Waste Flooding: Ferramenta para Retaliação de Phishing

Cristoffer Leite, João J. C. Gondim[1], Priscila Solis Barreto [1], Eduardo A. Alchieri [1]

[1] Departamento de Ciência da Computação (CIC)
Universidade de Brasília (UnB) – Brasilia, DF – Brazil

cristoffer@aluno.unb.br,{gondim,pris,alchieri}@unb.br

Abstract. *Phishing is a well known attack technique but it is still a growing threat. The Internet rising popularity potentialized phishing possibilities giving attackers a group of new instruments and allowing closer contact to its focus, the user. By applying social engineering methods, phishing thrives on misinformation. That's why, current main phishing response methods focus only on educating users or blocking phishing attempts, without any response to derail already implemented attacks. These conditions may leave targeted users unprotected, as any leaked information can not be tracked to determine who suffered from phishing and compromised data can not be saved or easily detected. In this paper, we present and discuss a new response tool that aims to furtively retaliate by automatic detecting phishing forms and using them to clutter phishing databases.*

Resumo. *O phishing é um tipo de ataque bem conhecido, mas que ainda é uma ameaça crescente. A popularidade da Internet potencializou as possibilidades de phishing, dando aos atacantes um grupo de novos instrumentos e permitindo um contato mais próximo ao seu foco, que é o usuário. Ao aplicar métodos de engenharia social, o phishing prospera pela desinformação. É por isso que atualmente os principais métodos de resposta a phishing se concentram apenas em educar os usuários ou bloquear tentativas de phishing, sem nenhuma resposta para atrapalhar os ataques já implementados. Essas condições podem deixar os usuários desprotegidos, pois qualquer informação previamente vazada não pode ser rastreada para determinar quem sofreu phishing, deixando os dados comprometidos sem possibilidade de serem salvos ou facilmente detectados. Neste artigo, apresentamos e discutimos uma nova ferramenta de resposta que visa retaliar furtivamente, detectando automaticamente os formulários de phishing e os usando para confundir os bancos de dados de phishing.*

1. Introdução

Phishing é uma técnica maliciosa que tenta enganar pessoas e obter informações privadas. Atacantes contactam pessoas através de emails, mensagens SMS ou até mesmo através de chamadas telefônicas. O objetivo principal é induzir as pessoas a inserir dados pessoais em formulários fraudulentos, que enviam os dados coletados para bancos de dados usados por criminosos virtuais O objetivo é roubar o dinheiro da vítima ou até mesmo usar as informações coletadas em atividades criminosas. O problema fundamental dos ataques de *phishing* reside no fato de tentar pescar pessoas usando engenharia social e, no caso de páginas fraudulentas, enganar pessoas com pouco conhecimento sobre sistemas de computadores, fazendo-as acreditar que estão inserindo suas informações em sites confiáveis.

De acordo com [Kaspersky 2019], o número de ataques de *phishing* em todo o mundo mais do que dobrou no ano passado, chegando a 500 milhões. O Brasil é o país com o maior número de ataques de *phishing*, com 28 % de seus cidadãos recebendo pelo menos uma tentativa de *phishing* em 2018.

Embora as técnicas de detecção de *phishing* tenham melhorado muito na última década, sua força ainda reside na aplicação de engenharia social e na atração de pessoas com front-ends familiares adaptados a contextos específicos [Jagatic et al. 2005]. Essa característica ajuda os invasores a contornar as ferramentas de detecção de *phishing* por causa do contato direto com o usuário. Uma estratégia popular para se defender contra o poder social de *phishing* é a educação contínua dos usuários sobre novos métodos de *phishing* [Williams et al. 2018], por vezes considerada uma abordagem cara e pouco eficiente. Barras de ferramentas anti-*phishing*, filtros de SPAM, monitores de *phishing* e até sistemas preditivos recentes baseados em Machine Learning e Data Mining não conseguem verificar todas as conexões com perfeição [Qabajeh et al. 2018]. Além disso, nenhuma dessas técnicas consegue identificar os dados já fornecidos pelo usuário para páginas maliciosas.

Com base no exposto, a contribuição deste artigo está na introdução de uma nova ferramenta que detecta formulários de *phishing* em páginas maliciosas e, com uma variedade de proxies, inunda o banco de dados do invasor com dados inúteis, desorganizando seus registros e mascarando a origem dos envios para evitar possíveis respostas de negação de serviço. A intenção não é substituir as ferramentas atuais nem propor um método de resposta completa para tentativas de *phishing*, mas permitir um sistema de retaliação automatizado e de difícil detecção. A retaliação é viável pois a detecção é precisa, já que o sistema utiliza links de *phishing* recebidos. Assim, é mais difícil para os atacantes de *phishing* usarem os dados reunidos em ataques, ocultando-os em meio a informações inúteis.

Este artigo está estruturado da seguinte forma: a Seção 2 apresenta um resumo sobre o *phishing* na Web, técnicas tradicionais e uma visão geral de seus componentes, seguida de pesquisa de ponta sobre detecção e operação de resposta de *phishing*. A seção 3 mostra detalhes da arquitetura da ferramenta com uma descrição de cada módulo. A Seção 4 detalha os benefícios de tecnologias adotadas e métodos definidos. A seção 5 apresenta os requisitos de implementação da ferramenta e um grupo de testes que aplica a ferramenta em um ambiente real, com uma análise de como ela pode ser medida. O objetivo desses testes é demonstrar a eficácia da ferramenta proposta e verificar se os resultados confirmam a ocultação da fonte e a confusão na base de dados. A seção 6 apresenta conclusões e desenvolvimentos futuros da pesquisa.

2. Base Teórica e Estado da Arte

2.1. *Phishing* na Internet

Ataques de *phishing* têm o objetivo de enganar a vítima e fazer com que ela revele dados pessoais, como senhas e outras informações confidenciais. Para realizar um ataque de *phishing*, os agentes maliciosos usam três componentes principais: um meio, geralmente usando a Internet, SMS ou chamadas de voz; um vetor, como e-mails, redes sociais, mensagens instantâneas ou *Vishing* (*phishing* baseado em voz); e as estratégias técnicas, utilizando uma infinidade de opções aqui [Chiew et al. 2018]. De acordo com a Symantec

[Symantec 2019], os e-mails de *phishing* (*phishing* orientados) prevalecem como o vetor de ataque mais difundido, com 65% de todos os grupos ou organizações que foram alvo de ataques tendo recebido eles. Este trabalho se concentra em ataques de *phishing* que usam a Internet como meio principal.

Para implementar ataques de *phishing* baseados na Web, os agentes mal-intencionados geralmente aplicam componentes da Web conhecidos como kits de *phishing*, que servem para imitar páginas legítimas da Web. É fácil encontrar um kit de *phishing* à venda no mercado negro [Symantec 2019] ou até mesmo disponível gratuitamente, mas alguns deles vêm com funcionalidades escondidas que vazam os dados coletados para o desenvolvedor original. Esses kits geralmente usam formulários HTTP mascarados com interfaces visuais familiares para enganar usuários e coletar dados. Os métodos básicos de detecção de *phishing* podem reconhecer e bloquear o acesso a URLs de kits simples, mas a maioria das instâncias do kit de *phishing* aplica uma combinação de técnicas de ocultação que permite que eles sobrevivam por uma média de 10 horas sem serem detectadas [APWG 2014]. É comum em esquemas de *phishing* utilizem uma combinação de extensos pontos de acesso, permitindo com que essas operações coletem dados por meses [Cui et al. 2017].

2.2. Trabalho Relacionados

Há poucos resultados observados em outros estudos sobre realizar retaliações de *phishing* voltada para embaralhar a base de dados, já que a literatura clássica foca na detecção de ataques de *phishing* sem respostas reativas direcionadas. [Moore and Clayton 2007] mostram que a remoção de sites de *phishing* da Internet, apesar de ajudar a combater ataques, não funciona em um ritmo viável para resolver o problema sozinho. De acordo com [Qabajeh et al. 2018], a maioria das abordagens anti*phishing* bloqueia apenas o conteúdo de *phishing* utilizando listas negras e listas brancas ou detecção automatizada de *phishing*.

Classificação é o principal problema na detecção de *phishing*. Assim, uma tendência recente de pesquisa é usar abordagens baseadas em Inteligência Artificial para encontrar páginas de *phishing* na Web. Em [Khade and Shinde 2014] os autores utilizam mineração fuzzy de dados para detectar tentativas de *phishing*, gerando imediatamente uma notificação para o administrador da rede e para o host do site sobre a página. Esse método usa a alta resiliência do fuzzy à ambigüidade para identificar e-mails de *phishing*. A segunda parte é um mecanismo de resposta ativa que tenta remover a página da Internet e impedir seu sucesso.

Uma abordagem semelhante é seguida por [Aburrous et al. 2008] para detectar *phishing* em bancos virtuais, utilizando lógica fuzzy na modelagem de subjetividade de sistemas de segurança bancária e produzindo medidas de risco de ataques com uma estrutura hierárquica de camadas. O sistema não gera respostas de ataque, sendo apenas uma classificação de ameaça baseada em critérios diversos.

[Mcrae et al. 2007] propõem o uso de bugs da web e técnicas anti-*phishing* baseados em honeypot com honeytokens para rastrear os invasores e descobrir sua localização. A ideia é criar credenciais de usuário limitadas que serão reconhecidas quando vazadas, permitindo fácil identificação de acesso não autorizado ao monitorar essas contas, ao mesmo tempo em que também permite acompanhar cada local de tentativa de login com essas credenciais.

O *framework* apresentado por [Li and Schmitz 2009] melhora ainda mais este conceito, criando um sistema de honeypot bancário equipado com honeytokens. A estrutura usa um sistema detector de *phishing* que rastreia o comportamento do usuário e os replica. Usando este sistema de replicação, os honeypots executados em máquinas virtuais, denominadas honeybots, enviam ativamente esses honeytokens para sites de *phishing*, distribuindo o mecanismo de detecção mais facilmente.

Até onde sabemos, não há nenhum trabalho que se concentre na implementação de uma ferramenta para a mitigação ativa de *phishing*, introduzindo uma abordagem de retaliação e, ao mesmo tempo, evitando a detecção.

3. Arquitetura e Visão da Solução

A solução proposta neste trabalho é uma ferramenta que analisa recursivamente padrões de formulários HTML em páginas da *Web* de *phishing*. A análise permite gerar conteúdo inútil para cada modelo de formulário detectado e inundar as tentativas de *phishing*. Essa retaliação de *phishing* pode estimular uma resposta do invasor e, por causa disso, a ferramenta também aplica uma combinação de mecanismos de ocultação e randomização de dados para evitar a detecção.

3.1. Visão Geral da Solução

A ferramenta possui quatro etapas principais: (1) Um atacante envia uma tentativa de *phishing*, tentando coletar dados. A URL fornecida pelo atacante pode ser utilizada pela ferramenta para gerar e enviar dados inúteis; (2) A ferramenta recursivamente identifica formulários HTML na página da URL dada e também em cada nova página encontrada a partir dela, e então continua gerando grupos de dados inúteis para cada modelo de formulário identificado; (3) Para cada formulário, a ferramenta cria solicitações válidas com os dados inúteis gerados, enviando cada uma para um *proxy* aleatório de sua coleção; (4) Esses proxies então enviam as informações inúteis para o site de *phishing* para inundar seu banco de dados. A Figura 1 ilustra cada passo descrito. É importante ressaltar que a ferramenta não faz a identificação do *phishing*, sendo necessário informar a URL.

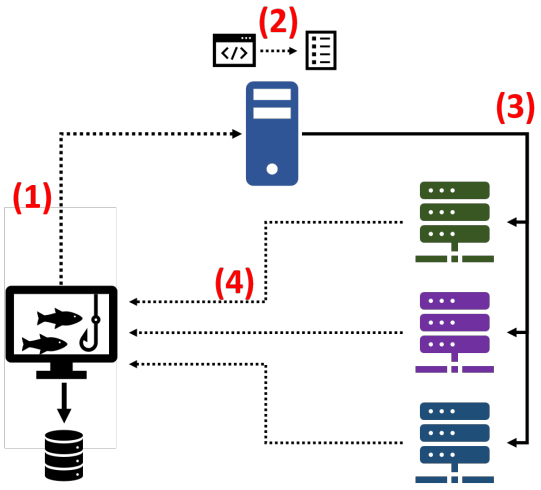

Figura 1. Arquitetura Geral da Ferramenta

3.2. Arquitetura do Sistema e dos Componentes

A ferramenta possui dois módulos principais: um com foco na análise de formulários HTML em uma página Web e gera um modelo para o formulário; e um que produz dados aleatórios para enviar por meio de proxies usando os modelos gerados. Esta divisão permite o uso de módulos de inundação em uma estrutura desacoplada para promover facilmente melhorias e adaptações futuras para cenários específicos sem que isso acarrete em mudanças severas. A ideia é criar um ambiente viável e permitir a adição de novos módulos como serviços consumidores da interface da ferramenta.

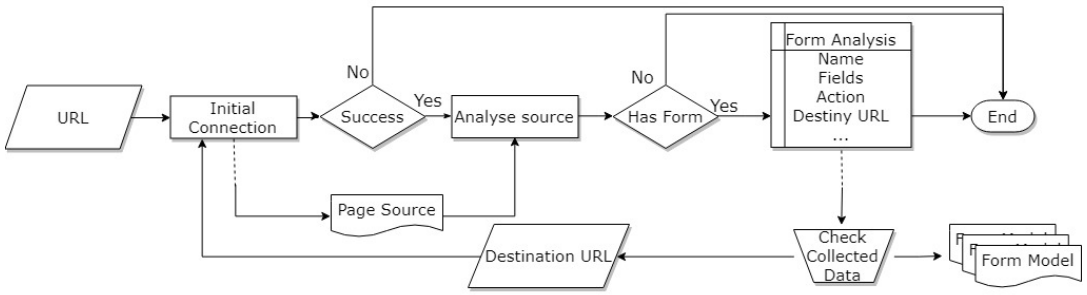

Figura 2. Arquitetura do Módulo de Formulário

A figura 2 mostra a arquitetura do módulo de formulário. Com uma URL, o módulo cria uma conexão inicial para verificar sua disponibilidade, além de extrair o código-fonte da página no processo. Em seguida, analisa o código extraído para verificar a existência de um formulário. A ferramenta cria um modelo para qualquer formulário detectado, com todas as suas entradas, sua URL de ação (o destino do formulário) e as validações identificadas. O modelo armazena o tamanho de entrada máximo e mínimo, o tipo de entrada e quaisquer outras validações detectadas. Depois de analisar um formulário, se algo relevante for encontrado, o módulo também verificará o código-fonte da URL de destino do formulário, repetindo o mesmo processo.

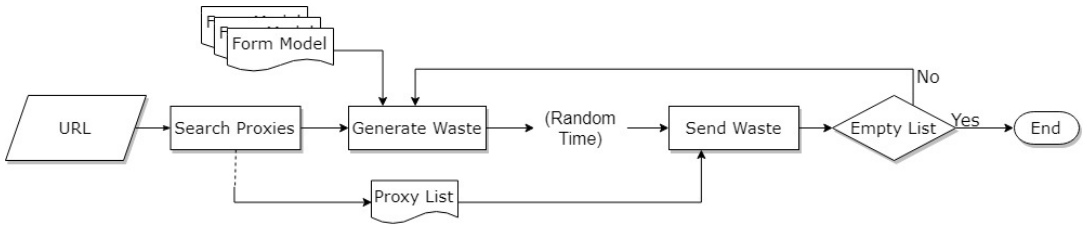

Figura 3. Arquitetura do Módulo de Flood

A figura 3 especifica a arquitetura do módulo de flooding. Para evitar a detecção, este módulo usa uma extensa lista de *proxy* de várias fontes. Para cada modelo de formulário gerado pelo módulo anterior, o sistema gera uma carga de dados inúteis, validando qualquer campo possível para ser indistinguível dos dados reais. Usando a lista de *proxy*, ela envia solicitações intermitentes em horários aleatórios para cada formulário, renovando os dados inúteis gerados a cada vez. Duas características importantes deste módulo são o uso de uma pausa aleatória entre as requisições para evitar uma detecção fácil por meio de análise de frequência de envios e o uso de uma extensa lista de proxies para evitar qualquer retaliação como, por exemplo, uma negação de serviço.

3.3. Características Gerais e Benefícios

Para suportar o uso em diferentes sistemas operacionais e manter a simplicidade de código, Python 3.7 foi usado como linguagem de desenvolvimento para executar a detecção de formulários e a automação de inundação. Esta escolha também foi feita para permitir que qualquer pessoa com habilidades de programação modifique o código e reutilize-o em outros casos com facilidade. A modularização foi adicionada para suportar adaptações adicionais com o uso das premissas da Arquitetura Orientada a Serviços (SOA, em inglês). As bibliotecas utilizadas pela ferramenta são: *gevent*, *requests* e *lxml*.

A ferramenta realiza pausas de comprimento aleatório para evitar saturar a comunicação e também para evitar a utilização um padrão de acesso, algo que seria facilmente identificável. A geração de lista de *proxy* cria um banco de dados de mais de cem mil proxies para contornar qualquer possibilidade de detecção de origem, evitando assim possíveis respostas de invasores. Quaisquer entradas inúteis geradas são validadas usando tipos de dados padrão, como Número de Registro e Número de Cartão de Crédito, definidos pelo seu comprimento mínimo e máximo. Esses parâmetros podem ser alterados e novos módulos podem implementar regras diferentes para cenários específicos. A ferramenta é baseada em linha de código e requer apenas um URL como entrada. O programa é projetado para mostrar cada tentativa de inundação em um esquema de visualização simples. Um recurso opcional permite a exibição de formulários detectados, seus campos e URLs de ação, como mostrado na Figura 4, ajudando a visualizar o cenário de ataque.

```
+ URL: ebApp_AplicationHome.php?brazil=01,13,49,AM,165,6,06,000000,15,1,2019,Saturday.seguro
|       Action: entification.php?brazil=04,21,17,AM,174,6,06,000000,24,4,2019,Monday.seguro
|       Inputs:
|               casa - Range: ('6', '6')
|               marca - Range: ('9', '9')
|               s8 - Range: ('8', '8')
+ URL: licationIdentification.php?brazil=04,21,17,AM,174,6,06,000000,24,4,2019,Monday.seguro
|       Action: webApp_AplicationCard.php
|       Inputs:
|               casa - Range: ('6', '6')
|               marca - Range: ('9', '9')
|               s8 - Range: ('8', '8')
|               cpf - Range: ('11', '11')
|               ddd - Range: ('2', '2')
|               numero - Range: ('9', '10')
|               s6 - Range: ('6', '6')
+ URL: webApp_AplicationCard.php
|       Action: envios/envio2.php
|       Inputs:
|               casa - Range: ('6', '6')
|               marca - Range: ('9', '9')
|               s8 - Range: ('8', '8')
|               cpf - Range: ('11', '11')
|               ddd - Range: ('2', '2')
|               numero - Range: ('9', '10')
|               s6 - Range: ('6', '6')
|               cc - Range: ('19', '19')
|               validade - Range: ('7', '7')
|               cvv - Range: ('3', '3')
|               sletra - Range: ('6', '6')
```

Figura 4. Visualização da Detecção de Formulários

4. Implementação e Resultados

A implementação do protótipo está disponível on-line[1] para testes e inspeção adicional. O repositório possui documentação completa sobre a instalação da ferramenta e todos

[1]https://github.com/imperador/WasteFlooding

os requisitos de software, incluindo cada plugin necessário. Uma coleção de domínios inseguros conhecidos com kits de *phishing* será usada na demonstração para confirmar a detecção correta de formulários e verificar o êxito das tentativas de requisições, mas um host local com um kit de *phishing* implantado também pode ser usado para verificar se o banco de dados de *phishing* foi inundado com sucesso. Isso será necessário somente se os revisores quiserem verificar as oportunidades de desvio de inundação do banco de dados, mas também pode ser útil propor melhorias futuras para a ferramenta. A fonte de inundação na solução implementada será um computador normal, pois a ferramenta não requer uma fonte poderosa para realizar requisições. Outro script python no host monitora todos os dados enviados e também mapeia cada tempo de solicitação, permitindo assim uma melhor visualização do fluxo de dados criado e da aleatorização do tempo de espera.

A ferramenta permite a detecção de formulários de *phishing* e o mapeamento exaustivo em kits de *phishing* pode ser testado usando a aplicação em uma coleção de kits vulneráveis. O mascaramento de fonte, a geração de dados inúteis e aleatórios usando solicitações válidas e a inundação do banco de dados também podem ser verificadas usando o mesmo esquema. Os dados podem ser analisados utilizando o tempo entre as solicitações observadas pelo servidor com o kit de *phishing*.

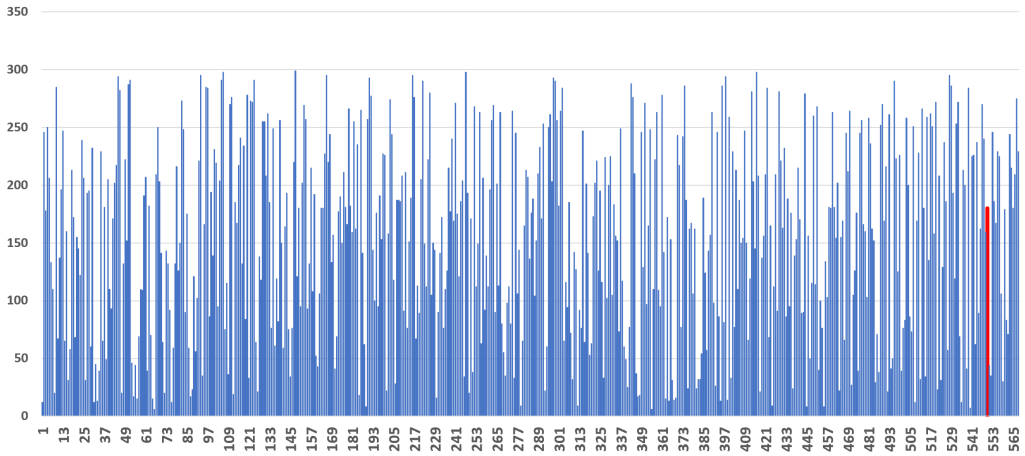

Figura 5. Tempo de Espera entre Requisições, e Colisão de *Proxy* em Vermelho

A figura 5 exemplifica uma amostra das requisições recebidas pelo servidor com o script sendo executado por um dia inteiro e aguardando tempos aleatórios entre 5 segundos e 3 minutos, gerando mais de 500 requisições. Outro ponto importante na ocultação de origem é a randomização do *proxy* usado. Nos dados analisados, uma colisão, ou seja, o uso da mesma fonte outra vez, ocorreu após 23 horas e 15 minutos, ou 552 solicitações, conforme mostra a Figura 5 em vermelho. É importante mencionar que os dados gerados para cada formulário não tiveram nenhuma repetição durante os testes preliminares.

5. Conclusões e Trabalhos Futuros

Para realizar ataques de *phishing*, os agentes maliciosos tentam persuadir usuários aleatórios ou selecionados a inserir seus dados em sites falsos. Frequentemente, esses usuários não têm informações para distinguir completamente essas páginas das reais e, em seguida, fornecer seus dados sem saber. Respostas comuns a *phishing* envolvem geralmente treinamento de usuários e detecção de páginas mal-intencionadas, não lidando

com informações já vazadas de maneira viável, portanto, usuários que não têm conhecimento suficiente sobre essas tentativas representam alvos fáceis.

A solução prospota neste artigo visa mitigar vazamentos de informações do usuário e também criar respostas proativas para tentativas de *phishing*. O sistema proposto combina análise de formulários HTML, geração de dados aleatórios inúteis e anonimização da fonte. O resultado é uma solução resistente à detecção e, consequentemente, uma maneira viável de misturar dados reais do usuário com pedaços aleatórios de informações, sem a fácil divulgação da fonte. Com a ferramenta fornecida, mesmo que o invasor tente procurar dados úteis, levaria muito tempo. A ferramenta desenvolvida não pode detectar dinamicamente cada tipo de campo possível. Mas, em particular, permite a criação e integração de novos módulos para identificar casos específicos. Para trabalhos futuros, planeja-se expandir essa ferramenta para que ela possa detectar mais tipos de campos e também implementar novas formas de detecção autônoma de *phishing*, permitindo que a ferramenta seja combinada com sistemas de rastreamento.

Referências

Aburrous, M. R., Hossain, A., Thabatah, F., and Dahal, K. (2008). Intelligent quality performance assessment for e-banking security using fuzzy logic. In *Fifth International Conference on Information Technology: New Generations (itng 2008)*. IEEE.

APWG (2014). Global Phishing Report 2H 2014. Technical report.

Chiew, K. L., Yong, K. S. C., and Tan, C. L. (2018). A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Systems with Applications*, 106:1–20.

Cui, Q., Jourdan, G.-V., Bochmann, G. V., Couturier, R., and Onut, I.-V. (2017). Tracking Phishing Attacks Over Time. pages 667–676.

Jagatic, T., Johnson, N., Jakobsson, M., and Menczer, F. (2005). Social Phishing *. Technical report.

Kaspersky (2019). Phishing attacks more than doubled in 2018 to reach almost 500 million.

Khade, A. and Shinde, S. (2014). Detection of phishing websites using data mining techniques. *International Journal of Engineering Research and Technology*, 2(12).

Li, S. and Schmitz, R. (2009). *A Novel Anti-Phishing Framework Based on Honeypots*.

Mcrae, C. M., Vaughn, R. B., and Research, S. (2007). Phighting the Phisher: Using Web Bugs and Honeytokens to Investigate the Source of Phishing Attacks. Technical report.

Moore, T. and Clayton, R. (2007). Examining the Impact of Website Take-down on Phishing. Technical report.

Qabajeh, I., Thabtah, F., and Chiclana, F. (2018). A recent review of conventional vs. automated cybersecurity anti-phishing techniques. *Computer Science Review*, 29:44–55.

Symantec (2019). ISTR - Internet Security Threat Report Volume 24. Technical report, Symantec Corporation.

Williams, E. J., Hinds, J., and Joinson, A. N. (2018). Exploring susceptibility to phishing in the workplace.