SMILE - Smart eMaIl Link Domain Extractor

Mattia Mossano(✉), Benjamin Berens, Philip Heller, Christopher Beckmann, Lukas Aldag, Peter Mayer, and Melanie Volkamer

Karlsruhe Institute of Technology, Karlsruhe, Germany
{mattia.mossano,benjamin.berens,philip.heller,christopher.beckmann, lukas.aldag,peter.mayer,melanie.volkamer}@kit.edu

Abstract. Phishing over email continues to be a significant threat, as such messages still end up in users' inboxes. Several studies showed that users rarely check the URL in the statusbar before clicking a link and that they have difficulties reading URLs. To support users, we propose SMILE (Smart eMaIl Link domain Extractor), a novel approach that provides the relevant information to distinguish between legitimate and phishing emails checking the links in them. Once applied, SMILE modifies all links in an email to contain the domain and top-level-domain of the URL behind them, e.g., "Click here" in an legitimate Amazon email is modified to "Click here [amazon.com]".

Keywords: Anti-phishing intervention · User support · URL analysis

1 Introduction

Phishing is still a growing threat, e.g., the Anti Phishing Working Group [6] shows that in 2020 the number of phishing websites doubled. Despite improved phishing detection tools, phishing emails still reach people's email inboxes. While there are various types of phishing emails, we focus on those containing dangerous links that could download malware or take victims to phishing webpages. Simple phishing emails can be detected through sender address or typos, but this is not the case for sophisticated attacks. Yet, all phishing emails with links can be identified through the URL behind each link. However, [23] showed that most people are not aware of this and [3] demonstrated that people have problems reading URLs correctly.

We propose SMILE, **S**mart e**M**a**I**l **L**ink domain **E**xtractor. SMILE checks the HTML code of an email to detect links and modifies them only to contain what we call a "SMILE-string". These strings are then the only clickable elements in the respective email. They can be the domain and top-level-domain (TLD) of the URL behind a link, an IP address, or include some subdomains (e.g., sites.google.com). This paper presents the SMILE concept and its working.

S. Katsikas et al. (Eds.): ESORICS 2021 Workshops, LNCS 13106, pp. 403–412, 2022.
https://doi.org/10.1007/978-3-030-95484-0_23

2 Related Works

There are various solutions to provide tool based anti-phishing support to users. Tools can, e.g., analyse and operate on malicious emails/websites content [1,16,25], work on the DNS side [4,7,10] or identify malicious websites with machine learning [2,8,12]. They either block or warn users if the risk is above a predefined threshold. Yet, there is no 100% guarantee for detection. SMILE differs from them as it helps users to identify phishing emails, not block the latter.

Researchers have proposed in [5,18,20,22] two different solutions to support users to analyse emails URLs. In [18,20,22], the authors show a tooltip just-in-time and just-in-place with the URL behind links. Some parts of SMILE, namely the SMILE-string resolution and SMILE special cases (see Sect. 5), are based on [20,22]. In [5], the authors propose a chat-bot that helps users to decide the legitimacy of a link through text interactions. SMILE advantage over them is placing the relevant information (and only that) whenever an email is opened.

Valve's videogame digital distribution service, STEAM, employs a SMILE-like security feature on their forum. In [21] there are some examples of the forum text formatting, however, we found no official documentation explaining this security feature. Thus, we conducted some tests on their platform, shown in the Appendix (Fig. 3). Valve adds domain and TLD of the URL after the link itself, between square parenthesis, as normal text with darker font colour and smaller character size. The feature only applies to textual links but only those where the text is a URL which does not start with http/https. SMILE modifies textual links (with and without protocol), image links, short URLs, and moves the clickable element to the SMILE-string.

3 Background on Link-Types

Our interest are links in emails. Thus, we focus on the four ways to create them:

- *Anchor-Element*, also known as a-tag
- *Form-Element*, that can send data to a link given by the "action" attribute
- *Formaction-Attribute*, special form-elements with the "formaction" attribute
- *Area-Element*, enable areas in a (possibly transparent) image to be clickable

We would like to make these remarks: (1) The term "link" usually indicates only anchor-elements, but we use it for all four for simplicity. (2) Links can be created with JavaScript, but the common web mail services and clients [15] block it [9]. Thus, we do not consider it further. (3) From the users' point of view, form-elements and formaction-attributes are indistinguishable.

There are three *link-types* for anchor-element, form-element, and formaction-attribute: *Image*, *URL-like*, and *Misc* (e.g., "Click here"). Area-tag is only applicable for the link-type Image (as we consider its usage in the email context).

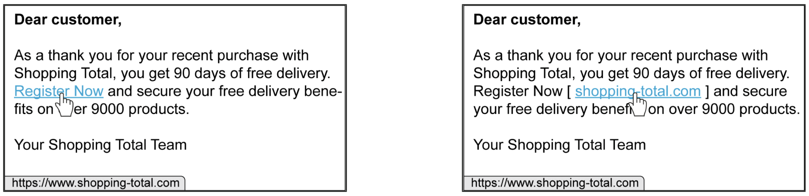
In summary, SMILE needs to cope with *ten different situations* (= 3 anchor-elements + 3 form-elements + 3 formaction-attributes + 1 area-element).

4 SMILE: General Idea

The general idea underlying SMILE is to enhance the transparency of every email link by substituting them with easy-to-read versions whenever the email is opened without further user action. After applying SMILE, the email contains textual links with the *SMILE-string*. This can be the domain and the TLD of the original link URL, an IP address, or include some subdomains (e.g., sites.google.com). The SMILE-string only provides the minimum information required to decide on a URL legitimacy (i.e., no automated URL analysis). Note, the statusbar is left untouched and it shows the entire URL on mouse-hover.

Our design principle has four motivations: (1) substituting every link at once saves time, as users do not have to check them independently (e.g., as with a tooltip). (2) Only placing the SMILE-string, instead of the URL, reduces the efficacy of phishing URLs with misplaced legitimate domains. (3) The relevant security indicator (SMILE-string) is in the email body, i.e., just-in-place, as recommended in [18]. (4) Limited information prevents conflicts/overlap with other tools, e.g., the solution in [22]: users wanting more information can combine SMILE with other tools. An example of an email modified by SMILE is shown in Fig. 1.

Toggle Function. SMILE might make complex emails unreadable. Thus, we implemented a toggle function to undo all substitutions on demand.



Dear customer,

As a thank you for your recent purchase with Shopping Total, you get 90 days of free delivery. Register Now and secure your free delivery benefits on over 9000 products.

Your Shopping Total Team

https://www.shopping-total.com

(a) Without SMILE

Dear customer,

As a thank you for your recent purchase with Shopping Total, you get 90 days of free delivery. Register Now [shopping-total.com] and secure your free delivery benefits on over 9000 products.

Your Shopping Total Team

https://www.shopping-total.com

(b) With SMILE

Fig. 1. Link in email, without and with SMILE.

5 SMILE: Algorithm

A high-level description of the algorithm is depicted in Fig. 2. Note, the credit for the processes described in “Resolve SMILE-string” (*short URL service*, *redirects*, *IP-address*, legitimate TLDs recognition and *punycode*) and “Resolve special case SMILE-string” (*programmed tooltips* and *dangerous files at cloud services*) goes to the authors of [20,22]. However, differently from TORPEDO, SMILE: (1) only shows the SMILE-string, not the entire URL, (2) adds the SMILE-string to the email text, not in a tooltip, (3) substitutions are situation based, and (4) substitutions are shown whenever the email is opened, not only on mouse-hover.

Identify Link-Type. First, SMILE searches for a link and identifies the link-type (see Sect. 3 for the different link-types).

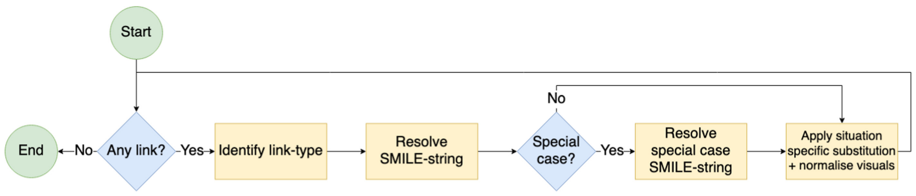

Start
End
No
Any link?
Yes
Identify link-type
Resolve SMILE-string
No
Special case?
Yes
Resolve special case SMILE-string
Apply situation specific substitution + normalise visuals

Fig. 2. Flow chart showing the algorithm behind SMILE.

Resolve SMILE-String. SMILE extracts the URL from the link and checks if it points to a known *short URL service* or a known *redirect service*. In these cases, the URL is meaningless to the user to determine whether it is a phishing one or not. SMILE resolves the final destination, if needed by repeatedly applying this step. Then, the resolved URL is set as actual URL.

For the *short URL service*, SMILE loads the headers of the service to check the target location following the HTTP 3xx server response. This can be a privacy issue, as it allows the link owner to get the user IP address Therefore, the users can configure SMILE to not send any request and to show domain and TLD of the short URL service.

For the *redirects*, SMILE resolves the URL from the path of the actual URL applying rules that recognise the structure used by known redirect services. For example, from google.de/url?url=https%3A%2F%2Fexample.com%2Fdocs to https://example.com/docs.

Afterwards, SMILE checks whether the URL is an *IP address*, i.e., IPv4 or IPv6. In this case, the SMILE-string is the complete IP address and SMILE stops processing it.

In case the URL is not an IP address, SMILE deduces from it the SMILE-string by extracting the domain. The information regarding actual TLDs and domains comes from the Mozilla Foundation's Public Suffix list [17] using the solution in the "publicsuffixlist.js" GitHub project [13].

Finally, SMILE checks the extracted SMILE-string for specific characters in the *Unicode character space*. This check is to prevent homographic techniques, i.e., using similar-looking characters from other character spaces in the domain or TLD, e.g., using a Cyrillic "e" for the domain "google.de". SMILE replaces non-ASCII characters in the SMILE-string with puny code, e.g., xn–googl-7of.de. This approach is already used in many programs, e.g., Google Chrome 51+ [11].

Resolve Special Case SMILE-String. SMILE addresses three special cases: (1) *programmed tooltips*, (2) *dangerous files at cloud services* and (3) *website creation and hosting tools*.

In the *first case*, the tooltip contains the legitimate URL for the link and is meant to distract users from the actual URL in the statusbar. If users do not know that a tooltip is not expected in a context, they can mistakenly consider the URL in the tooltip as a legitimate location, thus clicking on a fraudulent link. SMILE checks for programmed tooltips and blocks them from being displayed.

In the *second case*, phishers would store dangerous files at a cloud service and provide a link to them in the email. In this case, the SMILE-string alone might be misleading to users. Therefore, SMILE checks for the URL structure of well known cloud service providers (e.g., Dropbox, Google Drive, OneDrive). It then adds a warning before the SMILE-string link: "[Only click if you were expecting this email, as you are redirected to a cloud service:]".

In the *third case*, SMILE checks for well known website creation and hosting tools, e.g., Google Sites, Microsoft Azure. The SMILE string is then extended with the subdomain (e.g., sites.google.com) and a warning is added before the SMILE-string link: "[Only click if you were expecting this email, as you are redirected to a webpage that could have been set up by anyone:]".

Apply Situation Specific Substitution + Normalise Visuals. Each of the 10 situations identified in Sect. 3, is treated differently by SMILE:

- *Image link-type.* A link including an image as its first and only child. SMILE disable the link and adds the phrase "Image link:" above the image, followed by the SMILE-string between square parenthesis.
- *URL-like link-type.* A textual link appearing to be a URL. SMILE detects specific patterns (i.e., http, https, www, /) or a specific structure (i.e., *domain.tld*). Thus, SMILE works both on extended URLs, e.g., https://www.amazon.co.uk, and reduced ones, e.g., amazon.co.uk. In this case, SMILE substitutes the text content of the element with the SMILE-string.
- *Misc link-type.* Links whose content is neither an image nor URL-like, e.g., "Click here". To preserve the information in the misc type text, SMILE keeps the text but disables the link, i.e., it is just normal email text. SMILE adds the SMILE-string between square brackets as a new link right after this text.

These three link-types cover 9 of the 10 situations. The last situation are *maps with clickable areas over images.* SMILE adds a list of links above the image, analogously to the *Image link-type* described above. The map is then removed from the image to disable the clickable parts. This approach loses the original image area contextual information, but introduces a clear list of links.

Note that links created through form-elements and formaction-attributes (Sect. 3) do not show the URL behind them on mouse-hover. Without SMILE, the only way to check their URL is inspecting the HTML code of the webpage. However, since every SMILE-string is an anchor-element textual link, users can check their URL on mouse-hover, accessing otherwise hidden information.

Code examples in HTML for each of the ten situations are provided in the Appendix (Table 2). Examples of the substitutions can be seen in Table 1.

SMILE also normalises the font visuals. For example, it applies a minimum font size to prevent a too small to read SMILE-string and it checks for enough contrast with the background colour to make the SMILE-string easily legible.

6 Discussion

SMILE can work on both the receiving email server (*central approach*) or the email client, i.e., the software or app used by the user (*local approach*).

Central Approach. No user installation is required and SMILE is also available on mobile devices. However, the toggle function is either not available or users

Table 1. Examples of SMILE substitutions. Note, SMILE also works on buttons and button-like links, i.e., images of buttons and CSS modified anchor-elements.

		Without SMILE	*With SMILE*
Anchor, Form, Formaction	Image		Image Link: [example.org]
		Start Now	Image Link: [example.org] Start Now
	URL-Like	Register now at https://example.com/register.	Register now at [example.com].
		example.org/path	[example.org]
	Misc	Register now and secure benefits.	Register now [example.com] and secure benefits.
		Start Now	Start Now [example.org]

	Without SMILE	*With SMILE*
Area		Area Link: [example1.com] Area Link: [example2.com]

have to install an extension. Moreover, the DKIM authentication method has issues, as it uses digital signatures over the email body and any alteration leads to a client-side failed check. A workaround is for the server to perform the DKIM check and pass it on to the user, e.g., by changing the subject. End-to-end encryption mechanisms, e.g., PGP or S/MIME, can be used, but SMILE would not work, as the email text would not be accessible (i.e., encrypted).

Local Approach. DKIM authentication, end-to-end encryption and toggle function work. However, SMILE needs to be adapted to various email clients as an extension (i.e., add-on). Note, Outlook, Apple Mail and mobile clients require specific solutions.

As future work, we plan an evaluation of the SMILE concept, i.e., its effectiveness in different settings (e.g., mobile and desktop) as well as its performance in comparison to the tooltip proposed in [22]. This could be conducted in two ways: in a *non-interactive environment* and in an *interactive environment*. Both approaches have been used in the past (e.g., [14,19,22,24] used a non-interactive environment and [5,22] used an interactive one). We believe both options to be worthwhile, as each can potentially show different aspects. Hence, we plan to evaluate SMILE in both ways. Furthermore, we want to check SMILE in the real world to see how much the toggle function is required.

7 Conclusion

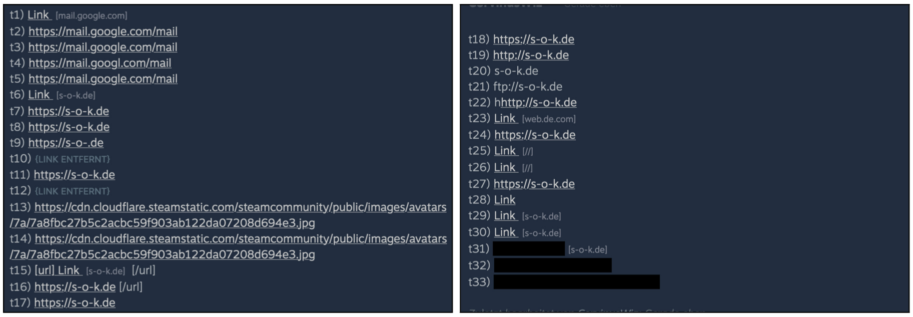
We propose SMILE, a new security intervention supporting users while detecting phishing emails. SMILE has various advantages over existing approaches: (1) it displays the relevant information immediately, not only on mouse-hover. (2) It only shows the SMILE-string, thwarting obfuscation techniques like subdomain-as-domain (e.g., google.com.domain.com). (3) It can work centrally (i.e., on the receiving email server) or locally (i.e., in the email client). As future work, we plan to conduct user studies to get empirical evidence of these advantages and to evaluate SMILE usability.

Acknowledgements. This work was supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF), by KASTEL Security Research Labs, by the Federal Ministry of Education and Research (BMBF) and the Baden-Württemberg Ministry of Science as part of the Excellence Strategy of the German Federal and State Governments, and by the Google Faculty Research Award 2019 for the Link-centric Phishing Warnings for Online Email Clients project.

Appendix

Table 2. Examples in html code for every substitution situation. The *formaction* attribute for the *input* and *button* is optional. If no *formaction* attribute is given, the *action* attribute of the *form* is used.

	Without SMILE	*With SMILE*
Anchor -Image	`<a href="https://example.org/"><img ↪ src="./example.png" /></a>`	`Image Link: [ <a ↪ href="https://example.org/">example.org</a> ↪ ]<br><img src="./example.png"/>`
Anchor -URL-Like	`<a href="https://www.example.org/path">https://sub.exa ↪ mple.org/path</a>`	`[ <a href="https://www.example.org/path">example.org</ ↪ a> ↪ ]`
Anchor -Misc	`<a class="button" href="https://example.org/">Start ↪ Now</a>`	`Start Now [ <a class="button" ↪ href="https://example.org/">example.org</a> ]`
Form -Image	`<form action="https://sub.example.com/path" ↪ method="POST"> <button type="submit"><img ↪ src="./example.png"/></button></form>`	`<form action="https://sub.example.com/path" ↪ method="POST">Image Link: [ <button type="submit">example.org</button> ]<br><img ↪ src="./example.png" /></form>`
Form -URL-Like	`<form action="https://sub.example.com/path" ↪ method="POST"> <button ↪ type="submit">https://sub.example.org/path</button> <input value="https://example.org/" ↪ type="submit"/></form>`	`<form action="https://sub.example.com/path" ↪ method="POST"> [ <button type="submit">example.org</button> ] [ <input value="example.org" type="submit"/> ]</form>`
Form -Misc	`<form action="https://sub.example.com/path" ↪ method="POST"> <button type="submit">Submit Now</button> <input value="Submit Now" type="submit"/></form>`	`<form action="https://sub.example.com/path" ↪ method="POST"> Submit Now [ <button ↪ type="submit">example.org</button> ] Submit Now [ <input value="example.org" ↪ type="submit"/> ]</form>`
Formaction -Image	`<form action="https://sub.example.com/path" ↪ id="form1"></form> <button type="submit" ↪ formaction="https://example.com/path" form="form1"><img src="./example.png" ↪ /></button>`	`<form action="https://sub.example.com/path" ↪ id="form1"></form> Image Link: [ <button type="submit" ↪ formaction="https://example.com/path" form="form1">example.org</button> ]<br><img ↪ src="./example.png" />`
Formaction -URL-Like	`<form action="https://sub.example.com/path" ↪ id="form1"></form> <input type="submit" form="form1" ↪ value="https://page1.example.co.uk/path1" formaction="https://page1.example.co.uk/path1" /> <button type="submit" ↪ formaction="https://page2.example.de/path2" form="form1">https://page2.example.de/</button>`	`<form action="https://sub.example.com/path" ↪ id="form1"></form> <input value="example.co.uk" type="submit" form="form1" formaction="https://page1.example.co.uk/path1" ↪ /> <button formaction="https://page2.example.de/path2" type="submit" form="form1">example.de</button>`
Formaction -Misc	`<form action="https://sub.example.com/path" ↪ id="form1"></form> <input type="submit" form="form1" value="Submit Now" ↪ formaction="https://page1.example.co.uk/path1"/> <button type="submit" form="form1" ↪ formaction="https://page2.example.de/path2" ↪ >Submit Now</button>`	`<form action="https://sub.example.com/path" ↪ id="form1"></form> Submit Now [<input value="example.co.uk" ↪ type="submit" form="form1" ↪ formaction="https://page1.example.co.uk/path1" />] Submit Now [<button type="submit" form="form1" ↪ formaction="https://page2.example.de/path2"> ↪ example.de</button>]`
Area -Image	`<img src="./example.png" usemap="#map"/><map name="map "> <area shape=".." coords=".." ↪ href="https://example1.com/"> <area shape=".." coords=".." ↪ href="https://example2.com/"></map>`	`Area Link: [ <a ↪ href="https://example1.com/">example1.com</a> ]<br> Area Link: [ <a ↪ href="https://example2.com/">example2.com</a> ]<br> <img src="./example.png"/>`



t1) Link [mail.google.com]
t2) https://mail.google.com/mail
t3) https://mail.google.com/mail
t4) https://mail.googl.com/mail
t5) https://mail.google.com/mail
t6) Link [s-o-k.de]
t7) https://s-o-k.de
t8) https://s-o-k.de
t9) https://s-o-.de
t10) {LINK ENTFERNT}
t11) https://s-o-k.de
t12) {LINK ENTFERNT}
t13) https://cdn.cloudflare.steamstatic.com/steamcommunity/public/images/avatars/7a/7a8fbc27b5c2acbc59f903ab122da07208d694e3.jpg
t14) https://cdn.cloudflare.steamstatic.com/steamcommunity/public/images/avatars/7a/7a8fbc27b5c2acbc59f903ab122da07208d694e3.jpg
t15) [url] Link [s-o-k.de] [/url]
t16) https://s-o-k.de [/url]
t17) https://s-o-k.de

t18) https://s-o-k.de
t19) http://s-o-k.de
t20) s-o-k.de
t21) ftp://s-o-k.de
t22) hhttp://s-o-k.de
t23) Link [web.de.com]
t24) https://s-o-k.de
t25) Link [//]
t26) Link [//]
t27) https://s-o-k.de
t28) Link
t29) Link [s-o-k.de]
t30) Link [s-o-k.de]
t31) [redacted] [s-o-k.de]
t32) [redacted]
t33) [redacted]

Fig. 3. Tests of Valve's STEAM forum formatting of links. The substitution is added only for some of them. T31, 32 and 33 use the "spoiler" function, that allows to obscure some of the text, then visible only through mouse-hover.

References

1. Afroz, S., Greenstadt, R.: PhishZoo: detecting phishing websites by looking at them. In: International Conference on Semantic Computing, pp. 368–375 (2011). https://doi.org/10.1109/ICSC.2011.52
2. Al-Janabi, M., de Quincey, E., Andras, P.: Using supervised machine learning algorithms to detect suspicious URLs in online social networks. In: International Conference on Advances in Social Networks Analysis and Mining, pp. 1104–1111 (2017). https://doi.org/10.1145/3110025.3116201
3. Albakry, S., Vaniea, K., Wolters, M.K.: What is this URL's destination? Empirical evaluation of users' URL reading. In: Conference on Human Factors in Computing Systems (CHI), pp. 1–12 (2020). https://doi.org/10.1145/3313831.3376168
4. Ali, M., Nelson, J., Shea, R., Freedman, M.J.: Blockstack: a global naming and storage system secured by blockchains. In: USENIX Annual Technical Conference (USENIX ATC), pp. 181–194 (2016). https://www.usenix.org/conference/atc16/technical-sessions/presentation/ali
5. Althobaiti, K., Vaniea, K., Zheng, S.: Faheem: explaining URLs to people using a Slack bot. In: Symposium on Digital Behaviour Intervention for Cyber Security, pp. 1–8 (2018). https://vaniea.com/papers/aisb2018.pdf
6. APWG: Phishing activity trends report, 4th quarter 2020 (2021). https://docs.apwg.org/reports/apwg_trends_report_q4_2020.pdf
7. Bin, S., Qiaoyan, W., Xiaoying, L.: A DNS based anti-phishing approach. In: 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing, pp. 262–265 (2010). https://doi.org/10.1109/NSWCTC.2010.196
8. Chiew, K.L., Chang, E.H., Sze, S.N., Tiong, W.K.: Utilisation of website logo for phishing detection. Comput. Secur. **54**, 16–26 (2015). https://doi.org/10.1016/j.cose.2015.07.006
9. Englehardt, S., Han, J., Narayanan, A.: I never signed up for this! Privacy implications of email tracking. Proc. Priv. Enhancing Technol. **2018**(1), 109–126 (2018)
10. Gastellier-Prevost, S., Gonzalez Granadillo, G., Laurent, M.: A dual approach to detect pharming attacks at the client-side. In: 4th IFIP International Conference on New Technologies, Mobility and Security, pp. 1–5 (2011). https://doi.org/10.1109/NTMS.2011.5721063

11. Google: Internationalized Domain Names (IDN) in Google Chrome (2021). https://chromium.googlesource.com/chromium/src/+/refs/heads/main/docs/idn.md
12. Hajgude, J., Ragha, L.: Phish mail guard: phishing mail detection technique by using textual and URL analysis. In: 2012 World Congress on Information and Communication Technologies, pp. 297–302 (2012). https://doi.org/10.1109/WICT.2012.6409092
13. Hill, R.: Public suffix list (2020). https://github.com/gorhill/publicsuffixlist.js
14. Lastdrager, E., Gallardo, I.C., Hartel, P., Junger, M.: How effective is anti-phishing training for children? In: Thirteenth Symposium on Usable Privacy and Security (SOUPS), pp. 229–239 (2017). https://www.usenix.org/conference/soups2017/technical-sessions/presentation/lastdrager
15. Litimus Email Analytics: Email client market share (2021). https://emailclientmarketshare.com/
16. Marchal, S., Armano, G., Grondahl, T., Saari, K., Singh, N., Asokan, N.: Off-the-hook: an efficient and usable client-side phishing prevention application. IEEE Trans. Comput. **66**, 1717–1733 (2017). https://doi.org/10.1109/TC.2017.2703808
17. Mozilla: Public Suffix List (2020). https://publicsuffix.org/
18. Petelka, J., Zou, Y., Schaub, F.: Put your warning where your link is: improving and evaluating email phishing warnings. In: Conference on Human Factors in Computing Systems (CHI), pp. 1–15 (2019). https://doi.org/10.1145/3290605.3300748
19. Reinheimer, B., et al.: An investigation of phishing awareness and education over time: when and how to best remind users. In: Sixteenth Symposium on Usable Privacy and Security (SOUPS), pp. 259–284 (2020). https://www.usenix.org/conference/soups2020/presentation/reinheimer
20. SecUSo: TORPEDO-Webextension (2021). https://github.com/SecUSo/TORPEDO-Webextension
21. Valve: Text formatting (2021). https://steamcommunity.com/comment/Recommendation/formattinghelp
22. Volkamer, M., Renaud, K., Reinheimer, B., Kunz, A.: User experiences of TORPEDO: TOoltip-poweRed Phishing Email DetectiOn. Comput. Secur. **71**, 100–113 (2017). https://doi.org/10.1016/j.cose.2017.02.004
23. Wash, R.: How experts detect phishing scam emails. In: Proceedings of the ACM on Human-Computer Interaction, pp. 1–28 (2020). https://doi.org/10.1145/3415231
24. Wen, Z.A., Lin, Z., Chen, R., Andersen, E.: What. Hack: engaging anti-phishing training through a role-playing phishing simulation game. In: Conference on Human Factors in Computing Systems (CHI), pp. 1–12 (2019). https://doi.org/10.1145/3290605.3300338
25. Zhou, Y., Zhang, Y., Xiao, J., Wang, Y., Lin, W.: Visual similarity based anti-phishing with the combination of local and global features. In: 13th International Conference on Trust, Security and Privacy in Computing and Communications, pp. 189–196 (2014). https://doi.org/10.1109/TrustCom.2014.28