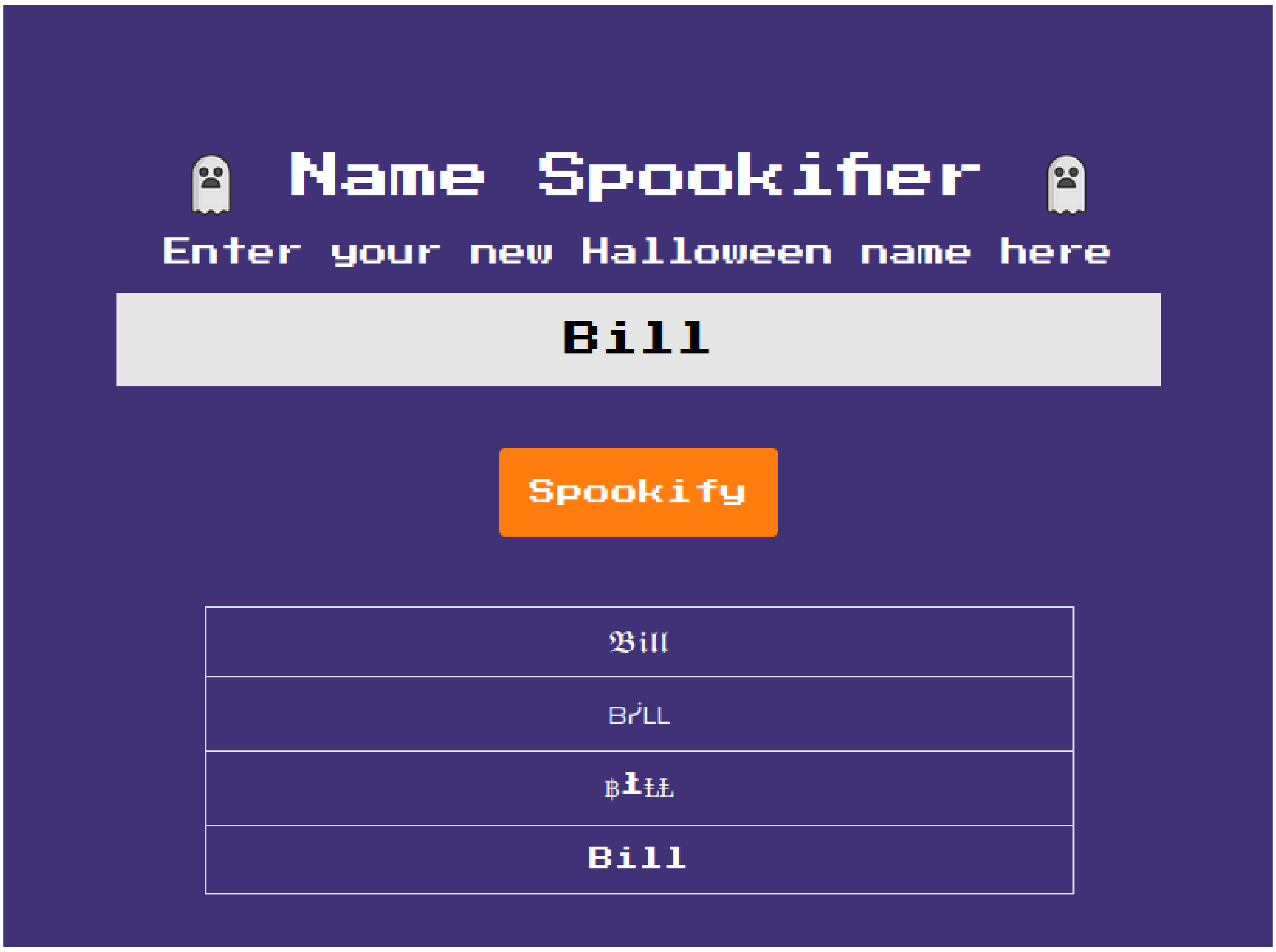
Spookifier

Background

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.
XSS attacks can generally be categorized into two categories: reflected and stored. There is a third, much less well-known type of XSS attack called DOM Based XSS. However in this write-up we focus only on the reflected XSS.
Reflected cross-site scripting (or XSS) arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.



Bill

𝔅𝔦𝔩𝔩
BіLL
฿ł₤₤
Bill

Reflected attacks are those where the injected script is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other website. When a user is tricked into clicking on a malicious link, submitting a specially crafted form, or even just browsing to a malicious site, the injected code travels to the vulnerable web site, which reflects the attack back to the user’s browser. The browser then executes the code because it came from a “trusted” server. Reflected XSS is also sometimes referred to as Non-Persistent or Type-I XSS (the attack is carried out through a single request / response cycle).

Template injection

Template engines are widely used by web applications to present dynamic data via web pages and emails. Unsafely embedding user input in templates enables Server-Side Template Injection, a frequently critical vulnerability that is extremely easy to mistake for Cross-Site Scripting] (XSS), or miss entirely. Unlike XSS, Template Injection can be used to directly attack web servers' internals and often obtain Remote Code Execution (RCE), turning every vulnerable application into a potential pivot point.

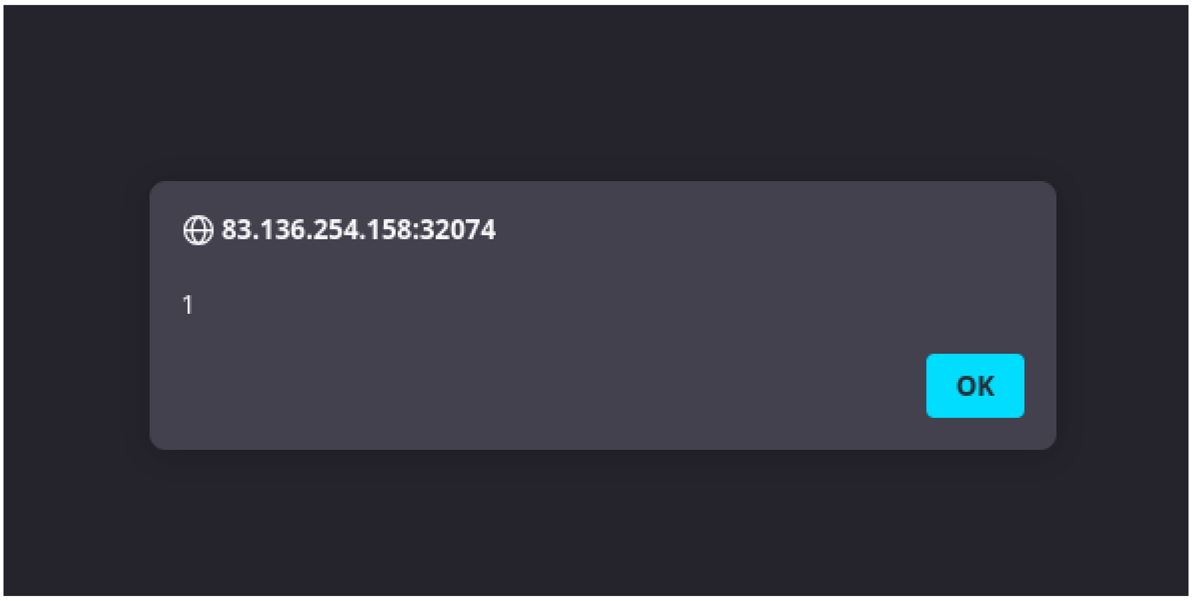
Vulnerability

The machine is vulnerable to Server-Side Template Injection (SSTI). Thanks to the "mako" template engine it is possible to write arbitrary python commands. Mako allows execution of Python code via its templates, making it a powerful vector for exploitation when improperly sanitized inputs are injected.

Solution

Visiting the site I noticed that It reflected the user input so I firstly tried a basic XSS payload to confirm the vulnerability.

`


83.136.254.158:32074

1

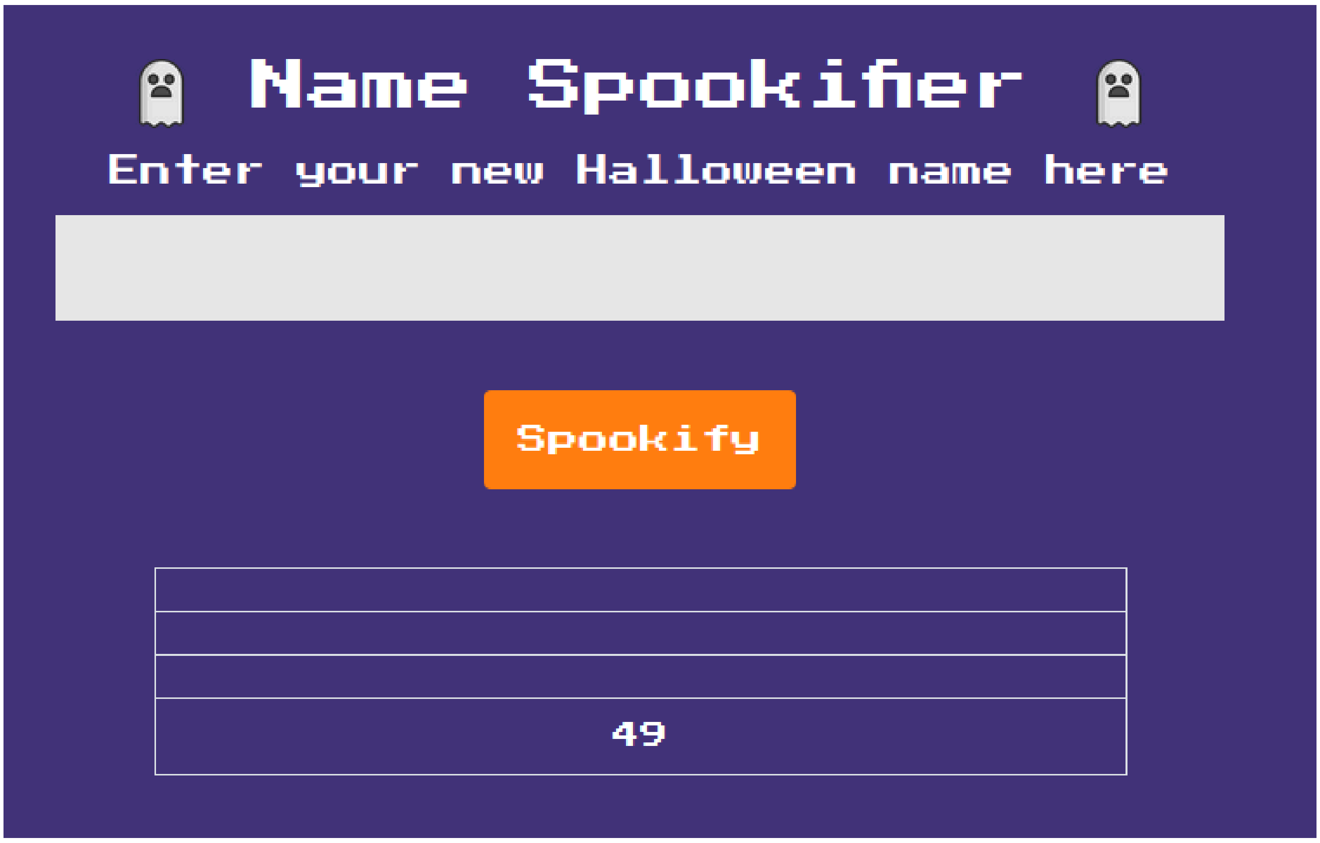
The site is vulnerable to reflected XSS, so I thought was a XSS challenge, but I wasn't sure so I checked out the source code and I found something interesting:

```
from mako.template import Template
```

In the application/util.py file I found this line so I immediately tried for a template injection using this payload:

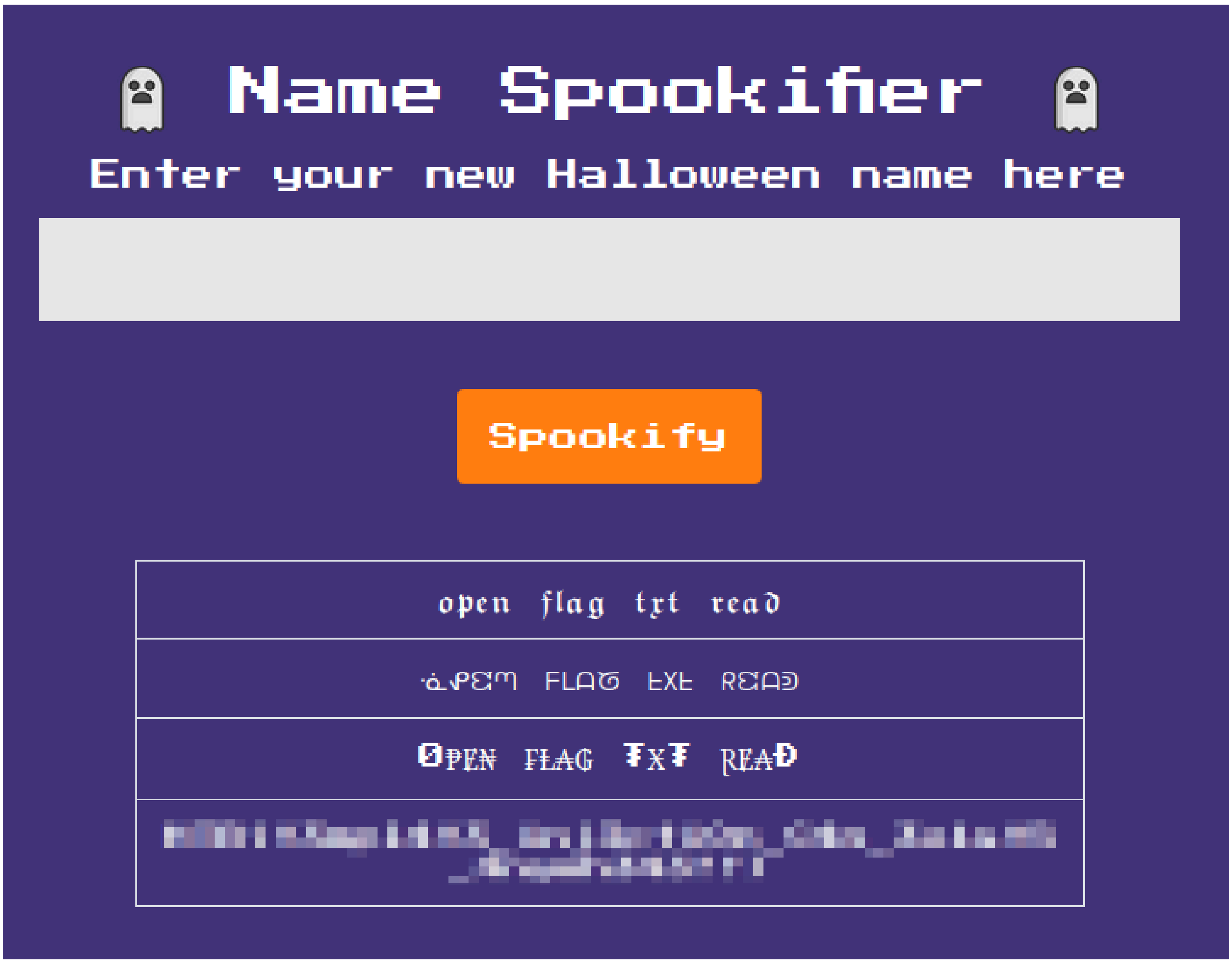
`${7*7}`

It turned out that the application was vulnerable to SSTI infection since it printed:



Then I crafted this payload to retrieve the flag:

`${open('/flag.txt').read()}`



𝔬𝔭𝔢𝔫 𝔣𝔩𝔞𝔤 𝔱𝔵𝔱 𝔯𝔢𝔞𝔡
ᓍᑭᕮᘻ FLᗩᘜ ᒪXᒪ ᖇᕮᗩᗫ
ØPEN FⱠAG ŦxŦ REAĐ
[redacted]

References

https://portswigger.net/web-security/cross-site-scripting/reflected
https://owasp.org/www-community/attacks/xss/
https://portswigger.net/research/server-side-template-injection