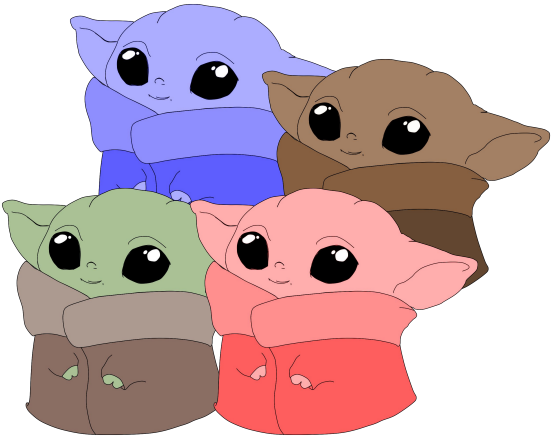
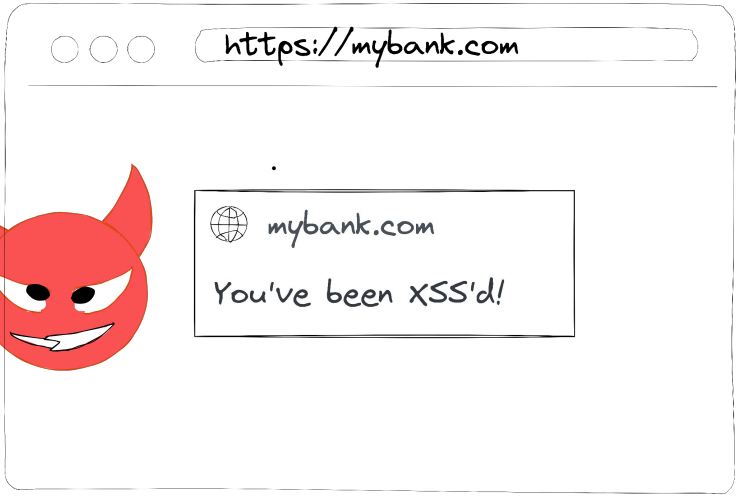
Exclusive

# Lessons on XSS from Yoda

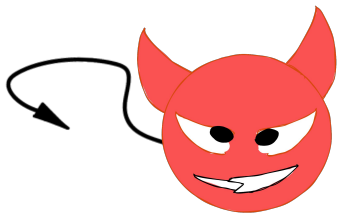





Made with ❤ by SecurityGOAT

# Today, Yoda is going to teach everything you ever wanted to know about XSS!

- What is XSS?
- Is there more to XSS than simple pop-ups?
- The significance of pop-ups
- Root cause of XSS
- Consequences of XSS
- Common XSS variants
- How to fix XSS issues?
- Who's responsible of fixing it?




Please tell us more about XSS.

Since the time we started with our Infosec 101 class, there's been a craze about the XSS bugs.

We want to learn more about it.

I will teach you folks.

Lean in, for a deep-dive into XSS.

Yes, master.

(Everyone's super excited!)

# What is XSS?

Cross-Site Scripting or XSS happens when the enemy has power to execute the code of their choosing, inside a trusted website!

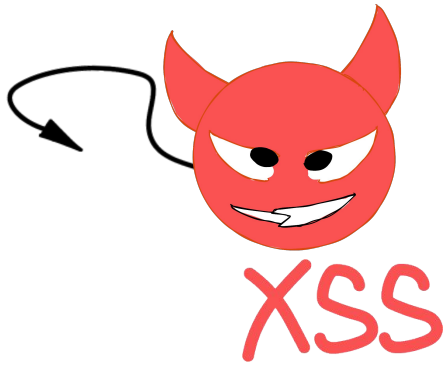
That's scary, master.

(Peter)

Yes, correct Peter, it indeed is.

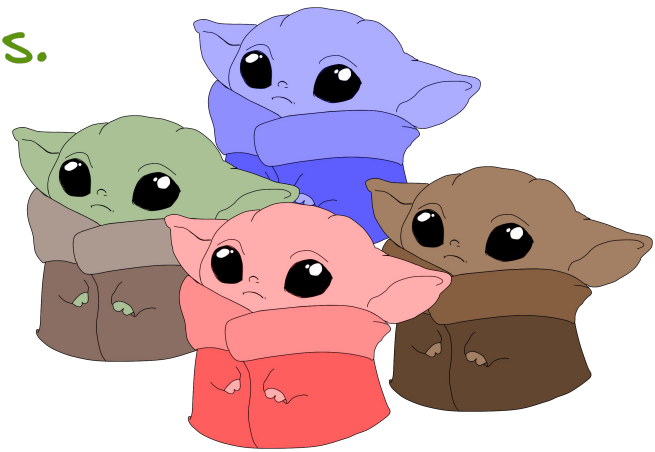
But how is it too scary, master? I just see pop-ups in my browser. Didn't look harmful to me, master.

(Amanda)




Don't confuse XSS for the bening pop-ups, Amanda. There's more to it than meets the eye.
Let me share the tale of an XSS issue that brought havoc to an innocent merchant.

Lean in students.



(All listen very closely)

# An XSS Tale

Once upon a time, in a far land operated a wealthy merchant. He took all the orders digitally and made good profit out of every sale. Everything was going good until one day he received an email from the enemy disguised as his banking assistant.

Not realizing the URL could be malicious, the merchant quickly opened it in the browser and got the login page to his bank. After a few minutes of logging in, he discovered his bank balance dropped to zero!

How is that possible master?

(Mary)

The merchant had logged into a fake bank website. The malicious URL was a trap that looked real and was meant to lure the merchant to steal his credentials. Once the enemy got his credentials, the enemy transferred all the money from the merchant's account.

Woah! That's indeed scary, master.

Yes students and that is why I said XSS is scary, and sometimes costly as well.

We want to learn more, master. Please teach us, please teach us.

Ok students. You all seem pretty excited. Let me teach you all.

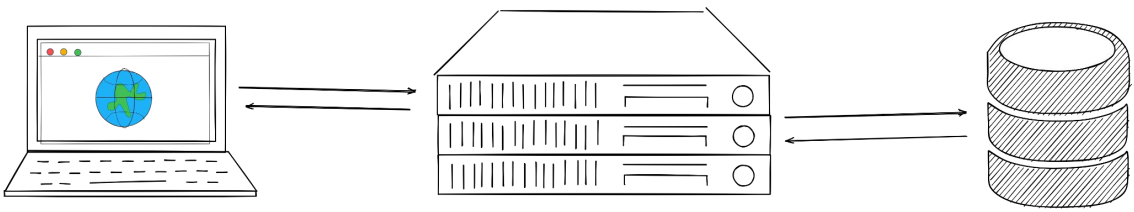
Thank you master.

# XSS Lessons from Yoda

XSS is a class of bugs which happens when the enemy can inject their malicious code in a website you trust.

Modern websites require user input in some place - login, logout, signup, contact us form, etc.

When user passes input to the website, it processes the data and sends back some response and occasionally saves the data in a database.



Yes master. We learnt about this in our class.

Good. Now if an enemy passes evil input to the server and the server sends it back, what will happen?

The input will be rendered in the browser, master.

Correct! And that means?

The evil input is rendered in the browser! Oooh.

And that's how the enemy can execute their code as well.

(Martin)

Very good, Amanda and Martin.

Oh, now we get it!

Pretty smart students, I have.

Master.. Master.. How can the enemy use it, besides the tale you told?

If a web page is vulnerable to XSS, all the enemy has to do is get you to click a malicious link.*

Master, in that case I feel very safe. I don't click links shared by untrusted people.

Good Peter. There's a caveat though.

There is an XSS variant in which the enemy just has to plant the payload, say in a tweet or a comment and anyone who views it is attacked.

Ooops, never thought of that, master!


Now it's getting scarier, master.

Yes, students!

Master, so what were those pop-ups for? A confirmation of the existence of XSS I suppose?

Correct, Amanda!

When I was a budding pentester, I used to send a bunch of alert-based payloads with different messages all over the website and when I used to get a popup while surfing through the website, I knew where the issue was!

That was very wise of you, master :)

That's the easiest way I could come up with, then.

Okay, so now that you all know of XSS, who can tell me what this issue boils down to?

Me, master! Me!

Shall I, master?

Okay, you go first Amanda, and then you can tell Peter.

Master, I think XSS boils down to an implicit trust placed by the browser on the data sent back by the server.

And what do you think, Peter?

Master, I think the root cause is the server-side code that trusts user-supplied input in the first place and sends the data without any sanitization.

Very good, both of you. It's when the user-supplied data is not sanitized and it ends up in your browsers, it brings all kinds of troubles we have learned about so far.

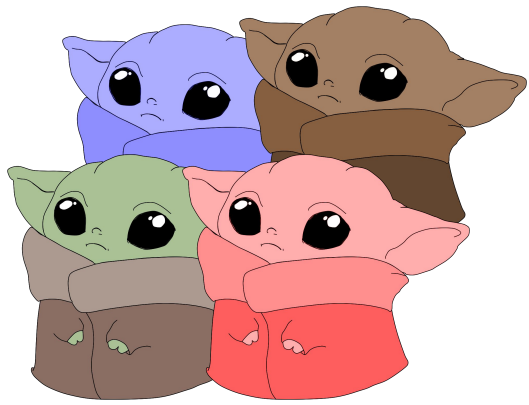
Master, if someone can get code execution using XSS, can they even mine crypto-currency?

(All listen carefully)

It's good you asked, Peter.

Yes, they can! Since the enemy can run Javascript in your browser, he can very well execute code to mine crypto-currencies using your system resources, he can also log your keystrokes, monitor your activity, access your location, scan your internal networks. He can send requests to your bank website, using your user session, he can steal the cookies, he can show you a fake page and steal your login credentials, and more.

Omg! Master that is super scary.



Master, can we fix it?
There must be
some way to
fix it, right?
Finger's crossed

Yes, it can be fixed.

How, master?
Please tell..,
please tell..

(Everyone's excited!)

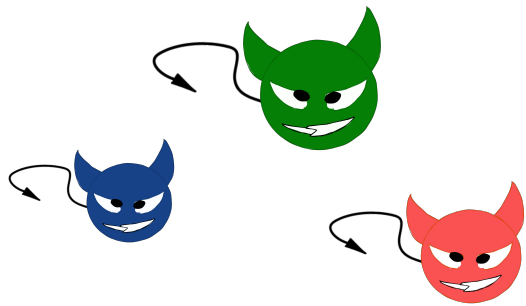
# Common XSS variants

Before I tell you that, let me teach you the common variants of XSS. Then you can better understand the defenses.

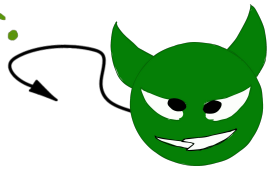
Okay master. Please teach us.

XSS has three common variants:

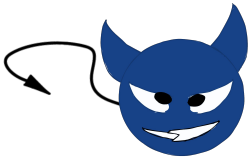
- Reflected XSS
- Persistent XSS
- DOM-based XSS

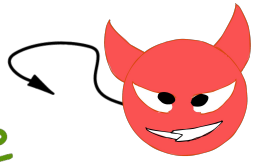


In Reflected XSS, as the name suggests, whatever evil input the enemy passes, it is rendered on the page, as is, by the server.



In Persistent XSS, the evil input provided by the enemy is stored as is, in the database. Whenever a user visits the page where this data is shown, they get attacked.

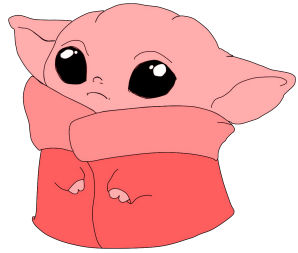




In DOM-based XSS, it's the client-side code that is vulnerable and it passes the unsanitized user-input to some vulnerable function (an XSS "sink") which results in XSS.

Oooh. So it's not just the server-side code but even the client-side code that can do harm.

Yes Martin. End of the day, anything that is rendered in the browser, if it is evil, it will do harm. So it can be both server and the client-side code that you should worry about.



Master, if XSS is scary and happens when the browser renders the evil payload, why can't browser vendors fix it and end this issue at once, for all?

Good question, Mary.

XSS is indeed scary and can have serious consequences, as we discussed, but XSS just means the attacker can run their code in your browser.

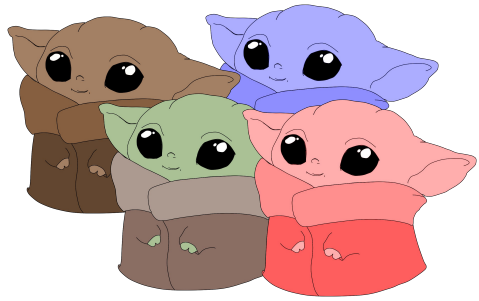
How can a browser tell which code is good and which is not, provided that the actions attacker's do using XSS are seemingly normal - like sending web requests, rendering some content on the page for spoofing purposes, etc. Browsers in such situations are not the best judges and thus, can't help you.

Understood master. So the browser's can't help us with this.

Yes Mary, not entire help but some help they do. Browsers enforce some policies like the Same-Origin Policy (SOP) and Cross Origin Resource Sharing (CORS) which reduced the impact XSS used to have.

Students, we will learn about SOP and CORS in our next class. For today, let's keep it to XSS only. It's already too much for one day.

Yes master.



But master, please tell us how to fix it? And who is incharge of fixing it for us?

(Everyone's curious!)

Very good questions. As I always say, to find answers to anything, go back to the fundamentals of the topic.

How does XSS happen, students?

When the user input is not sanitized by the server and sent back to the client and rendered or when the client-side code passes user input to some unsafe function.

How do you think it can be fixed?

By filtering user input, master.

Very good, Martin. That is a part of the solution. To complete it, one MUST also escape and encode user-input so that it is not rendered on the page.

That sounds like it, master. Now that you mentioned, it seems like an obvious answer.

# Execution context

Since you all are very intelligent, let's go one step further - the rendered content has three places it can go to:

- HTML
- CSS
- Javascript

of these, the most dangerous place is _____?

We know it master: its Javascript!

(All speak together)

Correct! If direct access to JS, the enemy can do more harm. But do not consider other places benign.

To fix XSS when HTML is the sink, what can we do?

I think we can do HTML encoding, master.

Excellent, Martin. What about Javascript sink?

Javascript content escaping would help there, master.

Fantastic, Peter! Who can tell me the remedy for CSS-based sinks?

Escaping user-input, master. Besides that, a whitelist to prevent arbitrary stylesheets loading should help.

Well done, Amanda.
So you all see it now - it's filtering + escaping + encoding which will save you in the end.

(All chanting the magic formula)

Filtering + Escaping + Encoding

Who is incharge of fixing it, master?

What do you think, Mary?

I think it is the website owners, since they control the code. They can make changes to it and add the filtering and input encoding code. That way, the browser would only receive sanitized output, which can be rendered safely.

Excellent, Mary.

Students, there is one more tool that browsers have, to prevent XSS.

What is it master?

(Everyone's curious!)

It is Content-Security Policy (CSP).
It's a lesson for some other day.

Yes master, we all are very curious to learn about it.

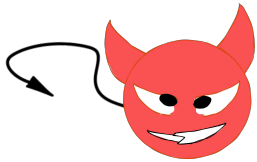
Indeed, students. It is quite an interesting topic.

Thank you master, for this interesting and quite eye-opening lesson on XSS. We now understand all these things:

- What is XSS?
- What can possibly go wrong?
- How it can be exploited?
- The root cause
- Potential fixes and remediations

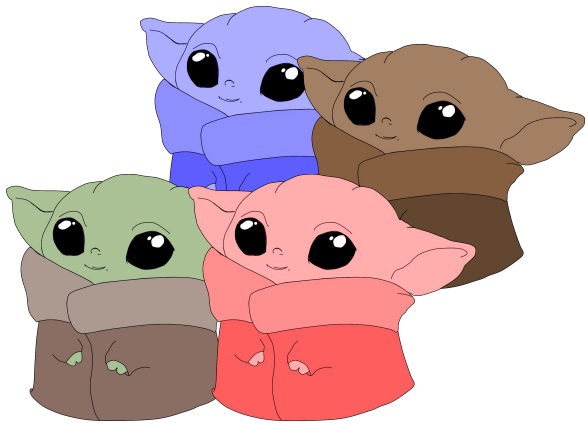
You're all very smart. It's time to take a break. See you next time!

Exclusive



# XSS

A comprehensive story-telling book on XSS, filled with the knowledge and experience of master Yoda.



All your common XSS doubts should be cleared once you finish your learning journey with master Yoda and his brilliant students!

May the force be with you!

Made with ❤ by SecurityGOAT