Prototype Pollution Guide

By Vansh Khanna

Step 1: Set up the Environment

First, we need to set up the environment for the pentest. We will use the vulnerable web application NodeGoat, which is an intentionally vulnerable Node.js application designed for practicing and testing various web vulnerabilities.

To install NodeGoat, we need to run the following commands:

```
bash                                                                    Copy code

sudo apt-get update
sudo apt-get install -y git nodejs npm
sudo git clone https://github.com/OWASP/NodeGoat.git /opt/NodeGoat
sudo chown -R $USER:$USER /opt/NodeGoat
cd /opt/NodeGoat
npm install
```

Step 2: Configure NodeGoat

Once NodeGoat is installed, we need to configure it by accessing it through the browser. To do this, we need to open the browser and enter the following URL: `http://localhost:4000/`.

The NodeGoat application will prompt us to create a new user account. We need to enter the required details and create a new account.

Step 3: Set up Burp Suite

Burp Suite is a popular web application testing tool that helps us in testing web applications for vulnerabilities. We can download Burp Suite from its official website and install it on our Kali Linux machine.

After installing Burp Suite, we need to configure our browser to use Burp Suite as a proxy. To do this, we need to open the browser settings and set the proxy to `localhost` and port `8080`.

Step 4: Identify the Vulnerable Input Field

Now, we need to identify the vulnerable input field in NodeGoat. In this case, we will use the `Vulnerable Packages` page, which has a vulnerable input field that is susceptible to Prototype Pollution attacks.

Step 5: Perform Prototype Pollution Attack

We will now use Burp Suite to perform the Prototype Pollution attack. To do this, we need to follow these steps:

1. Open Burp Suite and click on the `Proxy` tab.
2. Start the proxy by clicking on the `Intercept is On` button.
3. Open the browser and navigate to the `Vulnerable Packages` page in NodeGoat.
4. Enter the following text in the `Package Name` input field:

```
css                                                                    Copy code

$[systemSync('touch /tmp/pwned')]
```

5. Click on the `Search` button.
6. Burp Suite will intercept the request and display it in the `Proxy` tab.
7. Right-click on the request and select `Send to Repeater`.
8. In the `Repeater` tab, we will see the intercepted request, and we can modify it to perform the Prototype Pollution attack.
9. Modify the payload by entering the following text in the `Package Name` input field:

```
css                                                                    Copy code

__proto__.polluted=$[systemSync('touch /tmp/pwned')]
```

10. Click on the `Go` button to send the modified request to the server.
11. The server will process the request and execute the injected code, creating a file named `/tmp/pwned`.

Step 6: Validate the Results

Finally, we need to validate the results of the Prototype Pollution attack. We can do this by checking if the injected code is executed on the server-side and created a file named `/tmp/pwned`.

In this case, the validation is successful, and we have successfully demonstrated how a Prototype Pollution attack can be performed using Kali Linux and Burp Suite.

Step 7: Mitigation

To mitigate Prototype Pollution attacks, we need to sanitize and validate all user input before using it in the application. We can also use a JavaScript library or framework that provides built-in protection against Prototype Pollution attacks, such as sanitizing user input and validating object properties.

Additionally, we can use a Content Security Policy (CSP) to restrict the execution of unauthorized JavaScript code and prevent the injection of malicious scripts.

Conclusion

In this tutorial, we have demonstrated how to perform a Prototype Pollution attack using Kali Linux and Burp Suite. We have also discussed the mitigation techniques that can be used to prevent such attacks.

It is essential for developers and security professionals to understand the various web vulnerabilities and how they can be exploited. By conducting regular penetration testing and vulnerability assessments, we can identify and remediate potential security risks in our web applications.

Additional Resources

1. OWASP - Prototype Pollution: **https://owasp.org/www-community/attacks/Prototype_Pollution**
2. Prototype Pollution: **https://portswigger.net/research/prototype-pollution-a-new-era-of-exploitation**
3. Prototype Pollution Attacks in NodeJS Applications: **https://research.securitum.com/prototype-pollution-attacks-in-nodejs-applications/**
4. Prototype Pollution in AngularJS: **https://snyk.io/blog/prototype-pollution-angularjs-cleanup/**
5. Prototype Pollution Attacks in the Wild: **https://www.acunetix.com/blog/web-security-zone/prototype-pollution-attacks-in-the-wild/**
6. Pentesting with Burp Suite: **https://portswigger.net/burp/documentation/desktop/testing**
7. OWASP Testing Guide - Testing for Prototype Pollution: **https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/15-Testing_for_Client_Side_Script_Injection/01-Testing_for_Prototype_Pollution/**