# IFS4103 Lab 4:
# Using Burp Suite’s Live Tasks, Intruder, Comparer & Decoder

## Objectives:

In Lab 4, you will look at Burp Suite’s **other modules/components**, including the powerful **Burp Intruder**. More specifically, you will perform the following:

1. To set up and launch Burp Suite’s **live tasks**;
2. To use Burp **Intruder** for ***fuzzing/brute-forcing*** field(s) in a request;
3. To use Burp **Comparer** for performing a ***visual “diff”*** between any two related data items, e.g. pairs of similar HTTP messages;
4. To use Burp **Decoder** for decoding and encoding of application data.

## Task 1: Setting up and Launching Burp Suite’s Live Tasks

As previously mentioned, Burp 2 UI is more **task oriented**. In Burp 2, you can thus set up and launch ***live tasks*** from Burp’s Dashboard tab.

***Live tasks*** are used to **process traffic** from specific Burp tools (e.g. Proxy, Repeater or Intruder), and **perform defined actions** on it, including conducting:

- A ***live audit***: which performs a vulnerability scan;
- A ***live passive crawl***: which adds entries to the Target’s site map.

To use live tasks in Burp Suite, please refer to the following YouTube video:

- “How to use live tasks in Burp Suite”: https://www.youtube.com/watch?v=Y6p5AhZjmFc

From observing the video, you should be able to use Burp Suite's live tasks, for instances, to perform the following:

- To **passive audit/scan** all traffic that passes through **Burp Proxy**, including URLs with a certain domain-name prefix if needed;
- To **passive crawl** all traffic with a certain domain-name prefix that passes through **Burp Proxy**;
- To **active audit/scan** only **in-scope items**.

You can additionally check the documentation about "Live scans" at: https://portswigger.net/burp/documentation/desktop/automated-scanning/live-tasks.

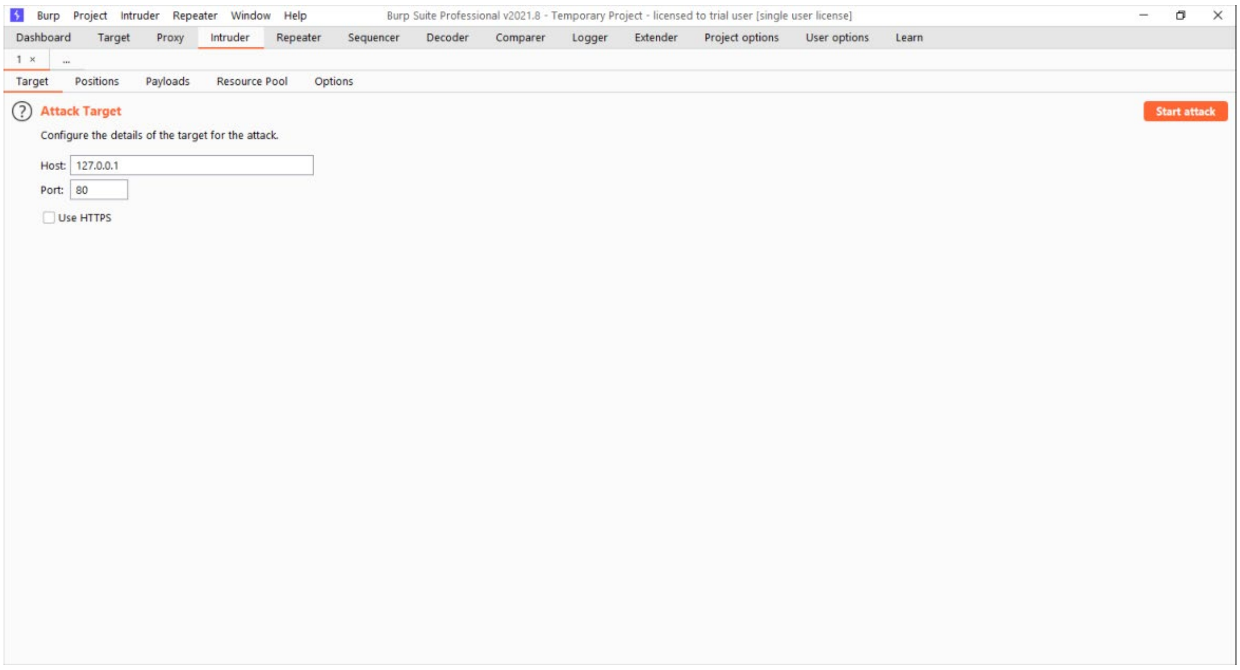
## Task 2: Using Burp Intruder

Burp Intruder is a **powerful tool** for carrying out ***automated customized attacks*** against target web applications. It is highly configurable and can be used to perform a **huge range of tasks**, including: account name enumeration, password guessing, web directory guessing, and even an active exploitation of complex blind SQL injection vulnerabilities.

In attacking a target web application, Burp Intruder basically works in a workflow with the **following stages**:

1. **Taking** a HTTP request (called the "***base request***");
2. **Modifying** the request in various systematic ways;
3. **Issuing** each modified version of the request; and
4. **Analyzing** the application's responses to identify interesting results/features.

Moe specifically, for each ***attack target*** (see Figure 1), you must first specify one or more ***payload sets*** (see Figure 2), and the ***positions*** in the base request where the payloads are to be placed/assigned (see Figure 3).


Burp Project Intruder Repeater Window Help
Burp Suite Professional v2021.8 - Temporary Project - licensed to trial user [single user license]
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Target Positions Payloads Resource Pool Options
Start attack

Attack Target

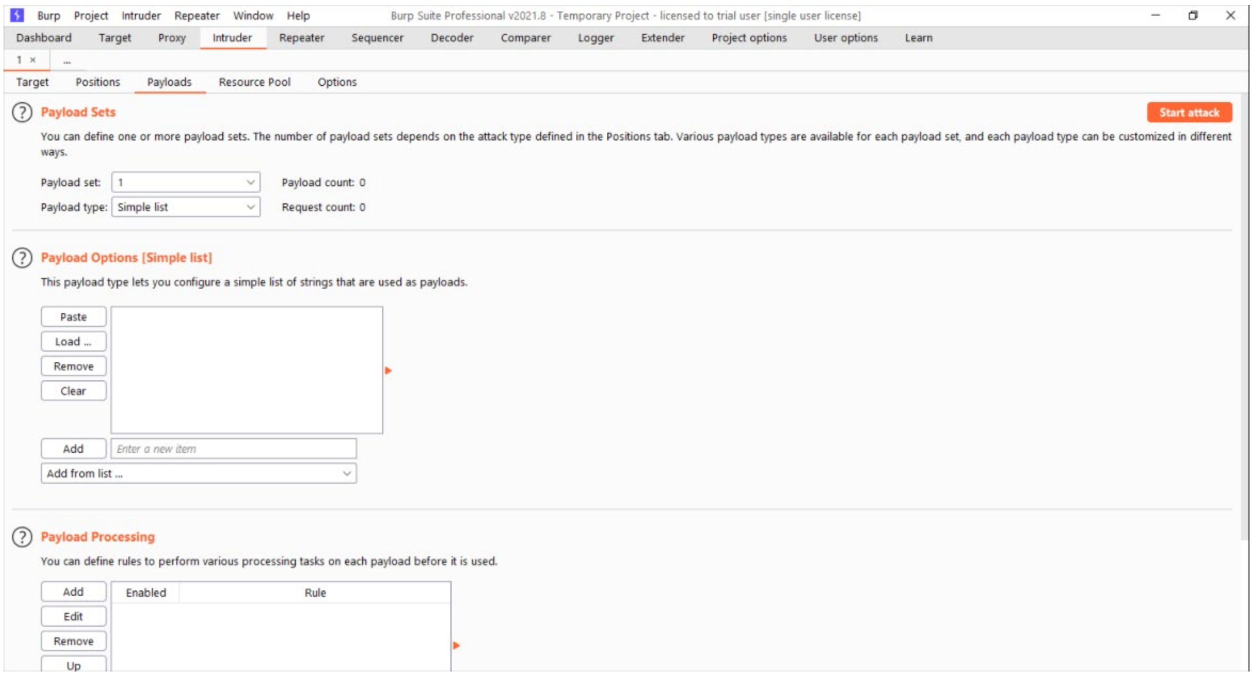
Configure the details of the target for the attack.

Host: 127.0.0.1

Port: 80

Use HTTPS

Figure 1: "Intruder – Target" Tab


Burp Project Intruder Repeater Window Help
Burp Suite Professional v2021.8 - Temporary Project - licensed to trial user [single user license]
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Target Positions Payloads Resource Pool Options
Start attack
Paste
Load ...
Remove
Clear
Add
Enter a new item
Add from list ...
Add
Edit
Remove
Up

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 0

Payload type: Simple list Request count: 0

Payload Options [Simple list]

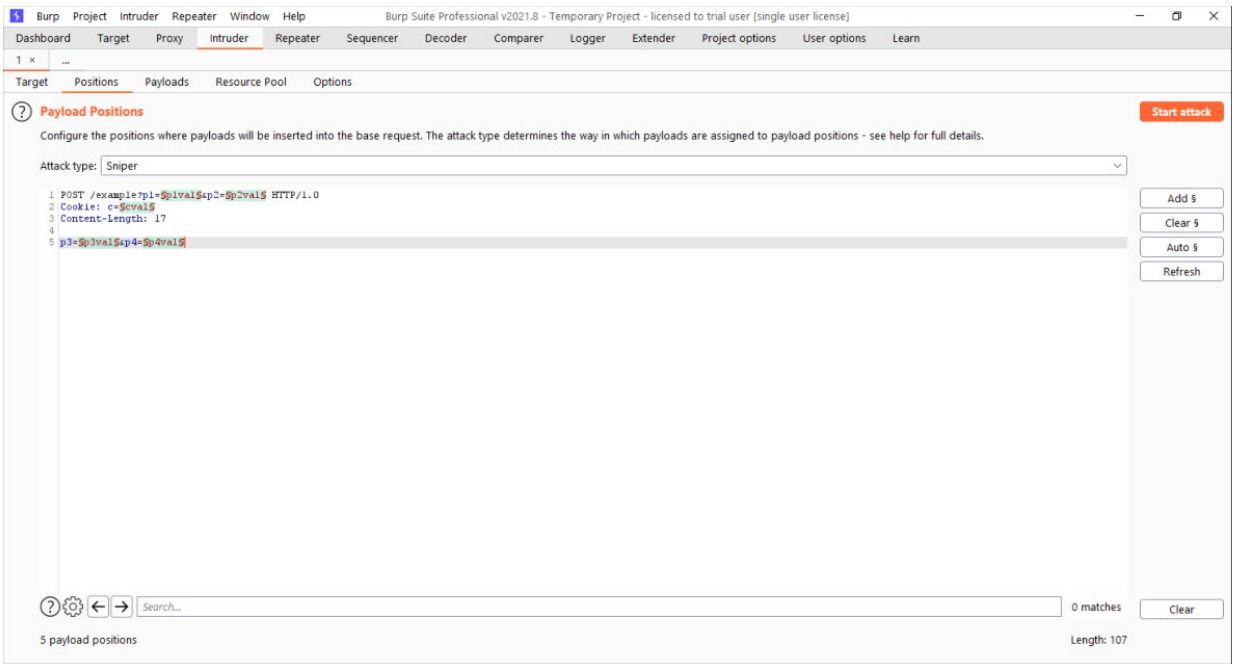
This payload type lets you configure a simple list of strings that are used as payloads.

Payload Processing

You can define rules to perform various processing tasks on each payload before it is used.

| Enabled | Rule |
|---|---|

Figure 2: "Intruder – Payloads" Tab


Burp Suite Professional v2021.8 - Temporary Project - licensed to trial user [single user license]
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Target Positions Payloads Resource Pool Options
Start attack
Add §
Clear §
Auto §
Refresh
Search...
0 matches
Clear

**Payload Positions**

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

```
POST /example?p1=§p1val§&p2=§p2val§ HTTP/1.0
Cookie: c=§cval§
Content-Length: 17

p3=§p3val§&p4=§p4val§
```

5 payload positions

Length: 107

Figure 3: "Intruder – Positions" Tab

The **positions** of the payloads in the base request can be determined by manually placing ***payload markers*** (see: Burp Intruder payload positions.)

When setting the **payloads**, numerous ***payload-generating methods*** are available, including: simple lists, numbers, dates, brute forcer, bit flipper (see: Burp Intruder payload types). The payloads generated, in fact, can be further **manipulated** using various ***payload processing rules*** and ***payload encoding techniques*** (see: Burp Intruder payload processing).

The specified **payloads** are then automatedly assigned to the defined payload **positions** using several following "***attack types***" (see: Burp Intruder attack types):

- ***Sniper***: This attack type uses a ***single*** set of payloads. It is useful for fuzzing a **payload position**, e.g. a parameter, within a request.
- ***Battering ram***: This attack type uses a ***single*** set of payloads. It is useful where an attack requires ***the same input value*** to be inserted simultaneously in ***multiple positions*** within a request (e.g. a username within a Cookie and a body parameter).

- ***Pitchfork***: This attack type uses ***multiple*** payload sets, with a ***different*** payload set for each defined position. It is useful where an attack requires ***different but related input*** to be inserted in ***multiple positions*** within a request (e.g. a username in one parameter, and a known ID number corresponding to that username in another parameter).
- ***Cluster bomb***: This attack type uses ***multiple*** payload sets, with a ***different*** payload set for each defined position. The attack iterates through each payload set in turn, so that **all payload combinations** are tested.
  It is useful where an attack requires ***different and unrelated/unknown*** input to be inserted in ***multiple positions*** within a request (e.g. when guessing credentials: a username in one parameter, a password in another parameter, and all username-password combinations need to be tested).

These different attack types can be **summarized** in the following tables:

| **Attack type** | **No of payload sets** | **No of payload positions** | **Assignment of payload sets to payload positions** |
|---|---|---|---|
| **Sniper** | single | single | 1 payload set to 1 position |
| **Battering ram** | single | multiple | 1 payload set to multiple positions *simultaneously* |
| **Pitchfork** | multiple | multiple | 1 payload set to each corresponding position; all payload entries *with the same index* in the payload sets are assigned simultaneously |
| **Cluster bomb** | multiple | multiple | 1 payload set to each corresponding position; test all *payload-entry combinations* |

Lastly, various tools are available to help ***analyze* the results** and even ***automatically identify*** interesting items/results for further investigation (see also: Analyzing attack results). For example, if a valid identifier returns a **different HTTP status code** or **response length**, you can ***sort*** the attack results on this attribute. Or if a valid identifier returns a response containing a **specific expression**, you can define a **"*match grep*"** item to pick out responses that match this expression

You can refer to PortSwigger's **documentation** about **Intruder** (https://portswigger.net/burp/documentation/desktop/tools/intruder) to see how Intruder can be utilized to perform the following **most common use cases**:

- Enumerating identifiers;
- Harvesting useful data;
- Fuzzing for vulnerabilities.

Additionally, you can refer to the following **YouTube videos** about using Burp Intruder with **different available attack modes**:

- "Intruder Tool - **Sniper** Mode": https://www.youtube.com/watch?v=QLDk5zI2cdM

- "Intruder Tool - **Battering Ram** Mode": https://www.youtube.com/watch?v=NPVddIPYn6M

- "Intruder Tool - **Pitchfork** Mode": https://www.youtube.com/watch?v=iG7003AC8ys

- "Intruder Tool - **Cluster** Mode": https://www.youtube.com/watch?v=ehGsDQbMXn8

## Task 3: Using Burp Comparer

**Burp Comparer** is a handy utility for performing a ***visual-based differential analysis*** between any two related data items, such as pairs of similar HTTP messages. Using Comparer, you can thus perform the following **common uses**:

- When looking for **username enumeration conditions**, you can compare responses to failed logins using valid and invalid usernames, looking for subtle differences in the responses.
- If you have very **large responses** with different lengths than your base response, you can compare these to quickly see where the differences lie.

You can refer to the following YouTube **video** about using Burp Comparer:

- "Comparer Tool": https://www.youtube.com/watch?v=lT56Z54K-Jo

You can additionally check PortSwigger's **documentation** on Comparer.

## Task 4: Using Burp Decoder

**Burp Decoder** is a useful tool for performing *manual or intelligent* **decoding and encoding** of application data. It can transform encoded data into its canonical form, or transform raw data into various encoded and hashed forms. In addition, with its **smart decode** feature, it is capable of intelligently recognizing several encoding formats using **heuristic techniques**.

You can refer to the following YouTube **video** about using Burp Decoder:

- "Decoder Tool": https://www.youtube.com/watch?v=KlWZ5pKg-PM

You can additionally check PortSwigger's **documentation** on Decoder.