente

baixo (entre 0 e 7); mas não foi possível identificar se todas seguem esse padrão, pois o score não foi considerado no levantamento de todas as vulnerabilidades devido à falta dessa informação nos arquivos XML obtidos no NVD.

Há vulnerabilidades da *Oracle MySQL*, grande e conhecida empresa, que não foram sequer mencionados. Tomando uma dessas vulnerabilidades como exemplo, de código *CVE-2016-0601*, ela permite que usuários logados dê lentidão à rede e até parar a mesma. Ela possui um *score* de 3.1, considerado leve.

Outro fator a ser notado é a quantidade de *tweets* que contêm vulnerabilidades documentadas no mesmo ano corrente da publicação, e aqueles que discutem de vulnerabilidades de anos anteriores. A Tabela 8 apresenta essa informação.

Tabela 8 – Tabela que descreve a quantidade de tweets citadas no ano vigente e nos demais.

Ano de busca	Nº de Tweets Totais	Nº Tweets Ano Vigente	Nº Tweets Outros Anos
2015	91691	74614	17077
2016	113709	91778	21931
2017	156472	113807	42665
2018	138079	105474	32605

A Tabela 8 apresenta as informações por ano. Tomando por base o ano de 2017, foram 156472 *tweets* falando sobre vulnerabilidades, desses 113807 *tweets* falaram das vulnerabilidades documentadas no ano de 2017 e outros 42665 falando de vulnerabilidades de outros anos. Isso indica que o número de *tweets* que mencionam vulnerabilidades documentadas em outros anos também é alto.

Seguindo ainda a ideia da Tabela 8, é possível identificar que o número de *tweets* que citam vulnerabilidades documentadas no ano vigente é muito maior comparado ao número de *tweets* que citam vulnerabilidades de outros anos. A Figura 28 reforça essa informação.

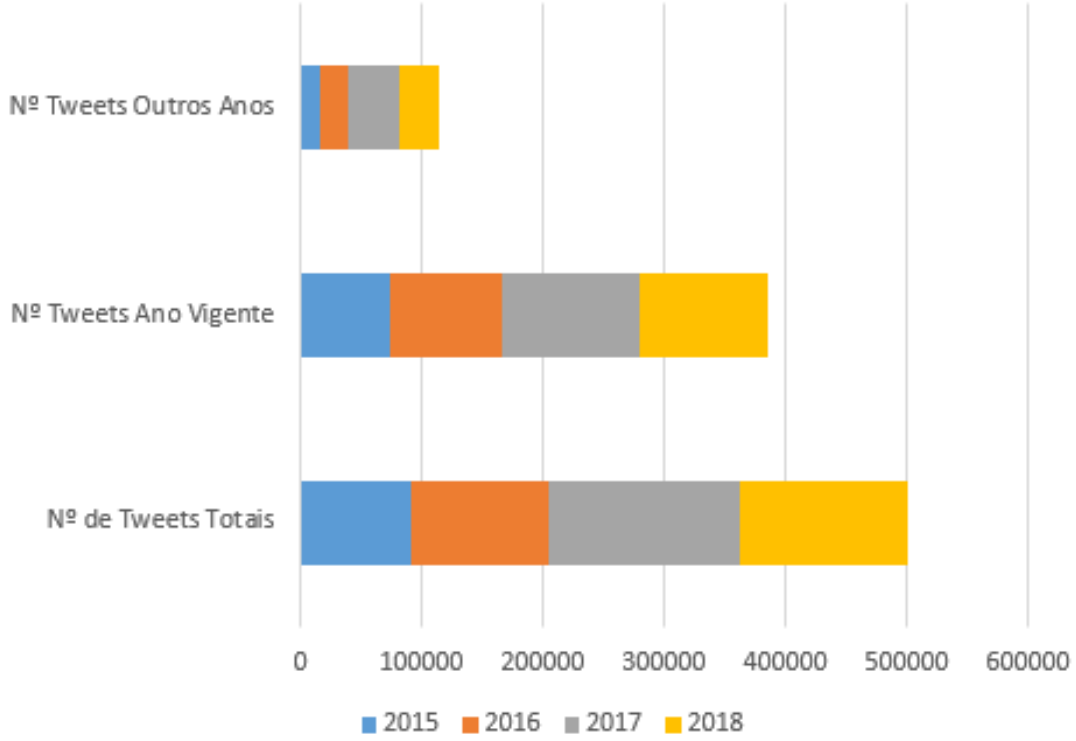


Figura 28 – Número de vulnerabilidades discutidas nos anos anteriores e no vigente

5 Conclusão

O objetivo desse trabalho foi desenvolver uma ferramenta para coletar *tweets* sobre vulnerabilidades de segurança. Com o auxílio da API GetOldTweets, uma ferramenta usando a arquitetura *Desktop* foi desenvolvida. Ela permite a busca de um conjunto de tweets à partir de diversos filtros como busca de texto, usuário, datas de início e fim e limite de tweets. Os dados coletados são organizados e exportados em diferentes saídas como JSON, CSV, XML e SQL.

Um objetivo secundário do trabalho foi avaliar como a comunidade do Twitter discute o assunto vulnerabilidades de segurança. Foram coletados dados entre 2015 e 2019 usando a palavra chave 'CVE'. Foi possível identificar há aumento no número de *tweets* que mencionam sobre vulnerabilidades de segurança ao longo dos anos. Além disso, também foram conduzidas análises sobre as vulnerabilidades mais mencionadas e a relação entre vulnerabilidades catalogadas na NVD e mencionadas no *Twitter*.

Durante o desenvolvimento do trabalho houve dificuldade por questões técnicas da API utilizada. Tal fato que a API depende de uma rede de Internet não oscilante, ou seja, que tenha um conexão estável e que não haja percas de pacotes. Ademais, o computador que conecte à essa API deve apresentar uma memória principal de no mínimo 8 gigas, pois a utilização de memória é alta. Ainda sim foi possível intervir no problema a nível de software, alterando a nível de código para identificar quando o retorno está com dados incorretos. Para identificar esses erros foi coletado a quantidade de resultados de cada dia pesquisado, se em um dia não houvesse resultado, a busca era feita novamente para o intervalo.

Durante o desenvolvimento, também foi possível elaborar formas para melhorar a interpretação dos dados coletados. Diversos gráficos foram gerados para melhorar a compreensão acerca dos resultados.

A partir dos resultados obtidos foi possível responder às perguntas apresentadas no Capítulo 1:

1. *A discussão sobre vulnerabilidades de segurança em redes sociais está aumentando ao longo do tempo, assim como a descoberta de novas vulnerabilidades?* Os resultados apresentados indicam que sim. Os dados apresentados no Capítulo 4 fornecem uma indicação de que com o passar do tempo grande parte das vulnerabilidades divulgadas pela NVD estão sendo discutidas na rede social.

2. *Quais são os tipos de vulnerabilidades com maior incidência de discussão no Twitter?* A análise conduzida entre os anos de 2015 e 2019 mostrou que vulnerabilidades associadas a serviços populares como *Remote Desktop Services do Windows* (antigo *Terminal Services*), compilador C e pacote Office foram discutidas em mais de 14 mil *tweets*. Outra

característica importante das vulnerabilidades mais discutidas é a alta severidade associada a elas. A vulnerabilidade CVE-2019-0708, por exemplo, foi a mais discutida no período de análise. As razões por trás desse resultado podem estar associadas as características de tal vulnerabilidade: i) presente em sistemas Windows, ii) permite a execução remota de código e acesso remoto a máquina vulnerável e iii) pode se propagar em rede.

3. *Qual é a razão entre vulnerabilidades de segurança presentes em redes sociais e vulnerabilidades descobertas?* Os resultados mostram que para um dado ano, em média 91% das vulnerabilidades catalogadas pela NVD foram citadas no Twitter, naquele ano. Em 2017 esse número foi menor (84,5%) e em 2018 esse número foi maior 97,6%. Isso mostra que o Twitter fornece uma boa cobertura sobre a menção de vulnerabilidades. Um ponto importante e que não faz parte do escopo desse trabalho, envolve estudar a forma com que tais vulnerabilidades são mencionadas no *Twitter*. Por exemplo, um *tweet* feito por um robô que sempre compartilha as novidades da NVD deveria ter um peso menor do que usuários discutindo o impacto de uma vulnerabilidade recém descoberta.

Trabalhos futuros podem ser divididos em duas frentes: atualizações na ferramenta e uso da ferramenta em outras análises e contextos. Um exemplo de melhoria está relacionada ao problema de limitação de rede e memória na coleta dos dados. Além disso, melhorias na interface gráfica e alterar o sistema para um modelo cliente-servidor *online* são outras possibilidades.

Com relação a aplicação da ferramenta em outras análises, um ponto importante seria identificar o alcance (*retweets* e grupos de usuários de interesse, por exemplo) da discussão de uma vulnerabilidade presente no *Twitter*. Os dados gerados pela ferramenta também poderiam ser utilizados para auxiliar análises em contextos diferentes daqueles apresentados aqui. Um exemplo seria investigar o sentimento dos usuários acerca de assuntos relevantes como eleições e pandemia.

Referências

ABNT. ABNT NBR ISO/IEC 17799. p. 1–120, 2005. Disponível em: <http://www.fieb.org.br/download/senai/NBR_ISO_27002.pdf>. Citado 2 vezes nas páginas 14 e 15.

BATISTA, C. F. A. Métricas de Segurança de Software. 2007. PUC RIO, Rio de Janeiro, p.106. Citado 4 vezes nas páginas 20, 21, 23 e 24.

CERT. *US-CERT | United States Computer Emergency Readiness Team*. 2018. Acesso em: 07 de Setembro de 2020. Disponível em: <https://www.us-cert.gov/>. Citado na página 17.

COMMUNITY, P. *GetOldTweets3 PyPI*. 2019. Acesso em: 07 de Setembro de 2020. Disponível em: <https://pypi.org/project/GetOldTweets3/>. Citado 6 vezes nas páginas 26, 30, 31, 32, 34 e 37.

CORREIA, A. M. *Aprendizagem Automática em Larga Escala nas Redes Sociais para a Descoberta de Ameaças de Segurança*. 75 p. Dissertação (Mestrado) — Faculdade de Ciências, Departamento de Informática, Universidade de Lisboa, Lisboa, 2016. Disponível em: <http://hdl.handle.net/10451/24882>. Citado 5 vezes nas páginas 20, 21, 22, 23 e 24.

CVE. *CVE - Common Vulnerabilities and Exposures (CVE)*. 2018. Acesso em: 07 de Setembro de 2020. Disponível em: <https://cve.mitre.org/>. Citado 6 vezes nas páginas 6, 12, 13, 17, 28 e 38.

DANTAS, M. L. *Uma Abordagem Focada em Gestão de Riscos*. 1. ed. Olinda, PE: Livro Rápido, 2011. 5–147 p. ISBN 9788540600478. Disponível em: <http://www.marcusdantas.com.br/files/seguranca{_}informacao.> Citado na página 15.

FONTES, E. *Segurança da Informação*. 1. ed. [S.l.: s.n.], 2017. São Paulo: Editora Saraiva. ISBN 9788502122192. Citado na página 14.

Graz University of Technology. *Meltdown and Spectre*. 2018. Acesso em: 18 de Setembro de 2020. Disponível em: <https://meltdownattack.com/>. Citado na página 48.

KROTH, J. A. *Estudo e Análise de vulnerabilidades em serviÇOS ESTUDO E ANÁLISE DE VULNERABILIDADES EM SERVIÇOS*. 75 p. Dissertação (Bacharel em Engenharia da Computação, UNIVATES), Lajeado, RS, 2018. Citado 3 vezes nas páginas 20, 23 e 24.

MARTINELO, C. A. G.; BELLEZI, M. A. Análise de Vulnerabilidades com OpenVAS e Nessus. *Revista TIS*, v. 3, n. 1, 2014. Citado na página 16.

MIANI, R. S. *Um estudo sobre métricas e quantificação em segurança da informação*. 202 p. Tese (Doutorado em Engenharia Elétrica na área de Telecomunicações e Telemática) — UFU, Uberlândia, MG, 2020. Citado na página 11.

NEWSOME, J. et al. Vulnerability-specific execution filtering for exploit prevention on commodity software. In: *NDSS*. [S.l.: s.n.], 2006. Citado na página 47.

NIST. *NVD - Home*. 2018. Acesso em: 07 de Setembro de 2020. Disponível em: <https://nvd.nist.gov/>. Citado 2 vezes nas páginas 12 e 17.

OffSec Services Limited. *Exploit Database*. 2020. Acesso em: 18 de Setembro de 2020. Disponível em: <https://www.exploit-db.com/>. Citado na página 47.

OSVDB. *OSVDB | Everything is Vulnerable*. 2002. Disponível em: <https://blog.osvdb.org/>. Citado na página 17.

PEOTTA, L.; GONDIM, P. Análise de Risco em Ambientes Corporativos na Área de Tecnologia da Informação. *SEGeT – Simpósio de Excelência em Gestão e Tecnologia*, p. 1–12, 2006. Disponível em: <http://ww.aedb.br/seget/artigos07/1049{_}risco.> Citado 3 vezes nas páginas 11, 17 e 18.

PFLEEGER, C. P.; PFLEEGER, S. L.; MARGULIES, J. *Security in Computing, ProQuest Safari Tech Books Online*. [S.l.: s.n.], 2015. ISBN 0975442201. Citado na página 17.

SABOTTKE, C.; SUCIU, O.; DUMITRAŞ, 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. Citado 5 vezes nas páginas 20, 21, 22, 24 e 28.

SECUNIA. *Computer Security Research - Secunia*. 2009. Acesso em: 07 de Setembro de 2020. Disponível em: <https://secuniaresearch.flexerasoftware.com/community/research/>. Citado na página 17.

SPANCESKI, F. R. Politica de segurança da informação–Desenvolvimento de um modelo voltado para instituições de ensino. *Monografia do Trabalho de Conclusão de Curso em Sistemas de Informação*, 2004. Citado na página 14.

STATISTA. *Twitter: number of active users 2010-2019 | Statista*. 2019. 1 p. Acesso em: 07 de Setembro de 2020. Disponível em: <https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/>. Citado 2 vezes nas páginas 6 e 21.

The MITRE Corporation. *CVE - Download CVE List*. 2018. Acesso em: 07 de Setembro de 2020. Disponível em: <https://cve.mitre.org/data/downloads/index.html>. Citado na página 37.

UTO, N.; MELO, S. Vulnerabilidades em aplicações Web e mecanismos de proteção. *Minicursos SBSeg*, p. 237–283, 2009. Citado na página 12.

WHITEMAN, M. E.; HERBERT, J. M. *Principles of Information Security, Second*. [S.l.]: Thomson Technology, India Edition, Pg.[198-199], 2011. Citado na página 16.

XIAO, C. et al. From patching delays to infection symptoms: using risk profiles for an early discovery of vulnerabilities exploited in the wild. In: *27th {USENIX} Security Symposium ({USENIX} Security 18)*. [S.l.: s.n.], 2018. p. 903–918. Citado 2 vezes nas páginas 6 e 16.