Identificação de Serviços e Dispositivos em Dados de Motores de Busca para o Enriquecimento de Análise de Vulnerabilidades

Lucas M. Ponce,[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 Ciência 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,indramatsiendra,etelvinaoliveira}@dcc.ufmg.br

{cunha,dorgival,meira}@dcc.ufmg.br {cristine,jessen,mhp}@cert.br

Abstract. *The enumeration of network-connected assets is an important step in vulnerability analysis. In this context, the use of search engines like Shodan has become popular for identifying services and devices accessible through the Internet. However, the information collected by these engines is incomplete and often does not keep pace with the speed at which new services emerge. This paper presents a solution for efficient service enumeration based on fingerprints. To validate our solution, we compared the information obtained by our framework with that provided by Shodan. For example, our solution enables the increase in the identification of services such as the operating system by 1.6 times and hardware information by up to 14 times. We also present two use cases of how our framework can assist in vulnerability analysis by providing more accurate information.*

Resumo. *A enumeração dos ativos conectados à rede é uma etapa importante na análise de vulnerabilidades. Nesse contexto, a utilização de motores de busca, como o Shodan, vem se tornando popular para a identificação de serviços e dispositivos acessíveis pela Internet. No entanto, as informações inferidas por esses motores nem sempre são completas e, muitas vezes, não acompanham a velocidade com que novos serviços surgem. O presente trabalho apresenta uma solução para a enumeração eficiente de serviços a partir de fingerprints. Para validar nossa solução, comparamos as informações obtidas pelo nosso arcabouço com as fornecidas pelo Shodan. Por exemplo, nossa solução permite o aumento da identificação de serviços, como o sistema operacional, em 1,6 vezes e informações sobre o hardware em até 14 vezes. Apresentamos também dois casos de uso que mostram como nosso arcabouço pode auxiliar na análise de vulnerabilidades fornecendo informações mais precisas.*

1. Introdução

A análise de vulnerabilidades consiste na identificação e avaliação de falhas, e potenciais ameaças à segurança de um sistema, que podem ser exploradas comprometendo a

integridade, disponibilidade ou confidencialidade das informações [Microservice 2022]. A primeira etapa desse processo envolve a identificação dos ativos e a enumeração de serviços para que, posteriormente, sejam identificadas suas vulnerabilidades. Nesse contexto, a utilização de motores de busca vem se tornando uma abordagem popular não apenas na identificação de dispositivos e serviços conectados à Internet, mas também para auxiliar etapas de identificação e priorização de vulnerabilidades [Markowsky e Markowsky 2015, Albataineh e Alsmadi 2019].

Atualmente, diversos motores de busca estão disponíveis no mercado, como o Censys, Shodan e o Zoomeye.[1] Ferramentas desse tipo sondam a Internet em busca de dispositivos acessíveis pela rede, coletando informações sobre os serviços identificados. No entanto, como motores de busca são geralmente fechados e comerciais, a capacidade de identificação de novos tipos de serviços não acompanha o surgimento de novas aplicações. Para contornar essas limitações, alguns trabalhos propõem combinar as informações de múltiplos motores de busca para maximizar os resultados [Markowsky e Markowsky 2015, Ponce et al. 2024]. Outros focam em criar seus próprios mecanismos de enumeração de serviços ou vulnerabilidades [Genge e Enăchescu 2016]. Trabalhos como estes geralmente envolvem duas etapas: (i) a identificação de um serviço, vulnerabilidade ou qualquer outro alvo de interesse; e (ii) o enriquecimento dos dados a partir da adição estruturada das informações relacionadas ao item identificado, como a versão de um produto. No entanto, atividades como estas, em dados de motores de busca, são complexas, pois, além do processo de enriquecimento, envolvem questões como a necessidade do usuário gerenciar uma arquitetura de processamento escalável, capaz de lidar com os grandes volumes de dados de entrada.

O presente trabalho propõe um arcabouço para a identificação de serviços e o enriquecimento de dados de motores de busca a partir da inferência de dados utilizando *fingerprints*. No contexto de cibersegurança, *fingerprints* são os registros que aplicações deixam quando se interage com o serviço, os quais podem ser utilizados para inferir informações sobre o dispositivo ou o serviço a partir da identificação de padrões. Um padrão, nesse contexto, é formado por uma condição de correspondência, como uma expressão regular, e por informações relacionadas ao item avaliado. Por exemplo, o *fingerprint* `SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu3` , gerado após uma requisição em um dispositivo, nos informa não apenas que se trata de um serviço SSH, mas também a versão do seu protocolo, a ferramenta e o sistema operacional utilizado.

Nosso arcabouço permite o processamento eficiente de *fingerprints* a partir de padrões obtidos de diversas fontes disponíveis na Internet ou criados manualmente por especialistas, e aplicáveis em dados de motores de busca. Na seção 4, avaliamos a capacidade de inferência de informações, comparando-as com os dados fornecidos pelo Shodan. Em geral, o nível de informação inferido supera o fornecido pelo Shodan. Apresentamos também dois casos de uso que destacam a importância dessas informações para análises de vulnerabilidades mais precisas (seção 5). Por exemplo, a partir de um censo sobre a diversidade de serviços SSH acessíveis na Internet brasileira, diversas aplicações desatualizadas puderam ser encontradas, algumas até consideradas críticas.

[1]Censys (https://censys.io); Shodan (https://shodan.io); Zoomeye (https://www.zoomeye.hk)

2. Motores de busca e a identificação de serviços

Atualmente, o Shodan é um dos motores de busca mais populares de dispositivos conectados à Internet [Daskevics e Nikiforova 2021]. Lançado em 2009, sua lógica consiste em se conectar a endereços IP e portas aleatórios e acessíveis pela Internet para coletar informações sobre diferentes serviços disponíveis. Embora o funcionamento de cada motor possa variar, eles compartilham um núcleo comum: centenas de módulos implementam protocolos de comunicação para coletar dados na forma de registros (*banners*) das aplicações em funcionamento nas portas sondadas, a fim de inferir informações sobre os serviços, que variam de repositórios de dados a páginas *web*.

Existem diversas pesquisas que se concentram na identificação automática de *fingerprints* para o enriquecimento de dados de serviços, a partir de dados provenientes de sondagens feitas por motores de busca [Sarabi et al. 2023, Wang et al. 2022, Wang et al. 2009]. As abordagens variam desde a utilização de redes neurais para treinar modelos capazes de identificar um *fingerprint* [Sarabi et al. 2023] à utilização de técnicas de mineração de dados para identificar serviços a partir de textos e capturas de telas [Wang et al. 2022]. Embora as abordagens desse tipo geralmente superem as disponibilizadas pelos motores de busca, elas apresentam dois problemas principais: (i) exigem um grande conjunto de dados para identificar os diversos padrões entre sondagens, o que pode atrasar a capacidade de identificação de novos tipos de ferramentas; e (ii) as técnicas empregadas e o volume de dados podem exigir grande poder computacional.

Apesar da lógica interna utilizada em motores como Shodan ou Censys não ser revelada, por serem ferramentas baseadas em soluções tradicionais como o ZMap [Durumeric et al. 2015],[2] conjecturamos que grande parte (senão todos) dos casamentos de padrões sejam feitos a partir da identificação de *fingerprints* conhecidos cadastrados em um catálogo. Motores como o Shodan são capazes de sondar a Internet inteira pelo menos uma vez por semana, essa taxa com que novos dados são gerados limitaria a utilização de modelos de processamento mais complexos. Além disso, por serem ferramentas comerciais, utilizar técnicas automáticas para a identificação de serviços poderia gerar muitos falsos positivos, comprometendo a precisão da ferramenta.

Nesse sentido, o cadastro de termos e expressões regulares capazes de identificar serviços a partir de *fingerprints* geralmente é realizado por especialistas do domínio. Após o casamento de padrão (*match*) entre um *fingerprint* com uma expressão regular e consequente identificação de um serviço, o resultado de uma sondagem pode ser estendido para conter campos específicos do serviço identificado. Exemplos de campos incluem: (i) campos textuais, como o nome ou versão do sistema operacional; (ii) campos numéricos, como a porta sondada; (iii) campos com lista de valores, como a lista de vulnerabilidades; e (iv) campos complexos (em uma subestrutura JSON), como o campo MongoDB no Shodan, que contém informações sobre o *status* do serviço e a lista de bases de dados.

Uma vez que o resultado de um *scan*, seja coletado, este se torna disponível através da interface *web* do motor para consultas ou a partir de *dumps* para o processamento local. Nosso arcabouço permite expandir as informações derivadas de motores como o Shodan enquanto é capaz de lidar com os grandes volumes de dados gerados por esses sistemas. No entanto, ele oferece a flexibilidade de utilizar padrões disponibilizados pela comunidade na Internet, padrões criados manualmente por especialistas ou padrões inferidos a

[2]Zmap (https://zmap.io)

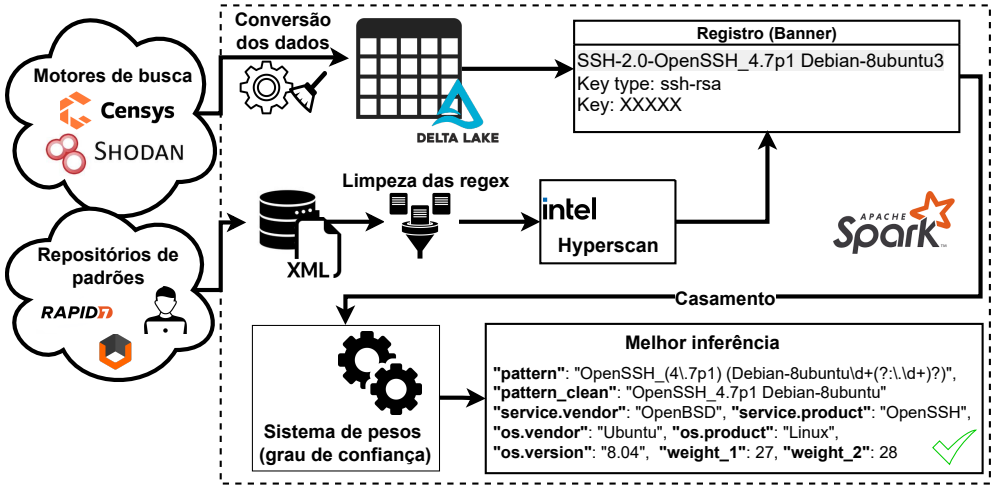

Figura 1. Arquitetura para a enumeração de serviços e dispositivos.

partir de análises sobre os próprios dados do motor, possibilitando o casamento de milhares de expressões regulares nos *fingerprints* de milhões de registros de forma eficiente. Na próxima seção, detalharemos a arquitetura e os componentes do nosso arcabouço, demonstrando como ele pode ser utilizado para aprimorar a análise de vulnerabilidades.

3. Arcabouço para enumeração eficiente de serviços e dispositivos

Nesta seção, apresentamos o arcabouço para a enumeração eficiente de serviços e dispositivos a partir das respostas das requisições presentes nos registros produzidos durante a sondagem de um dispositivo. Embora nossa pesquisa utilize registros produzidos por motores de busca como o Shodan, as técnicas propostas são diretamente aplicáveis a dados de outras ferramentas. Para isso, fornecemos uma interface comum para análise de dados, que reduz a complexidade de processamento de grandes volumes de dados para operadores de rede. Exemplificamos isso na figura 1: (i) dados dos motores de busca são convertidos em um formato mais eficiente; (ii) utilizamos uma ferramenta de processamento paralelo e distribuído para a execução de um motor de computação de expressões regulares; (iii) atribuímos um sistema de pesos que pode ser utilizado para ranquear os padrões encontrados em grau de especificidade do casamento realizado (*match*). Disponibilizamos o arcabouço como contribuição para a comunidade científica.[3]

3.1. Interface de consulta

Motores de busca são capazes de coletar informações de mais de 500 milhões de dispositivos ao redor do mundo várias vezes por mês. Processar grandes volumes de dados desse tipo pode ser desafiador, especialmente em tarefas de identificação de novas vulnerabilidades, onde o tempo é um fator importante. Para tornar o processamento de dados desses motores de busca mais eficiente, organizamos e comprimimos os dados no formato Delta Lake, um formato tabular, para processamento no Apache Spark.[4] Maiores explicações sobre os benefícios desse formato são discutidos em [Ponce et al. 2023].

Nosso arcabouço é baseado na abstração de dados tabulares do Spark (DataFrame), mas com a extensão de operadores e algoritmos que permitem aos usuários ex-

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

[4]Apache Spark (https://spark.apache.org/); Delta Lake (https://delta.io/)

pressar consultas complexas em dados de motores de busca em um nível de abstração elevado, delegando a construção e execução dos comandos Spark para nosso arcabouço. A identificação e o enriquecimento das informações proposto em nosso trabalho são exemplos de como atividades complexas podem ser encapsuladas a partir de operadores simples fornecidos pelo arcabouço. O funcionamento do algoritmo em questão pode ser dividido em três etapas: (i) coleta e limpeza dos padrões; (ii) execução das expressões regulares sobre os dados dos motores de busca; e (iii) priorização das inferências.

3.2. Coleta e limpeza dos padrões

Abordagens de identificação de serviços utilizando técnicas automáticas de descoberta de padrões, como as mencionadas na seção anterior, além de serem suscetíveis a erros, geralmente lidam apenas com a etapa de identificação, sem permitir inferir informações relacionadas. Mesmo inferências de informações simples derivadas dessa identificação, como a vinculação do sistema Windows ao saber que o produto sondado é o Microsoft Outlook, são limitadas. Por esse motivo, o enriquecimento de informações por identificação de serviços e dispositivos por casamento de *fingerprints* é uma abordagem promissora.

Nesse sentido, nossa abordagem se vale do cadastro de padrões em arquivos XML divididos por tipo de serviço. Cada elemento do XML corresponde a um *fingerprint*, com seu padrão regex e as informações que podem ser derivadas dele. Nessa primeira versão, atualmente, temos catalogados 2432 *fingerprints* obtidos a partir de fontes públicas na web; 2366 foram obtidas pelo projeto de código aberto Recog, um arcabouço em Ruby para identificação de serviços a partir de *fingerprints*, na qual as regras são selecionadas manualmente por especialistas do domínio; 46 foram obtidos na plataforma Vulners; e outros 20 foram implementados pela equipe.[5]

Como utilizamos padrões definidos por expressões regulares de diferentes fontes, realizamos um pré-processamento para limpar e padronizar as expressões regulares. Por exemplo, fontes como o Recog assumem que um registro apresentará apenas um tipo de serviço, o que nem sempre é válido para dados fornecidos por motores de busca. Por causa disso, nossa etapa de limpeza padroniza, por exemplo, a remoção de *case insensitive*, remoção de restrições das regexes para início e término do padrão e a possibilidade de remover expressões incorretas ou mal formatadas utilizando um arquivo de *blocklist*.

3.3. Execução das expressões regulares

Devido ao volume de dados gerado pelos motores de busca, nosso arcabouço utiliza o Apache Spark para tornar o processamento dos dados mais eficiente e escalável. Internamente, o Spark é capaz de dividir o processamento entre os diversos núcleos disponíveis, tornando toda a execução mais rápida. Embora o Spark tenha sua própria função para execução de expressões regulares, ela falha pelos seguintes motivos: (i) a função se resume a remover ou substituir *substrings* de um casamento, enquanto nossa proposta utiliza as expressões regulares como um mecanismo de busca, que, para um dado casamento, as informações derivadas sobre o *fingerprint* de entrada podem ser adicionadas como um novo campo no registro; (ii) em nosso contexto, um registro poderá ter mais de um *fingerprint* válido, sendo necessário adicionar o resultado em uma lista, uma função não disponível; (iii) a função nativa se resume a processar um padrão por vez, o que poderia ser inviável em cenários com centenas ou milhares de expressões. Por todos esses motivos, a utilização da abordagem adotada pelo Spark seria inviável.

[5]Recog (https://github.com/rapid7/recog); Vulners (https://vulners.com)

Para satisfazer esses requisitos, implementamos um novo operador para o Spark, que utiliza um motor de processamento de expressão regular que não apenas satisfaz nossos requisitos, mas é mais eficiente que o tradicional do Spark (o mesmo fornecido pela linguagem Java). Em nossa abordagem, utilizamos o Hyperscan [Project 2024], um mecanismo de expressões regulares implementado em C e projetado para oferecer alto desempenho. Cadastramos todas as expressões regulares no início da execução e a ferramenta aplica técnicas de otimização de autômatos para combinar múltiplas expressões, permitindo a execução simultânea de todas as expressões regulares em um texto de entrada.

Desse modo, o resultado fornecido pela ferramenta será uma lista dos identificadores de cada padrão que foram encontrados e o trecho do registro em que houve o casamento. Nosso algoritmo então agrupa todos os dicionários de informações a partir do ID em uma lista de dicionários (JSON). Após isso, utilizamos o trecho retornado pelo casamento de cada *regex* para a criação de um sistema de pesos que pode ser utilizado para priorizar ou atribuir um grau de certeza sobre os padrões encontrados. A ideia desse sistema de pesos é priorizar *fingerprints* cujas expressões regulares possuem mais termos constantes que outras. Nossa premissa é que as expressões cadastradas já foram previamente testadas, então um padrão que for encontrado sobre os dados já terá uma certa confiança. Mas, para casos onde mais de um padrão for encontrado, inclusive para um mesmo *fingerprint*, a melhor correspondência será aquela que trará mais especificidade na regra avaliada. Por exemplo, embora a melhor expressão regular vinculada ao registro exemplificado na figura 1 tenha sido `OpenSSH_(4\.7) (Debian-8ubuntu\d+(?:\.\d+)?)`, outras expressões como `OpenSSH_[\d\.]+` também seriam consideradas válidas. Para nós, a diferença entre os dois casos é que o primeiro, por exigir um maior número de termos fixados, garante uma maior especificidade sobre as inferências. Por exemplo, podemos derivar deste padrão o sistema operacional e a sua versão.

3.4. Sistema de pesos

A computação do peso para priorização se dá da seguinte forma: primeiramente, realizamos um processamento nas expressões regulares para remover caracteres que representam símbolos regulares como "\d" ou "\w". O resultado dessa etapa irá criar uma variável chamada "*pattern_clean*". Em seguida, contabilizamos os caracteres presentes em todas as subsequências contínuas entre o *fingerprint* e a *pattern_clean*. Nossa ideia é ponderar a quantidade de termos específicos presentes entre o casamento e a expressão. Em nossa abordagem, o tamanho da *pattern_clean* também atua como um segundo peso. Ainda na figura 1, o processamento da expressão regular utilizada no casamento nos retorna `OpenSSH_4.7 Debian-8ubuntu`, dessa forma dos 28 caracteres presentes nesse resultado, 27 foram encontrados no *fingerprint* `SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu3`, ou seja, apenas o caractere "3" não estava especificado na *pattern_clean*.

Utilizar esse sistema de pesos permite priorizar os padrões caso um mesmo tipo de serviço tenha sido identificado mais de uma vez. Padrões com pesos zero, por exemplo, indicam que o padrão possui partes totalmente aleatórias, um indicativo de que talvez o padrão possa ter uma baixa especificidade. No entanto, filtrar padrões por um limiar nem sempre é uma ideia viável. Por exemplo, o *fingerprint* `nginx/2.3` para a expressão `nginx/[\d\.]+` apontaria um peso de apenas 6, o que não significa que seja fraco. Nesse caso, a relação entre o peso 1 e o peso 2 nos ajudaria a indicar que 100% das partes constantes da expressão foram satisfeitas. O sistema de pesos criado permite que o usuário

Tabela 1. Relação do número de registros e IPs com a porcentagem de casamentos.

MÉTRICA	# BANNERS	# IPS
TOTAL	330.414.756	19.557.988
% INFERIDOS	59,8	68,9

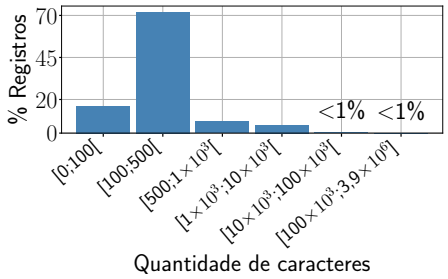

Figura 2. Quantidade de caracteres avaliados nos registros.

Tabela 2. Relação dos padrões por categoria de serviço.

FONTE	REGISTROS (%)	IPS (%)	PADRÕES (#)	PADRÕES (%)
HTTP (Servers)	41,34	36,25	402	90,33
DNS	9,15	25,01	57	81,42
HTML	8,53	19,61	195	43,33
SSH (Recog)	7,08	27,16	129	86,00
HTTP (Auth)	5,41	14,64	60	80,00
Apache (OS)	5,18	7,23	27	71,05
Telnet	4,32	18,63	57	39,58
HTTP (Cookies)	1,95	1,60	62	76,54
SNMP (SysDescr)	1,67	2,99	133	23,58
SMB (OS)	1,18	2,23	60	78,94
FTP	0,75	1,26	81	54,72
SSH (Outros)	0,25	0,51	65	98,48
SNMP (SysObjId)	0,02	0,12	7	16,66
SMB (LM)	0,01	0,05	5	62,50
NTP	$< 0{,}01$	$< 0{,}01$	1	1,33

decida a melhor abordagem para o seu contexto. Por exemplo, pode optar por selecionar apenas o padrão de maior peso caso esteja procurando *fingerprints* de um único tipo de serviço, ou pode utilizar como um ranqueamento, selecionando apenas os de interesse.

4. Avaliação experimental

Nesta seção, avaliamos os resultados das inferências obtidas utilizando nosso arcabouço. Inicialmente, apresentamos uma caracterização geral das inferências por número de banners e IPs. Na seção 4.2, discutimos mais a fundo sobre os tipos de padrões encontrados. Já nas seções 4.3 e 4.4 avaliamos a qualidade da solução na inferência de informações sobre o sistema operacional e sobre o tipo de dispositivo, respectivamente.

4.1. Caracterização inicial

Para a avaliação apresentada nesta seção, utilizamos dados do Shodan 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 conjunto de dados utilizado corresponde aos primeiros semestres de 2021 e 2023. A tabela 1 apresenta um sumário do número de IPs e registros da amostra, bem como uma relação com o número de casamentos obtidos pelas inferências dos 2432 padrões discutidos na seção anterior. Já o gráfico da figura 2 apresenta um histograma da quantidade de caracteres presentes no campo onde são armazenadas as sondagens do Shodan em formato de texto, utilizado para as nossas inferências. Cerca de 70% dos registros possuíam entre 100 e 500 caracteres em seu conteúdo de sondagem, no entanto, casos extremos de até 3,9 milhões de caracteres foram observados.[6] A execução completa do processo de inferência para os 330 milhões de registros (primeira coluna da tabela 1), salvando o resultado em um arquivo separado, levou aproximadamente 175 minutos.

Cerca de 55% dos padrões cadastrados (1341 dos 2432) tiveram pelo menos um casamento. Já o número de IPs com algum casamento foi de aproximadamente 69%. A

[6]Casos como esse representam sondagens sobre um repositório de dados como o ElasticSearch, onde um volume alto de informações é coletado sobre os nós e tabelas disponíveis.

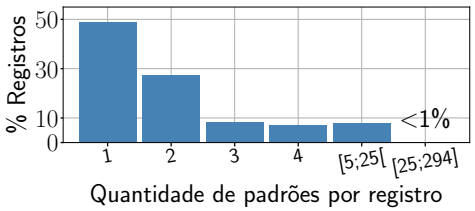

Figura 3. Histograma da quantidade de padrões por registro.

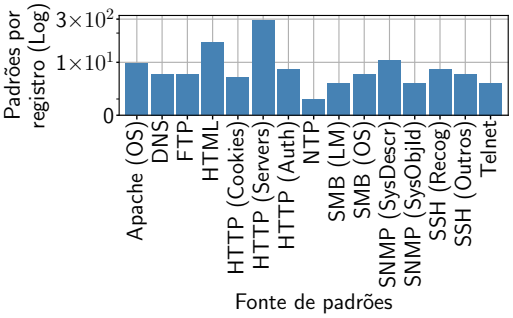

Figura 4. Número máximo de padrões de uma mesma fonte por registro.

diversidade dos padrões encontrados é sumarizado na tabela 2. Grande parte dos casamentos observados são para servidores *Web* (representado na primeira linha); ao todo, cerca de 41% dos registros (36% dos IPs) que tiveram algum casamento foram para esse tipo de serviço. No total, 402 padrões distintos foram encontrados no conjunto de dados, o que representa aproximadamente 90% dos padrões que foram encontrados por nossa ferramenta. Os casamentos a partir de serviços DNS ou em páginas HTML (p.ex., a partir dos títulos) ficam em segundo e em terceiro lugar, respectivamente. A maior quantidade de padrões encontrados para servidores HTTP está diretamente relacionada com o nosso contexto de análise: grande parte dos serviços identificados pelo Shodan são, de fato, serviços expostos pela Internet. Já outros serviços, como DNS ou SSH, embora sejam frequentes em número de IPs, a sondagem por parte do Shodan em portas que rodam tais serviços não é tão frequente, resultando em um número baixo de registros.

4.2. Relação dos padrões encontrados

A relação da quantidade de padrões com casamento em cada registro é capturada no histograma da figura 3. Em geral, foram encontrados apenas um ou dois casamentos por registro. No entanto, em alguns casos menos frequentes, até 294 casamentos foram encontrados em um único registro. Casos como esses representam cenários onde mais de uma categoria de serviço ou protocolo foi identificada, ou seja, mais de uma fonte de padrões (coluna 1 da tabela 2), ou até mesmo mais de um tipo de aplicação.

Cerca de 75% dos registros (Q3) tinham no máximo uma ou duas inferências de uma mesma fonte. No entanto, como se pode observar na figura 4, que relaciona o número máximo de casamentos de uma mesma fonte por registro, havia casos em que esse valor era maior. Padrões como HTML ou HTTP Servers são os casos mais extremos, para os quais foram observados registros com até 47 e 265 casamentos, respectivamente. Casos assim, no entanto, são esperados. Por exemplo, em um desses casos, a resposta da coleta do Shodan foi de aproximadamente 20 mil caracteres, contendo uma série de rastros de serviços *web*, servidores de impressora, serviços Windows e até serviços Linux. Esses casos provavelmente representam algum roteamento da rede, onde serviços de múltiplas máquinas foram acessíveis pelo Shodan a partir de um único IP.

Os cinco padrões com maior número de casamentos estão mostrados na tabela 3. A lista inclui páginas *web* não acessíveis, serviços DropBear (uma ferramenta leve que fornece um servidor e cliente SSH), roteadores Cisco, servidores de DNS BIND e gateways da Huawei. A identificação desses serviços pode ser utilizada para várias finalidades. Por exemplo, existem pelo menos 62 vulnerabilidades catalogadas no NVD que afetam diretamente versões do produto DropBear, que pode ser identificado a partir do

Tabela 3. Padrões mais frequentes.

PADRÃO	FONTE	# IPS
(?:404)?Not Found	HTML	3.178.775
*SSH-2.0-dropbear_.**	SSH (Outros)	2.454.469
(?:Basic|Digest) realm="Cisco_CCSP_CWMP_TCPCR"	HTTP (Auth)	1.667.539
(?:BIND)?([89]\.[\d\.]+(?:[ab]\d+)? \ (?:-ESV (?:-R\d+)?)?(?:[-SPW][\d\.]+)? \ (?:-REL)?(?:-W\d+)?(?:rc\d)?)(?:-NOESW)?	DNS	1.523.558
(?:Basic|Digest) realm="HuaweiHomeGateway"	HTTP (Auth)	592.601

Tabela 4. Relação dos principais tipos de informações obtidas pelas inferências.

TIPO	# PADRÕES	# CAMPOS	TOP 3 CAMPOS
APACHE	35	3	variant (10)
HARDWARE	297	9	vendor (261), device (245), product (81)
SIST. OPERACIONAL	674	11	vendor (604), product (516), family (446)
SERVIÇO	800	17	product (767), vendor (739), cpe (495)
SNMP	125	2	snmp.fpmib.oid.1 (1) e snmp.fpmib.oid.2 (1)

segundo padrão na tabela. Já o terceiro padrão pode ser utilizado para identificar dispositivos com CWMP acessíveis na Internet, um protocolo que tem sido usado pelo botnet Mirai em ataques a roteadores domésticos.[7] Embora os ataques não tenham sido causados por vulnerabilidades no próprio CWMP, os invasores conseguem tirar proveito de erros de configuração e de versões antigas: o protocolo original utiliza um serviço baseado em HTTP para gerenciamento remoto, que é inerentemente inseguro.

Nos casos apresentados na tabela 3, com exceção da quarta linha, os padrões só identificam um serviço, sem permitir atribuir uma versão ou outra informação mais específica, necessitando de uma segunda etapa de avaliação ou de padrões mais específicos. Isso pode ocorrer pois nem sempre existem informações nas *fingerprints* que forneçam esse tipo de detalhe. No geral, dos 1341 padrões que tiveram pelo menos um casamento, 800 possuem informações sobre algum serviço identificado, 674 sobre o sistema operacional e 297 permitem identificar algum tipo de informação sobre o hardware, como exemplificado na tabela 4 a partir dos cinco tipos de informações mais frequentes. A tabela também relaciona a quantidade de campos derivados em cada tipo de informação; por exemplo, para as informações sobre o sistema operacional, são divididas em até 11 campos, sendo que o campo *vendor* (fabricante) é o mais comum, presente em 604 padrões.

Informações derivadas, como as do sistema operacional, possuem um grau de detalhamento maior do que o reportado pelos motores de busca. Por exemplo, enquanto em muitos casos o Shodan fornece apenas o nome do sistema operacional, como "Linux" ou "Ubuntu Linux", utilizar a inferências a partir de padrões como `SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu3`, como já exemplificado anteriormente, permite não apenas identificar informações sobre a ferramenta SSH, mas também vincular ao sistema Ubuntu e a sua versão (8.04, codinome Hardy). A próxima seção discute com mais detalhes o enriquecimento de informações sobre o sistema operacional.

4.3. Identificação de informações sobre o sistema operacional

Como mostrado na tabela 4, cerca de 50,2% dos padrões identificados em nosso conjunto de dados fornecem pelo menos uma informação sobre o sistema operacional. Informações

[7] https://t.ly/hiV1R

Tabela 5. Inferências sobre o SO.

SHODAN	INFER.	% IPS	% BANNERS
✗	✗	69,08	75,96
✗	✓	11,90	18,21
✓	✗	2,70	1,77
✓	✓	16,32	4,06

Tabela 6. Inferências sobre o dispositivo.

SHODAN	INFER.	% IPS	% BANNERS
✗	✗	80,16	91,57
✗	✓	18,43	7,81
✓	✗	0,92	0,32
✓	✓	0,49	0,30

desse tipo complementam as informações dos motores de busca. A tabela 5 apresenta uma comparação da quantidade de IPs e de registros para os quais o nosso método conseguiu inferir informações de sistema operacional em relação ao Shodan. Nossa solução complementa os dados do Shodan com a informação do sistema operacional em cerca de 2,3 milhões de IPs (11,9% do total), enquanto que, em 16,3% (3,2 milhões), tanto o Shodan quanto a nossa solução foram capazes de identificar o sistema operacional. No total, nossa solução aumenta a informação sobre o sistema operacional nos IPs em 1,62 vezes.

Avaliar o conflito de informações entre o sistema operacional inferido e o fornecido pelo Shodan é uma tarefa complexa. Em nossa análise, consideramos apenas o padrão de maior peso para cada IP. Desta forma, a maior parte dos conflitos acontecem quando o Shodan fornece um sistema genérico, como "Linux" ao invés de sua distribuição "Ubuntu". A partir de um mapeamento dessas relações, grande parte dos conflitos deixa de existir. No entanto, ainda existem conflitos provocados por casos, como já exemplificado na seção anterior, onde as informações capturadas pelo Shodan consistem em um conjunto de serviços redirecionados a partir de diversas máquinas. Em um desses casos, enquanto o Shodan inferiu o sistema operacional do dispositivo sondado como sendo Windows, provavelmente por causa do *fingerprint* `ASP.NET` , um dos nossos padrões resultou na descoberta do produto `Helix Server Version 9.0.4.960 (linux-2.2-libc6-i586-server)` , e, por isso, foi atribuído um sistema Linux. Em casos assim, diversos serviços podem ser inferidos, pelo Shodan ou pela nossa solução, sendo necessário um pós-processamento para identificar esses casos e resolver ou atribuir prioridades em alguns tipos de padrão.

4.4. Identificação de informações sobre o *hardware*

Identificar o tipo de um dispositivo, seja um roteador, uma câmera IP ou um dispositivo de propósito geral (como nomeado pelo Shodan), é uma tarefa complexa. Em alguns casos, essa identificação pode ser realizada a partir de *fingerprints* semelhantes aos já mostrados anteriormente. O *fingerprint* `Digest realm="Hikvision"` é um exemplo de padrão frequente encontrado em nosso conjunto de dados que pode ser utilizado para identificar um dispositivo DVR, utilizado para gerenciar e gravar imagens de câmeras IP, fabricado pela Hikvision. Dispositivos como esse podem estar susceptíveis ao CVE-2014-4880 (CVSS 7,5), que permite que atacantes remotos executem códigos arbitrários junto ao cabeçalho de autorização em uma solicitação RTSP PLAY. Como exemplificado na tabela 6, o Shodan é capaz de inferir informações sobre o dispositivo em apenas 1,41% dos IPs sondados pela ferramenta. Nossa solução, por outro lado, o supera com uma taxa de inferência de 19,8%, identificando o tipo de dispositivo em 14,1 vezes mais IPs que o Shodan.

5. Casos de uso

O ganho de informações, na descoberta de um serviço em execução, no detalhamento do sistema operacional ou na descoberta do tipo de dispositivo, pode ser utilizado em diversos contextos na análise de vulnerabilidades. Esta seção apresentará dois casos de uso para ilustrar como informações obtidas pelo nosso arcabouço podem ser utilizadas

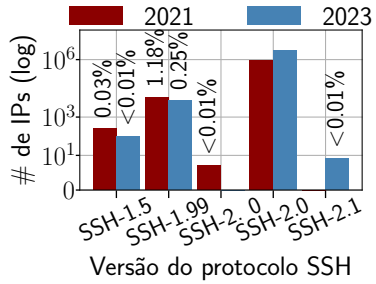

Figura 5. Diversidade do protocolo SSH no Brasil.

Tabela 7. Top 5 produtos SSH em 2023.

PRODUTO	% IPS	TOP (2016)
DropBear	77,72	2^o
OpenSSH	16,86	1^o
ROSSSH	2,25	4^o
Ausente	0,71	7^o
Comware	0,51	12^o

para avaliar vulnerabilidades na utilização de SSH na internet brasileira (seção 5.1) e para avaliar a popularidade de um sistema operacional (seção 5.2).

5.1. Revisitando o census sobre o protocolo SSH na Internet

Em 2014, um dos primeiros artigos a analisar a diversidade de serviços SSH na Internet a partir de sondagens a esses dispositivos foi publicado [Gasser et al. 2014]. Após isso, outras pesquisas, como o Internet Census de 2016, também foram publicadas [Popescu 2016]. A importância de tais pesquisas está relacionada ao fato do SSH ser um protocolo essencial em muitos contextos, e vulnerabilidades nesses serviços podem permitir que invasores tenham acesso às máquinas remotas. Um exemplo recente é a vulnerabilidade CVE-2024-3094 (*XZ vulnerability*), descoberta em março de 2024, que, a partir de falhas de segurança na biblioteca de compressão XZ, permite que algumas versões do OpenSSH sejam utilizadas como um *backdoor* malicioso para contornar o sistema de autenticação. O presente caso de uso revisita os censos anteriores, a partir de um novo mapeamento de informações utilizando dados mais recentes (de 2021 e 2023), e, dessa vez, focando em IPs do espaço de endereçamento brasileiro.

No trabalho de [Popescu 2016], 15.646.188 IPs espalhados pelo mundo com servidores SSH foram escaneados em 2016. Em nossa pesquisa, encontramos 3.643.220 IPs apenas no Brasil, dos quais 914.990 foram sondados pelo Shodan no primeiro semestre de 2021 e 2.957.615 em 2023 (229.385 apareceram nos dois períodos). É esperado que o número de instâncias SSH no geral tenha aumentado desde 2016; no entanto, é interessante observar que o número de instâncias no Brasil representa cerca de 23% do total de instâncias SSH sondadas naquela época globalmente. Quando comparamos nossos dados de 2021 e 2023, também há um aumento de 3,23 vezes mais instâncias sondadas pelo Shodan, embora o intervalo de tempo tenha sido igual nos dois períodos. Isso pode estar relacionado ao aumento da capacidade do Shodan em sondar novos dispositivos, mas o aumento de dispositivos conectados à Internet também pode ter sido um fator importante.

Também em 2016, os autores contabilizaram as versões dos protocolos. Comparamos algumas dessas informações com nossos dados, e elas estão exemplificadas na figura 5: (i) cerca de 95,8% dos serviços utilizavam o SSH de versão 2.0, essa quantidade é próxima ao observado em nossos dados (p.ex., em 2023, a razão era de 99,7%); (ii) SSH de versão 1.99 representava cerca de 4,12%, enquanto que em nossos dados, percebe-se uma redução de 2021 para 2023 (0,25%); (iii) algumas poucas instâncias (<0.01), mesma proporção do estudo de 2016, foram sondadas como "SSH-2. 0";[8] (iv) por fim, também encontramos versões do protocolo 1.5, grande parte relacionadas a dispositivos da Cisco e da Huawei, respectivamente. Dado que o SSH-1 possui inúmeras falhas de design que o tornam vulnerável, atualmente, essa versão é considerada obsoleta e deveria ser evitada.

[8]O espaçamento é um erro no OpenSSH de versão 3.8p1, lançado em 2004.

Tabela 8. Relação dos IPs com DropBear em 2023 e o ano da versão.

ANO	≤ 2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2022
% IPS	0.38	0.05	0.07	0.60	0.16	1.96	1.00	0.03	93.78	0.03	< 0.01

A popularidade das principais ferramentas de SSH é talvez uma das maiores diferenças entre a pesquisa de 2016 e os nossos dados (exemplificado na tabela 7, desta vez, considerando apenas o ano de 2023). Enquanto em 2016, o OpenSSH e o DropBear representavam mundialmente cerca de 74,4% e 13,8% do número de instâncias, respectivamente, em nossos dados, percebemos uma clara inversão: cerca de 77,7% dos IPs estão vinculados ao DropBear, e 16,8% ao OpenSSH. Para os outros casos, embora a posição nos ranqueamento sofra algumas alterações, as proporções são parecidas.

A tabela 8 apresenta a popularidade de cada versão DropBear encontrada no Brasil no ano de 2023. Nesse ano, quase a totalidade dos dispositivos com DropBear utilizavam algumas das versões de 2019 (como a versão 2019.78). O motivo dessa popularidade é desconhecido; talvez essa versão tenha sido disponibilizada como padrão em alguns gerenciadores de pacotes. Uma investigação mais aprofundada seria necessária nesse sentido. No entanto, além do fato de que em 2023 essa versão já estaria desatualizada, ainda encontramos versões mais antigas, algumas lançadas em 2003, vinte um anos atrás.

Como mencionado na seção 4.2, existem pelo menos 62 vulnerabilidades catalogadas no NVD que afetam diretamente versões do DropBear. Algumas delas são críticas, como o CVE-2016-7406 e o CVE-2016-7407, ambos com CVSS v3 de 9,8, que afetam dispositivos com DropBear abaixo da versão 2016.74 e permitem que invasores remotos executem código arbitrário durante a autenticação por meio de manipulação do nome de usuário ou argumento de host. Outro exemplo é o CVE-2023-48795 (também conhecido como ataque Terrapin), descoberto mais recentemente em 2023, que afeta versões abaixo da 2022.83 e permite que invasores remotos contornem as verificações de integridade, omitindo alguns pacotes e consequentemente estabelecendo uma conexão para a qual alguns recursos de segurança podem ser rebaixados ou desabilitados. Apesar do CVSS do CVE-2023-48795 ser de 5,9 (médio), ele possui atualmente um score EPSS de 95,9%, o que representa uma alta probabilidade de ser explorado.

Apesar do grande número de dispositivos vulneráveis, devido a essas versões desatualizadas do DropBear, nenhuma vulnerabilidade foi vinculada em nenhum dos registros coletados pelo Shodan. Pelo que avaliamos, quando o Shodan identifica algum serviço SSH suportado pelo motor, suas inferências se concentram em coletar informações das chaves públicas, protocolos de criptografia disponíveis, entre outras informações. No entanto, informações como versão ou nome da ferramenta utilizada são ignoradas, provavelmente devido à complexidade de criar padrões capazes de se adequar à diversidade de serviços. Nossa ferramenta, neste contexto, pode ser utilizada para complementar os dados do Shodan, enriquecendo-os com tais informações.

5.2. Caso de uso 2: Utilização de sistemas operacionais desatualizados

Assim como no caso anterior sobre o versionamento do SSH, o motor de busca ainda falha em identificar informações sobre o sistema operacional em muitos cenários. Informações sobre o sistema e suas versões podem ajudar equipes a direcionar esforços de segurança, uma vez que sistemas desatualizados podem refletir sistemas com uma série de aplicações desatualizadas. Avaliar o versionamento do Ubuntu é outro exemplo de como nossa solução pode complementar as informações do Shodan.

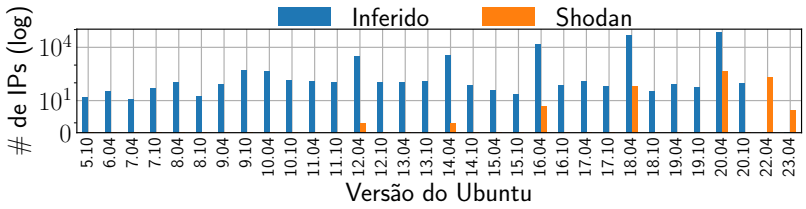

Figura 6. Quantidade de dispositivos com o sistema Ubuntu.

Como já mencionado, a informação sobre o sistema operacional, quando disponibilizada pelo Shodan, é apresentada em um único campo textual, que pode variar desde a inferência genérica de "Linux" até descrições mais detalhadas como "Ubuntu 20.04.4 LTS (Focal Fossa) (Linux 5.4.0-139-generic)". Nosso sistema de inferências, por outro lado, possui uma estruturação dessas informações, quando disponíveis, o que permite uma maior granularidade. Em nosso conjunto de dados, para o ano de 2023, por exemplo, o Shodan foi capaz de inferir 3,7 milhões de IPs com algum tipo de sistema operacional. Destes, 2,3 milhões possuíam o termo "Linux" associado ao campo e apenas 198 mil IPs com o termo "Ubuntu", sendo que destes, apenas 749 IPs possuíam informações sobre a versão utilizada.[9] Por outro lado, nossa solução conseguiu identificar 3,4 milhões de IPs com sistemas baseados em Linux. Destes, 274 mil eram distribuições Ubuntu (sem versão definida) e 147 mil versões do Ubuntu com a versão inferida. A figura 6 apresenta a quantidade de dispositivos em cada versão do Ubuntu, sondados pelo Shodan no ano de 2023. As barras azuis representam os dispositivos onde nossa solução foi capaz de identificar, enquanto as barras laranjas representam as informações fornecidas pelo Shodan.

Nesse exemplo, enquanto o Shodan conseguiu identificar distribuições mais recentes, como a versão 22.04 (abril de 2022) e a 23.04 (abril de 2023), nossa abordagem supera em diversidade de versões e em quantidade. Grande parte de nossas inferências vieram a partir do casamento de padrões relacionados ao SSH. Portanto, um esforço inicial em cadastrar padrões mais recentes sobre o SSH pode ajudar nossa abordagem a lidar com esses outros casos. O resultado encontrado é impactante. Existe uma grande variedade de versões do Ubuntu sem qualquer tipo de suporte há anos. Por exemplo, o suporte estendido da Canonical (empresa responsável pelo Ubuntu) acabou em 2007 para a versão 5.10, enquanto o suporte padrão para a versão 12.04 (LTS) teve fim em 2014.

6. Trabalhos Relacionados

Motores de busca são amplamente utilizados para pesquisas em diversos contextos, como a análise de vulnerabilidades de uma região [Al-Alami et al. 2017, Novianto et al. 2021], a identificação de dispositivos industriais [Samtani et al. 2018], ou pesquisas mais gerais para avaliar o comportamento das organizações na Internet [Nogueira et al. 2023, Moriot et al. 2022]. No entanto, devido ao grande volume de dados, muitos estudos sobre motores de busca de dispositivos são limitados a análises utilizando apenas a interface *web* do motor, sem a possibilidade de realizar pós-processamento nos dados, aplicar algoritmos mais complexos ou integrar com outras bases de dados, o que pode limitar os resultados obtidos [Al-Alami et al. 2017, Raikar e Maralappanavar 2021].

Pesquisas promovendo o enriquecimento de informações de motores de busca têm sido realizadas em diversos contextos. Por exemplo, em [Cheng et al. 2021], os autores

[9]Devido à forma como as informações são salvas no Shodan, nem todos os IPs com o termo Ubuntu estão incluídos no conjunto de IPs com o termo Linux.

utilizaram capturas de tela do Shodan para ajudar a distinguir o tipo de dispositivo sondado. Em [O'Hare et al. 2019], similar ao que é provido pelo nosso arcabouço, os autores motivam a integração com bases de vulnerabilidades para complementar as inferências realizadas pelo motor. No contexto específico da utilização de padrões a partir de expressões regulares, uma parte importante desses trabalhos se resumem a tornar o processamento mais eficiente enquanto fornecem garantias dos casamentos inferidos. Por exemplo, existem pesquisas, como em [Majumder et al. 2008], que focam em utilizar técnicas de otimização de expressões regulares para criar motores de processamento mais eficientes. Já trabalhos como o de [Wang et al. 2009], cria um mecanismo de pré-filtragem de padrões baseado nos termos do *fingerprint*, para reduzir o espaço de comparação.

Embora nossa proposta tenha semelhança à ferramenta Recog, apresentada inicialmente na seção 3, existem diferenças de desenho. Por exemplo, por causa da ferramenta ser pensada para usos diretos em etapas de Pentest, isto é, se conectando a um serviço para inferência de informações, o desempenho não é um ponto-chave. Nossa abordagem, no entanto, por ter como alvo o processamento em grandes volumes de dados gerados pelos motores de busca, pode ser utilizado via linguagem de programação em um ambiente distribuído, além de possuir mecanismos de otimizações para a computação dos padrões. Nossa abordagem fornece também métricas para priorizar e validar as inferências computadas, além de ser alimentado por padrões de diversas fontes.

7. Conclusão e Trabalhos Futuros

Este trabalho apresenta um arcabouço capaz de processar padrões em larga escala para o reconhecimento de aplicações e dispositivos a partir de diversas fontes de dados. Utilizando esse arcabouço, realizamos o enriquecimento de dados provenientes do Shodan para a inferência de aplicações e dispositivos conectados à Internet. Em nossos resultados, observamos que o conjunto de informações inferidas geralmente supera as disponibilizadas pelo motor de busca. Não apenas fomos capazes de identificar um número maior de serviços, como o sistema operacional (em 1,6 vezes mais IPs), e informações sobre dispositivos (14,1 vezes), mas também a qualidade das informações é mais estruturada. Também ilustramos como informações desse tipo podem ser utilizadas na análise de vulnerabilidades a partir de dois casos de uso. No primeiro cenário, realizamos um censo sobre a utilização do protocolo SSH na Internet, comparando com pesquisas de anos anteriores. Nesse cenário, identificamos uma série de ferramentas desatualizadas que possuíam vulnerabilidades publicadas, no entanto, a falta de identificação desse tipo de ferramenta por parte do Shodan fazia com que diversas dessas vulnerabilidades não pudessem ser investigadas. No segundo caso de uso, fomos capazes de identificar uma série de versões do sistema operacional Ubuntu sem suporte desde 2007. Nosso arcabouço permite complementar as informações dos motores de busca a partir da enumeração de serviços e identificação das vulnerabilidades relacionadas. Como trabalhos futuros, pretendemos incluir novos padrões de *fingerprints*. Avaliações mais focadas em análises de risco podem ser feitas relacionando os *fingerprints* às vulnerabilidades catalogadas em fontes como o NVD. Além disso, nossa identificação de serviços pode ser utilizado em outros contextos como uma forma para diferenciar dispositivos em IPs dinâmicos.

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., Hadi, A., e Al-Bahadili, H. (2017). Vulnerability scanning of IoT devices in Jordan using Shodan. In *2017 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes & Systems (IT-DREPS)*, pages 1–6, Jordan. IEEE.

Albataineh, A. e Alsmadi, I. (2019). IoT and the Risk of Internet Exposure: Risk Assessment Using Shodan Queries. In *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks"*, pages 1–5, EUA. IEEE.

Cheng, H. et al. (2021). Identify IoT Devices through Web Interface Characteristics. In *2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS)*, pages 405–410. IEEE.

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 *2021 Second International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, pages 38–45, Estonia. IEEE.

Durumeric, Z. et al. (2015). A Search Engine Backed by Internet-Wide Scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 542–553, EUA. ACM.

Gasser, O., Holz, R., e Carle, G. (2014). A deeper understanding of SSH: Results from Internet-wide scans. In *2014 IEEE Network Operations and Management Symposium*, pages 1–9, Poland. IEEE.

Genge, B. e Enăchescu, C. (2016). ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services. *Security and Communication Networks*, 9(15):2696–2714.

Majumder, A., Rastogi, R., e Vanama, S. (2008). Scalable regular expression matching on data streams. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 161–172, EUA. ACM.

Markowsky, L. e Markowsky, G. (2015). Scanning for vulnerable devices in the Internet of Things. In *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, volume 1, pages 463–467. IEEE.

Microservice (2022). O que é análise de vulnerabilidade e qual sua importância? https://www.microserviceit.com.br/analise-vulnerabilidade/. Acessado em 31/05/2024.

Moriot, C. et al. (2022). How to build socio-organizational information from remote ip addresses to enrich security analysis? In *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, pages 287–290. IEEE.

Nogueira, M. et al. (2023). A Large Scale Characterization of Device Uptimes. *IEEE Transactions on Emerging Topics in Computing*, 11(3):553–565.

Novianto, B., Suryanto, Y., e Ramli, K. (2021). Vulnerability analysis of internet devices from indonesia based on exposure data in shodan. In *IOP Conference Series: Materials Science and Engineering*, volume 1115, page 012045. IOP Publishing.

O'Hare, J., Macfarlane, R., e Lo, O. (2019). Identifying vulnerabilities using internet-wide scanning data. In *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)*, pages 1–10.

Ponce, L. et al. (2023). Um Arcabouço para Processamento Escalável de Vulnerabilidades e Caracterização de Riscos à Conformidade da LGPD. In *Anais do XXIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 15–28, Porto Alegre, RS, Brasil. SBC.

Ponce, L. et al. (2024). Arcabouço Multi-motor para Detecção de Vulnerabilidades na Internet Brasileira. In *Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 197–210, Porto Alegre, RS, Brasil. SBC.

Popescu, M. (2016). Internet Census, 2016. https://t.ly/aQRM2. Acessado em 31/05/2024.

Project, H. (2024). Hyperscan - ultra-fast regular expression matching library. https://hyperscan.org/. Acessado em 31/05/2024.

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

Samtani, S. et al. (2018). Identifying SCADA Systems and Their Vulnerabilities on the Internet of Things: A Text-Mining Approach. *IEEE Intelligent Systems*, 33(2):63–73.

Sarabi, A., Yin, T., e Liu, M. (2023). An LLM-based Framework for Fingerprinting Internet-connected Devices. In *Proceedings of the 2023 ACM on Internet Measurement Conference*, IMC '23, pages 478–484, EUA. ACM.

Wang, R. et al. (2022). WYSIWYG: IoT Device Identification Based on WebUI Login Pages. *Sensors*, 22(13).

Wang, X. et al. (2009). Extraction of fingerprint from regular expression for efficient prefiltering. In *2009 IEEE International Conference on Communications Technology and Applications*, pages 221–226, China. IEEE.