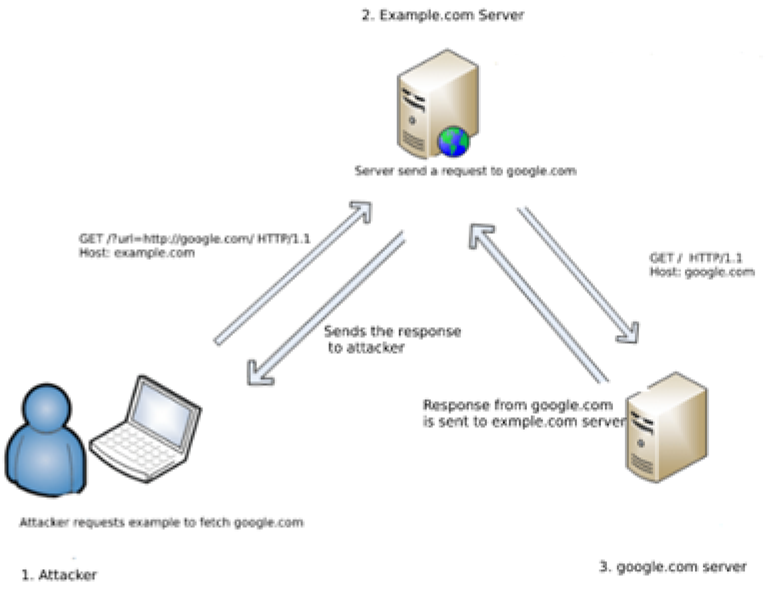
SSRF: Server-side request forgery

What is SSRF?

Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests to an arbitrary domain of the attacker's choosing.

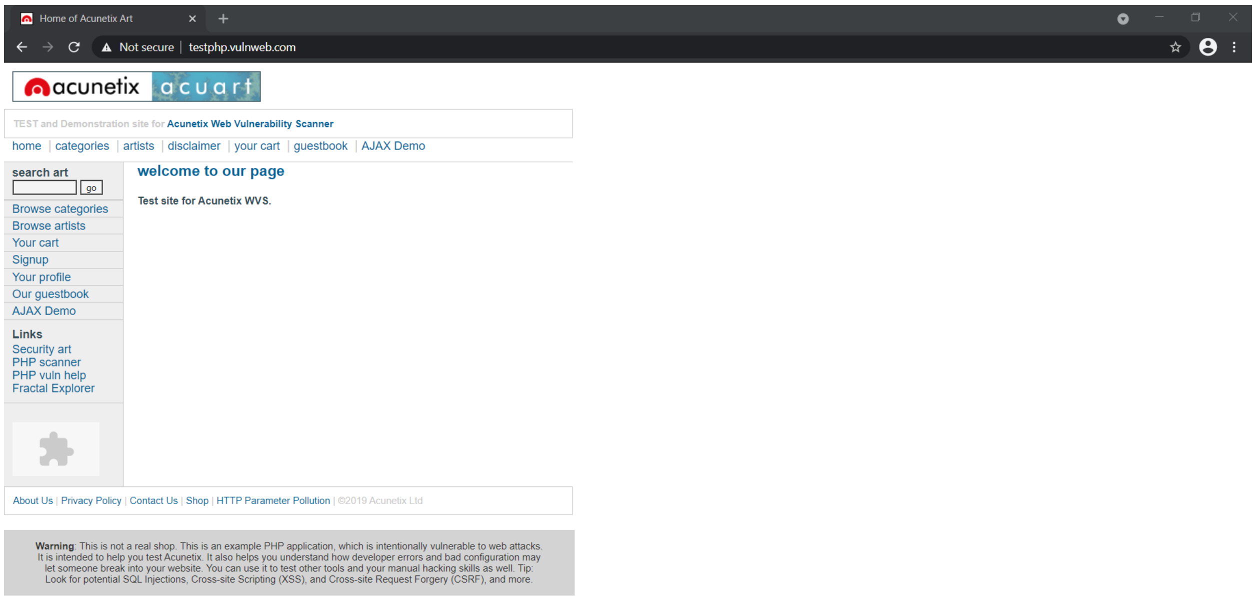
In a typical SSRF attack, the attacker might cause the server to make a connection to internal-only services within the organization's infrastructure. In other cases, they may be able to force the server to connect to arbitrary external systems, potentially leaking sensitive data such as authorization credentials.

In simple words, Server-Side Request Forgery (SSRF) refers to an attack, wherein an attacker can send a crafted request from a vulnerable web application. SSRF is mainly used to target internal systems behind WAF (web application firewall), that are unreachable to an attacker from the external network. Additionally, it’s also possible for an attacker to mark SSRF, for accessing services from the same server that is listening on the loopback interface address called (127.0.0.1).


2. Example.com Server
Server send a request to google.com
GET /?url=http://google.com/ HTTP/1.1
Host: example.com
GET / HTTP/1.1
Host: google.com
Sends the response to attacker
Response from google.com is sent to exmple.com server
Attacker requests example to fetch google.com
1. Attacker
3. google.com server

Let's understand using an example

So currently I am on a vulnerable website which is: `http://testphp.vulnweb.com`


Home of Acunetix Art
Not secure | testphp.vulnweb.com
acunetix acuart
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
About Us | Privacy Policy | Contact Us | Shop | HTTP Parameter Pollution | ©2019 Acunetix Ltd

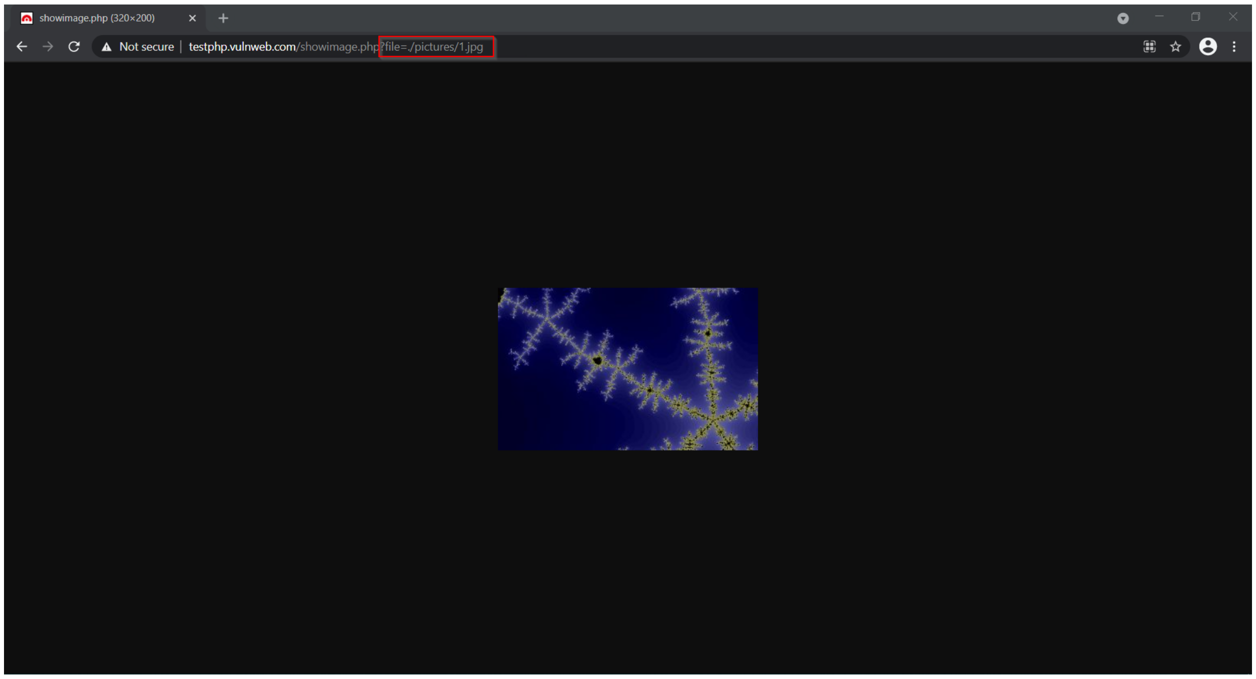
TEST and Demonstration site for **Acunetix Web Vulnerability Scanner**

welcome to our page

Test site for Acunetix WVS.

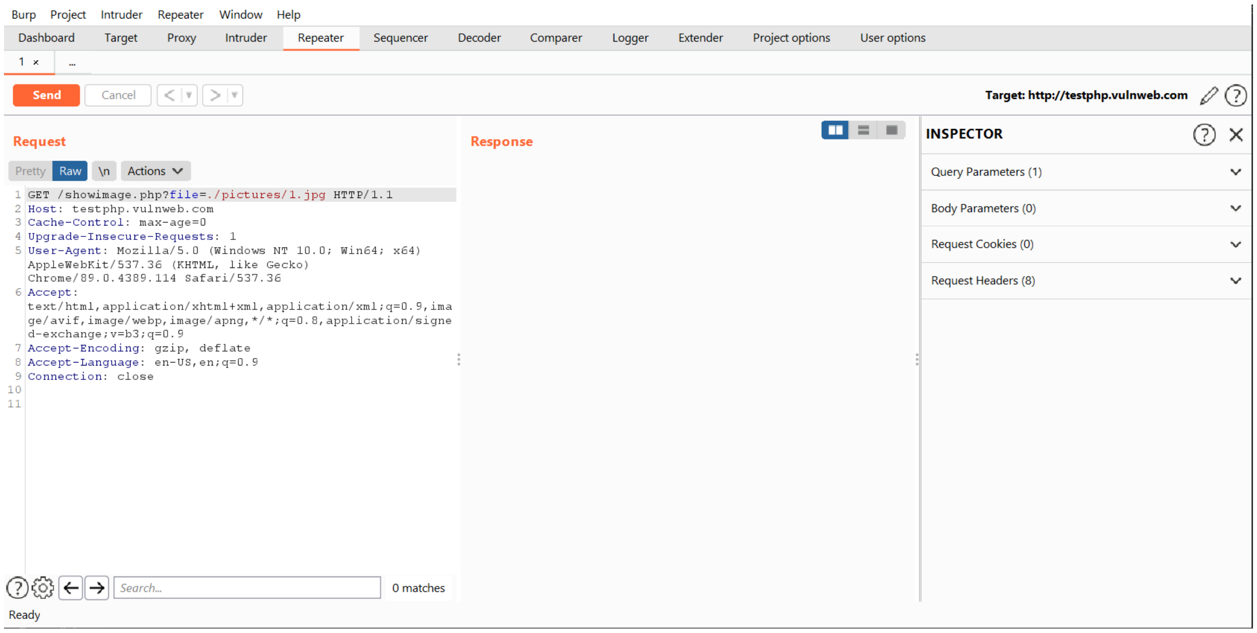
Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

So let's go to the endpoint `http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg`



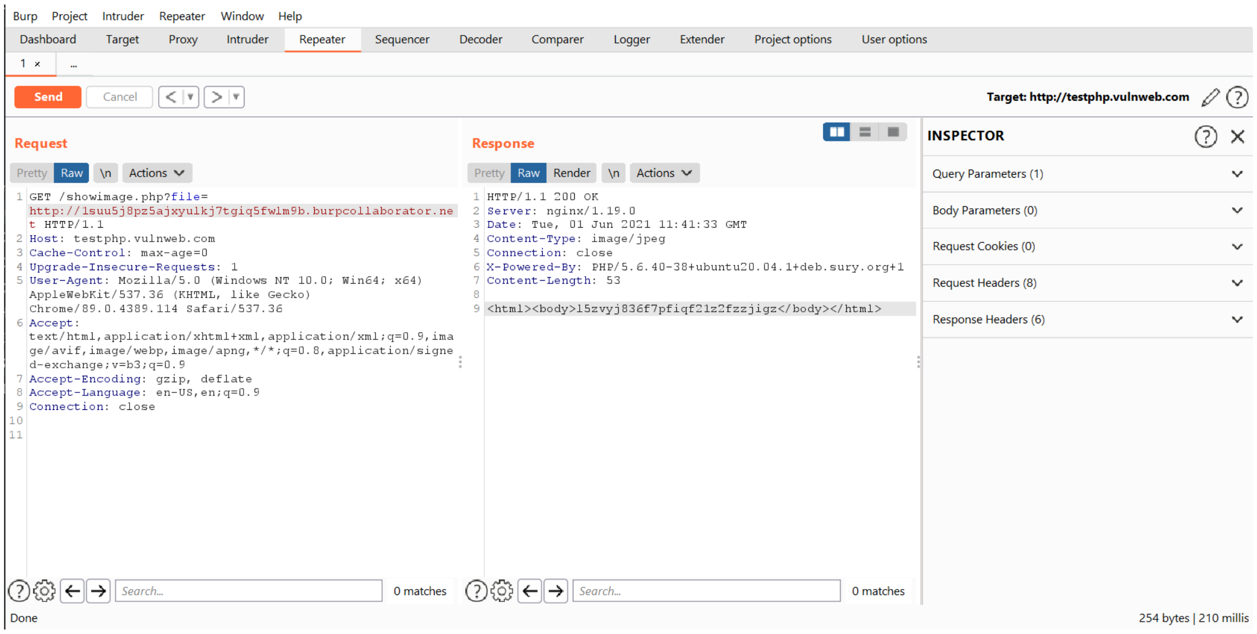
Notice the parameter `?file=` while performing SSRF attacks always remember that you need a parameter which interacts with the system. The `?file=` is one of those parameters which selects an image file from the server and displays it.

Let's quickly intercept this request and send it to our `Repeater`.


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options
Send
Cancel
Target: http://testphp.vulnweb.com
Request
Pretty Raw \n Actions
Response
INSPECTOR
Query Parameters (1)
Body Parameters (0)
Request Cookies (0)
Request Headers (8)
Search...
0 matches
Ready

```
GET /showimage.php?file=./pictures/1.jpg HTTP/1.1
Host: testphp.vulnweb.com
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/89.0.4389.114 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,ima
ge/avif,image/webp,image/apng,*/*;q=0.8,application/signe
d-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Alright so to perform an `SSRF` attack let's change the value of `file` parameter to our Burp Collaborator Client ID (Generating this is shown in the later part).



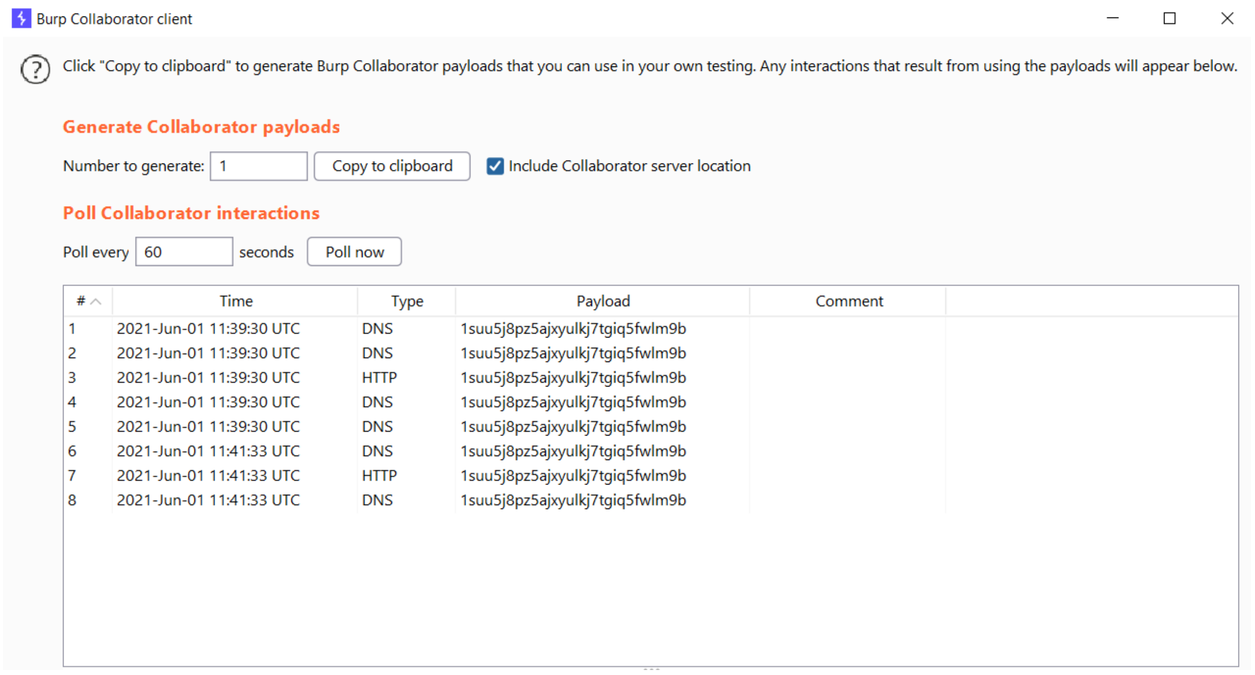
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options
Send Cancel
Target: http://testphp.vulnweb.com
Request
Pretty Raw \n Actions
Response
Pretty Raw Render \n Actions
INSPECTOR
Query Parameters (1)
Body Parameters (0)
Request Cookies (0)
Request Headers (8)
Response Headers (6)
0 matches
0 matches
Done
254 bytes | 210 millis

```
GET /showimage.php?file=
http://1suu5j8pz5ajxyulkj7tgiq5fwlm9b.burpcollaborator.ne
t HTTP/1.1
Host: testphp.vulnweb.com
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/89.0.4389.114 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,ima
ge/avif,image/webp,image/apng,*/*;q=0.8,application/signe
d-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Tue, 01 Jun 2021 11:41:33 GMT
Content-Type: image/jpeg
Connection: close
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: 53

<html><body>15zvyj836f7pfiqf21z2fzzjigz</body></html>
```

Notice the vulnerable application accepted the server address and interacted with it.



Burp Collaborator client
Generate Collaborator payloads
Copy to clipboard
Poll Collaborator interactions
Poll now

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the payloads will appear below.

Number to generate: 1

[✓] Include Collaborator server location

Poll every 60 seconds

#	Time	Type	Payload	Comment
1	2021-Jun-01 11:39:30 UTC	DNS	1suu5j8pz5ajxyulkj7tgiq5fwlm9b	
2	2021-Jun-01 11:39:30 UTC	DNS	1suu5j8pz5ajxyulkj7tgiq5fwlm9b	
3	2021-Jun-01 11:39:30 UTC	HTTP	1suu5j8pz5ajxyulkj7tgiq5fwlm9b	
4	2021-Jun-01 11:39:30 UTC	DNS	1suu5j8pz5ajxyulkj7tgiq5fwlm9b	
5	2021-Jun-01 11:39:30 UTC	DNS	1suu5j8pz5ajxyulkj7tgiq5fwlm9b	
6	2021-Jun-01 11:41:33 UTC	DNS	1suu5j8pz5ajxyulkj7tgiq5fwlm9b	
7	2021-Jun-01 11:41:33 UTC	HTTP	1suu5j8pz5ajxyulkj7tgiq5fwlm9b	
8	2021-Jun-01 11:41:33 UTC	DNS	1suu5j8pz5ajxyulkj7tgiq5fwlm9b	

These are the logs created. Notice we got a HTTP request on our server which proves that the web application `http://testphp.vulnweb.com` is vulnerable to SSRF Attack.

Burp Collaborator and Alternatives!

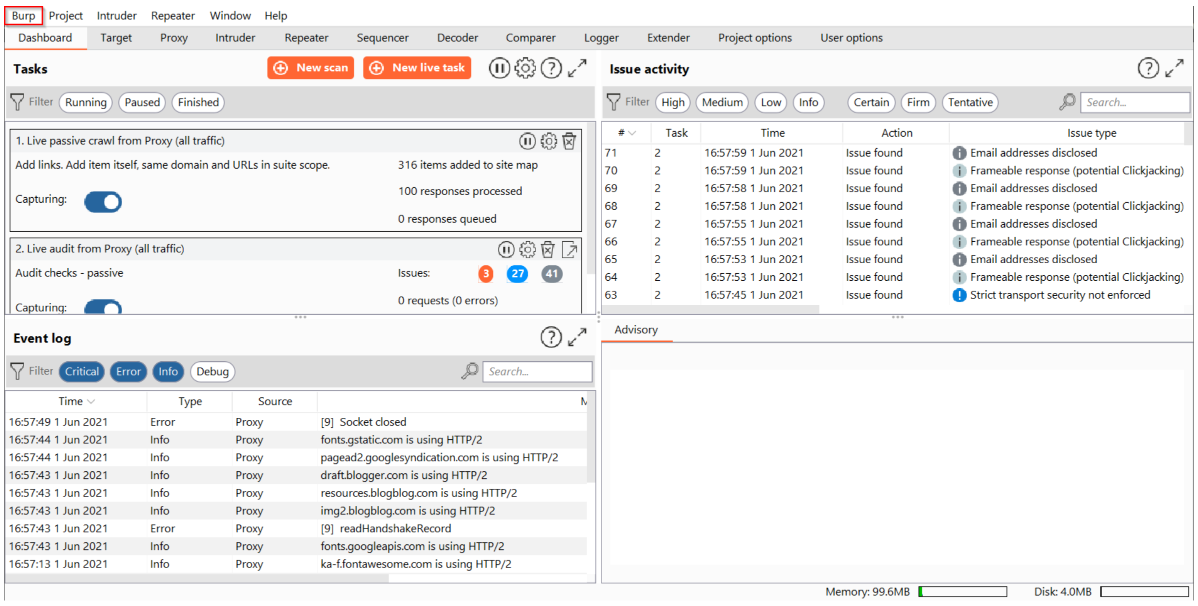
Burp Collaborator is a network service that Burp Suite uses to help discover many kinds of vulnerabilities. In simple words it basically creates a server for you which you can use to redirect requests. Services provided by Burp Collaborator are as follows:

- It uses its own dedicated domain name, and the server is registered as the authoritative DNS server for this domain.
- It provides a DNS service that answers any lookup on its registered domain (or subdomains) with its own IP address.
- It provides an HTTP/HTTPS service, and uses a valid, CA-signed, wildcard TLS certificate for its domain name.
- It provides an SMTP/SMTPS service.

Note: Burp Collaborator comes with the Pro Version of Burp Suite!

Steps to set up Burp Collaborator

1. Click on `Burp`



1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, same domain and URLs in suite scope.
Capturing:
316 items added to site map
100 responses processed
0 responses queued

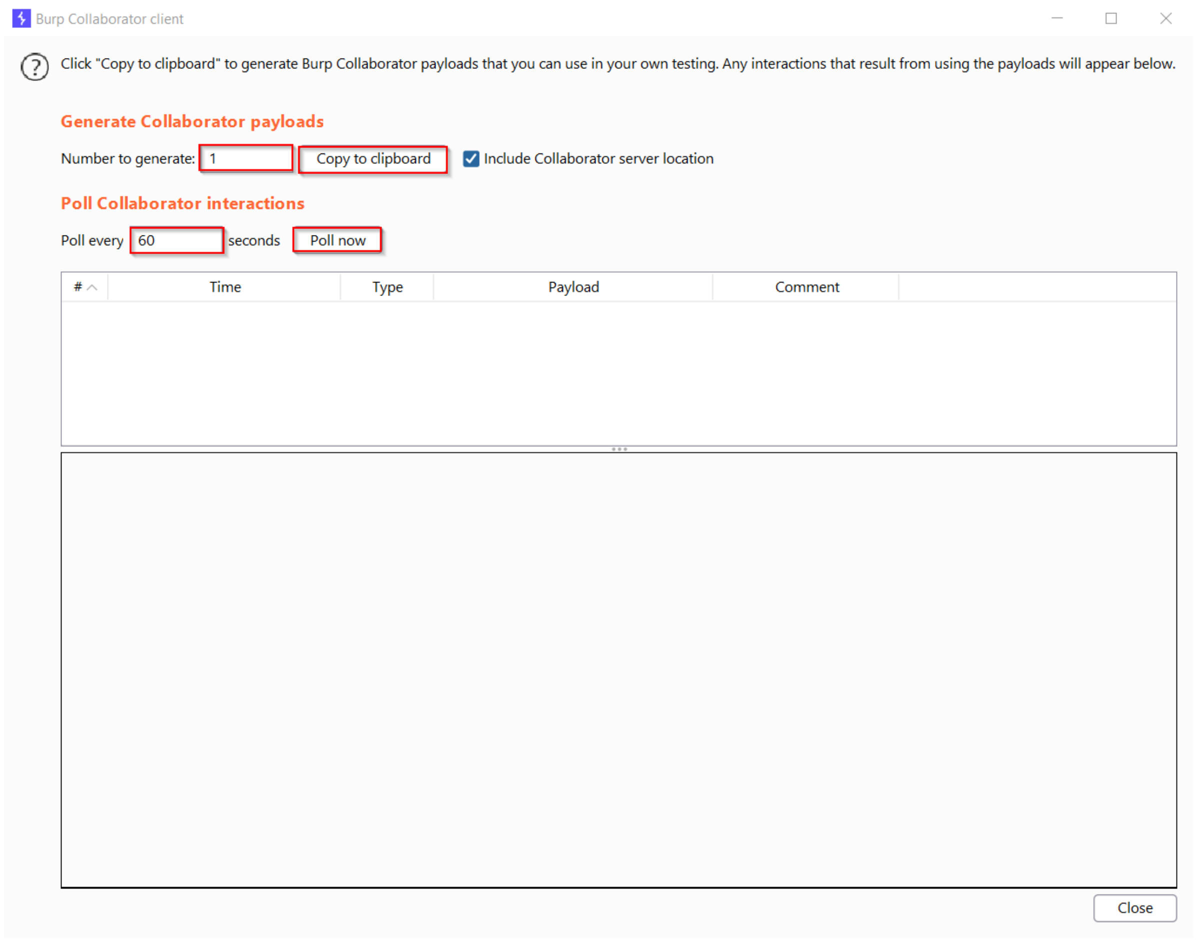
2. Live audit from Proxy (all traffic)
Audit checks - passive
Capturing:
Issues: 3 27 41
0 requests (0 errors)

#	Task	Time	Action	Issue type
71	2	16:57:59 1 Jun 2021	Issue found	Email addresses disclosed
70	2	16:57:59 1 Jun 2021	Issue found	Frameable response (potential Clickjacking)
69	2	16:57:58 1 Jun 2021	Issue found	Email addresses disclosed
68	2	16:57:58 1 Jun 2021	Issue found	Frameable response (potential Clickjacking)
67	2	16:57:55 1 Jun 2021	Issue found	Email addresses disclosed
66	2	16:57:55 1 Jun 2021	Issue found	Frameable response (potential Clickjacking)
65	2	16:57:53 1 Jun 2021	Issue found	Email addresses disclosed
64	2	16:57:53 1 Jun 2021	Issue found	Frameable response (potential Clickjacking)
63	2	16:57:45 1 Jun 2021	Issue found	Strict transport security not enforced

Time	Type	Source	M
16:57:49 1 Jun 2021	Error	Proxy	[9] Socket closed
16:57:44 1 Jun 2021	Info	Proxy	fonts.gstatic.com is using HTTP/2
16:57:44 1 Jun 2021	Info	Proxy	pagead2.googlesyndication.com is using HTTP/2
16:57:43 1 Jun 2021	Info	Proxy	draft.blogger.com is using HTTP/2
16:57:43 1 Jun 2021	Info	Proxy	resources.blogblog.com is using HTTP/2
16:57:43 1 Jun 2021	Info	Proxy	img2.blogblog.com is using HTTP/2
16:57:43 1 Jun 2021	Error	Proxy	[9] readHandshakeRecord
16:57:43 1 Jun 2021	Info	Proxy	fonts.googleapis.com is using HTTP/2
16:57:13 1 Jun 2021	Info	Proxy	ka-f.fontawesome.com is using HTTP/2

Memory: 99.6MB Disk: 4.0MB

2. `Click on Burp Collaborator Client`


Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the payloads will appear below.

Generate Collaborator payloads

Number to generate: 1 | Copy to clipboard | ✓ Include Collaborator server location

Poll Collaborator interactions

Poll every 60 seconds | Poll now

#	Time	Type	Payload	Comment

3. The various options over here are:

   1. In `Number to generate` enter the number of servers you want to create.
   2. You can copy the address of your server using the `Copy to Clipboard` button
   3. `Poll every` will basically check for the requests on the specified time
   4. `Poll Now` will check for request whenever that button is hit

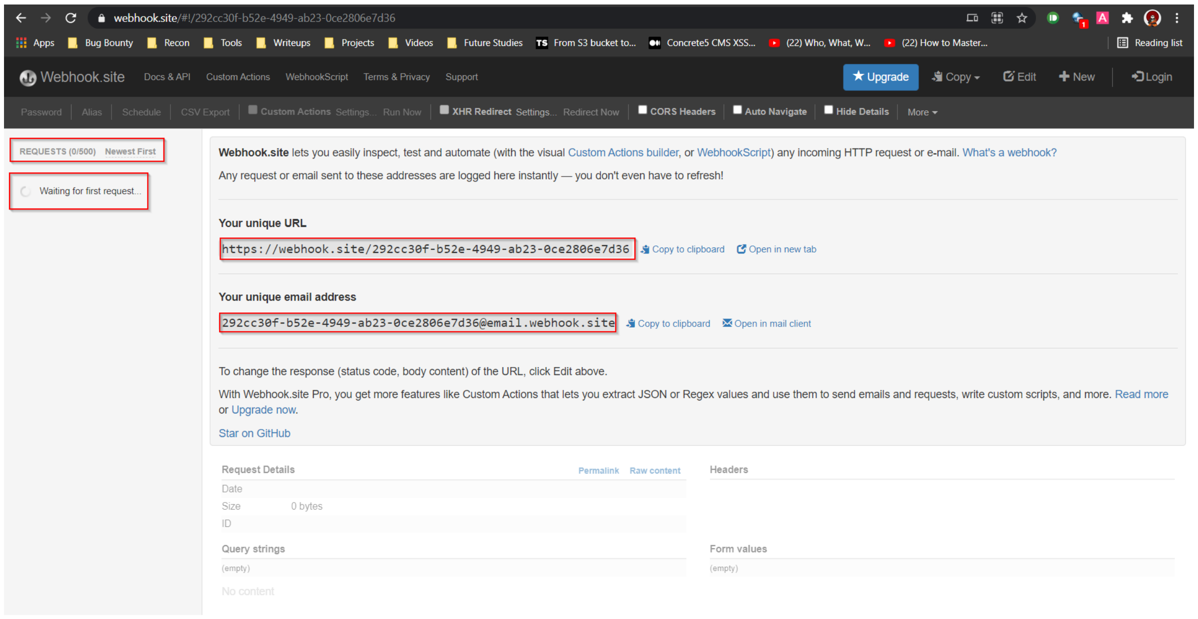
Alternatives to Burp Collaborator

For all those who do not use Burp Suite Pro we got you covered. Following are some of the alternatives you can use for Burp Collaborator

1. Webhook.site (https://webhook.site/)

   Webhook.site is an external alternative one can use. It allows you to create your own server as well as provides you with your unique email address and a mail client for it. It allows you up to 500 requests which is more than enough.

Although you don't need an account to create you own server but having an account will let you save your work.


webhook.site/#!/292cc30f-b52e-4949-ab23-0ce2806e7d36

Webhook.site lets you easily inspect, test and automate (with the visual Custom Actions builder, or WebhookScript) any incoming HTTP request or e-mail. What's a webhook?

Any request or email sent to these addresses are logged here instantly — you don't even have to refresh!

REQUESTS (0/500) Newest First

Waiting for first request...

Your unique URL

`https://webhook.site/292cc30f-b52e-4949-ab23-0ce2806e7d36`

Your unique email address

`292cc30f-b52e-4949-ab23-0ce2806e7d36@email.webhook.site`

To change the response (status code, body content) of the URL, click Edit above.

With Webhook.site Pro, you get more features like Custom Actions that lets you extract JSON or Regex values and use them to send emails and requests, write custom scripts, and more. Read more or Upgrade now.

Star on GitHub

Request Details	
Date	
Size	0 bytes
ID	

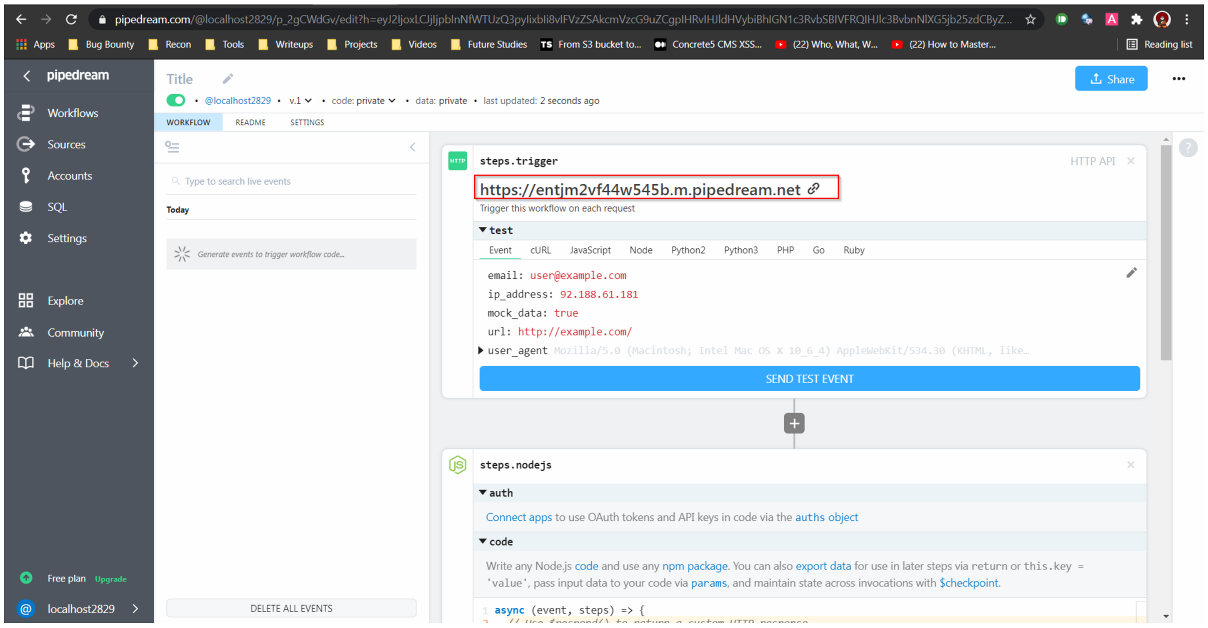
Query strings: (empty)

Form values: (empty)

No content

2. Request Bin (https://requestbin.com/)

Request Bin allows you to create a trigger from variety of sources. The only disadvantage over here is that you compulsory need an account in order to generate a server address


pipedream.com/@localhost2829/p_2gCWdGv/edit?h=eyJ2ljoxLCJjljpblnNfWTUzQ3pylixbli8vlFVzZSAkcmVzcG9uZCgplHRvIHJldHVybiBhlGN1c3RvbSBIVFRQIHJlc3BvbnNlXG5jb25zdCByZ...

Title • @localhost2829 • v.1 • code: private • data: private • last updated: 2 seconds ago

steps.trigger

https://entjm2vf44w545b.m.pipedream.net

Trigger this workflow on each request

```
email: user@example.com
ip_address: 92.188.61.181
mock_data: true
url: http://example.com/
user_agent Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_4) AppleWebKit/534.30 (KHTML, like…
```

steps.nodejs

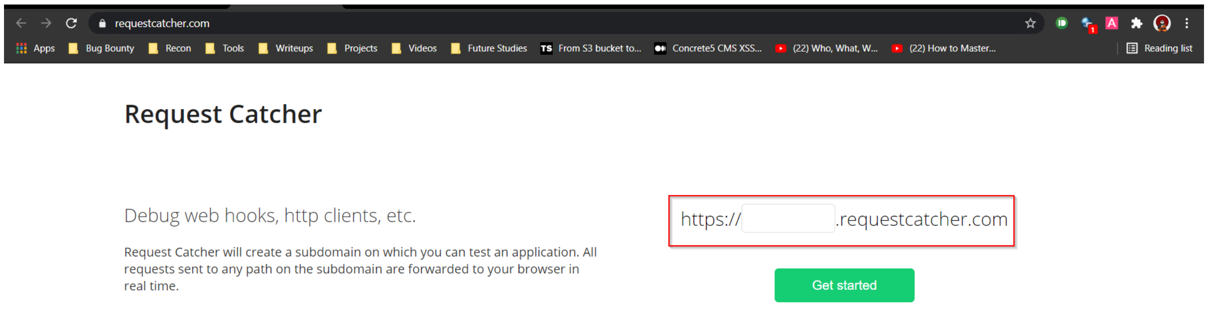
auth: Connect apps to use OAuth tokens and API keys in code via the auths object

code: Write any Node.js code and use any npm package. You can also export data for use in later steps via return or `this.key = 'value'`, pass input data to your code via params, and maintain state across invocations with $checkpoint.

```
async (event, steps) => {
```

3. Request Catcher (https://requestcatcher.com/)

Another beautiful tool is Request Catcher. It allows you to create your own custom subdomain. The best part is you do not require any Sign Up!


Request Catcher

Debug web hooks, http clients, etc.

Request Catcher will create a subdomain on which you can test an application. All requests sent to any path on the subdomain are forwarded to your browser in real time.

Exploiting SSRF vulnerability

SSRF attacks often exploit trust relationships to escalate an attack from the vulnerable application and perform unauthorized actions. These trust relationships might exist in relation to the server itself, or in relation to other back-end systems within the same organization.

The steps to exploit this vulnerability are:

- Exploit a web application to induce requests to attackers controlled server
- Configure Burp Collaborator as third party server
- Wait for interaction
- Attack Successful

The actual working scenario can be something like this:

- `Intercept` the Request
- Send the request to `Repeater`
- Right Click -> `Engagement Tools` -> `Discover Content`

- Discover and identify URLS
- Choose the URL with the parameter you want
- Set Up `Burp Collaborator`
- Copy the payload and replace it with the value of the parameter you have taken
- Hit Send
- In the Burp Collaborator you will be able to see the logs.

SSRF Attacks against the server itself

Many times administrative functionality is available only to the authenticated users. When any attacker makes a request through a local machine, he gets access to administrative functionality. A request to local machine can be made using payloads such as `localhost`, `127.0.0.1` etc.

SSRF Attacks with black list based input filters

Developers often blacklist filters to block certain characters as input. Many times applications block hostnames like localhost and 127.0.0.1. Certain sensitive URLS like /admin are also blocked. A good alternative can be different types of SSRF Payloads. A list of SSRF Payloads can be found here .

In certain cases converting the IP address also works. This can be done using https://www.vultr.com/resources/ipv4-converter/

SSRF Attacks with white list based input filters

Some applications only allow input that matches, begins with, or contains, a whitelist of permitted values. In this situation, you can sometimes circumvent the filter by exploiting inconsistencies in URL parsing. In such a case

- You can embed credentials in a URL before the hostname, using the `@` character. For example: `https://expected-host@evil-host`.
- You can use the `#` character to indicate a URL fragment. For example: `https://evil-host#expected-host`.
- You can leverage the DNS naming hierarchy to place required input into a fully-qualified DNS name that you control. For example: `https://expected-host.evil-host`.

Severity

The severity of SSRF varies and depends on case to case basis.

Impact of SSRF

A successful SSRF attack can often result in unauthorized actions or access to data within the organization, either in the vulnerable application itself or on other back-end systems that the application can communicate with. In some situations, the SSRF vulnerability might allow an attacker to perform arbitrary command execution.

An SSRF exploit that causes connections to external third-party systems might result in malicious onward attacks that appear to originate from the organization hosting the vulnerable application.

In simple words, By this attack, an attacker can gather information about ports, IP addresses, Remote Code Execution (RCE), and can also discover the IP addresses of servers running behind a reverse proxy, etc.

Prevention of SSRF

1. **Whitelist Domains in DNS :** The easiest way to remediate SSRF is to whitelist any domain or address that your application accesses. Blacklisting and regex have the same issue, someone will eventually find a way to exploit them
2. **Do Not Send Raw Responses:** Never send a raw response body from the server to the client. Responses that the client receives need to be expected.
3. **Enforce URL Schemas:** Allow only URL schemas that your application uses. There is no need to have ftp://, file:/// or even http:// enabled if you only use https://. And if you do use other schemas make sure that they're only accessible from the part that needs to access them and not from anywhere else.
4. **Enable Authentication on All Services:** Make sure that authentication is enabled on any service that is running inside your network even if they don't require it. Services like memcached, redis, mongo and others don't require authentication for normal operations, but this means they can be exploited.

5. **Sanitize and Validate Inputs:** Never trust user input. Always sanitize any input that the user sends to your application. Remove bad characters, standardize input (double quotes instead of single quotes for example). After sanitization make sure to validate sanitized input to make sure nothing bad passed through.

References

- SSRF by Port Swigger : https://portswigger.net/web-security/ssrf
- SSRF by OWASP: https://owasp.org/www-community/attacks/Server_Side_Request_Forgery
- SSRF by Acunetix: https://www.acunetix.com/blog/articles/server-side-request-forgery-vulnerability/
- Port Scanning with SSRF: https://www.acunetix.com/blog/articles/ssrf-vulnerability-used-to-scan-the-web-servers-network

🧩 Lab Documentation