Monitoramento e Identificação de Páginas de Phishing

Heitor Damasceno*[1] Vitor Freire*[2] Welton Santos*[2] Elverton Fazzion[2,1]
Osvaldo Fonseca[1] Ítalo Cunha[1] Cristine Hoepers[3] Klaus Steding-Jessen[3]
Marcelo H. P. C. Chaves[3] Dorgival Guedes[1] Wagner Meira Jr.[1]

[1]DCC/UFMG [2]DCOMP/UFSJ [3]CERT.br/NIC.br

{heitor.damasceno,vitorfreire,weltonsantos,osvaldo.morais,cunha,dorgival,meira}@dcc.ufmg.br
{mhp,cristine,jessen}@cert.br fazzion@ufsj.edu.br

* Contribuíram de forma equitativa para este trabalho, listados em ordem alfabética.

Abstract. *Phishing campaigns frequently employ Web pages that mimic legitimate pages to fool victims into providing sensitive information. Despite the research community's continued effort to address these malicious activities, phishing becomes ever more sophisticated and continues making victims. In this paper we present a new phishing Web page monitoring framework that combines multiple techniques to achieve scalability and effectiveness. We also study trade-offs in the complex task of training models for identifying phishing Web pages. We show that representative datasets and features about the content of phishing Web pages are crucial for building general models. Our framework has been applied to hundreds of thousands of daily e-mails, and identified about one hundred* phishing *pages, a reduction of three orders of magnitude, and also serves as starting point for future* phishing *mitigation efforts.*

Resumo. *Campanhas de* phishing *frequentemente utilizam páginas Web que imitam páginas legítimas para enganar as vítimas. Apesar dos esforços da comunidade científica em combater essa atividade, o* phishing *fica cada vez mais sofisticado e continua fazendo vítimas. Neste artigo apresentamos um novo arcabouço de monitoramento de páginas de* phishing *que combina técnicas que proveem escalabilidade e efetividade. Também estudamos os compromissos existentes na complexa tarefa de construir modelos para identificar páginas de* phishing*. Mostramos que bases de dados representativas e atributos do conteúdo das páginas são cruciais para construir modelos gerais. Nosso arcabouço de monitoramento e identificação de páginas de* phishing *foi aplicado a centenas de milhares de e-mails diários, identificando uma centena de páginas de* phishing*, uma redução de três ordens de magnitude, e serve também de ponto de partida para direcionar esforços futuros de combate ao* phishing*.*

1. Introdução

A Internet tem impulsionado ou mesmo definido diversos setores de nossa sociedade e, por isso, alcança uma fração cada vez maior da população e faz-se cada vez mais importante em nosso cotidiano. Essa popularização tornou a Internet um campo fértil de exploração por agentes maliciosos. Um crime comum é o *phishing*, que tenta ludibriar vítimas através de engenharia social para obter dados pessoais ou informações sensíveis. Um exemplo de *phishing* são mensagens de e-mail falsas que se assemelham a mensagens de um remetente legítimo (p.ex., bancos) enviadas às vítimas para induzi-las a acessar uma página Web falsa e fornecer informações confidenciais [Sahingoz et al. 2019].

Apesar dos esforços de combate a este tipo de crime, é notório que o *phishing* ainda tem sucesso. Uma das razões é que o *phisher* aprimora estratégias e mecanismos continuamente, reduzindo a efetividade dos métodos de combate existentes. Com a popularização das mídias sociais, que permitem acesso a informações específicas e pessoais de vítimas, o nível de personalização da engenharia social (p.ex., menção ao nome da vítima, familiares ou emprego) pode enganar até usuários experientes. Com isso, o *phishing* continua uma das atividades criminosas mais lucrativas na Internet, gerando bilhões de dólares de prejuízo para os usuários [Koetsier 2020].

O estudo e o combate ao *phishing* requerem a coleta e identificação de um grande número de páginas maliciosas para posterior caracterização e desenvolvimento de mecanismos de mitigação. Porém, diversos fatores dificultam a coleta e identificação de páginas de *phishing*. Páginas de *phishing* evadem a coleta usando longas cadeias de redirecionamento, carregamento indireto de conteúdo, disponibilidade por curtos intervalos de tempo, ou mesmo subvertendo os sistemas de monitoramento, levando-os a realizarem ações danosas [Gupta et al. 2017, Jain and Gupta 2017, Santos et al. 2019]. A identificação de páginas de *phishing* também é desafiadora, pois páginas de *phishing* são propositalmente misturadas a páginas legítimas e ativamente as imitam visando ludibriar possíveis vítimas [Jain and Gupta 2018].

Neste trabalho apresentamos um arcabouço de coleta de páginas de *phishing* que provê eficácia, eficiência, e segurança (seção 3). A eficácia do nosso arcabouço advém do uso de um navegador Web equivalente ao utilizado por vítimas, garantindo a verossimilhança do acesso às páginas de *phishing*; a eficiência advém da identificação e seleção de URLs redundantes, reduzindo o número de páginas monitoradas e a chance de detecção do monitoramento pelo *phisher*, sem perda significativa de informação; e a segurança advém da implementação de restrições de acesso que bloqueiam atividades maliciosas induzidas por páginas de *phishing*. As funcionalidades do nosso arcabouço são genéricas e aplicáveis a outros arcabouços existentes. Disponibilizamos nosso arcabouço para a comunidade científica, de operadores de rede e de monitoramento de incidentes.[1]

Também analisamos como construir modelos para identificar páginas de *phishing*, apresentando um estudo detalhado sobre o impacto de diferentes fatores na precisão e revocação (*recall*) de classificadores (seção 4). Primeiro, mostramos que a composição das bases de treino e teste é fator essencial para a geração de classificadores gerais aplicáveis a várias bases de dados, particularmente a diversidade das páginas e a semelhança entre páginas de *phishing* e páginas legítimas. Nossos resultados indicam que as bases de dados utilizadas em trabalhos anteriores superestimam a generalidade e utilidade dos classificadores em cenários reais. Segundo, avaliamos a importância de atributos (*features*) utilizados na construção desses classificadores. Nossos resultados indicam que atributos do conteúdo das páginas, que podem ser coletados por nosso arcabouço de monitoramento, contribuem significativamente para a precisão e revocação de classificadores.

2. Trabalhos Relacionados

Arcabouços de Monitoramento. Pesquisadores e operadores de redes de computadores desenvolveram arcabouços de monitoramento para diversas ameaças na Internet (p.ex., mensagens de spam [Spamhaus 2021, Steding-Jessen et al. 2008] e ataques de negação

[1] https://github.com/heitorps123/app-monitoramento-web-phish

de serviço [Krämer et al. 2015]). O arcabouço proposto por [Park et al. 2017] monitora páginas de *phishing* utilizando um Web *scraper* que, apesar de funcional e eficiente computacionalmente, é incapaz de monitorar uma grande variedade de páginas (p.ex., todas as páginas que utilizam cadeias de redirecionamento são implementadas em JavaScript) e ignora questões éticas inerentes ao processo de Web *scraping*. Soluções comerciais também existem para coletar páginas de *phishing* (p.ex., urlscan.io); apesar da riqueza de funcionalidade, detalhes de implementação não estão publicamente disponíveis. Nosso arcabouço supera estas limitações combinando e estendendo ferramentas de código aberto (seção 3).

Identificação de Páginas de *Phishing*. Diversos trabalhos recorrem a algoritmos de aprendizado de máquina, em particular classificação, para a identificação de páginas maliciosas [Xiang et al. 2011, Zhang et al. 2016, Jain and Gupta 2017, Chiew et al. 2019, Rao and Pais 2018, Sahingoz et al. 2019, Jain and Gupta 2018, Shirazi et al. 2018, Li et al. 2018, Sameen et al. 2020, Mohammad et al. 2012, Gowtham and Krishnamurthi 2014]. Para aumentar a eficiência desses classificadores, diversos trabalhos (i) avaliam diferentes algoritmos de aprendizado de máquina, como Random Forest e Árvores de Decisão, variando configurações e avaliando a eficácia dos classificadores resultantes e (ii) propõem novos atributos (*features*) para fornecer aos algoritmos de aprendizado de máquina mais informações sobre páginas maliciosas. Nossas contribuições são complementares, pois avaliamos o impacto da composição das bases de dados e de diferentes tipos de atributos na eficácia e generalidade de classificadores.

Caracterização de *Phishing*. A caracterização do *phishing* é uma importante frente no combate a este tipo de abuso, permitindo a identificação de tendências e subsidiando o desenvolvimento de novos mecanismos de mitigação. Caracterizações de *phishing* podem ser encontradas em relatórios empresariais (p.ex., [Cisco Systems 2019, Symantec 2019]) que indicam a prevalência de atividades maliciosas, porém utilizam dados proprietários e não publicam detalhes da metodologia utilizada na análise. Estudos científicos geralmente traçam perfis de comportamento do *phisher* a nível de rede, lançando luz sobre a infraestrutura de origem e hospedagem do *phishing*, e a nível de conteúdo, avaliando técnicas de engenharia social utilizadas para ludibriar as vítimas e contornar detectores de *phishing* [Ho et al. 2019, Santos et al. 2019].

Mecanismos de Combate ao *Phishing*. Mecanismos de combate ao *phishing* podem ser classificados em três frentes: (i) educação e treinamento do usuário, instruindo-o a realizar verificações simples para reconhecer tentativas de *phishing* e evitar golpes; (ii) mecanismos que identificam mensagens de e-mail de *phishing*, filtrando ou marcando o e-mail antes que o usuário seja exposto ao conteúdo da mensagem; e (iii) mecanismos de combate às páginas de *phishing* [Gupta et al. 2017, Mason 2002], identificando e desativando servidores que hospedam páginas maliciosas. Mecanismos de identificação de mensagens de *phishing* (frente ii) são frequentemente integrados à interface de e-mail de usuários devido à dificuldade de identificar e desativar servidores que hospedam páginas de *phishing*, seja devido a questões legais ou redes de hospedagem "à prova de bala"(*bulletproof hosting*) [Konte et al. 2015]. O arcabouço e análise de classificadores apresentados neste artigo contribuem para o desenvolvimento de novos mecanismos de combate ao *phishing*.

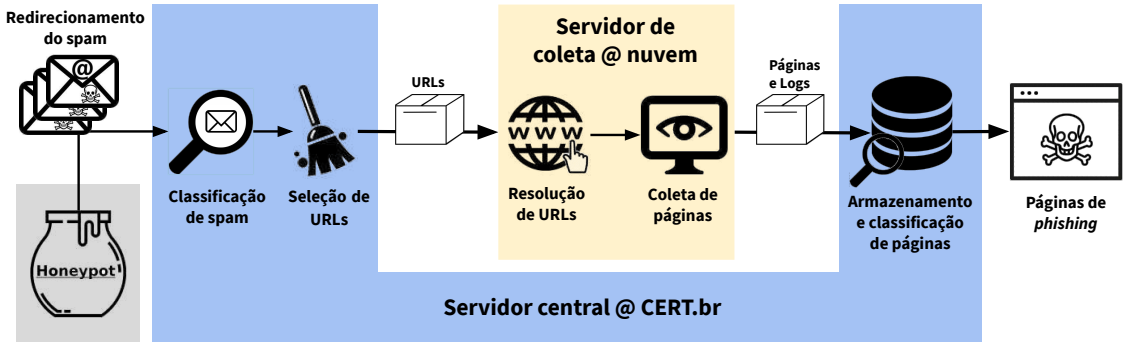

Figura 1. Visão geral do arcabouço e da metodologia de monitoramento de *phishing*.

3. Arcabouço de Coleta de Páginas de *Phishing*

Nesta seção apresentamos nosso arcabouço de coleta de páginas de *phishing*. O arcabouço resolve URLs observadas em campanhas de *phishing* reais disseminadas por e-mail. As mensagens de e-mail são coletadas por um conjunto de 11 *honeypots* distribuídos em 8 países, que emulam servidores de e-mail vulneráveis e recebem em média 335.000 mensagens de e-mail não requisitadas por dia (*spam* e *phishing*), que veiculam aproximadamente 180 mil URLs [Steding-Jessen et al. 2008]. A figura 1 sumariza o processo descrito nesta seção.

3.1. Seleção de URLs

Os números de mensagens enviadas e vítimas alvejadas em campanhas de *phishing* variam significativamente [Han and Shen 2016]. Campanhas de *phishing* de grande abrangência podem levar à captura de centenas de milhares de URLs que correspondem a páginas Web com conteúdo idêntico ou muito similar. Além do custo de acesso em si, caso o arcabouço realize uma quantidade de acessos aos servidores de *phishing* suficiente para atrair a atenção do *phisher*, este pode bloquear o acesso, impedindo o monitoramento. Para reduzir a quantidade de URLs a serem resolvidas bem como evitar a detecção e bloqueio do monitoramento, selecionamos um subconjunto representativo das URLs para evitar acesso a páginas Web similares.

Obtemos uma lista de URLs potencialmente ligadas a campanhas de *phishing* utilizando um classificador de e-mails [Santos et al. 2019] e extraindo todas as URLs contidas em e-mails de *phishing* recebidos pelos *honeypots* (ignoramos os e-mails de *spam*). Propomos e avaliamos três técnicas de seleção de URLs:

Seleção Sintática: Seleciona todas as URLs distintas observadas nas mensagens de *phishing* (cada URL distinta será acessada uma vez).

Seleção por Domínio: Seleciona uma URL (aleatoriamente) dentre URLs que possuem o mesmo domínio e os mesmos nomes de parâmetros (se houver). Parâmetros são adicionados à URL, p.ex., para realizar uma busca em um banco de dados ou personalizar a página a ser retornada. As URLs https://foo.com/?nome=Ana e https://foo.com/?nome=Bernardo possuem o mesmo domínio e apenas o parâmetro nome. Na seleção por domínio apenas uma delas seria selecionada para acesso.

Seleção por Subdomínio: Estende a seleção por domínio considerando apenas os dois primeiros níveis do nome de domínio, ou os três primeiros níveis se houver

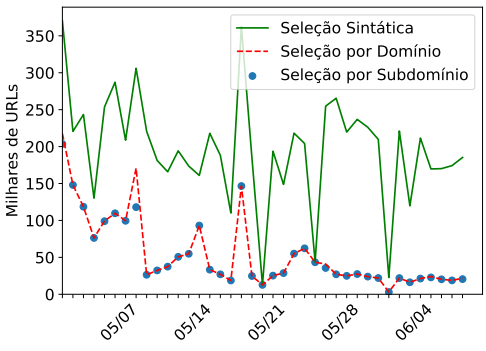

Figura 2. Comparação dos filtros de URLs propostos. A filtragem restritiva reduz significativamente a quantidade de URLs acessadas.

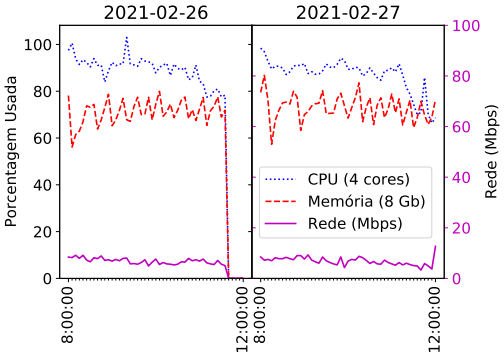

Figura 3. Análise do uso de recursos críticos no arcabouço de coleta ao longo de dois dias em um dado período de tempo.

código de país. Por exemplo, as URLs https://a8wq.bar.com.br/?id=1aso e https://asq9.bar.com.br/?id=hhqw estão hospedadas em bar.com.br e possuem apenas o parâmetro id; apenas uma delas seria selecionada para acesso.

A seleção é realizada sobre o conjunto de URLs observadas em mensagens de *phishing* de cada dia, isoladamente. Consequentemente, apesar de uma URL ser acessada no máximo uma vez por dia, ela pode ser acessada repetidas vezes em múltiplos dias. Esta abordagem permite o monitoramento longitudinal de campanhas de *phishing*.

A figura 2 mostra o número de URLs selecionadas para acesso por cada filtragem em cada dia do período entre 1 de maio e 8 de junho de 2020. Resultados para outros períodos são qualitativamente similares. A variação do número de URLs é grande, e depende da quantidade de mensagens e do conteúdo de campanhas de *phishing* observadas pelos *honeypots*. A seleção por domínio reduz significativamente a quantidade de URLs a serem acessadas na maior parte dos dias, frequentemente selecionando menos de 50% das URLs distintas. A seleção por subdomínio traz pouca redução no número de URLs para acesso além da seleção por domínio. Uma análise manual das URLs escolhidas indica que o número de URLs selecionadas depende das estratégias utilizadas pelas campanhas de *phishing* para construir URLs.

Os resultados da seleção por domínio e por subdomínio são muito similares. Decidimos conservadoramente pela seleção por domínio, que pode resultar na seleção de um número maior de URLs mas evita perda de informação caso diferentes subdomínios sejam utilizados para hospedar páginas de *phishing* diferentes.

Para avaliar a qualidade da seleção por domínio, todos os dias adicionamos URLs redundantes aleatoriamente à seleção, com uma probabilidade de 0,005, um valor que não impacta significativamente a carga e funcionamento do arcabouço mas possibilita a coleta de dados suficientes para análise. Após a coleta, utilizamos a biblioteca html-similarity[2] para verificar a similaridade entre páginas que a seleção por domínio considera equivalentes [Gowda and Mattmann 2016]. A biblioteca suporta três tipos de similaridade combinando a estrutura (HTML) e o estilo (CSS) das páginas. Nossos resultados mostram que as similaridades entre páginas redundantes e suas equivalentes são altas nas três categorias: 95% dos pares de páginas apresentaram similaridades acima de 90%. Reali-

[2]https://pypi.org/project/html-similarity/

zamos uma comparação visual dos dez pares de páginas mais *dis*similares e confirmamos que são páginas com o mesmo conteúdo, com diferenças irrelevantes para as vítimas. Esse resultado indica que a redução do número de páginas acessadas não compromete a representatividade da cobertura das páginas de *phishing*.

Complementarmente, a redução do número de páginas a serem coletadas (i) provê maior escalabilidade do arcabouço caso a quantidade de URLs aumente muito no futuro, bem como (ii) permite aumentar o limite de tempo (timeout) de acesso a uma página e, indiretamente, a taxa de sucesso de coleta.

3.2. Resolução de URLs e Coleta de Páginas

O desafio central no monitoramento de páginas de *phishing* é capturar o mesmo conteúdo que seria exibido a uma vítima. Superar este desafio requer executar códigos JavaScript, contornar verificações de navegadores Web e carregar objetos embutidos, como vídeos. Em particular, a execução de código JavaScript é um requisito para coletar páginas ofuscadas por cadeias de redirecionamento [Santos et al. 2019].

Para garantir o acesso às URLs de forma similar à experiência da vítima, utilizamos o navegador Firefox sem interface gráfica (*headless*)[3] para realizar os acessos. A utilização do Firefox incorre sobrecarga computacional, mas supera todas as dificuldades enumeradas acima. Além disso, a utilização de um navegador completo permite o arcabouço acompanhar avanços em tecnologias Web de forma transparente.

Nosso arcabouço executa múltiplas instâncias do Firefox para acelerar o acesso às páginas. Cada instância do Firefox acessa múltiplas páginas ao longo do dia de forma sequencial. Utilizamos o *web-driver* Selenium[4] para gerenciar a lista de URLs a serem coletadas e controlar as instâncias do Firefox para sequenciar os acessos.

As requisições HTTP geradas pelas instâncias do Firefox são realizadas através do BrowserMob,[5] um *proxy* que coleta metadados (data, hora, informações de rede, URLs, cabeçalhos HTTP das requisições e respostas) bem como o conteúdo recebido dos servidores durante o acesso às páginas. Os dados coletados pelo BrowserMob permitem a reconstrução não apenas da página de *phishing*, mas também da sequência de requisições realizadas, p.ex., devido a uma cadeia de redirecionamento.

3.3. Medidas de Ética e de Segurança

As URLs observadas em mensagens de *phishing* e selecionadas para coleta (seção 3.1) incluem URLs legítimas e URLs de *phishing*. Um fator complicante, específico ao monitoramento de páginas de *phishing*, é a frequente referência a objetos (p.ex., figuras com logomarcas) hospedados em servidores legítimos operados por empresas alvo de campanhas de *phishing*. Um requisito ético para operação de arcabouços de monitoramento é não impactar negativamente os servidores hospedando as páginas monitoradas, especialmente servidores hospedando páginas legitimas. Além disso, verificamos na prática que a utilização do arcabouço sem medidas de segurança e seleção de URLs leva a uma alta taxa de bloqueio do servidor de coleta pelos *phishers*, comprometendo o monitoramento.

[3]https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Headless_mode

[4]https://www.selenium.dev

[5]https://github.com/lightbody/browsermob-proxy

Para garantir acesso ético a servidores legítimos e evitar o bloqueio em servidores de páginas de *phishing*, implementamos no BrowserMob um controle de acesso sobre requisições geradas pelo Firefox. Regras de controle de acesso são especificadas em código Java e novas regras podem ser facilmente integradas ao arcabouço.

Limitação da Taxa de Acesso. O acesso a uma URL pode levar à execução de diversas requisições HTTP, p.ex., para resolver cadeias de redirecionamento ou recuperar objetos como folhas de estilo CSS e código JavaScript. Além disso, diversas URLs selecionadas para acesso podem estar hospedadas em um mesmo servidor. Para não sobrecarregar servidores com um grande número de requisições, limitamos a taxa de requisições que podem ser enviadas a um domínio. Adicionamos ao BrowserMob uma janela deslizante que mantém o histórico recente de requisições HTTP enviadas a cada domínio acessado pelo arcabouço. Nossa abordagem classifica domínios e limita o número de requisições HTTP enviadas a um domínio da classe c nos últimos T_c segundos a um máximo N_c. Requisições geradas pelo Firefox após o limite ser atingido são rejeitadas. Classes de domínio, a duração do histórico T_c e o número de requisições N_c são configuráveis. Em nossa implantação atual classificamos domínios como operados por redes de distribuição de conteúdo (CDNs) ou não utilizando uma base pública.[6] Para os domínios operados por CDNs permitimos uma taxa maior de requisições considerando que estes serviços concentram um grande número de requisições e são especializados em atender altas cargas de trabalho. Empiricamente configuramos $T_{\text{CDN}} = 900\text{s}$ e $N_{\text{CDN}} = 1000$ requisições e para os demais domínios $T_\star = 900\text{s}$ e $N_\star = 100$ após realizarmos diversos experimentos variando os valores de T e N. A configuração corrente mantém a taxa de bloqueio de acesso em aproximadamente 25%, um compromisso entre a diminuição do número de requisições e completude do monitoramento; diferentes implantações podem configurar o arcabouço de forma mais agressiva ou conservadora ajustando classes de domínios e os valores de T e N. Requisições rejeitadas são respondidas com código HTTP 429 (*too many requests*) diretamente pelo BrowserMob e nenhum acesso é realizado ao servidor.

***Blocklist* de Domínios.** Mantemos uma *blocklist* de domínios que solicitaram não serem incluídos no acesso de URLs diário do arcabouço. Requisições para estes domínios são respondidas diretamente pelo BrowserMob com código HTTP 410 (*gone*). Até o momento nenhum domínio solicitou inclusão na *blocklist*, que permanece vazia.

Detecção de Serviços Anti-DDoS. Alguns serviços de proteção contra ataques de negação de serviço adicionam metadados a algumas respostas HTTP. Por exemplo, o serviço BitNinja adiciona um cabeçalho 'BN-Fontend: captcha-https' quando uma página de verificação é servida face a um grande número de requisições feitas por um cliente. Construímos um conjunto de expressões regulares para detectar serviços de proteção a ataques de negação de serviço. Estendemos o BrowserMob para verificar se algum cabeçalho HTTP de uma resposta satisfaz alguma expressão regular e, em caso positivo, descarta a resposta original e envia uma resposta HTTP 418 (*I'm a teapot*) para o Firefox, evitando requisições futuras.

Bloqueio de Requisições POST. Observamos que algumas páginas monitoradas subvertem o funcionamento do arcabouço para realizar ataques a servidores legítimos. Muitas campanhas de *phishing* utilizam requisições HTTP com o método POST para explorar

[6] https://github.com/dyne/domain-list

vulnerabilidades em formulários de páginas Web para furtar, alterar e destruir dados de instituições. Por exemplo, observamos páginas de *phishing* com URLs em *tags HTML link*, geralmente utilizadas para acessar automaticamente folhas de estilo CSS, que realizam uma tentativa de autenticação em servidores Wordpress. Como o objetivo do nosso arcabouço é apenas recuperar informações de servidores e jamais realizar alguma atualização, implementamos um filtro no BrowserMob que responde todas as requisições que não utilizam o método GET com uma resposta HTTP 405 (*method not allowed*).

Isolamento e Tempo Limite de Execução. Páginas de phishing podem conter código Javascript arbitrário, inclusive malicioso, que será executado pelo navegador no processo de coleta de páginas. Os seguintes fatores restringem o dano causado por código Javascript malicioso: (i) os acessos à rede são mediados pelo BrowserMob e as políticas de segurança anteriores; (ii) o navegador isola páginas umas das outras; e (iii) o tempo de acesso a uma página, e consequentemente de execução de código Javascript malicioso, é limitado (configurável, 15 segundos na configuração atual do arcabouço).

3.4. Desempenho e Gerenciamento de Recursos

A figura 3 mostra a série temporal do uso de recursos de CPU, memória e rede pelo arcabouço para coletar URLs durante dois dias consecutivos em um servidor com interface 1Gbps, 8 GiB de memória RAM e 4 processadores Xeon E5-2650 v4. A queda brusca do uso de recursos observado no primeiro dia corresponde ao fim do processamento de URLs naquele dia, o que depende majoritariamente do número de URLs selecionadas. Observamos que o recurso computacional mais utilizado pelo arcabouço é memória RAM: cada instância do Firefox utiliza centenas de MiB de memória. Configuramos o Selenium para manter todas as instâncias em memória principal e reiniciar o Firefox quando a utilização de memória excede uma fração da memória disponível (configuramos em 80% em nossa implantação). Além disso, observamos que a taxa de coleta depende fortemente do tempo máximo de acesso (*timeout*) configurado para cada página e da quantidade de instâncias do Firefox em execução. Nossa implantação executando 5 instâncias do Firefox utilizando um *timeout* de acesso de 15s consegue coletar aproximadamente 6800 páginas/hora.[7]

3.5. Modularidade e Implantação

As funcionalidades de nosso arcabouço são implementadas de forma modular. Em nossa implantação utilizamos a modularidade estrategicamente (figura 1). Os *honeypots* de coleta de e-mail são distribuídos em várias redes [Steding-Jessen et al. 2008]. O processamento de dados é realizado em um *servidor central* hospedado no CERT.br. O *servidor de coleta* é hospedado em provedores de computação em nuvem para evitar identificação e possível retaliação a nossas instituições. Além disso, a hospedagem do servidor de coleta na nuvem também nos permite trocar facilmente o endereço IP do servidor quando detectamos bloqueio do monitoramento.

4. Identificação de Páginas de *Phishing*

Nesta seção analisamos como bases de dados e atributos (seções 4.1 e 4.2) utilizados para treinar modelos de identificação de páginas de *phishing* afetam a generalidade, precisão

[7] Apesar de desnecessário para o volume de URLs monitoradas atualmente, ajustes no *timeout* e no número de instâncias do Firefox permitiriam aumentar significativamente a taxa de coleta, se necessário.

Tabela 1. Composição e origem das bases de dados.

	Páginas de Phishing		Páginas Legítimas	
Base de Dados	Origem	Quantidade	Origem	Quantidade
DB1	Phishtank	1007	Majestic Top milhão	1038
	Honeypot	31		
DB2	Phishtank	1007	Honeypot	474
	Honeypot	31	Páginas de pagamento	139
			Crawler	425

e sensibilidade (*recall*). Nossos resultados indicam que ambos, base de dados e atributos, têm impacto significativo na acurácia e generalidade dos modelos resultantes e direcionam a construção de novas bases para treinamento de modelos futuros (seções 4.3 e 4.4).

4.1. Construção das Bases de Dados

Diversos trabalhos na literatura [Rao and Pais 2018, Gowtham and Krishnamurthi 2014, Zhang et al. 2016, Jain and Gupta 2018, Jain and Gupta 2017] constroem bases de dados para treinamento de modelos para identificação de páginas de *phishing* combinando uma fonte de páginas de *phishing* (p.ex., PhishTank) com uma fonte de páginas legítimas (p.ex., índices de páginas populares como Alexa ou Majestic). Neste trabalho construímos uma base de dados, chamada DB_1, seguindo esta estratégia, selecionando 1007 páginas de *phishing* do Phishtank e 31 páginas de *phishing* capturadas pelo nosso arcabouço, bem como 1038 páginas legítimas obtidas através do índice de popularidade Majestic Top Million. Utilizamos o Majestic porque a base Alexa Top Million é comercial. As 31 páginas de *phishing* capturadas por nosso arcabouço fazem parte de um conjunto de 505 páginas dissimilares (seção 3.1) selecionadas aleatoriamente e classificadas manualmente: 31 de *phishing* e 474 legítimas. Apesar de serem poucas páginas de *phishing* elas correspondem a campanhas distintas; cada página distinta representa milhares de mensagens de e-mail.

Construímos também uma base de dados, chamada DB_2, com páginas mais similares a páginas de *phishing* de fontes diversas. A base DB_2 utiliza o mesmo conjunto de páginas de *phishing* da DB_1, mas substituímos as páginas do Majestic por páginas legítimas de três tipos: (i) 425 páginas com palavras frequentemente utilizadas em páginas de *phishing*; (ii) 139 páginas legítimas similares a páginas alvo de *phishing* (p.ex., bancos online, login em sítios de provedores de serviço e páginas secundárias de sítios Web legítimos); e (iii) 474 páginas legítimas referenciadas por mensagens de e-mail de *phishing* coletadas por nosso *honeypot* (seção 3). A seleção de páginas legítimas da base DB_2 foi manual e teve o auxílio de um *crawler* desenvolvido para recuperar sub-páginas, evitando adicionar à base páginas iniciais de domínios legítimos. A tabela 1 sumariza as informações das bases de dados.

Para realizarmos uma comparação justa entre as bases de dados, as bases DB_1 e DB_2 são balanceadas e possuem o mesmo número de páginas; a principal diferença está nas fontes das páginas legítimas.

4.2. Conjuntos de Atributos

Diversos trabalhos na literatura propuseram atributos de páginas que podem ser úteis na identificação de páginas de *phishing* (seção 2). Neste trabalho implementamos todos os 101 atributos propostos em quatro trabalhos anteriores [Mohammad 2013,

Tabela 2. Conjuntos de atributos utilizados na avaliação, contendo referência original e número de atributos em cada categoria.

	Referência	URL	HTML	JS/CSS	Ext.
$\mathcal{A}_1$	[Mohammad 2013]	5	4	3	5
$\mathcal{A}_2$	[Chiew et al. 2019]	27	17	4	0
$\mathcal{A}_3$	[Sahingoz et al. 2019]	38	0	0	1
$\mathcal{A}_4$	[Jain and Gupta 2017]	8	8	1	0
$\mathcal{A}_{\text{all}}$	—	65	26	5	5

Chiew et al. 2019, Sahingoz et al. 2019, Jain and Gupta 2017].[8] Agrupamos os atributos implementados em quatro categorias: (i) 65 atributos da URL como protocolo, número de subdomínios e número de caracteres; (ii) 26 atributos do código fonte como número de *links* para domínios externos e existência de formulários; (iii) 5 atributos de código JavaScript e folhas de estilo CSS como existência de pop-ups e se o CSS está hospedado em um domínio externo; e (iv) 5 atributos baseados em dados externos como a popularidade do domínio no ranque Alexa e validade de certificados digitais.

Avaliamos diferentes conjuntos de atributos, os conjuntos $\mathcal{A}_1$–$\mathcal{A}_4$ correspondem aos utilizados em trabalhos anteriores, como mostrado na tabela 2. Avaliamos também o conjunto completo com todos os 101 atributos denominado $\mathcal{A}_{\text{all}}$. Para cada um dos conjuntos, consideramos também um subconjunto que contém apenas atributos das URLs.

4.3. Metodo de Avaliação

Treinamos e avaliamos modelos usando as bases DB_1 e DB_2. Cada modelo utiliza um conjunto distinto de atributos. Consideramos os conjuntos de atributos $\mathcal{A}_1$–$\mathcal{A}_4$ e $\mathcal{A}_{\text{all}}$ (seção 4.2), bem como subconjuntos destes contendo apenas os atributos de URL, denotados com um sufixo indicando o filtro, p.ex., $\mathcal{A}_3|\text{URL}$. Os modelos são treinados e avaliados sobre uma base de dados utilizando validação cruzada com 10 grupos.

Para avaliar a generalidade dos modelos gerados com diferentes base de dados e conjunto de atributos, também avaliamos cada modelo na base de dados oposta: modelos treinados na base DB_1 são avaliados na DB_2 e vice-versa.

Para cada combinação de base de dados e conjunto de atributos, treinamos modelos usando cinco técnicas de aprendizado de máquina implementadas no SciKit-Learn: *Random Forest, Support Vector Machine, Logistic Regression, Decision Tree* e *AdaBoost Classifier*. Por questões de espaço, apresentamos resultados apenas para modelos treinados com *Random Forest*, que foram os melhores; porém, os resultados para modelos treinados com outras técnicas de aprendizado de máquina são qualitativamente similares.

4.4. Análise dos Modelos de Identificação de Páginas de *Phishing*

A tabela 3 mostra a acurácia e o F1-score nos diferentes cenários, incluindo combinações das bases utilizadas para treinamento dos modelos e para avaliação. Como mencionado, o treinamento e a avaliação são realizados utilizando validação cruzada; a avaliação dos modelos sobre outra base de dados é realizada aplicando o modelo a todas as páginas da base. Cada linha da tabela considera um conjunto de atributos diferente. Também são apresentados os desvios-padrão da validação cruzada, onde percebemos a robustez dos modelos. A tabela 3 mostra dois resultados importantes que discutimos a seguir.

[8]Nossa implementação de cada um dos atributos se encontra pública e disponível no link do projeto.

Tabela 3. Acurácia e F1-score, em percentagem, para diferentes combinações de base de treinamento (primeira linha), base de teste (segunda linha) e conjunto de atributos (primeira coluna). Colunas em cinza mostram o desvio padrão da acurácia e F1-score.

Treinamento	DB_1				DB_2							
Teste	DB_1				DB_2		DB_1		DB_2			
Atributos	Ac.	F1	Ac_{dp}	$F1_{dp}$	Ac.	F1	Ac.	F1	Ac.	F1	Ac_{dp}	$F1_{dp}$
$\mathcal{A}_1$	89,84	89,41	1,83	2,05	74,37	77,52	86,99	86,73	83,86	83,07	2,60	2,54
$\mathcal{A}_1 \vert \text{URL}$	80,44	77,88	1,73	2,33	65,75	66,79	69,17	58,16	68,68	57,79	1,91	2,92
$\mathcal{A}_2$	96,29	96,22	1,18	1,23	70,32	77,11	97,73	97,78	83,86	83,07	2,10	2,22
$\mathcal{A}_2 \vert \text{URL}$	93,15	92,94	1,94	2,06	66,81	73,99	90,22	90,63	85,40	85,29	1,82	1,98
$\mathcal{A}_3$	94,70	94,63	1,01	1,12	67,14	75,01	90,12	90,93	89,11	89,03	2,83	2,97
$\mathcal{A}_3 \vert \text{URL}$	94,07	93,93	1,69	1,81	68,88	75,96	87,13	86,96	87,09	86,92	3,01	3,16
$\mathcal{A}_4$	90,55	90,62	2,31	2,34	84,63	86,41	92,29	92,63	85,79	85,79	1,94	1,74
$\mathcal{A}_4 \vert \text{URL}$	81,93	79,89	1,90	2,56	66,90	68,32	79,62	76,51	70,85	69,14	4,23	4,79
$\mathcal{A}_{all}$	96,86	96,82	0,94	0,96	69,94	76,88	97,49	97,55	93,97	93,82	1,90	2,00
$\mathcal{A}_{all} \vert \text{URL}$	94,26	94,12	1,81	1,96	68,73	75,88	92,96	93,33	83,86	83,07	2,87	3,04

A base utilizada para treinamento é determinante na generalidade dos modelos. Apesar dos modelos treinados na base DB_1 terem desempenho satisfatório quando avaliados na base DB_1 (primeiro par de colunas), seus desempenhos degradam significativamente quando avaliados na base DB_2. Os modelos treinados na base DB_1, mais simples, não generalizam para cenários mais desafiadores como os representados pela base DB_2.[9] A tabela 3 também mostra que os modelos treinados na base DB_2 tem desempenho *superior* quando avaliados nas bases DB_1 e DB_2. Em particular, quando avaliamos os modelos na base DB_1, vários modelos treinados na base DB_2 têm desempenho superior aos modelos correspondentes treinados na própria base DB_1. Este resultado mostra que os modelos treinados na base DB_2 são mais gerais do que os modelos treinados na base DB_1, e indicam que a base utilizada para treinamento dos modelos é determinante na generalidade dos mesmos.

Uma implicação prática deste resultado é que trabalhos anteriores reportando alta acurácia na identificação de páginas de *phishing* com modelos treinados em bases simples podem não generalizar para utilização em cenários práticos. Para reforçar este ponto, a figura 4 mostra a importância dos atributos dos modelos treinados nas bases DB_1 e DB_2 usando todos os atributos ($\mathcal{A}_{\text{all}}$). Para modelos treinados na base DB_1, vemos que atributos sobre o tamanho da URL como o tamanho do caminho, número de caracteres e número de barras estão entre os quatro atributos mais importantes. Estes atributos podem diferenciar páginas de *phishing* de páginas como https://www.facebook.com, mas não generalizam para outras páginas legítimas como https://www.facebook.com/coronavirus_info/facts/1255533528116513/. Em contrapartida, os oito atributos mais importantes do modelo treinado na base DB_2 são relativos ao conteúdo da página, código JavaScript ou fatores externos.

Com base nas análises anteriores evidenciamos que os modelos treinados em bases construídas com fontes como Majestic e Alexa Top Sites podem possuir forte viés oriundo

[9] A acurácia obtida por nossos modelos está alinhada à acurácia de modelos treinados em bases similares à base DB_1 em outros trabalhos na literatura. Mais especificamente, modelos treinados e avaliados em bases contendo primariamente páginas legítimas populares apresentaram acurácia entre 92% e 99% [Mohammad 2013, Chiew et al. 2019, Rao and Pais 2018, Zhang et al. 2016, Shirazi et al. 2018].

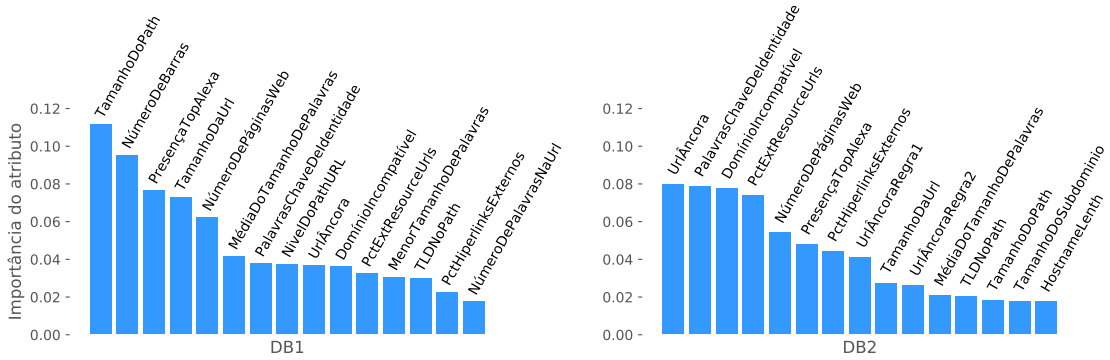

Figura 4. Índice de Gini dos 15 atributos mais importantes em modelos treinados com todos os atributos ($\mathcal{A}_{\text{all}}$) nas bases DB_1 e DB_2.

da base de dados, supervalorizando atributos como o tamanho da URL. A construção de bases com páginas legítimas diversas e similares a páginas de *phishing*, como a base DB_2, podem levar a modelos mais gerais e com melhor desempenho em cenários reais.

Atributos do conteúdo, código, estilo e propriedades da página contribuem para identificação de *phishing*. Trabalhos anteriores avaliaram modelos para identificação de páginas de *phishing* treinados utilizando apenas atributos de URLs [Sahingoz et al. 2019, Shirazi et al. 2018]. Estas abordagens possuem a vantagem de não dependerem de dados de difícil obtenção, como o conteúdo da página ou bases externas.

A tabela 3, porém, mostra que modelos construídos utilizando apenas atributos das URLs ($\mathcal{A}_\star|\text{URL}$), no geral, têm desempenho inferior aos modelos correspondentes utilizando atributos adicionais (conteúdo, estilo, código e propriedades externas). O modelo $\mathcal{A}_3|\text{URL}$ é a exceção. Porém, o ganho de desempenho relativo ao modelo utilizando $\mathcal{A}_3$ é pequeno e acontece apenas em casos de treinamento na base DB_1, que supervaloriza atributos das URLs, e avaliação na base DB_2.

Este resultado indica que atributos adicionais contribuem para a construção de modelos com melhor desempenho. Para tanto, nosso arcabouço de monitoramento de páginas de *phishing* é uma contribuição significativa, pois contribui para aliviar o problema da escassez de dados e auxiliar na difícil tarefa de construção de bases de dados representativas (seção 4.1). A coleta de páginas por nosso arcabouço permite computarmos todos os 101 atributos propostos em trabalhos anteriores, e provavelmente vários outros que sejam eventualmente propostos.

5. Conclusão

Neste trabalho descrevemos um arcabouço de coleta de páginas de *phishing*, apresentando mecanismos para garantir escalabilidade e efetividade. Também construímos um classificador de páginas de *phishing* e mostramos que o conjunto de páginas que compõem uma base de dados bem como o conjunto de atributos extraídos das páginas são fatores determinantes para generalidade e acurácia de modelos.

Nosso arcabouço recebe em média 335K e-mails por dia, dos quais selecionamos 17K URLs, entre as quais identificamos 120 que levam a páginas de *phishing*. Esta redução significativa (3 ordens de magnitude) entre o número de mensagens de e-mail e o número de páginas de *phishing* permite cobertura de vastas campanhas de *phishing* e direcionamento de ações de mitigação.

Nossos resultados podem direcionar futuros esforços de monitoramento e construção de modelos para combate ao *phishing*. Em particular, recomendamos que trabalhos futuros (i) construam bases de dados com páginas de *phishing* de diferentes fontes, cobrindo campanhas de *phishing* de diferentes tipos; (ii) construam bases de dados com páginas legítimas diversas e similares a alvos de *phishing*; e (iii) utilizem atributos complexos, como atributos do conteúdo e propriedades de páginas. Nosso arcabouço de monitoramento, que disponibilizamos para operadores de rede e pesquisadores, provê informações que permitem a aplicação destas recomendações.

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

Referências

Chiew, K., Tan, C. L., Wong, K., Yong, K., and Tiong, W. (2019). A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. *Information Sciences*, 484:153–166.

Cisco Systems (2019). Email: Click with Caution. Technical report.

Gowda, T. and Mattmann, C. A. (2016). Clustering Web Pages Based on Structure and Style Similarity (Application Paper). In *Proc. of IEEE International Conference on Information Reuse and Integration*.

Gowtham, R. and Krishnamurthi, I. (2014). A Comprehensive and Efficacious Architecture for Detecting Phishing Webpages. *Computers & Security*, 40:23–37.

Gupta, B. B., Arachchilage, N., and Psannis, K. (2017). Defending against Phishing Attacks: Taxonomy of Methods, Current Issues and Future Directions. *Telecommunication Systems*, 67(2):247–267.

Han, Y. and Shen, Y. (2016). Accurate Spear Phishing Campaign Attribution and Early Detection. In *Proc of. ACM Symposium on Applied Computing*.

Ho, ., Cidon, A., Gavish, L., Schweighauser, M., Paxson, V., Savage, S., Voelker, G. M., and Wagner, D. (2019). Detecting and Characterizing Lateral Phishing at Scale. In *Proc. of USENIX Security Symposium*.

Jain, A. and Gupta, B. B. (2017). Towards detection of phishing websites on client-side using machine learning based approach. *Telecommunication Systems*, 68(04):687–700.

Jain, A. and Gupta, B. B. (2018). A machine learning based approach for phishing detection using hyperlinks information. *Journal of Ambient Intelligence and Humanized Computing*, 10(5):2015–2028.

Koetsier, J. (2020). Scammers Send 3.1 Billion Domain Spoofing Emails A Day. Here's How To Protect Yourself (And Your Company).

Konte, M., Perdisci, R., and Feamster, N. (2015). ASwatch: An AS Reputation System to Expose Bulletproof Hosting ASes. In *Proc. of ACM Conference on Special Interest Group on Data Communication*.

Krämer, L., Krupp, J., Makita, D., Nishizoe, T., Koide, T., Yoshioka, K., and Rossow, C. (2015). AmpPot: Monitoring and Defending Against Amplification DDoS Attacks.

In *Proc. of RAID International Symposium on Research in Attacks, Intrusions, and Defenses*.

Li, Y., Chen, X., Yuan, H., and Liu, W. (2018). A stacking model using URL and HTML features for phishing webpage detection. *Future Generation Computer Systems*, 94:27–39.

Mason, J. (2002). Filtering spam with SpamAssassin. In *HEANet Annual Conference*.

Mohammad, R. (2013). Predicting Phishing Websites based on Self-Structuring Neural Network. *Neural Computing and Applications*, 25(2):443–458.

Mohammad, R., Thabtah, F., and Mccluskey, T. (2012). An assessment of features related to phishing websites using an automated technique. In *Proc. of IEEE International Conference for Internet Technology and Secured Transactions*.

Park, A. J., Quadari, R. N., and Tsang, H. H. (2017). Phishing website detection framework through web scraping and data mining. In *Proc. of IEEE Information Technology, Electronics and Mobile Communication Conference*.

Rao, R. and Pais, A. (2018). Detection of Phishing Websites Using an Efficient Feature-based Machine Learning Framework. *Neural Comp. and Applic.*, 31(8):3851–3873.

Sahingoz, O. K., Buber, E., Demir, Ö., and Diri, B. (2019). Machine learning based phishing detection from URLs. *Expert Syst. Appl.*, 117:345–357.

Sameen, M., Han, K., and Hwang, S. (2020). PhishHaven -An Efficient Real-Time AI Phishing URLs Detection System. *IEEE Access*, 8:83425–83443.

Santos, W., Fazzion, E., Fonseca, O., Cunha, I., Chaves, M., Hoepers, C., Steding-Jessen, K., Guedes, D., and jr, W. M. (2019). Uma Metodologia para Agrupamento e Extração de Informações de URLs de Phishing. In *Proc. of Brazilian Symposium on Information and Computational Systems Security (SBSeg)*.

Shirazi, H., Bezawada, B., and Ray, I. (2018). "Kn0w Thy Doma1n Name": Unbiased Phishing Detection Using Domain Name Based Features. In *Proc. of Symposium on Access Control Models and Technologies*.

Spamhaus (2021). https://www.spamhaus.org/.

Steding-Jessen, K., Vijaykumar, N., and Montes, A. (2008). Using Low-Interaction Honeypots to Study the Abuse of Open Proxies to Send Spam. *Journal of Computer Science*, 7(1):44–52.

Symantec (2019). Internet Security Threat Report. Technical report.

Xiang, G., Hong, J., Rose, C. P., and Cranor, L. (2011). CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites. *Transaction on Information System Security*, 14(2):21:1–21:28.

Zhang, W., Jiang, Q., Chen, L., and Li, C. (2016). Two-stage ELM for phishing Web pages detection using hybrid features. *World Wide Web*, 20(4):797–813.