Node Security

Group Research

Seraphine, Francisca, Hasanin and Raj

Code Your Future

July 21, 2018

Outline

- https (Raj)
- same origin (Hasanin qasem)
- cookies (Seraphine)
- xss (Francisca)

Recently, Nodejs has become a popular choice to develop backe-end applications as many other server-side languages.

Equally, many factos should be taken into account when it comes to secuering the applications.

- ✓ JavaScript as a language,
- ✓ Node.js and its API (event loop and error handling),
- ✓ Third-party modules interactign with the application and
- ✓ HTTP request(CSRF) and response (XSS) cycles of the applications.

Node and 3rd party modules

Response

As HTTP is a clear-text protocol it must be secured via SSL/TLS tunnel, known as HTTPS. Nowadays high grade ciphers are normally used, misconfiguration in the server can be used to force the use of a weak cipher - or at worst no encryption

Request

same origin

covered...

Hasanin

cookies

covered...

Seraphine

XSS

Francisca: contents

- What's XSS?
- XSS vulnerabilities
- XSS Goal and Types of XSS
- Usual types of attacks
- Methods of prevent XSS
- XSS Quiz

What's XSS?

Cross Site Scripting, otherwise known as XSS is a code injection attack allowing the injection of malicious code into a website. XSS is currently one of the most common website attacks, with almost every website requiring the user to have JavaScript turned on. Rather than being an attack on the website itself, it uses the website as a means to attack the users of that website. When you can get your XSS permanently on a website all those who visit that page will have the JavaScript executed by their browser.

XSS vulnerabilities

XSS vulnerabilities are especially dangerous because an attacker exploiting an XSS attack can gain the ability to do whatever the user can do, and to see what the user sees including passwords, payment and financial information, and more. Worse the victims, both the user and the vulnerable application, often wont be aware theyre being attacked.
The possibility of JavaScript being malicious becomes more clear when you consider the following facts:
- JavaScript has access to some of the user's sensitive information, such as cookies. - JavaScript can send HTTP requests with arbitrary content to arbitrary destinations by using XMLHttpRequest and other mechanisms. - JavaScript can make arbitrary modifications to the HTML of the current page by using DOM manipulation methods.

Goal and Types of XSS

While the goal of an XSS attack is always to execute malicious JavaScript in the victim's browser, there are few fundamentally different ways of achieving that goal.
XSS attacks are often divided into five types:
a) Stored or Persistent XSS, which is when malicious script is injected directly into the vulnerable application. The malicious string originates from the website's database.
b) Reflected XSS or Non-persistent XSS, which involves reflecting malicious script into a link on a page, The attack will activate once the link has been clicked ,where the malicious string originates from the victim's request.
c) DOM-based XSS, where the vulnerability is in the client-side code rather than the server-side code.
d) Self-XSS, is a form of XSS vulnerability which relies on Social

Usual types of attacks

Among many other things, the ability to execute arbitrary JavaScript in another user's browser allows an attacker to perform the following types of attacks:
a) Cookie theft The attacker can access the victim's cookies associated with the website using document.cookie, send them to his own server, and use them to extract sensitive information like session IDs.
b) Keylogging The attacker can register a keyboard event listener using addEventListener and then send all of the user's keystrokes to his own server, potentially recording sensitive information such as passwords and credit card numbers.
c) Phishing The attacker can insert a fake login form into the page using DOM manipulation, set the form's action attribute to target his own server, and then trick the user into submitting sensitive

Methods of prevent XSS

Recall that an XSS attack is a type of code injection: user input is mistakenly interpreted as malicious program code. In order to prevent this type of code injection, secure input handling is needed. In order to be truly vigilant against XSS and other common, debilitating vulnerabilities, like the rest of the OWASP Top 10, its important to use a mix of code review, automated static testing during development and dynamic testing once the application is live, in addition, of course, to using secure coding practices that will help prevent vulnerabilities like cross-site scripting.
For a web developer, there are three fundamentally different ways:
a) Escaping
The first method you can and should use to prevent XSS vulnerabilities from appearing in your applications is by escaping user input. Escaping data means taking the data an application

References

- ✓ *https* : *//www.youtube.com/watch?v* = *M_nIIcKTxGk*
- ✓ *https* : *//www.checkmarx.com/*2017*/*10*/*09*/*3 − *ways* − *prevent* − *xss/https* : *//excess* − *xss.com/*
- ✓ *https* : *//www.incapsula.com/web* − *application* − *security/cross* − *site* − *scripting* − *xss* − *attacks.html*
- ✓ *https* : *//en.wikipedia.org/wiki/Crosssite_scripting*

Conclusion

we have looked at basic security issues...

- learned to work as a group,
- learned to organize and manage a group work using GitHub and research to collect informtion and
- have created some level of security awareness while developing web applitcation using Nodejs