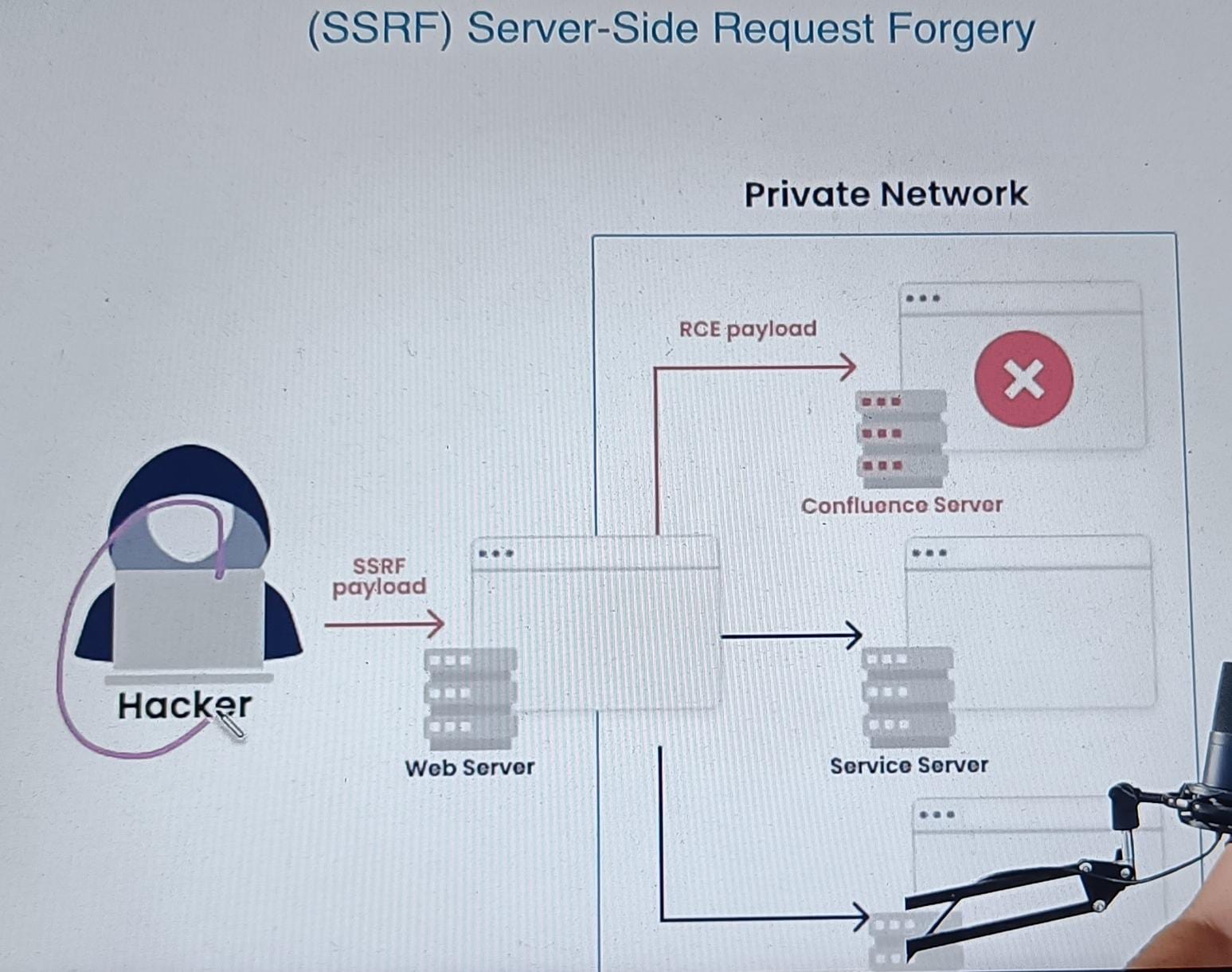
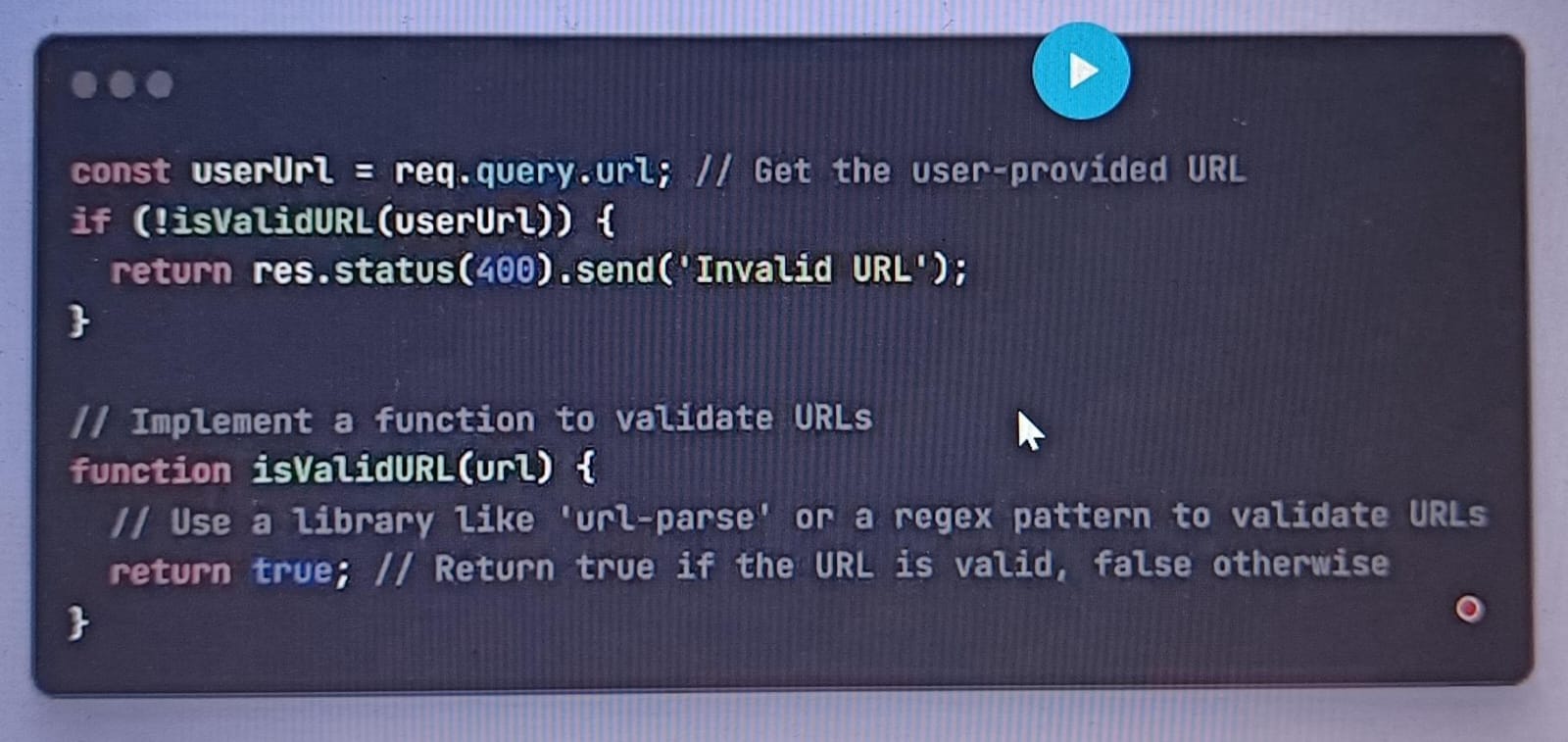
Server-Side Request Forgery –

Consider Example you are able to execute something on the server, which helps to get any data from the internet that data can be internal data or that can be external data that can be helps you to get the data from the database execute some system level operations it can be any damage that can be caused, now most of the cases when we talk about ssrf which is consider example you have a domain and you have access to that domain and you are able to do that because that domain is available to bigger infrastructure of the network system of any company, then most of the time it has access to the internal private network consider example also if someway I am able to execute the code on the server talking to the internal private network having the ip address or the internal access.



So you have a hacker and he can see the website which is publicly available now this website is taken care of lot off, what happen this website because it is part of big network or company so this machine has a private network also which is allow now consider example this hacker somehow able to put payload, this payload can now internally access or execute some resources which are on different ip of this machine which you are able to access or execute basically this become something horrible.

* Unvalidated User Input –



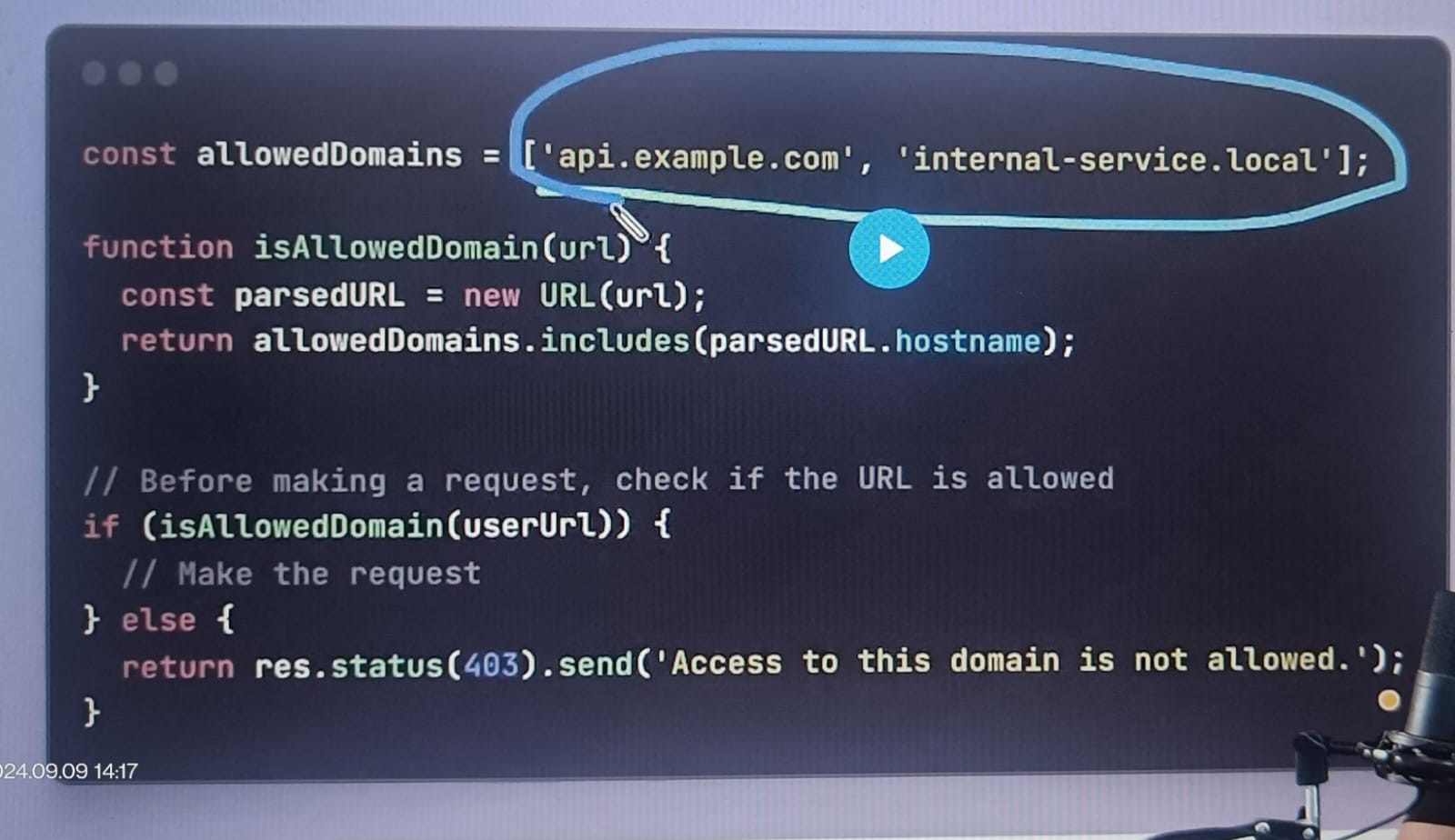
In this case ssji (server side JavaScript injection) this is wider problem where we are not just talking about vulnerabilities which is caused by JavaScript, but it can any dam vulnerabilities in any programming language which is actually caused this ssji.

Now what is not correct in the above image is, in first line we can see that basically directly consuming it you should always validate the have validateURL kind of thing where you actually ensure the url you are getting from the network is something valid so consider example somehow you get the malicious url like this

<https://localhost:3001/user/image?imgUrl=http://169.254.169.254/latest/meta-data/iam/security-credentials/>

so if you notice it has something called localhost user image, and what you are passing please show me a image and you have given a capability where instead of just giving the image id and what not you are saying I will give the url of the image and you just load that image and show me, interesting thing is I did not gave the url of the public network this is the ip address and this is the path basically I got to know for your interna server,

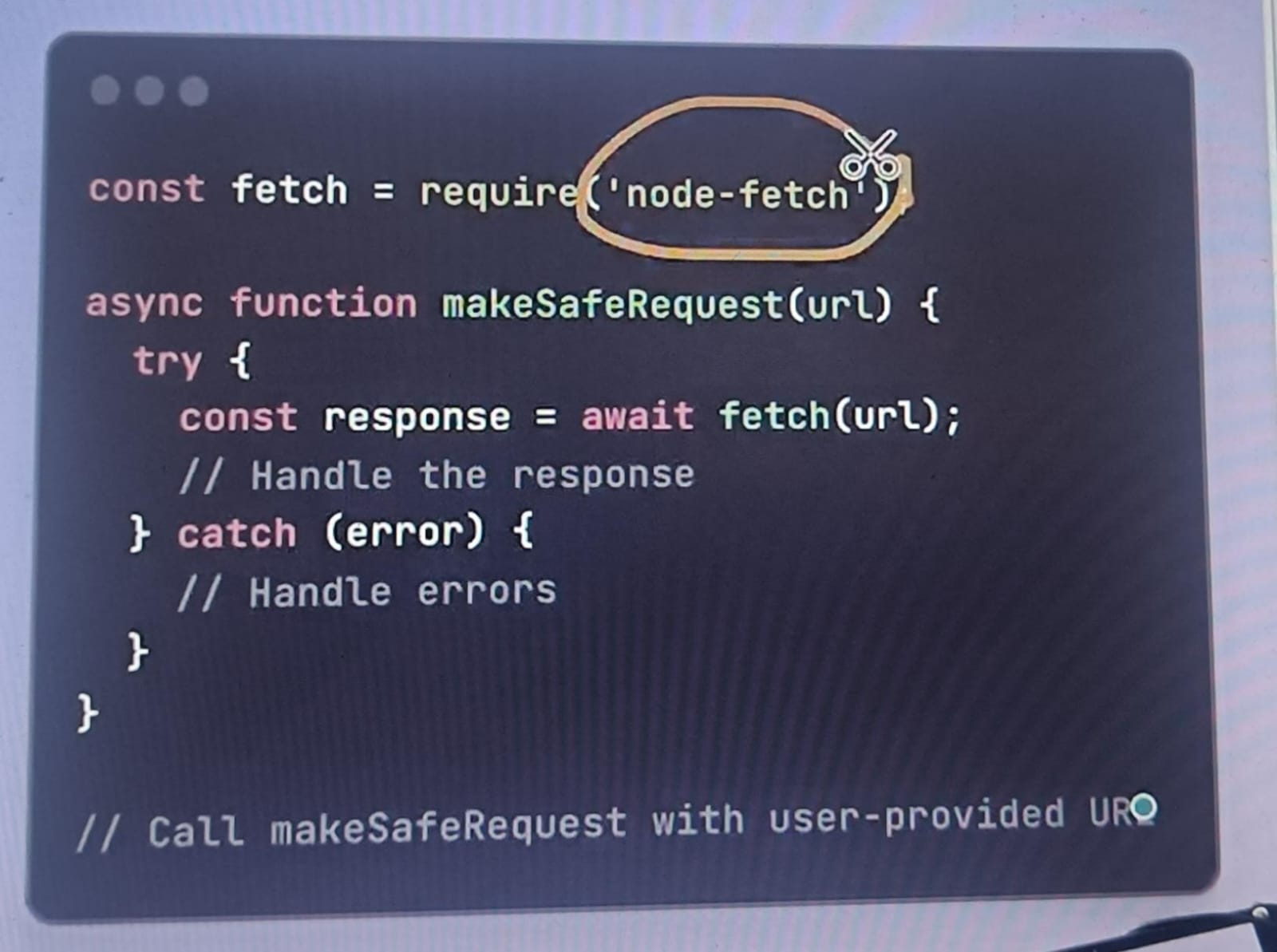
* Lack of whitelisting –



So if you have something called list of allowed apis and before basically generating a url and basically what you create, you create a basic function which says that a url which is you looking for or you wanted to make a req from the server does that belongs to a whitelisted one, if that does not belong to basically before making a req, then please say that access denied, you cannot access resource which you are looking for which you should send basically as an outcome.

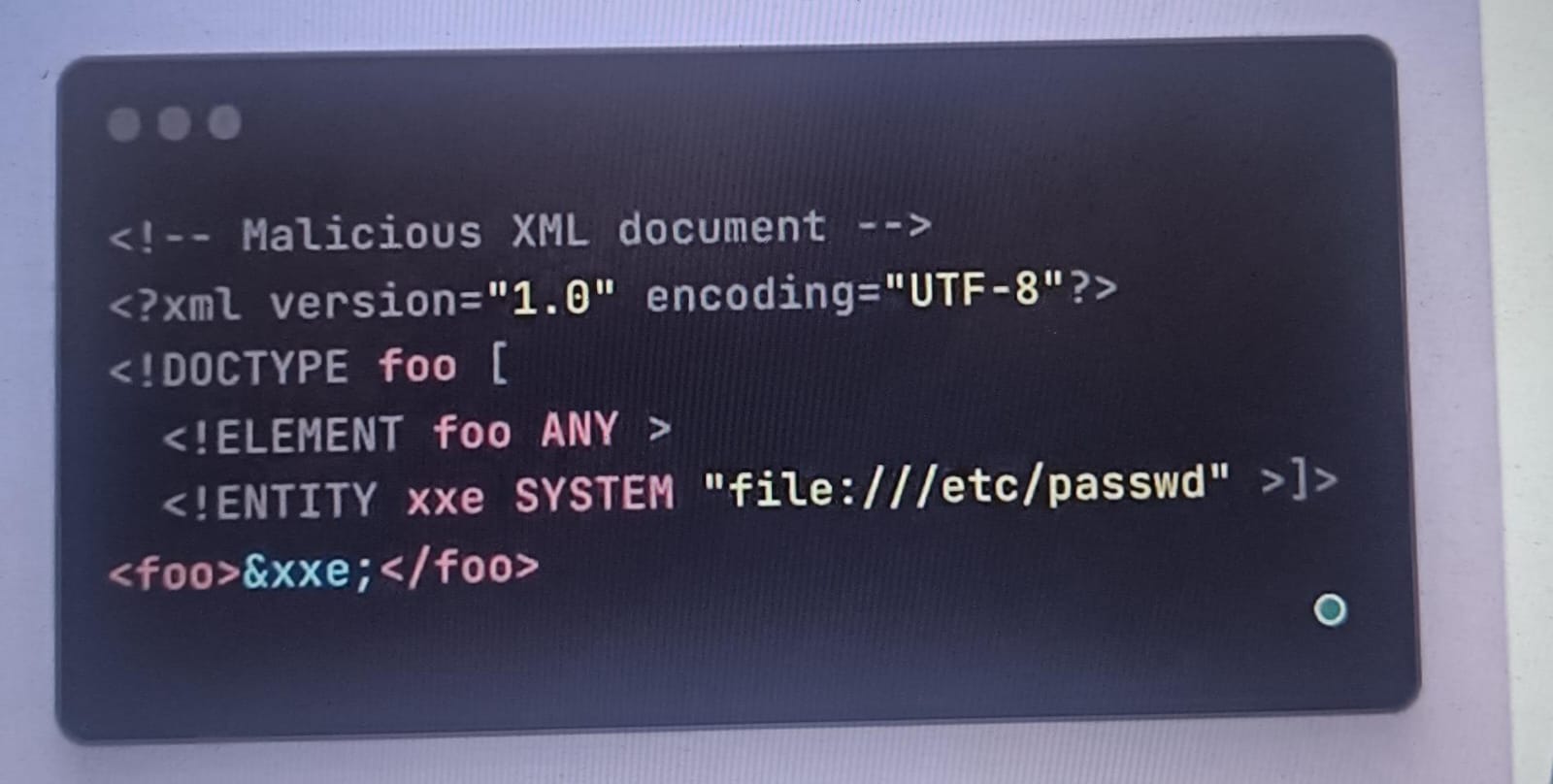
* Insufficient access control –

So did you create policies for what around can be accessed from the file system? What all can be accessed from the operating system, what all can be accessed from the database, what all can be done at a network layer? All these places we have something called as access control.



If you use some libraries, like node fetch kind of thing, axios kind of things what it helps? It helps you to provide first class SSRF protection where it not going to give you 100% protection but it will give you initial set of bare minimum validation or under the hood if you use xml kind of thing or fetch kind of thing that does basically provide any of the security but such libraries are well tested and keep on enhancing and adding lot of solutions as a first class.

* XML External Entity (XXE) –



A input can be a XML or malicious XML, now they are other kind of data which looks like XML data example even the HTML Looks like a XML even SVG that we have, the PDF or the RTF basically there are many things which looks like a XML and does parses the deserialization and serialization technique that we have regarding the XML they are not able to distinguish either we are talking about the XML or which is something is coming as an external thing.

So, what we have in this above example, there are some entities that need to be execute at a system level to get the file which is etc password, what you are saying if this code is get executed on your server, you are saying boss give me the etc password file that you have in your system and that is something going to be horrible right?

So, the XXE is the way where you get the user input in the format of XML or the XML looking like payload and you execute it and that was having a vulnerability, and you ended up leaking some information of your internal server or executing some code which can actually hamper or damage basically of your server that kind of malicious code you can think of what could be done?

So the solution for this is you should have a proper deserialization, and what to expect in the xml you cannot blindly say that just I have a XML parser, and if it is a XML kind of thing.