Open Redirects, Proxies, and LinkShim: The Issue Facebook Doesn't Want You To Think About

Matt McMahon
Computer Science
Department George
Mason University
Fairfax, Va. USA
https://orcid.org/0000-0002-8303-5481

***Abstract*—URL Wrapping or 'link shimming' is a process where websites redirect traffic through an intermediary endpoint. Engineers employ this technique for navigation security, privacy, and analytical purposes. This paper will outline the general purposes of link shimming, then will go into detail of how Facebook implements their own LinkShim system and vulnerabilities of the past. Finally a shortcoming of the Facebook LinkShim system will be explored and possible mitigation strategies will be offered.**

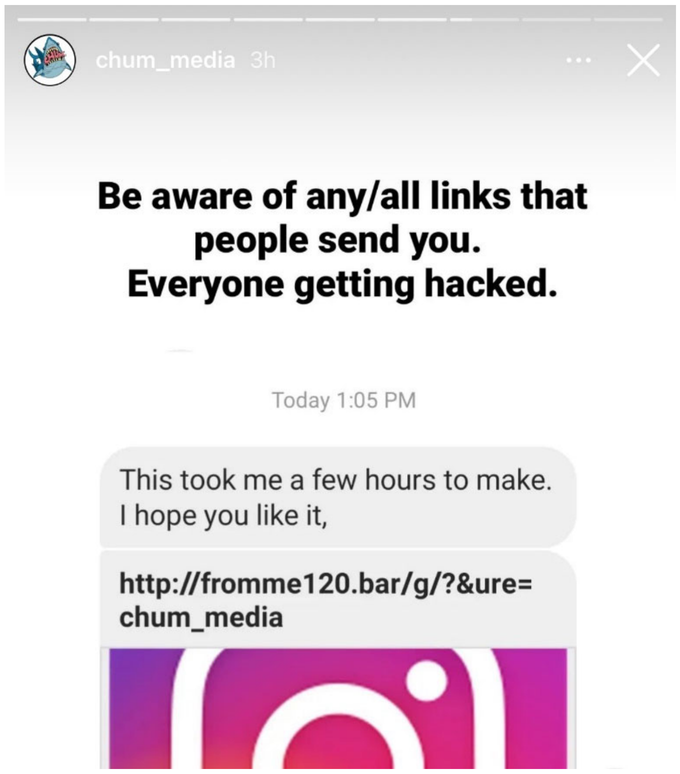
Keywords—Cyber Security, SecDevOps, LinkShim, CWE 601: Open Redirects, URL Wrapping

I. Introduction

Ever since the first BBS servers the internet has brought people together and created social networks. During the turn of the millennium advances in graphics and networking enabled the rise of more advanced social media applications. Facebook, and later Instagram, came to dominate the US social media market. Now in 2021 both platforms are almost ubiquitous in modern life. Consumers not only connect with friends, but stay informed of news, watch entertaining content and make purchases through the various platforms.

Due to the concentration of potential assets within a social media account for an attacker to exploit, social media accounts are frequently targeted by attackers. Through social media accounts attackers can login to other linked accounts that may have payment information. Some have payment information saved directly to their social media account for in-application purchases.

Frequently once an account is compromised, the attacker will impersonate the victim and request cash (through apps like Cashapp) to be sent because of contrived emergencies purported by the attacker. “My car broke down I need money for groceries” is a common request. Attackers also pretend to be the victim’s friend and promote various schemes such as requesting the victims send them money and they promise to return substantially more money, “send me $200 I’ll send you $2000, it’s my your old Pal Frank Kindergarten.”


chum_media 3h

Be aware of any/all links that people send you.
Everyone getting hacked.

Today 1:05 PM

This took me a few hours to make. I hope you like it,

http://fromme120.bar/g/?&ure=chum_media

An Instagram user warns his friends of mailicous links in October 2021 [McMahon2021]

Facebook and it’s child companies have taken a concerted effort in identifying and preventing cyber attacks. One area of focus has been preventing XSS, XSRF, open redirect and other attacks through the use of malicious links. In 2019 there was an increase of cyber attacks through social media. During the pandemic this trend continued as people were forced to make ends meet through other means, and more turned to cyber crime.

Facebook has developed their own in-house system for filtering links called LinkShim. While linkShim has enhanced its filtering capabilities since its development, there is still room for improvement. This paper will outline linkShim’s development, vulnerabilities, and possible mitigation strategies.

II. Context

A. *Development of LinkShim*

Facebook has it's own proprietary version of handling redirects with it's own implementation of URL wrapping; the project at Facebook is called linkShim. In 2008 LinkShim was first created [Elsobky2014].

There are three general reasons why linkShim was developed at Facebook [Facebook2012]. First was the need to anonymize referrer links. Before Facebook implemented their own linkShim advertisers and trackers were able to determine exactly which user clicked on a link. This is because Facebook referrer header contains the UserID of the user clicking the link. By channeling the user's traffic through an intermediary endpoint, the third party cannot determine the user who clicked on the link, only that the user is coming from Facebook.

The second general reason for creating the LinkShim system is for analytical purposes. Every click is tracked and the data is attached to both the user's advertising profile and that of the link. This way Facebook can track which links are being clicked, by who, and how frequently [Facebook2020].

The third reason for implementing the LinkShim URL wrapping system at Facebook is for security purposes. All links in Facebook, or it's affiliated companies like Instagram or WhatsApp, pass through the LinkShim filter.

Facebook LinkShim doesn't filter links dynamically or in real time. Instead the method compares the link to a database of existing known malicious links. The reason why LinkShim can't scan in real time is because by comparing links against a database, malicious URLs can be identified, added to the database, and links that have not been opened can be retroactively blocked.

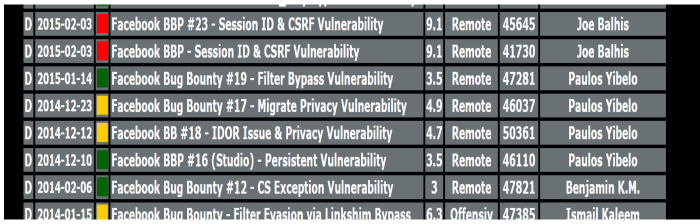
Facebook can retroactively block links that have been sent to users by having a database and not dynamic filtering. This is especially helpful for email messages or other services that have been sent but not necessarily opened immediately.

In 2020 Facebook improved LinkShim by actively scanning links before they are clicked by the user, not just scanning their header or URL. Per the Facebook documentation "we now check every link on the page before it's sent to the browser." This is computationally intensive and will be touched on later in this paper after past vulnerabilities are outlined.

III. Vulnerabilities

Every year Facebook publishes the names of a few hundred individuals who have collected the Facebook bug bounty[Facebook2021]. Ever since LinkShim was created people have been finding exploits.

One of the most successful researchers has been Paulos Yibelos, who found three during a six month stretch in 2015.



D	2015-02-03	🟥	Facebook BBP #23 - Session ID & CSRF Vulnerability	9.1	Remote	45645	Joe Balhis
D	2015-02-03	🟥	Facebook BBP - Session ID & CSRF Vulnerability	9.1	Remote	41730	Joe Balhis
D	2015-01-14	🟩	Facebook Bug Bounty #19 - Filter Bypass Vulnerability	3.5	Remote	47281	Paulos Yibelo
D	2014-12-23	🟨	Facebook Bug Bounty #17 - Migrate Privacy Vulnerability	4.9	Remote	46037	Paulos Yibelo
D	2014-12-12	🟨	Facebook BB #18 - IDOR Issue & Privacy Vulnerability	4.7	Remote	50361	Paulos Yibelo
D	2014-12-10	🟩	Facebook BBP #16 (Studio) - Persistent Vulnerability	3.5	Remote	46110	Paulos Yibelo
D	2014-02-06	🟩	Facebook Bug Bounty #12 - CS Exception Vulnerability	3	Remote	47821	Benjamin K.M.
D	2014-01-15	🟨	Facebook Bug Bounty - Filter Evasion via Linkshim Bypass	6.3	Offensiv	47385	Ismail Kaleem

Vulnerability Lab Search Engine[Vulnerability2021]

A. *Yibelos Exploits*

The first time Yibelos bypassed LinkShim was in 2014. He did so by altering the URL from a Facebook mobile feed story. The URL had a 'continue' parameter that could be altered by the client and was not checked at all by Facebook.

```
https://m.facebook.com/feed_menu/?story_fbid=808015282566492&id=100
000740832129&confirm=h&continue=../http://evilzone.org&perm&no_fw=1
&_rdr
```

Fig. 1: The URL with Payload highlighted in yellow

The second Bug Yibelos found is not relevant to securing LinkShim and how to prevent future attacks because the method is not possible. Now all of Facebook cookies are encrypted and tampering with them is much more difficult than in 2014. In 2014 Yibelos used a restricted account (60 day) and changed ‘name’ and ‘service’ updates by using a session tamper. While this is tangentially relevant it does not exploit open redirects but instead is XSRF attack where client data can be changed.

For the third exploit, Yibelos leveraged the fact that Facebook had been sanitizing LinkShim URLs by comparing them against a database. To quote Yibelos “the payload needed to be a valid & safe-URI, but with our payload.” Therefore the first exploit was done by creating a JS alert and payload all in one line that looks like a URI[Yibelos2016].

```
javascript://google.com/?x=%0Aalert`Hi!`;document.body.firstElement
Child.children[0].firstElementChild.firstElementChild.nextElementSi
bling.nextElementSibling.nextElementSibling.children[2].children[1]
.firstElementChild[4].click
```

Fig 1. The JS alert is in grey while the DOM payload is highlighted in yellow

Paulos Yibelos’s contributions are important to note from a methodological standpoint. He essentially focused on DOM manipulation to circumvent LinkShim. The next researcher was able to bypass LinkShim in another way.

B. *Anees Khan*

Anees Khan is a researcher who has also collected the Facebook bug bounty by bypassing LinkShim in a new way than before. After Anees Khan’s report, the way of bypassing LinkShim outlined is the most common way LinkShim is still bypassed.

Anees Khan showed that by using a link shortener attackers could bypass the LinkShim protections[Khan2017]. A link shortener changes the URL of a given link into a new URL. Therefore since LinkShim relies on a database of blacklisted URLs, a link shortener can provide an endless number of cheap and easy to create ‘backdoors.’ This is a really pernicious problem because there is not an easy fix. The version of LinkShim that Khan was able to exploit is essentially the same version of LinkShim that is operating today, 4 years later. As mentioned earlier, as of 2021 LinkShim now supposedly visits and scans links before they are clicked. This is computationally expensive but possibly the only way to prevent such an attack.

C. *2017-2021 LinkShim BugBounties*

In 2017 Elsallamy found that LinkShim could be bypassed by substituting ‘.’ Instead of ‘/’ in the URI scheme. This bug was recorded but no bounty was paid for it[Elsallamy2017].

In 2018 the security group Servicenger found an unsecured endpoint using fb://webview[Servicenger2018].

```
https://mbasic.facebook.com/a/feed_menu.php?story_fbid=xx&id=10000xx&m
enu_id=u_0_0&continue=fb%3A%2F%2Fwebview%2F%3Furl%3Dhttps%3A%2F%2Fevil
zone.org&action=us&gfid=xx
```

In 2020 Neilmark Ochea found the push notifications endpoint was not secured. The URL could have a payload set to the ‘ref’ value [Ochea2020].

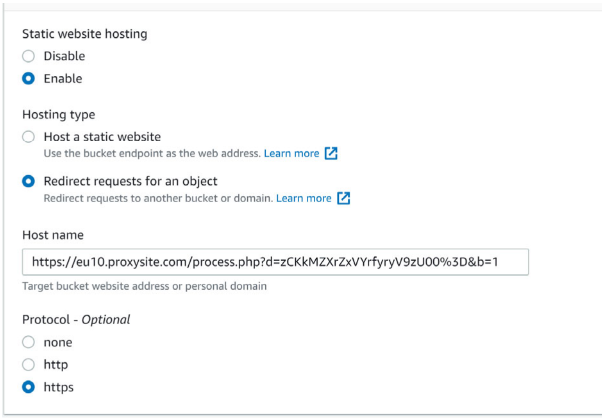
```
`https://facebook.com/notifications
/client/push/enabled/?ref=
```

Since the push notification endpoint was not properly sanitizing URLs it would be possible for an attacker to pass in a malicious URL without being detected.

D. *Personal Exploration*

The Bug Bounty Report that got me thinking the most was Khan2017 because it was the most simple. Just by changing the URL an attacker can pass in a malicious URL to a Facebook user and bypass the LinkShim filter. I wondered how these attacks could be prevented since this is potentially such a large vector for attack.

Another way to prevent these attacks is to prevent all 3xx types of redirects. 302 redirects are temporary redirects while 301 redirects are permanent. I created an s3 bucket on AWS and provisioned it to act as a static website. Then I set the bucket to function as a permanent redirect to a URL, a 301 redirect.



Static website hosting
Disable [✓ Enable]

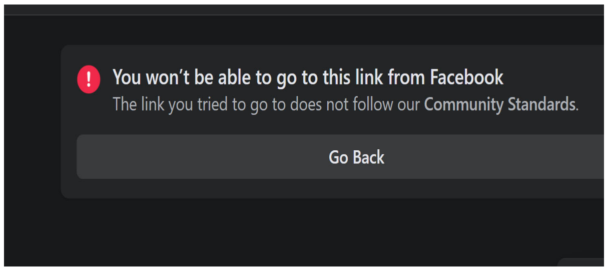
Hosting type
Host a static website
Use the bucket endpoint as the web address. Learn more
[✓ Redirect requests for an object]
Redirect requests to another bucket or domain. Learn more

Host name
https://eu10.proxysite.com/process.php?d=zCKkMZXrZxVYrfyryV9zU00%3D&b=1
Target bucket website address or personal domain

Protocol - *Optional*
none
http
[✓ https]

S3 bucket with proxy redirect

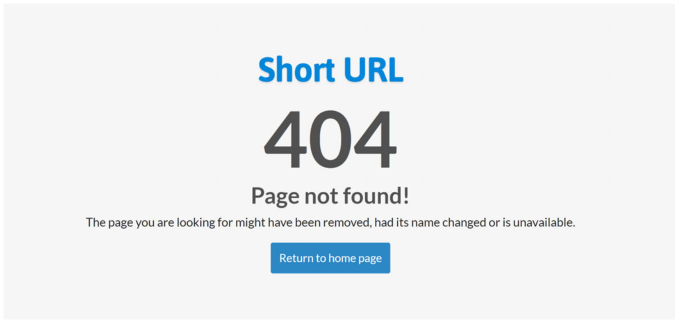
The s3 bucket with 301 redirect returned a the basic linkShim error. This showed me that LinkShim was preventing both 301 and 302 redirects.



You won't be able to go to this link from Facebook
The link you tried to go to does not follow our Community Standards.

Classic LinkShim Error Message

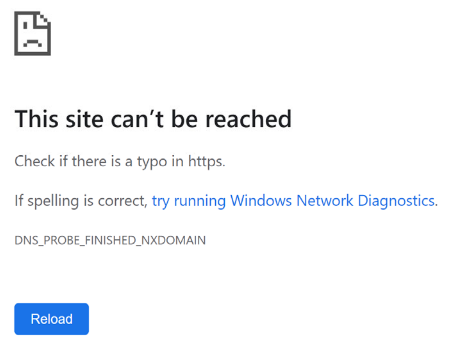
I wanted to see if I could reproduce Khan2017 by using a URL shortener. To my surprise most mainstream URL shorteners did not work with Facebook (Bit.ly/ Google/ Twitter).



Short URL
404
Page not found!
The page you are looking for might have been removed, had its name changed or is unavailable.

Internal Error Message when Using Mainstream Shorteners

After trying mainstream link shorteners I started to try smaller ones. Trying other shorteners and proxies would sometimes return DNS_PROBE_FINISHED_NXDOMAIN error.



This site can't be reached
Check if there is a typo in https.
If spelling is correct, try running Windows Network Diagnostics.
DNS_PROBE_FINISHED_NXDOMAIN

The reason why the NXDOMAIN error is coming is because the two IP address from the link are not matching up. The internal 404 errors from the link shortening sites is from something inside those sites that is preventing the routing. A regular proxy may return a the header information from the proxy site, and then LinkShim is picking up that info.

1) *In Browser Proxy*

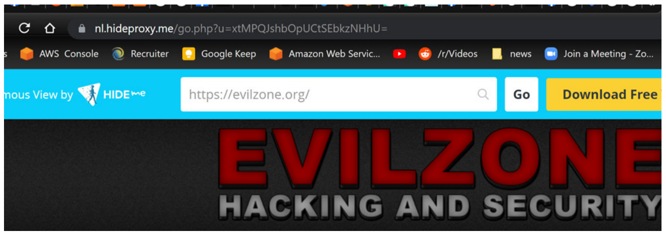
An ‘in-browser’ proxy is a type of proxy that does not record the HTTP header of the destination URL to the client. Headless proxies are proxies that do not load the GUI for a webpage. Headless proxies are often used for web-scraping because it’s harder for a server to detect a bot is scraping the site because the security headers can be spoofed or are just missing. A CSP header or Content Security Header is the most particular for telling the browser (and server) which resources to load (or pass through). A regular HTTP-header may have a ‘Referrer Policy’ or there may be a standalone ‘Referrer Header’.

By Using a headless browser we are decreasing the surface area for LinkShim to scan the link and determine if it’s safe. Per Khan’s contribution, we can reuse the victim’s hash value because Facebook reuses hashes for users.

https://l.facebook.com/l.php?u=https%3A%2F%2Fnl.hideproxy.me/go.php?u=xtMPQJshbOpUCtSEbkzN
HhU%3D&b=29%3Flr86-1%26fbclid%3DIwAR1CtMz5K0CNu0VHya8CPVMO7kz1Y2c5G-i5-vY_OTJxoydu-
VZU_43Mk2Y&h=AT3BWDOyLdfrclIteQJ3IZHqnBgWTT-
lb4W2ieLRZUKl0lVC_DJ3zmNXuu6XPbNz9YnFRU6Ds2_AzgzFZwbB6QT01mYFqCZVcADtEaR5l010WR_L2G
AzSVg7CDgtanoM2G4&__tn__=*I&c[0]=AT1BqcEaY_BEzI8T-OG-yupZ_6rm88yyQ8ny6G7YKa4wN2oM-
n39MnCz0STKC3Dzr4nk_U4JUyTP3qvX66SxAAVyqf7TZF_YnQdSKe4GJMZnk9iAJLXy72DvivzgNCoJZgJ61gC
AWAoL6v_Z3mBdaGvdSjSitM0rhWw6lNb8XZXhjqI9oYtOl-bqv7jTPukS5ODRnRiSErhOOMTC654YyVvx7w

My Endpoint, Payload, and Hash

Following the above URL will route us *through* Facebook LinkShim and to a malicious site.


nl.hideproxy.me/go.php?u=xtMPQJshbOpUCtSEbkzNHhU=

AWS Console · Recruiter · Google Keep · Amazon Web Servic... · /r/Videos · news · Join a Meeting - Zo...

https://evilzone.org/ Go Download Free

EvilZone.org is recommend to test Facebook because it’s listed as a known malicious URL. If you are able to travel to evilzone.org then you were able to pass through LinkShim.

IV. Mitigating Techniques

Facebook considers all open redirects using LinkShim to be sanitized. According to the Facebook Whitehat Education documentation “If LinkShim is used, we do not consider an issue a valid open redirect” [Facebook Whitehat2021]. There is an entire section in the false positive section of the bug bounty explaining types of false positives for open redirects bypassing LinkShim. Using a short URL is considered a false positive.

This makes total sense because Khan already collected that bounty in 2017. Also logically an attacker could ‘layer’ shortened links to achieve obfuscation. Even if Facebook searched one ‘layer’ deep the attacker could add another ‘layer’ and always be out of reach. In this same vein URLs not being normalized is also not accepted as evidence of bypassing LinkShim. Also a cracker could link to an infected website and then launch the open redirect attack from the infected site.

Most particularly Facebook explicitly states that methods that obfuscate or hide the IP address of malicious sites are not considered valid exploits.

Even though using an IP address obfuscation method like a proxy are not suitable for collecting a bounty from Facebook, there are still valid security concerns that could be addressed.

A. Checking HTTP Headers More Thoroughly

One mitigation technique Facebook could employ would be to more thoroughly vet HTTP headers. If a request is missing a referrer header or CSP header that should be a red flag. Second if the referrer/ CSP header is from a list of known proxies or link shortening sites than the request can be denied. It appears as though Facebook and the major link shortening companies have come to an internal agreement (bit.ly links return internal 404 errors.) Perhaps the major in-browser proxy companies can come to a similar agreement where their headers are known and flagged. Even though this will not stop all attackers it will eliminate a significant portion of attackers who have limited technical skills. Like Facebook said, if an attacker wanted to link to an infected site or layer shortening methods there is little to no recourse, but scanning Headers could decrease the total number of attacks.

B. *GeoFencing*

Many malicious open redirect requests come from phishing campaigns [Li2021]. Many phishing campaigns involve geographically separated attackers and victims. By geofencing links to certain clusters it may be possible to 'contain' the spread of malicious links.

One possible implementation of geofencing LinkShim could be allowing links as they are currently, but only for a specific geographic location. Then once the link is clicked a waiting period, or 'quarantine' begins. If after a set period of time the link appears benign, then it is able to be shared more widely over the network.

C. *One-Time Hashes*

The LinkShim system relies on two parts, the URL (u=) and the hash value (h=). If the hash values were not reused then these attacks would become much more difficult to perform.

One time hash values would be more difficult to stop because the act of revealing the hash would render it useless. Currently a user shares a link, the other user will be warned that the link is to an external site because the sender and receiver's hash value do not match. To really run a secret open redirect on a Facebook user you need to first gain their hash value. This could be gained from links that they have shared previously. If the hash values were one time only then previously shared links could not be used to create payloads that would appear to the user as if they are coming from themselves.

V. THE FUTURE OF LINKSHIM

Since it's development in 2008 LinkShim is a powerful tool of the modern web. LinkShim protects users' privacy while also providing valuable data to website operators such as Facebook. Most importantly LinkShim provides security against XSS, XSRF, open redirect and other attacks involving malicious links.

LinkShim does a good job at preventing the spread of malicious links but it could improve. Currently LinkShim considers any type of URL obfuscation method to bypass LinkShim a false positive. This appears to be negative thinking because though there are not ways to entirely prevent the transmission of malicious links, it could be possible to reduce the number of links through the implementation of certain methods. In particular if more thorough header scanning was implemented, then the use of proxies that obfuscate or remove header information could be diminished and the overall security of Facebook and it's corresponding sites would be improved.

ACKNOWLEDGMENT

I would like the thank Dr. Wheeler for giving me the opportunity to write this paper. I would also like to thank the Interdisciplinary Center for Economic Science for allowing me to manage their social media and become more aware of social media vulnerabilities.

REFERENCES

[1]
[Jaran202] S. Jaran, "Open-redirect on Facebook (Bypass LinkShim)," *Medium*, Feb. 16, 2020.

https://dwisiswant0.medium.com/open-redirect-on-facebook-bypass-linkshim-4050f680d45c (accessed Oct. 22, 2021).

[2]
[Yibelos2016] P. Yibelos, "Instagram Stored OAuth XSS," *Paulos Yibelo - Blog*, 2016. https://www.paulosyibelo.com/2016/11/instagram-stored-oauth-xss.html (accessed Oct. 22, 2021).

[3]
[AWS2021] AWS, "Configuring a webpage redirect - Amazon Simple Storage Service." https://docs.aws.amazon.com/AmazonS3/latest/userguide/how-to-page-redirect.html (accessed Oct. 22, 2021).

[4]
[Yibelo2015] P. Yibelo, "Facebook: Another LinkShim Bypass," *Paulos Yibelo - Blog*, 2015. https://www.paulosyibelo.com/2015/03/facebook-another-linkshim-bypass.html (accessed Oct. 24, 2021).

[5]
[Facebook2012] Facebook, "Link Shim - Protecting the People who Use Facebook from Malicious URLs," *Link Shim - Protecting the People who Use Facebook from Malicious URLs*, 2012. https://www.facebook.com/notes/10157814493891886/ (accessed Oct. 24, 2021).

[6]
[FacebookEngineering2020] Facebook Engineering, "A faster, better link shim," 2020. https://m.facebook.com/nt/screen/?params=%7B%22note_id%22%3A10158791573417200%7D&path=%2Fnotes%2Fnote%2F&_rdr (accessed Oct. 24, 2021).

[7]
[Scheme2011] "Scheme/Host/Port: Referrer (sic)," 2011. https://www.schemehostport.com/2011/11/referer-sic.html?fbclid=IwAR0-bh8ZOXUKei21ggprBcWlYfUfxB1bvHi88aIqxaj8bhNQGZFpTSV2wmE (accessed Oct. 24, 2021).

[8]
[FacebookWhitehate2021] "Thanks! | Facebook," *Bug Bounty Program*, 2021. https://www.facebook.com/whitehat/thanks/ (accessed Oct. 24, 2021).

[9]
[Elsobky2014] A. Elsobky, "On Evading Facebook's LinkShim Mechanism," 2014. http://0xsobky.github.io/evading-facebook-linkshim/ (accessed Oct. 24, 2021).

[10]
[Vulnerability2021] "VULNERABILITY LAB SEARCH ENGINE - SECURITY VULNERABILITY RESEARCH LABORATORY - VULNERABILITY DATABASE." https://www.vulnerability-lab.com/search.php?page=1&search=facebook (accessed Oct. 24, 2021).

[11]
[Vulnerability2015] "Facebook Bug Bounty #19 - Filter Bypass Vulnerability," *Facebook Bug Bounty #19 - Filter Bypass Vulnerability*, 2015. https://www.vulnerability-lab.com/get_content.php?id=1381 (accessed Oct. 24, 2021).

[12]
[Yibelos2014] Yibelos, "Facebook Bug Bounty #17 - Migrate Privacy Vulnerability," *Facebook Bug Bounty #17 - Migrate Privacy Vulnerability*, 2014. https://www.vulnerability-lab.com/get_content.php?id=1370 (accessed Oct. 24, 2021).

[13]
[Yibelos2014] Yibelos, "Facebook BB #18 - IDOR Issue & Privacy Vulnerability," *Facebook BB #18 - IDOR Issue & Privacy Vulnerability*, 2014. https://www.vulnerability-lab.com/get_content.php?id=1371 (accessed Oct. 24, 2021).

[14]
[Khan2018] A. Khan, "How I bypassed Facebook's LinkShim Protection.," *Medium*, Apr. 02, 2018. https://medium.com/@aneeskhan/how-i-bypassed-facebooks-linkshim-protection-a0ad4c494575 (accessed Oct. 24, 2021).

[15]
[CWE2021] "CWE - CWE-601: URL Redirection to Untrusted Site ('Open Redirect') (4.5)." https://cwe.mitre.org/data/definitions/601.html (accessed Oct. 24, 2021).

[16]
[Elsallamy2021] S. Elsallamy, "Rolling around and Bypassing Facebook's LinkShim protection on iOS," *Seekurity*, 2017. https://seekurity.com/blog/2017/07/26/seif-elsallamy/general/rolling-around-and-bypassing-facebook-linkshim-protection-on-ios (accessed Oct. 24, 2021).

[17]
[Ochea2020] N. Ochea, "Facebook Push Notification LinkShim Bypassed," *Medium*, Dec. 2020. https://infosecwriteups.com/facebook-push-notification-linkshim-bypassed-385fe471516 (accessed Oct. 24, 2021).

[18]
[Servicenger2021] servicenger, "Facebook: LinkShim protection bypass using fb://webview - SERVICENGER," 2018. https://servicenger.com/mobile/facebook-linkshim-protection-bypass-using-fb-webview/ (accessed Oct. 24, 2021).

[19]
[Li2021] F. Li, "Shim Shimmeny: Evaluating the Security and Privacy Contributions of Link Shimming in the Modern Web," 2020, pp. 649–664. Accessed: Oct. 24, 2021. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/li-frank

[20]
[Detectify2021] Detectify, "A guide to HTTP security headers for better web browser security," *Detectify Blog*, Feb. 05, 2019. https://blog.detectify.com/2019/02/05/guide-http-security-headers-for-better-web-browser-security/ (accessed Oct. 24, 2021).

[21]
[Whitehat2021]Facebook Whitehat, "Facebook Whitehat Education," 2021. https://m.facebook.com/whitehat/education/false-positives/ (accessed Oct. 24, 2021).

[22]

[Hawe2021] "Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness | USENIX." https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/akhawe (accessed Oct. 24, 2021).

[23]
[Seebug2021] "List of bug bounty writeups." https://paper.seebug.org/802/ (accessed Oct. 24, 2021).

[24]
[LinkShim2021] "Link Shim - Protecting the People who Use Facebook from Malicious URLs." https://m.facebook.com/nt/screen/?params=%7B%22note_id%22%3A10157814493891886%7D&path=%2Fnotes%2Fnote%2F&_rdr (accessed Oct. 24, 2021).

[25]
[McMahon2021]M. K. McMahon, *linkShim*. 2021. Accessed: Oct. 28, 2021. [Online]. Available: https://github.com/mkMcMahon/linkShim