Um Arcabouço para Processamento Escalável de Vulnerabilidades e Caracterização de Riscos à Conformidade da LGPD

Lucas M. Ponce,[1] **Matheus Gimpel,**[1] **Indra Ribeiro,**[1] **Etelvina Oliveira,**[1]
Ítalo Cunha,[1] **Cristine Hoepers,**[2] **Klaus Steding-Jessen,**[2]
Marcelo H. P. C. Chaves,[2] **Dorgival Guedes,**[1] **Wagner Meira Jr.**[1]

[1]Departamento de Ciências da Computação
Universidade Federal de Minas Gerais (UFMG)

[2]CERT.br - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança
NIC.br - Núcleo de Informação e Coordenaçãoo do Ponto BR

{lucasmsp,matheusgimpel,indramatsiendra,etelvinaoliveira}@dcc.ufmg.br
{cunha,dorgival,meira}@dcc.ufmg.br {cristine,jessen,mhp}@cert.br

Abstract. *Search engines like Shodan play an important role in the device mapping process and vulnerability tracking. However, the integration, the coding and the analysis processing by domain experts may be complex due to the large volume of data generated by these systems. Our work features a new, high-level abstraction for efficient analysis and an API for easy data integration. In our abstraction, code complexity is hidden by using a set of functional operators close to the domain area. We validated the usability of our library based on a case study involving critical vulnerabilities to the LGPD. Our results identified many accessible databases and systems being exposed for months with multiple high-risk confidentiality flaws.*

Resumo. *Sistemas de busca como o Shodan desempenham um papel importante no processo de mapeamento de dispositivos e no rastreamento de vulnerabilidades. No entanto, a integração, a codificação e o processamento das análises pelos especialistas do domínio podem ser complexas devido ao grande volume de dados gerado por esses sistemas. Nosso trabalho apresenta uma nova abstração, de alto nível, para análises eficientes, e uma API de bases externas para fácil integração. Em nossa abstração, as complexidades do código são ocultadas a partir de operadores funcionais próximos ao contexto da área de domínio. Validamos a usabilidade da nossa biblioteca a partir de um estudo de caso envolvendo vulnerabilidades críticas para a LGPD. Nossos resultados identificam uma série de bancos de dados acessíveis na internet e sistemas sendo expostos por meses com várias falhas de alto risco de confidencialidade.*

1. Introdução

Hoje em dia, o processamento de grandes volumes de dados, também conhecido como *Big Data*, vem sendo aplicado em diversos tipos de aplicações e áreas de conhecimento. Uma dessas áreas é a descoberta de ameaças e vulnerabilidades de rede, impulsionada pelo aprimoramento de técnicas de monitoramento [Durumeric et al. 2015] e pela necessidade das organizações de se protegerem do aumento de ataques cibernéticos [IBM Security 2021].

Nesse contexto, sistemas de buscas como Censys[1], Shodan[2] e Zoomeye[3] desempenham um papel importante na análise de vulnerabilidades, sendo amplamente utilizados por pesquisadores e operadores de redes. Sistemas desse tipo varrem (*scan*) a Internet em busca de dispositivos executando aplicações acessíveis pela rede, permitindo o reconhecimento em larga escala de dispositivos, aplicações e suas vulnerabilidades. A utilização desses sistemas, no entanto, apresenta dois desafios: a integração de dados e a interface de processamento. Apesar desses sistemas focarem na coleta de informações factuais sobre os dispositivos e aplicações, muitas vezes, tais informações precisam ser enriquecidas com outras fontes de dados. Por exemplo, relacionar informações sobre a organização responsável pelo IP sondado permite avaliar melhor os riscos de uma vulnerabilidade e identificar padrões. Informações sobre aplicações, como datas de atualização de uma versão, podem auxiliar no entendimento de como as medidas de segurança são adotadas e a diagnosticar potenciais riscos além dos já identificados durante a sondagem. No entanto, apesar desses sistemas disponibilizarem uma interface *web* para a busca de registros, essa interface é limitada, essencialmente, à filtragem por campos. Integração com bases externas, expressões regulares ou outras transformações dos dados não são suportadas.

Devido a essas limitações, a aquisição de dados via API e o processamento local são necessários para a maioria das análises realizadas nesses motores de buscas. Um problema dessa abordagem, no entanto, é a necessidade do usuário, geralmente especialista em redes ou em segurança, em lidar também com as complexidades da escalabilidade do processamento. Por exemplo, o Shodan é capaz de coletar informações de aproximadamente 500 destinos por segundo. Nesse cenário, mesmo utilizando técnicas de amostragem, uma simples operação de filtragem pode ser inviável em uma arquitetura tradicional. Mesmo soluções baseadas no processamento a partir de repositórios, como o PostgreSQL ou o ElasticSearch, apresentam desafios devido à presença de ruídos, codificação de consultas complexas ou a integração com outras ferramentas.

O objetivo deste trabalho é propor uma plataforma para o processamento eficiente de dados de motores de busca, com uma interface mais próxima do contexto de análise de vulnerabilidades e de caracterização de rede (Seção 3). Nossa abstração de dados esconde diversas complexidades, especialmente os aspectos do processamento paralelo e distribuído dos dados, dos operadores de redes e analistas de segurança. Além disso, nossa solução provê mecanismos para integração de dados de fontes externas a partir de uma interface de acesso único, facilitando o enriquecimento de dados. Avaliamos que o ganho de desempenho da nossa abordagem pode ser ordens de magnitude mais rápido em relação a um processamento tradicional (Seção 4). Além disso, outra contribuição do nosso trabalho é o estudo de vulnerabilidades e desafios impostos pela Lei Geral de Proteção de Dados Pessoais (LGPD) na integridade e na confidenciabilidade dos dados na Internet brasileira utilizando nosso arcabouço (Seção 5). Para conhecimento, este é o primeiro estudo publicado deste tipo, realizado em larga escala (Seção 6). Neste contexto, nosso resultado permitiu a identificação de potenciais riscos de vazamentos de dados, diversos repositórios de dados expostos por meses sem proteção e sites infectados com códigos maliciosos. Informações desse tipo podem ajudar órgãos regulatórios a fiscalizarem a conformidade das organizações com as diretrizes da lei.

[1] https://censys.io

[2] https://shodan.io

[3] https://www.zoomeye.org

2. Mecanismos de busca de dispositivos e vulnerabilidades

Nesta era da Internet das Coisas, muitos mecanismos de busca de dispositivos vêm se popularizando. Um dos pioneiros foi o Shodan, lançado em 2009, capaz de identificar dispositivos e aplicativos em execução e acessíveis na Internet, incluindo endereço IP, sistema operacional, softwares e portas abertas. Com o tempo, outras soluções como o Censys e o ZoomEye surgiram, e embora tenham tido motivações diferentes, hoje em dia, eles vêm se convergindo em aspectos de funcionalidades. Atualmente, a distribuição geográfica dos IPs sondados, o intervalo de portas e, por sua vez, os tipos de aplicações detectadas são as principais diferenças de funcionalidades [Li et al. 2020, Bennett et al. 2021].

Em geral, motores de busca possuem um núcleo de funcionamento em comum: Centenas de módulos implementam lógica específica para conectar e coletar informações detalhadas de diferentes aplicações. Os resultados de uma sondagem possuem campos de diversos tipos, que dependerão da aplicação coletada durante a sondagem: campos textuais, como nome da organização responsável; campos numéricos, como a porta sondada; campos com lista de valores, como a lista vulnerabilidades; e campos complexos (sob uma subestrutura de JSON), como o campo MongoDB, no caso do Shodan, com informações sobre o *status* do serviço, a lista de *databases* e outras informações relevantes. Esses motores também utilizam catálogos de informações para enriquecer os dados com campos derivados, como o nome e versão de um produto ou o sistema operacional do dispositivo.

Devido à diversidade dos campos e ao volume de dados, o processamento desse tipo de dado é um desafio. É esperado que análises de vulnerabilidades suportem consultas complexas, a integração com fontes externas de dados e que sejam escaláveis na realização de tais consultas. Em nossa experiência, mesmo análises diretas como a relação das vulnerabilidades de um dispositivo, feitas por muitos utilizando apenas a interface *web*, podem ser beneficiadas por uma solução integrada. Por exemplo, vulnerabilidades podem evoluir ao longo do tempo, tendo seu risco ou sua documentação alterados.

3. Abstração de dados e biblioteca de funções

Nesta seção, apresentamos a abstração de dados e a biblioteca de funções desenvolvidas para auxiliar nas análises de vulnerabilidades, minimizando a complexidade de processamento de grandes volumes de dados para os especialistas do domínio. O *framework* completo é ilustrado na Figura 1. Dados sobre os dispositivos sondados são *convertidos* e persistidos em um formato mais eficiente (Seção 3.1). Informações podem ser complementadas a partir de outras bases externas, disponibilizadas a partir de *Crawlers*, responsáveis pela coleta e o arquivamento (Seção 3.2). Durante as análises, dados como os do Shodan podem ser acessados a partir de uma *abstração de dados* mais próxima do especialista do domínio (Seção 3.3) e integrados com as demais bases. Nossa abstração, bem como os coletores e funções desenvolvidas, estão disponíveis para pesquisadores e operadores de redes interessados em utilizá-los ou estendê-los para outros domínios.[4]

3.1. Transformação e armazenamento dos dados

Análise de vulnerabilidades geralmente envolve a busca por um evento específico (*e.g.,* um serviço ou uma vulnerabilidade), por um período, ou análises mais amplas, envolvendo todo o conjunto de dados, com ou sem a necessidade de integração com outras bases. Nosso objetivo é seguir uma abordagem em que seja mantido um bom nível de compressão dos dados, que possua escalabilidade e que seja flexível à diversidade de cenários.

[4]Disponível em: https://github.com/lucasmsp/tlhop-library

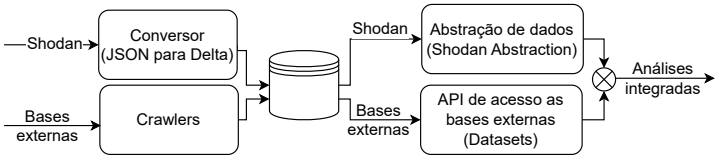

Figura 1. Framework de análises de segurança e redes integradas.

Para satisfazer estes requisitos, utilizamos o Apache Spark,[5] para a escalabilidade do processamento, e o formato Delta Lake para o armazenamento.[6] Uma etapa inicial consiste em adaptar os dados de origem dos motores de busca nesse formato mais eficiente. Para isso, implementamos um conversor responsável pela transformação dos dados enquanto realiza operações de limpeza e de padronização dos dados como a inferência das cidades e regiões brasileiras vinculadas aos dispositivos sondados, além da criação de novos atributos como o nome padronizado da organização vinculado ao IP sondados. Do ponto de vista de desempenho, nossa abordagem permite técnicas de otimizações de leitura, filtragem de registros e controles transacionais, como as descritas a seguir:

Otimização da leitura. Registros de motores de busca possuem grande diversidade de informações, o que tornaria o processo de leitura e inferência dos dados custoso se mantivéssemos os dados em um formato textual como o JSON. Nesse sentido, utilizar um formato binário comprimido e colunar, como o Delta Lake, garante ganhos de desempenho significativos, uma vez que a inferência da tipagem não é mais necessária a cada leitura. Formatos colunares também suportam outras otimizações, como o *Projection Pushdown*, que permite ler apenas as colunas necessárias para uma consulta.

Particionamento dos dados. A capacidade de filtragem dos registros a serem lidos durante uma tarefa, *i.e., Predicate Pushdown*, é outro ponto importante em consultas buscando um determinado evento ou durante junções. Embora arquivos tradicionais não possuam índices como em um banco de dados, formatos como o Delta implementam o particionamento de dados: Durante o processo de escrita, são criadas subpastas para cada valor das colunas utilizadas como chaves de partição. Dessa forma, ao se buscar por dados a partir dessas colunas, a ferramenta lê apenas os arquivos das pastas selecionadas.

A escolha das colunas para o particionamento é importante. É necessário que a dimensionalidade de cada coluna não seja alta, caso contrário, será criado um número muito grande de diretórios. Uma forma natural de particionamento é por ano e dia. Dessa forma, como o número de registros coletados geralmente é uniforme, a distribuição desses registros fica balanceada. Adicionamos também uma coluna que relaciona o módulo utilizado durante a sondagem. A ideia é que as informações de um registro estão relacionadas ao módulo utilizado, por exemplo, apenas os registros coletados pelo módulo *mongodb* terão informações acerca de dispositivos com MongoDB em execução. No entanto, como a frequência dos módulos é desbalanceada, ao invés de utilizar o nome do módulo como chave, criamos uma nova coluna, chamada de `meta_module`, a qual recebe como valor o mesmo nome do módulo de coleta utilizado pelo motor, caso sejam os módulos *"https"* (47,1% dos registros), *"http"* (22,5%), *"http-simple-new"* (11,4%), *"https-simple-new"* (5,1%), *"auto"* (4,4%), *"rdp"* (3,1%), e para todos os demais, que são menos frequentes (6,4%) receberá o valor fixo de *"mixed-modules"*.

Evolução do esquema e controle transacional. Como novos dados são lidos a cada dia,

[5]Apache Spark: https://spark.apache.org/
[6]Delta Lake: https://delta.io/

a evolução do esquema permite que novos campos possam ser adicionados. O suporte à atualização de registros e o controle transacional são outros pontos importantes. Por exemplo, é comum a presença de campos com falhas (*e.g.,* como a ausência da região de um estado brasileiro vinculada ao dispositivo sondado), sendo necessário a presença de processos de refinamentos dos dados *a posteriori*. Atualizações sobre os CVEs identificados, ao longo do tempo, também são frequentes. Sem esse suporte, haveria perda de informações ou a necessidade de reprocessamento dos dados.

3.2. API's para bases externas

Em nossa experiência, frequentemente é necessário mesclar dados de outras fontes, tanto para enriquecer as informações, quanto para possibilitar a computação de uma determinada métrica. No entanto, esse processo apresenta alguns desafios, como a coleta e verificação de atualizações dos dados externos, a conversão de dados, e a dificuldade do gerenciamento das diversas bases. Por exemplo, bases de vulnerabilidades como a do National Institute of Standards and Technology (NIST) são organizadas em arquivos anuais, atualizados diariamente. Novas atualizações precisam ser coletadas, mescladas com os anos anteriores, e convertidas em um formato estruturado. Tendo em vista esses desafios, implementamos duas APIs para gerenciar o uso das bases externas: *Crawlers*, onde implementamos todo o ferramental necessário para extrair os dados da Internet, notificar a existência de uma nova atualização, automaticamente, e persisti-la em um formato pós-processado/eficiente; e *DataSets*, que funciona como um ponto de acesso único.

Atualmente, são implementados 7 *crawlers*, conforme a Tabela 1 exemplifica. Todos os coletores fornecem métodos como `describe`, para descrever as principais informações da base, e `download` para cópia local, o pós-processamento da base e a sua persistência. Além dessas bases, ainda disponibilizamos outras 5 bases menores criadas pela equipe, que variam desde bases com o mapeamento dos códigos de requisições HTTP a um dicionário da língua portuguesa, utilizado em métodos internos da nossa API para a correção de determinados campos do Shodan mal-formatados.

3.3. Abstração de dados e funções

A representação de dados tabular tem se tornado popular em diversos *frameworks* de propósito geral de Ciência de Dados como Spark, Pandas e R. No entanto, ao utilizar tais sistemas na análise de vulnerabilidades, existem ainda uma série de operações de transformação, frequentes no processamento de dados de motores de busca, ausentes em sistemas gerais desse tipo. Abstrações genéricas podem também exigir uma lógica mais complexa envolvendo mais de um operador alinhado, o que torna o desenvolvimento do software mais laborioso. Nesse contexto, buscamos por uma abstração mais próxima das atividades relacionadas à análise de vulnerabilidades e de rede. Nossa solução é uma extensão da abstração de DataFrame do Spark com operadores comumente utilizados em análises desse tipo. Os operadores disponibilizados atuam como um *wrapper* de códigos internos mais complexos. Dessa forma, trechos de códigos mais frequentes ou operações alinhadas podem ser facilmente transformados em um novo operador.

Atualmente, são disponibilizados 18 operadores que variam desde a filtragem de campos específicos à transformações dos dados. O Código 1 exemplifica um dos nossos operadores para a filtragem de registros a partir de uma lista de CVEs. No Código 1(a), foi necessário utilizar apenas o nosso operador `contains_vulnerability`, enquanto na versão com apenas funções do Spark, Código 1(b), a filtragem demanda quatro funções,

Tabela 1. Bases de dados externas.

BASE DE DADOS	MODO
AS Rank da CAIDA	*crawler*
AS Classification da CAIDA	*crawler*
Banco de Vulnerabilidade do NIST	*crawler*
Cadastro de Pessoas Jurídicas do Brasil	*crawler*
Código de idiomas (ISO-639)	interno
Dicionário de palavras *pt-br*	interno
Informações de cidades brasileiras	*crawler*
Lançamentos dos S.O. da Mikrotik	*crawler*
Lista de palavras acentuadas (*pt-br*)	interno
Mapeamento de códigos UTF-8	interno
Registro de código de estado do HTTP	interno
Vulns. Exploradas Conhecidas da CISA	*crawler*

```
target = ["CVE-2021-26855",…,"CVE-2021-26885"]
from shodan_abstraction import DataFrame
df.contains_vulnerability(target, mode="any")
```
(a)

```
target = ["CVE-2021-26855",…,"CVE-2021-26885"]
import pyspark.sql.functions as F
tmp = F.array(*[F.lit(c) for c in target])
df.filter(F.arrays_overlap("cves_col", tmp))
```
(b)

Código 1. Exemplo de utilização do operador *contains_vulnerability*.

exigindo do usuário um conhecimento técnico maior da ferramenta. Além disso, se for alterada a condição para recuperar os registros com todas as vulnerabilidades (ao invés de pelo menos uma), uma outra sequência de passos tem que ser realizada. Um outro exemplo de operador é o `gen_meta_events`, responsável pela geração de um novo atributo, *meta-events*, que descreve informações importantes encontradas em cada registro. Esses itens mapeiam desde a presença de uma informação no registro como o rótulo *"has_vulns"*, se o registro possui uma lista de vulnerabilidades, ou o rótulo *"has_database"*, se o registro apresenta algum banco de dados exposto; mas também podem ser criados a partir de padrões mais complexos, como o rótulo *"has_malware_mars"*, se o registro possui um protocolo SMB exposto na Internet e a lista de compartilhamento inclui arquivos com extensão *"mars"*, identificando uma assinatura do *malware MARS* (discutido na Seção 5.4). Ao todo, implementamos 28 eventos a partir da nossa análise de dados, e novos eventos podem também ser estendidos por outros usuários.

4. Experimentos de desempenho

Nesta seção, avaliamos o desempenho do nosso arcabouço e sua capacidade de lidar com grandes volumes de dados. Todos experimentos descritos no presente trabalho foram executados em servidor Debian 11 com um ambiente Spark, versão 3.3.1, utilizando 20 *cores* de processadores Intel Xeon Silver 4114 e 40 GiB de RAM.

Nesse primeiro momento, implementamos apenas o conversor, discutido na Seção 3.1, para dados Shodan, no entanto, tudo o que foi realizado pode ser estendido para outros motores. O que mudará entre um conversor para o outro serão os nomes das colunas de entrada, que deverão ser padronizados para uma interface única. Os dados utilizados do Shodan são do período entre 01/01/2021 e 30/06/2021 compartilhados pelo Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br) em uma parceria para mapear vulnerabilidades no espaço de endereçamento IPv4 do Brasil. O período da amostra contém 165.167.269 registros JSON, separados por dia de coleta, e com um total de 118 campos distintos (média de 89 campos por registro), totalizando aproximadamente 347,5 GB comprimidos com bzip2.

O processo de conversão e refinamento dos dados obteve um custo médio de 301,6 segundos por dia. Mesmo com todo o enriquecimento e recursos do formato de dados escolhido, o tamanho da base resultante foi 324 GB, aproximadamente 7% menor que o original. Apesar desse custo inicial de conversão, o ganho de desempenho proporcionado

Tabela 2. Relação dos impactos por CVE.

IMPACTO	COMPLEX.	CONFID.	INTEG.
Nenhum	0,0%	32,7%	36,9%
Baixo	**85,2%**	19,1%	16,7%
Alto	14,8%	**48,2%**	**46,4%**

Tabela 3. Gravidade por CVE.

GRAU	QUANTIDADE
Média	80 (25,9%)
Alta	109 (35,4%)
Crítica	**119 (38,6%)**

Tabela 4. CVEs por ano de publicação.

ANO	2013	2014	2015	2016	2017	2018	2019	2020	2021
QUANTIDADE	2	2	2	**131**	**102**	45	20	3	1

por esse formato mais eficiente justifica todo esse processo. Comparamos o desempenho de dois tipos de experimentos envolvendo dados do Shodan: o primeiro avalia o impacto das otimizações de leitura do formato Delta em comparação ao formato original (JSON) e o segundo avalia o impacto do particionamento de dados.

No primeiro experimento, utilizamos uma consulta de contabilização de organizações com um determinado conjunto de vulnerabilidades. Nesse cenário, nossa abordagem reduz o tempo de execução em 10× (originalmente de 42,3 segundos). No segundo experimento realizamos uma junção dos dados do Shodan com uma segunda base que continha 110 *ids* de registros alvo. Diferentemente de uma abordagem tradicional utilizando apenas os *ids*, nossa abordagem utiliza internamente também os campos do particionamento, como dia de sondagem, para reduzir o espaço de busca, alcançando um ganho médio de 197,8× em execuções originalmente de 486,8 minutos. Essa grande diferença de desempenho se dá pela redução dos registros processados. Na primeira abordagem, todos os 165 milhões de registros (aprox. 300 GB) precisam ser lidos, já em nossa abordagem, apenas 24 milhões de registros são lidos (aprox. 17 GB).

5. LGPD: Caso de uso

A Lei Geral de Proteção de Dados Pessoais é uma legislação brasileira que entrou em vigor em setembro de 2020 visando regulamentar o tratamento de dados pessoais pelas empresas e instituições públicas e privadas no país.[7] Dentre diversos aspectos abordados por essa lei, a confidencialidade e a integridade dos dados são dois tópicos-chave.[8] Devido à recente implementação da LGPD, o presente trabalho avalia vulnerabilidades que impactam esses dois principais aspectos encontradas na rede brasileira e discute seus desdobramentos, demostrando a capacidade do nosso *framework* para o processamento de dados integrados de segurança. Nesse contexto, nossa base de dispositivos sondados pelo Shodan, no espaço de endereçamento brasileiro em 2021, permite obter um panorama da conformidade das organizações durante os primeiros anos de vigência da lei.

5.1. Caracterização das vulnerabilidades

As vulnerabilidades podem ser categorizadas por diversos critérios, incluindo a complexidade do ataque, o impacto na confidencialidade e na integridade dos dados. Segundo a base de dados de vulnerabilidades do NIST, das 60.440 vulnerabilidades com CVSS v3 calculado e com vetor de ataque via rede publicadas até julho de 2021, cerca de 80% são de baixa complexidade de exploração e com impacto alto ou crítico na integridade ou na confidencialidade dos dados. Na base do Shodan, a relação também é alta, cerca de 308 entre os 465 CVEs identificados no período (66%) impõem riscos à LGPD. Conforme

[7]LGPD: https://www.gov.br/cidadania/pt-br/acesso-a-informacao/lgpd

[8]A confidencialidade pode ser definida como a capacidade de garantir que os dados sejam mantidos em sigilo, enquanto a integridade como a garantia que os dados sejam corretos, autênticos e confiáveis.

Tabela 5. Vulnerabilidades do CISA.

CVE	CVSS	# IPs	DIAS ± STD
CVE-2019-0708	9,8	24.561	32,4 ± 53,7
CVE-2014-0160	7,5	8.863	56,8 ± 70,5
CVE-2020-0796	10,0	8.013	22,6 ± 44,6
CVE-2020-1938	9,8	2.614	69,6 ± 70,6
CVE-2021-26855	9,8	1.210	24,1 ± 37,6

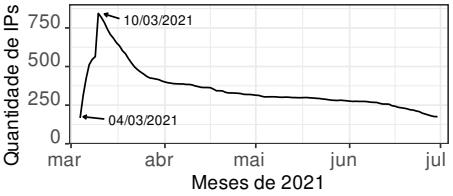

Figura 2. Quantidade do CVE-2021-26855 ao longo do tempo.

a Tabela 2, 85% dos CVEs possuem uma complexidade de ataque baixa, 48% com impactos altos na confidenciabilidade e 46% com impactos altos na integridade dos dados. Desses casos, a maioria são classificados com grau crítico ou alto (Tabela 3).

Quando avaliamos o ano de publicação dessas vulnerabilidades (Tabela 4), o cenário é ainda mais problemático. Grande parte dessas vulnerabilidades são conhecidas pela comunidade há 4 ou 5 anos, com potenciais *exploits* já bem documentados. Em geral, as vulnerabilidades são relacionadas a serviços *web* populares como Lighttpd (com 37,9% dos IPs com vulnerabilidades), Apache HTTP Server (26,9%), OpenSSH (20,7%), libGd (14,1%) e o PHP (14,1%). O Lighttpd, por exemplo, é um servidor *web* que atrai desenvolvedores por ser mais leve que o Apache. Observamos que as vulnerabilidades encontradas do Lighttpd estavam relacionadas às versões de 1.4.11 (ano de 2006) a 1.4.49 (2018), sendo que 85% dos IPs executando o serviço vulnerável possuíam a versão 1.4.39 (2016).Uma dessas vulnerabilidades é o `CVE-2014-2323` (CVSS 9,8), que permite que invasores remotos executem comandos maliciosos via injeção de SQL. Vulnerabilidades como essa possuem diversos *exploits* e provas de conceito disponíveis na Internet. Se considerarmos o número de vulnerabilidades distintas entre os serviços monitorados, o PHP apresentou o maior número, 103 (33% das vulnerabilidades avaliadas).

5.2. Vulnerabilidades conhecidas

Embora nossa caracterização inicial se limite a vulnerabilidades de severidade alta e com baixa complexidade, nem todas são conhecidas por serem exploradas na prática. Existem diversos fatores que restringem o potencial de exploração de uma vulnerabilidade, seja a necessidade de uma configuração específica ou até de uma autenticação inicial. A CISA (do inglês, Agência de Segurança Cibernética e Infraestrutura) dos EUA mantem um catálogo com as vulnerabilidades descobertas sendo utilizadas no mundo real.[9] Utilizando essa base externa,[10] identificamos 10 vulnerabilidades (das 308 iniciais) que possuem confirmações de serem exploradas em outros países. A Tabela 5 apresenta as 5 vulnerabilidades com maiores números de IPs.

O Bluekeep (`CVE-2019-0708`) está relacionado a um *bug* no Windows Remote Desktop que permite que usuários não autenticados executem código arbitrário por meio de uma solicitação mal-intencionada. *Patches* foram fornecidos pela Microsoft para o Windows 7 e o Server 2008, no entanto, outras versões afetadas como o XP, Vista e o Server 2003 não foram corrigidos. Embora o Heartbleed (`CVE-2014-0160`), segundo mais frequente, tenha o *score* de 7,5 hoje em dia, quando ele foi inicialmente descoberto, causou uma comoção na comunidade devido ao seu alto potencial de exploração.

Alguns dispositivos são corrigidos de 1 a 2 dias, no entanto, grande parte dos dis-

[9]Base de dados explorações conhecidas: https://www.cisa.gov/known-exploited-vulnerabilities-catalog
[10]Assim como o base do NIST, também suportada pela nossa API.

positivos permanecem vulneráveis por mais de 30 dias e alguns permanecem vulneráveis durante toda a amostra. Em nosso entendimento, há organizações em espectros distintos: aquelas com políticas de atualizações e correções frequentes e aquelas ainda sem uma metodologia bem estabelecida de segurança. O gráfico da Figura 2 exemplifica esse comportamento ao analisar a quantidade de IPs vulneráveis ao `CVE-2021-26855` ao longo do tempo. Esse CVE, relacionado a falhas no Microsoft Exchange Server, foi escolhido por estar na Tabela 5 e ser uma vulnerabilidade publicada durante o período da amostra. O CVE, publicado no dia 02/03/2021,[11] teve seu primeiro registro no endereçamento brasileiro pelo Shodan dois dias após sua publicação. O pico de IPs vulneráveis foi no dia 10 do mesmo mês. Logo após esse pico, grande parte dos dispositivos foram corrigidos nos 20 dias que se seguiram, provavelmente devido às atualizações automáticas no Windows. Após isso, é presenciado uma lenta taxa de decaimento ao longo do tempo.

5.3. Organizações com banco de dados expostos na Internet

Além de vulnerabilidades devido a brechas de segurança das aplicações, um dos grandes problemas relacionados a vazamentos de dados se dá por falhas durante a configuração. Em nossos experimentos, encontramos diversas aplicações de repositório de dados acessíveis pela Internet (Tabela 6), muitas delas com configurações padrão e até sem mecanismos de autenticação. Todos os eventos discutidos a seguir foram identificados pelo nosso operador `gen_meta_events`, apresentado na Seção 3.3, e que pode ser utilizado diretamente por outros pesquisadores em seus próprios conjuntos de dados.

CouchDB. Foram encontrados 380 instâncias do CouchDB acessíveis pela Internet. Até a versão 3.0, requisições para listar informações das tabelas disponíveis precisariam de autenticação, mas não exigiam permissões de administrador. Como o Shodan foi capaz de listar tabelas de 312 instâncias, significa que tais sistemas possuem falhas de autenticação ou utilizam configurações padrão. Uma vulnerabilidade publicada mais recentemente, `CVE-2022-24706` de *score* 9,8, aponta que versões anteriores a 3.2.2 estão susceptíveis a ataques em que é possível invadir uma instalação padrão e obter privilégios de administrador.[12] Dessa forma, grande parte dos dispositivos mapeados estariam vulneráveis, uma vez que a versão mais frequente foi a 1.7.1 (lançada em 2017).

ElasticSearch. Das 375 instâncias encontradas do ElasticSearch, o Shodan foi capaz de listar as tabelas (índices) e demais configurações do *cluster* em 316 IPs. Apesar de não termos certeza do nível de segurança desses sistemas, acreditamos que boa parte deles utiliza configurações padrão devido ao grau de informações coletadas. Por exemplo, a senha padrão de uma instância criada via Docker (169 instâncias) é "*changeme*". Já em uma configuração a partir do pacote de instalação, os mecanismos de segurança são desabilitados por padrão. Além disso, o Shodan possuiu pelo menos o privilégio de *monitor* durante as requisições para ser possível a listagem dos índices e do estado do *cluster*.[13]

InfluxDB. Foram encontrados 964 IPs executando a aplicação, dos quais 779 possuíam autenticação desativada. Embora não tenhamos acessado esses serviços por questão de ética, temos essa certeza, uma vez que o Shodan foi capaz de listar o nome das tabelas disponíveis, e no InfluxDB a operação de listar tabelas (`show`) só é permitida para usuários com permissões de leitura ou escrita. Além disso, a autenticação é desabilitada por padrão, todas as credenciais são ignoradas e todos os usuários têm todos os

[11] Mesmo dia do lançamento do *patch* de segurança da Microsoft.

[12] A exploração dessa vulnerabilidade já foi detectada pelo CISA.

[13] https://bit.ly/3VYEFsn

Tabela 6. Repositórios de dados.

APLICAÇÃO	# IPS	# IP VULNS
CouchDB	380	312
ElasticSearch	375	316
InfluxDB	964	779
MariaDB	2.223	2.220
MongoDB	1.537	194
MySQL	5.873	1.480
Redis	1.989	506
SMB	4.548	3677
SQL Server	28.357	8.408

Tabela 7. Lista dos *malwares* identificados.

MALWARE	# IPS
MEOW ATTACK	202
REDISWANNAMINE	279
MARS	344
BALADA INJECTOR	2

privilégios. Ou seja, pelo fato do Shodan ser capaz se listar as tabelas, significa que tais instâncias não exigem autenticação. Percebemos também que o tempo médio durante o qual uma instância fica em execução contínua é 70 dias, e em alguns casos encontramos instâncias há 714 dias sem nenhuma reinicialização, atualizações ou qualquer modificação do sistema. Ou seja, também potencialmente vulneráveis devido à falta de atualização.

MySQL e MariaDB. O MySQL talvez seja o sistema de banco de dados mais popular hoje em dia. Em nossa base, encontramos 5.873 instâncias que se comunicaram com o Shodan via método de autenticação por senha, o mesmo aconteceu para 2.223 instâncias do MariaDB (um *fork* do MySQL). Embora esse método tenha a compatibilidade com *drivers* antigos, ele é desencorajado pelos desenvolvedores. Além disso, um grande número de instâncias foram encontradas em versões já sem suporte. No caso do MySQL, 1.480 instâncias possuíam versões até a 5.5 (fim de suporte em dezembro de 2018) e, no caso do MariaDB, 2.220 instâncias possuíam versões sem suporte desde abril de 2020.

MongoDB. Das 1.537 instâncias detectadas, 194 não exigiam nenhum tipo de autenticação segundo o Shodan. Além disso, 354 instâncias possuíam versões sem suporte desde janeiro de 2020, sendo boa parte delas vítimas de *malwares* (Seção 5.4). Os repositórios possuíam em média 8,0 GB de dados armazenados com um valor máximo de 1,4 TB. Empresas que atuam no mercado de pagamentos digitais estão nessa lista.

Redis. Segundo a documentação do Redis, ele foi projetado para ser acessado dentro de ambientes confiáveis, não sendo recomendado expor a instância diretamente à Internet.[14] No entanto, 1.989 instâncias foram encontradas. Além disso, até a versão 6.0, o Redis não possuía mecanismos para criação de regras e limite de acesso a usuários. Em nossos dados, a versão mais frequente encontrada foi a 5.0 (72%). Embora o Redis tenha introduzido o *protected mode* (habilitado por padrão) na versão 3.2, na esperança de reduzir os riscos de segurança,[15] a realidade é que ainda existem administradores que desativam essa proteção, uma vez que, em 506 instâncias, o Shodan foi capaz de listar diversas informações, inclusive o nome das tabelas/chaves. Como será comentado na Seção 5.4, conseguimos detectar, inclusive, que algumas dessas instâncias foram vítimas de ataques.

SMB. Talvez seja o item mais crítico da tabela. Dos 4.548 IPs encontrados pelo Shodan com o protocolo acessível pela rede, pelo menos 3.677 deles possuem ao menos um arquivo sendo compartilhado com permissões de leitura e escrita acessível pela Internet. Desses 4.548 IPs, 94% deles executam em SMBv1, versão já obsoleta e desencorajada

[14] https://redis.io/docs/management/security

[15] Nesse modo, o Redis responde apenas às consultas das interfaces de *loopback* e retorna erro aos clientes que se conectam de endereços externos.

por várias instituições devido à presença de diversos ataques de *ransomware* relacionados, como será discutido na Seção 5.4. Em média, cada IP dessa lista compartilhou 96,1 arquivos (com máximo de 3.829). A lista de arquivos inclui *backups* de sistemas de gestão de empresas, *scripts* SQL, projetos do PowerBI e arquivos de nota-fiscais, alguns arquivos com mais de 20 GB e com casos extremos de 124 GB.

SQL Server. O Microsoft SQL Server é o sistema de banco de dados mais frequente em nossos dados, em cerca de 28 mil IPs. Embora as informações coletadas pelo Shodan não possuam muitos detalhes, identificamos que 8.408 instâncias executavam em Windows até a versão 7 SP1 (último suporte janeiro de 2020), versão obsoleta com pelo menos 2 mil vulnerabilidades conhecidas (sendo 423 classificadas como críticas). Versões mais antigas, como Windows XP e CE, também foram encontradas, mas em menores quantidades.

Discussão. Repositórios de dados expostos na Internet são uma grande ameaça a privacidade dos dados. Esses repositórios podem ser comprometidos por *hackers* que roubam dados confidenciais como informações de identificação pessoal ou dados corporativos. Sistemas sem autenticação comprometem, além do mencionado, a integridade dos dados, podendo ocasionar grandes prejuízos financeiros às corporações. Consideramos críticos até os cenários onde as permissões de acesso a esses bancos permitam apenas a exibição de tabelas, uma vez que tais informações podem servir de ajuda para ataques baseados em SQL *injection*. Por exemplo, foram encontradas diversas instâncias com nomes de tabelas criadas automaticamente por *softwares* conhecidos, facilitando a inferência do tipo de informação armazenada. Nesse contexto, nossa proposta pode ajudar as organizações a verificarem se há repositórios de dados acessíveis pela Internet em seus IPs.

5.4. Malwares

Por padrão, o Shodan é capaz de identificar a presença de certos tipos de *malwares*, como os *trojans* de Acesso Remoto (RAT). Em nossa amostra, por exemplo, seis *trojans* foram identificados, dentre eles estão o Bozok RAT (em 8 IPs), DarkComet (5) e o njRAT (4). Embora essa tabela aponte um número absoluto de IPs baixo, esse número pode não refletir o estado real da rede, uma vez que muitos *malwares* podem ter mecanismos diferentes de mascarar sua execução. Além disso, esses *trojans* apresentam um risco alto, por exemplo, o Bozok RAT, detectado em média por 48 dias em cada IP (em alguns casos durante todo o período da amostra), foi responsável por ondas de ataques na Internet entre 2015 e 2016 permitindo que criminosos obtivessem acesso total a um computador remoto.

Nessa seção, demostramos como nossa abstração de dados pode ser utilizada para identificar outros tipos de *malwares*, como os apresentados na Tabela 7: (i) *Meow attack*, um *ransomware* que substitui as tabelas de um repositório de dados por um recém-criado de sufixo *"-meow"*; (ii) *RedisWannaMine*[16], um ataque em repositórios Redis para a execução de *scripts* de mineração de criptomoedas enquanto busca por novos alvos; (iii) *ransonware MARS*,[17] *trojan* que explora pastas compartilhadas via protocolo SMB, criptografando diversos tipos de arquivos em uma extensão *".mars"* e deixando nota de resgate; e (iv) *Balada Injector*, que permite a geração de usuários administrativos em sites WordPress, coletando dados armazenados nos servidores e deixando *backdoors* para acesso persistente. Este último, por exemplo, ganhou atenção recentemente devido a relatórios de segurança relatando milhares de sites no mundo infectados desde 2017.[18]

[16]Servidores Redis: https://bit.ly/2HpZ6Xa

[17]Ransomware MARS: https://bit.ly/3HQSRxE

[18]Relatório de pesquisa de ameaças a sites: https://bit.ly/42zXyE5

Os três primeiros são baseados em ataques a arquivos, geralmente para fins de resgate. Por causa das assinaturas desses ataques, podemos identificar os sistemas violados a partir das informações já coletadas pelo Shodan, como o nome dos arquivos ou tabelas. Para o *Meow attack*, identificamos rastros em sistemas como o ElasticSearch (138 IPs), MongoDB (34) e CouchDB (30). Em muitos casos, o comportamento é o mesmo, durante todo o período da coleta, esses sistemas permaneceram com as tabelas infectadas. Em consequência, acreditamos que tais sistemas não estavam sendo utilizados ou monitorados. De qualquer forma, a situação ainda é alarmante, pois muitos deles possuem uma fraca ou nenhuma autenticação e estão expostos na rede, permitindo outros tipos de ataques e podendo servir de ponte de acesso à rede local do usuário.

Similar ao *Meow attack*, o *RedisWannaMine*, foi encontrado em 279 IPs com Redis. Na média, os dispositivos ficaram 44,6 dias infectados, com mínimo de 1 e máximo de 176 dias. No entanto, um dos tipos de ataques mais abrangentes na base foi o *ransonware MARS*, com 344 IPs infectados (de 4.548), totalizando cerca de 41 mil arquivos infectados. Nessa lista de arquivos foram encontrados arquivos de nota fiscais, arquivos de tesouraria, *backups* de banco de dados, alguns chegando a 29 GB. Outros tipos de ataques de resgate de dados, não vinculados a um *malware* em específico, também foram encontrados em outros 131 IPs executando ElasticSearch, 121 IPs com MongoDB e 28 IPs no CounchDB. Nesses casos, *hackers* apagam todo o repositório deixando notas de resgate como "*read_me_to_recover_your_data*" para pagamentos em bitcoins,[19] e dando às empresas, em média, dois dias para o pagamento (prazo relacionado ao limite de notificação de um vazamento de dados, segundo a regulamentação europeia).

Além dos ataques relacionados à base de dados, encontramos rastros de ataques como o do *Balada Injector* a partir de buscas de código mal-intencionado em páginas *web*. O código malicioso desse *malware* em específico consiste em um código Javascript, com a ofuscação via `String.fromCharCode`. Tal código é responsável por carregar *scripts* maliciosos em subdomínios aleatórios onde o ataque acontece. Na amostra, dois exemplos de sites foram encontrados, o primeiro de uma cantora brasileira, que ficou no máximo 48 horas com o *exploit* na sua página de entrada, já o segundo, de uma empresa de geração de energia elétrica, que ficou 53 dias com o *malware* em sua página.

Discussão. Grande parte do sucesso dos ataques cibernéticos se deve a falhas de segurança, por exemplo falta de atualização dos *softwares*, medidas de autenticação fracas ou topologia de rede. Com o número de ataques sendo cada vez mais frequentes, organizações e instituições precisam se preocupar mais com a segurança da rede e de seus dados. Hoje em dia, além do maior impacto de um vazamento de dados, pode haver penalizações legais para as organizações devido a LGPD. Uma recente pesquisa, por exemplo, aponta que cerca de 88% das empresas que já sofreram um ataque de *ransomware* admitem que optariam por pagar o resgate caso voltassem a ser atacadas.[20] Fato que exemplifica o impacto desse tipo ataque.

Nesse contexto, apresentamos nessa seção como nossa API pode complementar as capacidades de identificação de vulnerabilidades do Shodan ao mapear diversos traços de ataques de *malwares* e de vazamento de dados encontrados nos dispositivos sondados. Esse mapeamento pode ser estendido para outros tipos de ataques, agilizando o processo de identificação. Além disso, nosso arcabouço pode ser utilizado como uma fonte de

[19]Notas de resgates em servidores MongoDB: https://zd.net/44CcYcN

[20]Após ataque, 90% das empresas vítimas de ransomware pagariam resgate: https://bit.ly/42y0roU

dados alternativa para sistemas de mapeamento de incidentes cibernéticos,[21] uma vez que muitos desses sistemas dependem da notificação da própria organização envolvida.

5.5. Limitações dos resultados

Por questões de ética, todas as informações sobre os IPs vieram no Shodan. A utilização de dados desse tipo possui duas limitações: a frequência de coleta e a confidencialidade das detecções. Como sistemas desse tipo sondam IPs em intervalos de tempo aleatórios, quando avaliamos o tempo em que um dispositivo ficou vulnerável, assumimos que seu estado é constante no intervalo entre coletas consecutivas. Além disso, motores de busca como o Shodan vêm se especializando ao longo do tempo em suas capacidades de inferências, tornando os resultados mais confiáveis. No entanto, como os sistemas desse tipo geralmente não tentam explorar a falha, podem existir alguma contra medida dos administradores dos dispositivos que não é capturada pelo Shodan. Mesmo assim, as Seções 5.3 e 5.4 validam a vulnerabilidade dos dispositivos e confirmam suas explorações.

6. Trabalhos Relacionados

Grande parte dos estudos em motores de busca [Al-Alami et al. 2017, Raikar e Maralappanavar 2021] são limitadas a análises utilizando a interface *web*, sem a possibilidade de pós-processamento nos dados, algoritmos mais complexos ou a integração com outras bases. Esse fato talvez tenha contribuído para a baixa quantidade de pesquisas relacionados à LGPD utilizando tais sistemas. Atualmente, muitas dessas pesquisas geralmente se limitam a tópicos conceituais, como os aspectos de segurança envolvidos desde a coleta ao armazenamento de dados pessoais [Nobre et al. 2019] ou a discussão de metodologias para ajudar as empresas a entrarem em acordo com a lei [Silva e Borges 2021]. Mesmo as pesquisas no exterior [Daskevics e Nikiforova 2021] sobre o Regulamento Geral sobre a Proteção de Dados (GDPR) europeu, vem utilizando tais sistemas ainda em tarefas simples como a recuperação de IPs.

Embora existam diversas ferramentas de código aberto baseadas nas APIs nativas dos motores, até a data presente de publicação deste artigo, não encontramos nenhuma ferramenta que permita o processamento escalável de dados em lote como a da nossa proposta. O Grinder Framework[22] é talvez um dos arcabouços mais próximos do nosso trabalho. Por esse arcabouço, é possível que usuários façam requisições aos motores de busca a partir de uma sintaxe única, fazendo a mesclagem de algumas informações, gerando mapas e estatísticas. A principal diferença da nossa abordagem é o modo de acesso e de processamento aos dados. No Grinder, por exemplo, a cada consulta realizada é necessário requisições aos motores, não há suporte a reutilização de dados passados e cabe ao usuário estabelecer limites de resultados em cada consulta para evitar que os créditos se esgotem ou para evitar que a quantidade de dados extrapole o espaço de memória.

Por causa dessa falta de arcabouços para facilitar o processamento de grandes volumes de dados, algumas pesquisas armazenam seus dados em repositórios como o ElasticSearch ou Splunk [Novianto et al. 2021, Huq et al. 2017]. Embora o armazenamento escalável e a recuperação dos dados seja garantida a tais sistemas, o suporte a operações durante consultas são limitadas a transformações simples, filtragens ou agregações. Alinhar múltiplas operações de processamento em uma consulta, por exemplo, costuma exigir grande experiência dos usuários em sua sintaxe. Além disso, dados persistidos são

[21] Incidentes Notificados ao CERT.br: https://stats.cert.br/incidentes/

[22] Disponível em: https://github.com/sdnewhop/grinder

estáticos e não editáveis. Já a nossa abordagem, além de fornecer uma sintaxe amigável para execução de operações complexas sobre os dados, reduzindo o tempo de aprendizado da plataforma, permite a integração de outras fontes de dados e a atualização de registros.

7. Conclusão e Trabalhos Futuros

Este trabalho apresenta um arcabouço escalável para análises de vulnerabilidades utilizando dados de rastreamento como os do Shodan. A abstração de dados permite que operadores de redes analisem grandes volumes de dados, enquanto reduz a complexidade da codificação. Demonstramos como análises podem ser facilmente integradas a dados de diferentes fontes. Estudamos as principais vulnerabilidades relacionadas a impactos na LGPD, como vazamento de dados. Os resultados apontam uma série de aplicações vulneráveis com alto risco à integridade e confidencialidade de dados. Esperamos que a nossa proposta possa ser estendida por outras equipes e outros cenários de análises. Trabalhos futuros incluem novos operadores e um acompanhamento das vulnerabilidades encontradas em dados mais recentes. Esperamos que o tempo adicional da vigência da lei dê prazo para as organizações melhorarem a segurança de seus ativos.

Agradecimentos

Este trabalho foi parcialmente financiado pelo NIC.br, RNP/CTIC (2955), FAPEMIG, CNPq, CAPES, MASWEB, INCT-Cyber e IAIA (INCT para IA).

Referências

Al-Alami, H. et al. (2017). Vulnerability scanning of IoT devices in Jordan using Shodan. In *Int. Conf. on the Applications of Inf. Tech. in Developing Renewable Energy Processes Systems*.

Bennett, C. et al. (2021). Empirical scanning analysis of Censys and Shodan. In *Workshop on Measurements, Attacks, and Defenses for the Web*.

Daskevics, A. e Nikiforova, A. (2021). ShoBeVODSDT: Shodan and Binary Edge based vulnerable open data sources detection tool or what Internet of Things Search Engines know about you. In *Int. Conf. on Intelligent Data Science Technologies and Applications*, pages 38–45.

Durumeric, Z. et al. (2015). A Search Engine Backed by Internet-Wide Scanning. In *Proc. of ACM SIGSAC Conf. on Computer and Comm. Security*.

Huq, N. et al. (2017). US Cities Exposed in Shodan. Technical report, Trend Micro. Disponível em: https://bit.ly/3HOrXGXf. Acessado em 26/05/2023.

IBM Security (2021). Cost of a data breach report 2022. Technical report, IBM. Disponível em: https://ibm.co/3M7gXH6. Acessado em 26/05/2023.

Li, R. et al. (2020). A Survey on Cyberspace Search Engines. In *China Cyber Security Annual Conference*, pages 206–214.

Nobre, J., Lopes, R., Gomes, M., e de Oliveira, N. (2019). Segurança da Informação para Internet das Coisas (IoT): uma Abordagem sobre a Lei Geral de Proteção de Dados (LGPD). *Revista Eletrônica de Iniciação Científica em Computação*, 17(4).

Novianto, B. et al. (2021). Vulnerability Analysis of Internet Devices from Indonesia Based on Exposure Data in Shodan. *IOP Conf. Series: Materials Science and Engineering*, 1115(1).

Raikar, M. M. e Maralappanavar, M. S. (2021). Vulnerability assessment of MQTT protocol in Internet of Things (IoT). In *Int. Conf. Cyber Secur.*, pages 535–540.

Silva, R. D. e Borges, L. (2021). A implementação da Lei Geral de Proteção de Dados: Um estudo de caso sobre a LGPD na cooperativa de crédito na cidade de Franca-SP. *Revista Eletrônica de Computação Aplicada*, 2(2).