International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET218318

Server-Side Template Injection with Custom Exploit

Rushi Mamtora[1], Dr. Priyanka Sharma[2]
School of Information Technology, Artificial Intelligence, and Cyber Security, Rashtriya Raksha University, Gandhinagar, Gujarat, India

Article Info
Volume 8, Issue 3
Page Number: 105-108

Publication Issue :
May-June-2021

Article History
Accepted : 10 May 2021
Published: 16 May 2021

ABSTRACT

Cyber attacks are getting progressively incessant, causing a great deal of harm. Attackers take our valuable information by compromising web application security loopholes. Dynamic content that is being incorporated into the html that has been served to the client. assume when you open a site page then you see your name so that is dynamic substance for each client who additionally at any point visits that page. We can inject input fields and they are shipped off the web worker. So ,we need to check for all information handled whose worth is reflected in some structure to get the prepared payload. Then attempt to misuse it dependent on the layouts. This paper discusses the idea of an template injection and its impact on template based web application

Keywords: Cyber Security, Vulnerability, Templates, Websecurity, Remote Code Execution, Directory Path Traversal.

I. INTRODUCTION

Nowadays, Templates are widely used by web applications to present dynamic data via web pages and emails. Unsafely embedding user input in templates which empowers SSTI. It occurs when an attacker is able to use original template syntax to inject a malicious payload into a template, which is then executed server-side. SSTI vulnerabilities can reveal websites to a number of attacks relying upon the format of the template engine. In certain uncommon conditions, these vulnerabilities affect no real security risk. However, most of the time, the impact of server-side template injection can be great damage. An attacker can potentially achieve remote code execution by taking full control of the back-end server and using it to perform different attacks on internal infrastructure. SSTI is occurs when an attacker is able to use original template syntax to inject a malicious payload into a template, which is then executed server-side. This attack potentially allows attackers to upload custom exploits like, remote code execution, directory traversal, command execution, XML injection, etc

II. OBJECTIVE

Cyber attacks are getting progressively incessant, causing a ton of harm. Assailants take our important information by bargaining web application security loopholes .Dynamic content that is being incorporated into the html that has been served to the client. assume when you open a page then you see your name so that is a dynamic substance for each client who likewise at any point visits that page.

Typically it is being finished by templates like, twig (php), jinja2(python), smarty(php),etc. which utilize

fixed static templates for html records and the qualities are being supplanted with whatever esteem the server is resembling passing it.

However this is the symptom of XSS and more critical or serious vulnerability. So we need to deep dive into this and try to perform mathematical operations. This kind of code can perform it also.
Ex. custom_email={{3**3}}
9

This kind of stuff works depending on the templates used by the website and developers. We can check if the application is approving information fields by embedding characters that are utilized in the server side template to incorporate mandates, as : < ! # =/. " - > likewise we can check [a-zA-Z0-9].

Likewise we can check by certain pages with augmentations like .stm, .shtm and .shtml. That isn't sure if these expansions are there in the site is helpless in light of the fact that it is just weak when the web server licenses server side incorporate execution without legitimate approval. This can prompt access and control of the record framework and interaction over the web server. Attackers can get to delicate data, for example, secret phrase documents, and execute the shell commands. Attack result will be distinguishable whenever that the page is stacked for the client's program.

A. DETECT

1. PLAIN TEXT

Most template languages are freeform text where you can directly input html code.

Ex.
smarty = hello {user.name}
Hello user
freemarker = hello ${username}
hello user2
any=hello
hello

This kind of thing we can exploit by doing XSS[5]. But also it can allow for template injection.

Ex.
smarty = hello ${3*3}
hello 9

freemarker = hello ${3*3}
hello 9

2. CODE TEXT

User input can be also placed with template statements.
welcome_greeting = username
Hello user1

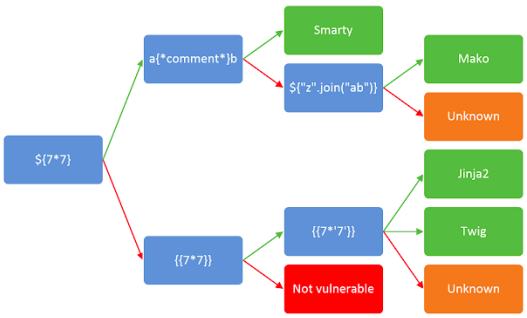
So, XSS is not obvious in this. We need to break out the tag after it.

welcome_greeting = username<tag>
Hello

Welcome_greeting = username}}<tag>
Hello user1 <tag>

B. IDENTITY

After detecting template injection our next step is to identify the template engine usage. Sometimes it is trivial. We need to do some try and error methods and try to identify. We can do this thing by using burpsuite tool to automate the process of identity.



C. EXPLOIT

First, we need to read the template engine's documentation. Like, list of built-in methods, filters, variables, extensions, plugins, etc. The second step is to explore the environment to find out what exactly we have to access. Suppose there are no built-in variables then we can brute force it by using a wordlist. The last step is to proceed with traditional security techniques, reviewing functions for exploitable vulnerabilities. This is an important approach in context. If the wider application is there then we can follow template injection to trigger arbitrary object creation, arbitrary read/write/upload, remote file inclusion, information disclosure, etc.

III. Result & Discussion

According to different kinds of literature, manual methodology consistently gives the best precision. However, to distinguish various kinds of layouts there must be a work process to recognize it and do misuse part of it. Already individuals had done research on various formats and other vulnerabilities. Various layouts have various personalities and from that, we can go further for the misuse part. That abuse will be covered by the most significant vulnerabilities like Remote Code Execution, Path Traversal, XSS, and so on.

An attempt has been made in 2018[1], to unmask the backend logic of an Ajax template injection, its process of exploitation by which it can be used to access database tables and columns. The results show that it currently poses a threat to Ajax-based web applications, but it can keep a footprint of the most sophisticated attack in the history of cyberspace attack. Therefore, the attempt was intended to help web developers, security experts, and penetration testers to consider it seriously even if it is a low-level Ajax-based security loophole.

File sharing and downloading activities using web applications have now become common, not just ensuring the easy distribution of various kinds of records and archives yet in addition hugely lessening the time and exertion of clients. Albeit the online administrations that are being utilized much of the time have made clients' life simpler, it has expanded the danger of misuse of local file disclosure (LFD) weakness in the web uses of various public-area associations due to unstable plan and indiscreet coding. In 2017[2], examinations examine the underlying driver of LFD weakness and its misuse procedures.

Kullback-Leibler distance (or divergence) (KLD) to identify the side effects of code injection attack ahead of schedule during program runtime. Bit of leeway of the perception that during code injection attack, the proposed structure goes amiss from the normal design. The KLD can be an appropriate measure to catch the deviation. That Includes the advancement of a worker-side framework to register KLD. They applied a smoothing algorithm to keep away from the limitless KLD distance during the assault recognition stage and assess the approach with three PHP applications having SQLI and XSS vulnerabilities. The underlying outcomes show that KLD can be a powerful estimation strategy to distinguish the event of a code injection attack. The methodology experiences lower false positive and negative rates, and forces unimportant runtime overhead. (March - 2014)[3].

Remote Code Execution (RCE) attacks in web applications. The examination reasons about the

string and non-string conduct of a program firmly. It initially makes two abstractions of the program to show the string and non-string behavior, respectively, which are encoded to limitations independently. A novel algorithm is created to determine the two sets of constraints together. The strategy handles a great deal of RCE explicit difficulties by expanding the deliberations. Our examination shows that the procedure is compelling in recognizing RCE vulnerabilities in certifiable PHP applications, delivering many fewer false positives contrasted with elective methods. Furthermore, the fundamental limitation tackling calculation is very efficient. (2013)[4].

A static analysis for finding XSS vulnerabilities that analyzes the root cause of XSS: weak input validation. In May-2008[5], analysis checked whether untrusted input to the server can invoke a client's JavaScript interpreter. We made a careful examination of the W3C recommendation, the Firefox source code, and other online sources to express this policy in formal language terms. We have demonstrated that our approach can scale to large codebases and can detect known and unknown XSS vulnerabilities in real-world web applications with manually written input validation routines.

IV. Conclusion

The goal of this research due to lack of input validation the templates are possibly vulnerable to server side template injection. The use input appears to be placed into dynamically evaluated inputs that allow an attacker to execute arbitrary code execution, directory path traversal, etc. Developer needs to pass user inputs into templates as parameters. Sanitize the input before passing it into the templates by removing unwanted and risky characters before parsing the data. This minimizes the vulnerabilities for any malicious probing of your templates. I will research first to detect the templates and then go for the execution step.

V. REFERENCES

[1]. Vijit Das Noyon, Yeahia Md Abid , Md. Maruf Hassan , Md. Hasan Sharif, Fabiha Nawar Deepa, Rayhanul Islam Rumel, Rafita Haque, Samia Nasrin, Moniruz Zaman. A Study of Ajax Template Injection in Web Applications. https://www.researchgate.net/publication/326668286_A_Study_of_Ajax_Template_Injection_in_Web_Applications

[2]. M. I. Ahmed, M. M. Hassan, and T. Bhuyian. Local File Disclosure Vulnerability: A Case Study of Public-Sector Web Applications https://www.researchgate.net/publication/322236714_Local_File_Disclosure_Vulnerability_A_Case_Study_of_Public-Sector_Web_Applications

[3]. Hossain Shahriar*, Sarah M. North, YoonJi Lee and Roger Hu. Server-side code injection attack detection based on Kullback-Leibler distance. https://www.researchgate.net/publication/280768581_Server-Side_Code_Injection_Attack_Detection_Based_on_Kullback-Leibler_Distance

[4]. Yunhui Zheng, Xiangyu Zhang. Path sensitive static analysis of web applications for remote code execution vulnerability detection https://ieeexplore.ieee.org/document/6606611

[5]. Gary Wassermann; Zhendong Su. Static detection of cross-site scripting vulnerabilities. https://ieeexplore.ieee.org/document/4814128

Cite this article as :

Rushi Mamtora, Dr. Priyanka Sharma "Server-Side Template Injection with Custom Exploit", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 8 Issue 3, pp. 105-108, May-June 2021. Available at doi : https://doi.org/10.32628/IJSRSET218318

Journal URL : https://ijsrset.com/IJSRSET218318