DAY3_advent_of_the_cyber

Learning Objectives

- Understanding Authentication
- Understand the use of default credentials and why they're dangerous
- Bypass a login form using BurpSuite

Authentication

Authentication is a process of verifying a users' identity, normally by credentials (such as a username, user id or password); to put simply, authentication involves checking that somebody really is who they claim to be. Authorization (which is fundamentally different to authentication, but often used interchangeably) determines what a user can and can't access; authorization is covered in tomorrow walkthrough, today's task focuses on authentication and some common flaws.

Default Credentials

You've probably purchased (or downloaded a service/program) that provides you with a set of credentials at the start and requires you to change the password after it's set up (usually these credentials that are provided at the start are the same for every device/every copy of the software). The trouble with this is that if it's not changed, an attacker can look up (or even guess) the credentials.

What's even worse is that these devices are often exposed to the internet, potentially allowing anyone to access and control it. In 2018 it was reported that a botnet (a number of internet-connected devices controlled by an attacker to typically perform DDoS attacks) called Mirai took advantage of Internet of Things (IoT) devices by remotely logging, configuring the device to perform malicious attacks at the control of the attackers; the Mirai botnet infected over 600,000 IoT devices mostly by scanning the internet and using default credentials to gain access.

In fact, companies such as Starbucks and the US Department of Defense have been victim to leaving services running with default credentials, and bug hunters have been rewarded for reporting these very simple issues responsibly (Starbucks paid $250 for the reported issue):

- https://hackerone.com/reports/195163 - Starbucks, bug bounty for default credentials.
- https://hackerone.com/reports/804548 - US Dept Of Defense, admin access via default credentials.

In 2017, it was reported that 15% of all IoT devices still use default passwords.

SecLists is a collection of common lists including usernames, passwords, URLs and much more. A password list known as "rockyou.txt" is commonly used in security challenges, and should definitely be a part of your security toolkit.

Dictionary Attacks using BurpSuite

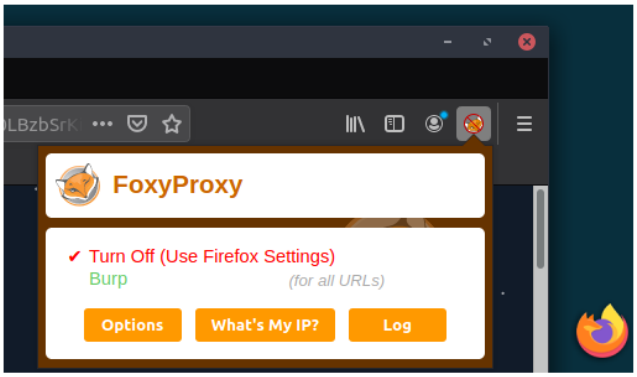
A dictionary attack is a method of breaking into an authenticated system by iterating through a list of credentials. If you have a list of default (or the most common) usernames and passwords, you can loop through each of them in hopes that one of the combinations is successful.

You can use a number of tools to perform a dictionary attack, one notable one being Hydra (a fast network logon cracker) and BurpSuite, an industry-standard tool used for web application penetration testing. Given day 3 is about web exploitation, we'll show you how to use BurpSuite to perform a dictionary attack on a web login form.

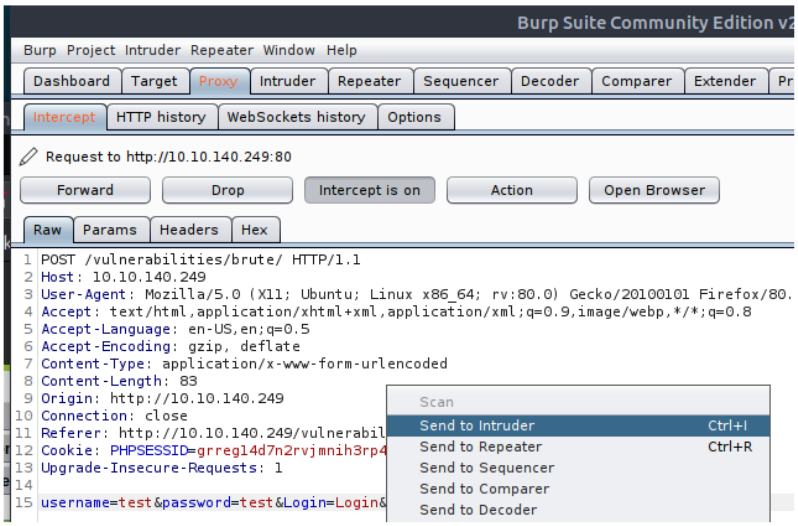
To download BurpSuite click here, otherwise, BurpSuite is pre-installed on our web-based AttackBox.

1. Start BurpSuite, you can do this on the AttackBox by clicking BurpSuite logo in the icon tray.

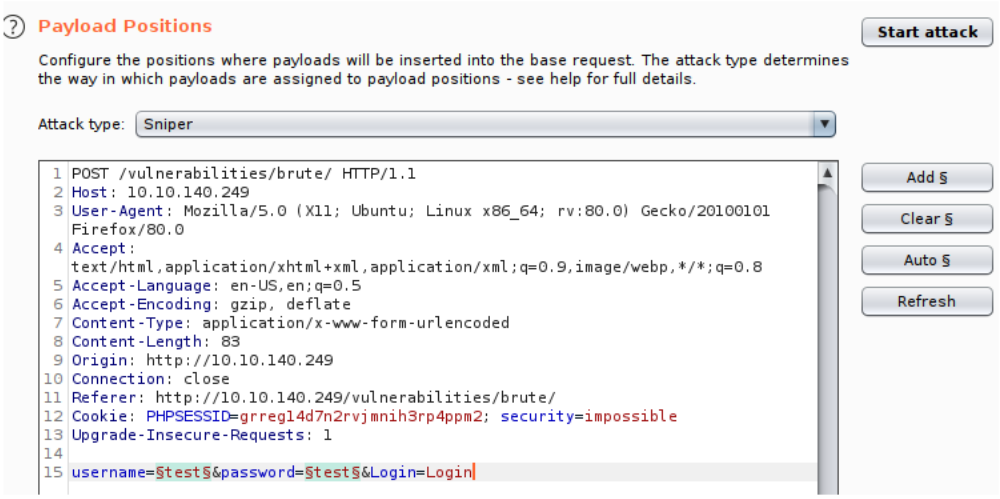
2. Once this has loaded, you want to "Intercept" your traffic by proxying it through the BurpSuite, which will then forward the request to the intended destination (in our case it will be a website) This will give you the ability to analyse and modify your browsers traffic.
    1. This example uses the AttackBox, and makes proxying traffic to BurpSuite easy (if you're using BurpSuite on your own machine, click here) to see how to proxy traffic to BurpSuite). On the AttackBox, open Firefox, click on the FoxyProxy browser extension, and select "Burp" - this will now proxy your traffic to BurpSuite.



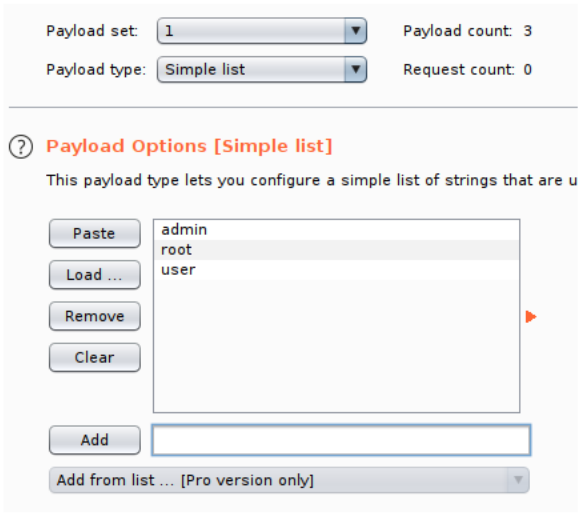
    2. Go to the BurpSuite application and click the Proxy tab, then click the button "Intercept is on".
3. Navigate to your chosen website, as you're intercepting your traffic, you will see BurpSuite has held your request and will not forward it on until you tell it to. Let's go to our web application and submit your details into a given form, in our case its a generic login form.
4. This captured request will show up in the Proxy tab. Right-click it, and click "Send to Intruder"; BurpSuite has a lot of functionality to repeat modify and manipulate requests, Burp Intruder is a tool to automate customize web attacks. We will use intruder to loop through and submit a login request using a list of default credential, in the hopes that one of the usernames and passwords in the list is correct.



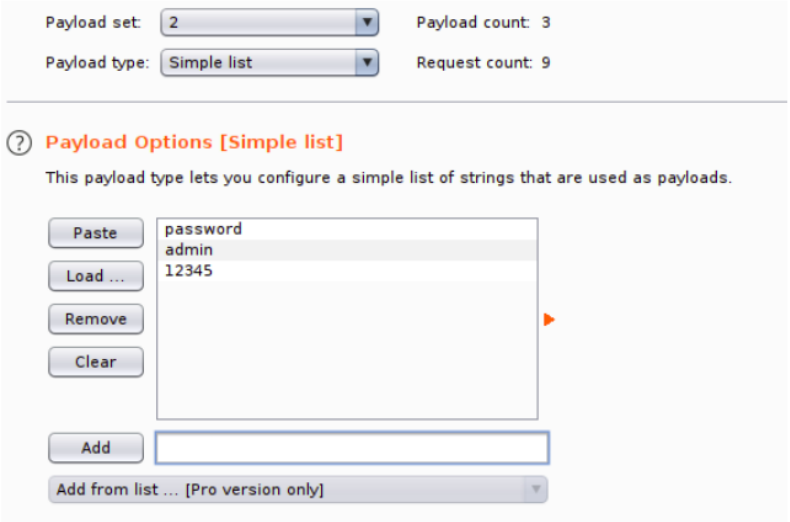
5. Go to the Intruder tab, you should see your request. Here we will insert "positions" (telling Burp which fields to update when automating a request), select a list per position and start the attack.
    1. Click the **"Positions"** tab, and clear the pre-selected positions.
    2. Add the username and password values as positions (highlight the text and click "Add")
    3. **Select "Cluster Bomb" in the Attack type dropdown menu**; this attack type iterates through each payloads sets in turn, so every combination of each set is tested.



6. We're going to tell each "Position" which Payload to use. In our example, we will select a list of usernames for the username field and a list of passwords for the password field.
    1. Click the "Payloads" tab, select your Payload set (set 1 is the username field, set 2 is the password field) and add select your list in the "Payload Options" section (or manually add entries).
    2. For set 1 (username), we will add a few common default username entries such as "admin", "root" and "user"



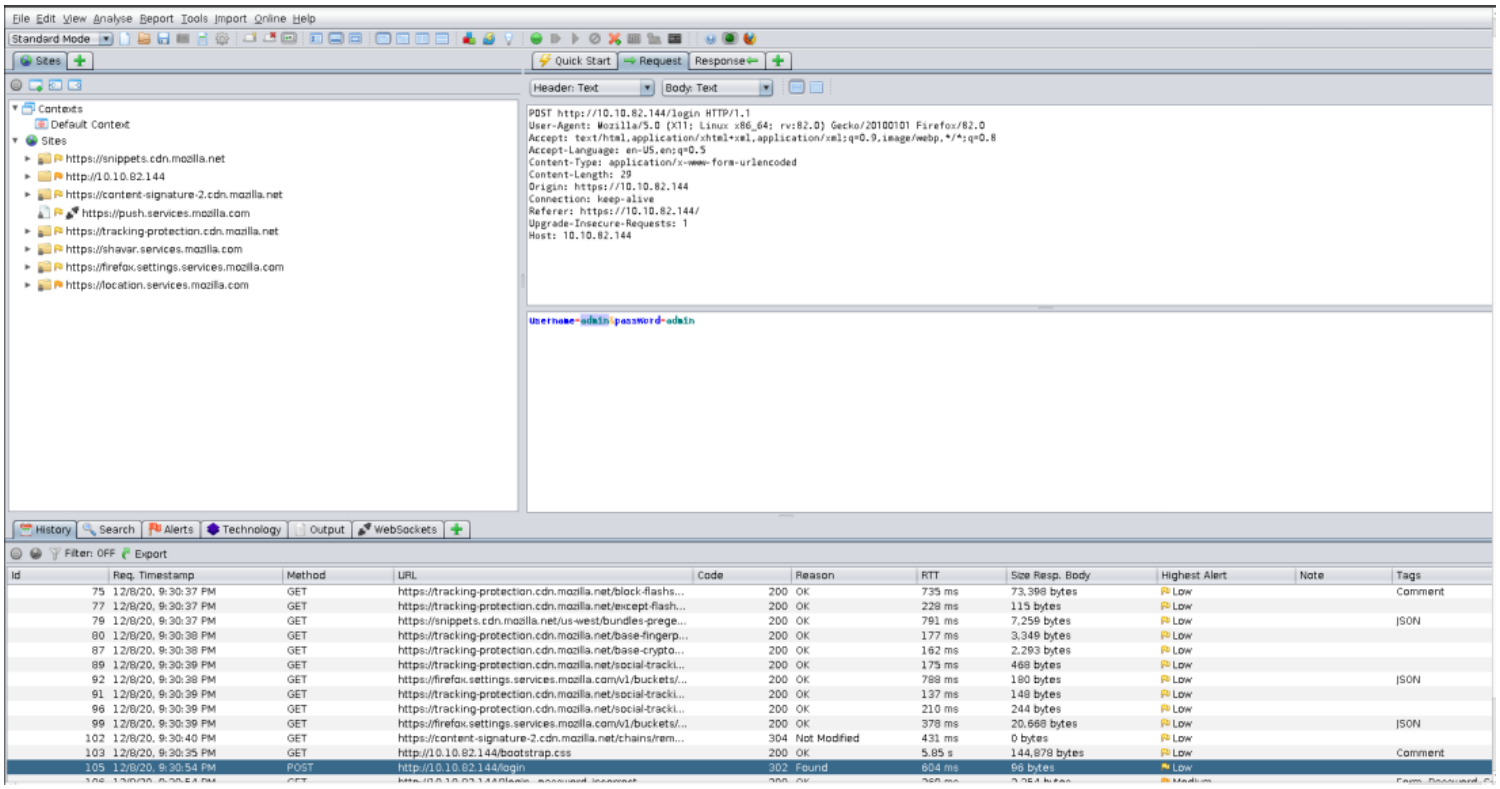
    3. For set 2 (password), we will add a few common default passwords such as "password", "admin" and "12345"



7. Click the "Start Attack" button, this will loop through each position list in every combination. You can sort by the "Length" or "Status" to identify a successful login (*typically all incorrect logins will have the same status or length, if a combination is correct it will be different.*

Though this challenge has to be done preferably using

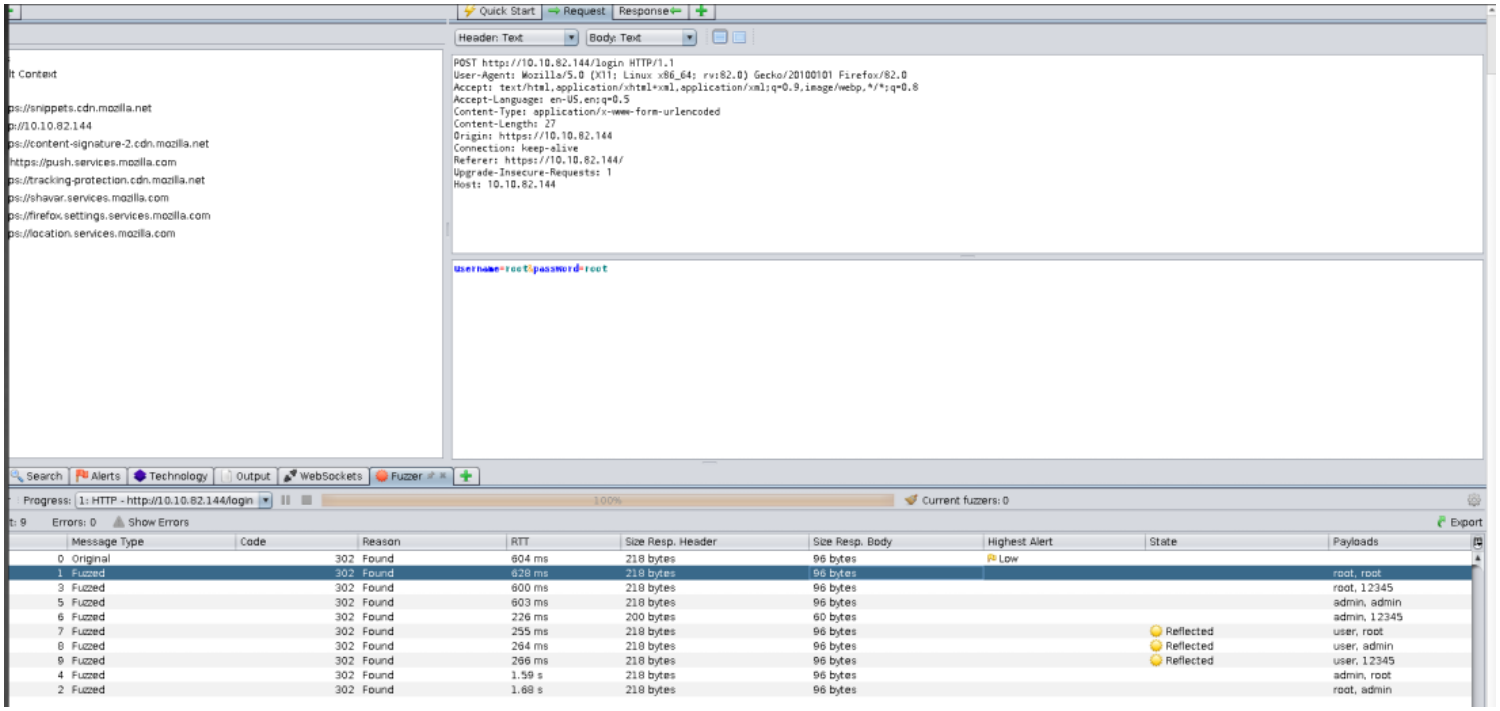
BurpSuite, I will use OWASP ZAP(due to various limitations in community edition of BurpSuite).

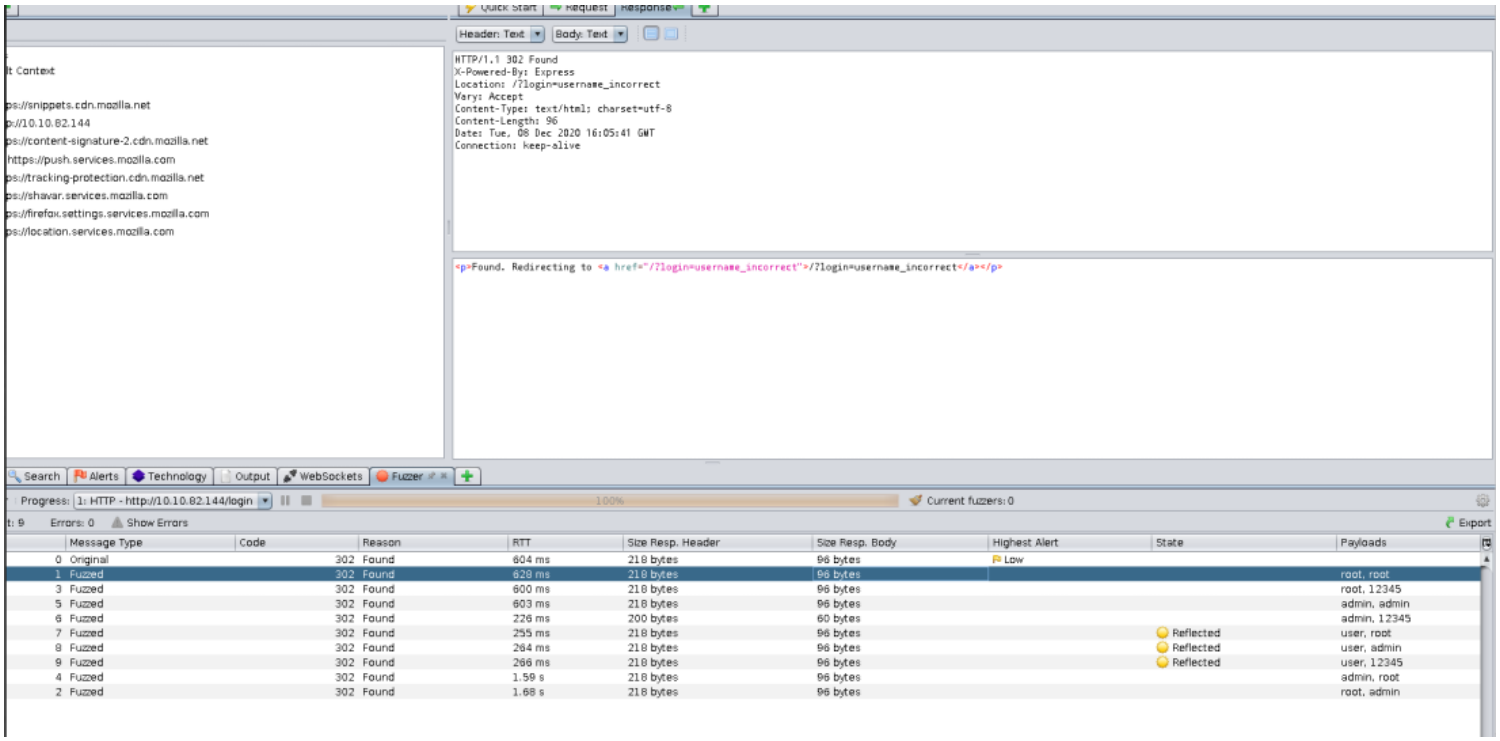


on seeing the request tab in above screenshot, we can see the attempted “username” and “password”. we will now use fuzzer(after highlighting and adding string/list one by one for each) to try various username and passwords combinations.

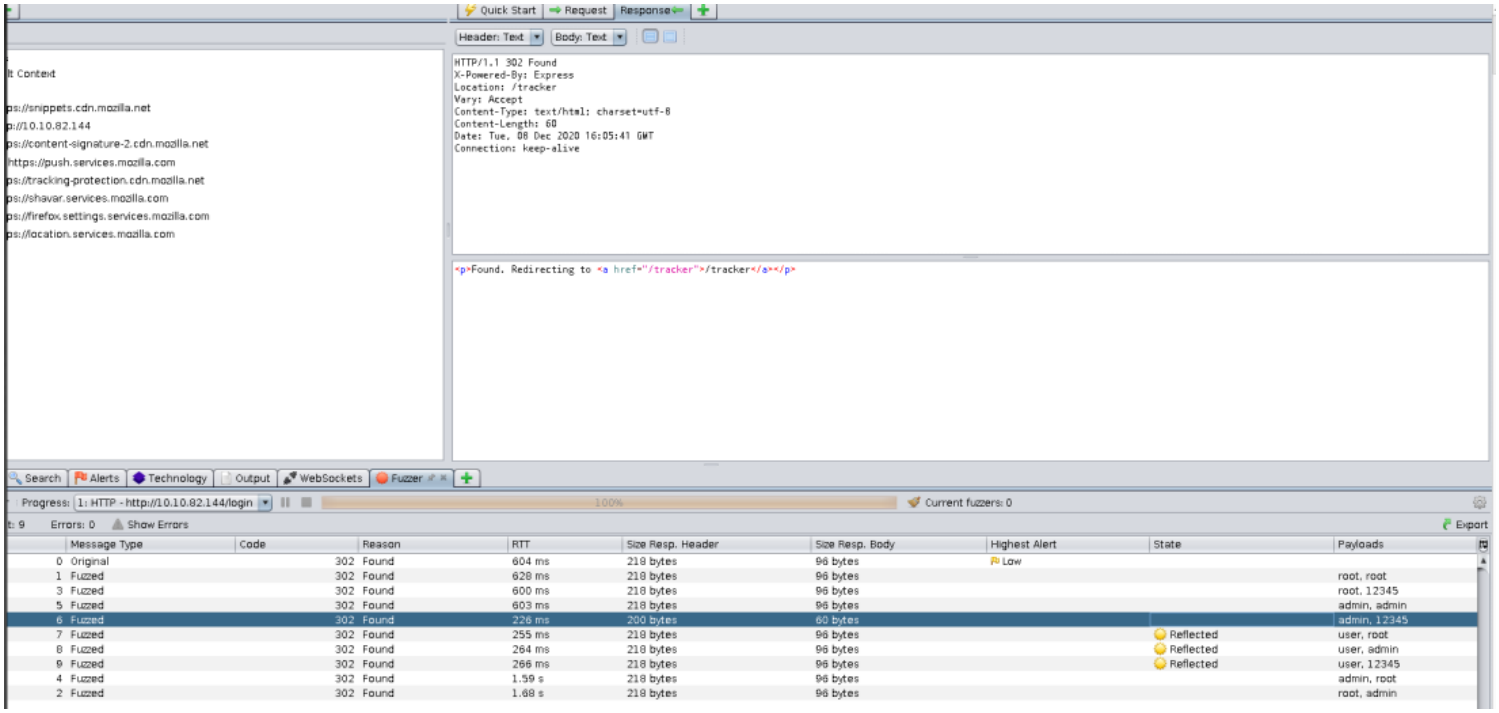
To get the correct combination we can see ‘response size’, ‘state’ , etc. Here all response sizes are 96 bytes , except one(60 bytes) .

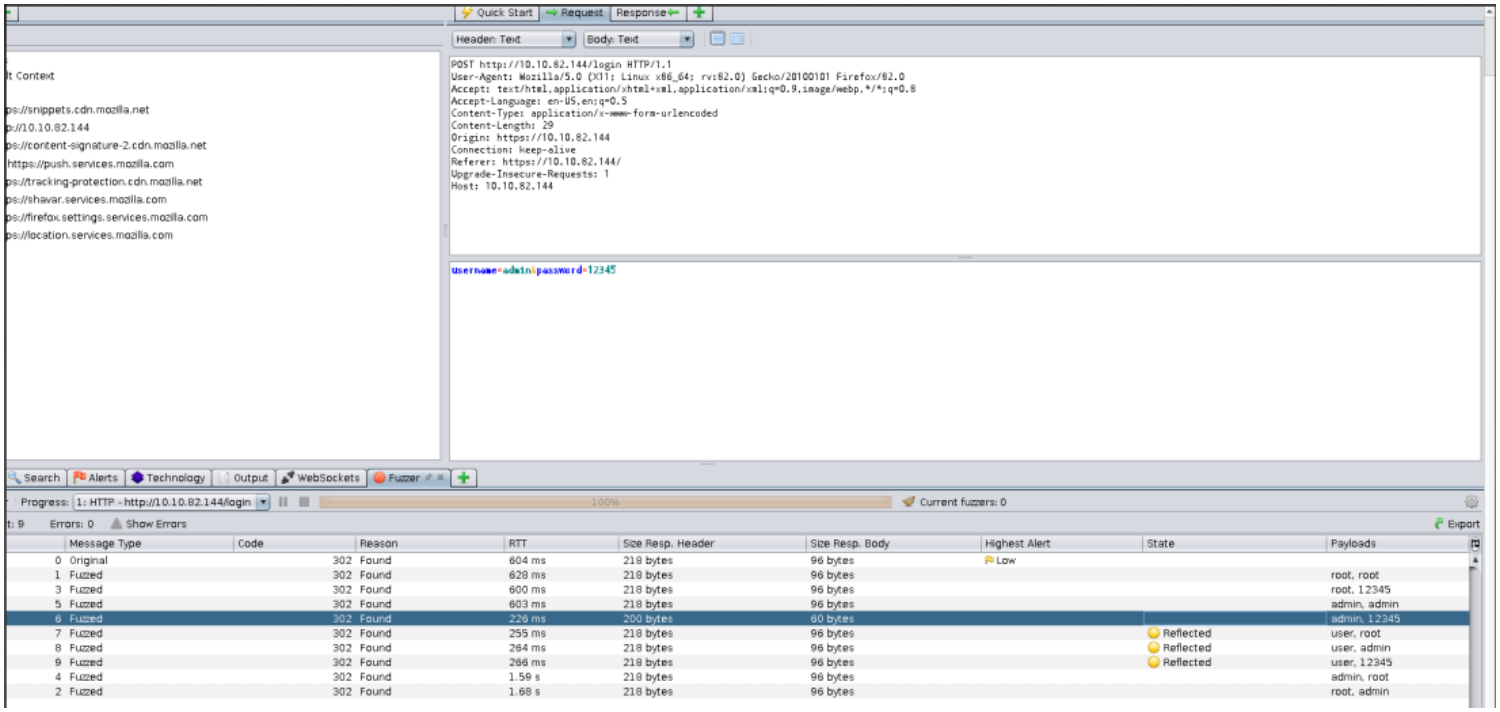
On seeing a response with size of 96 bytes, we can see the response to reach the conclusion that these combinations are not the correct ones .





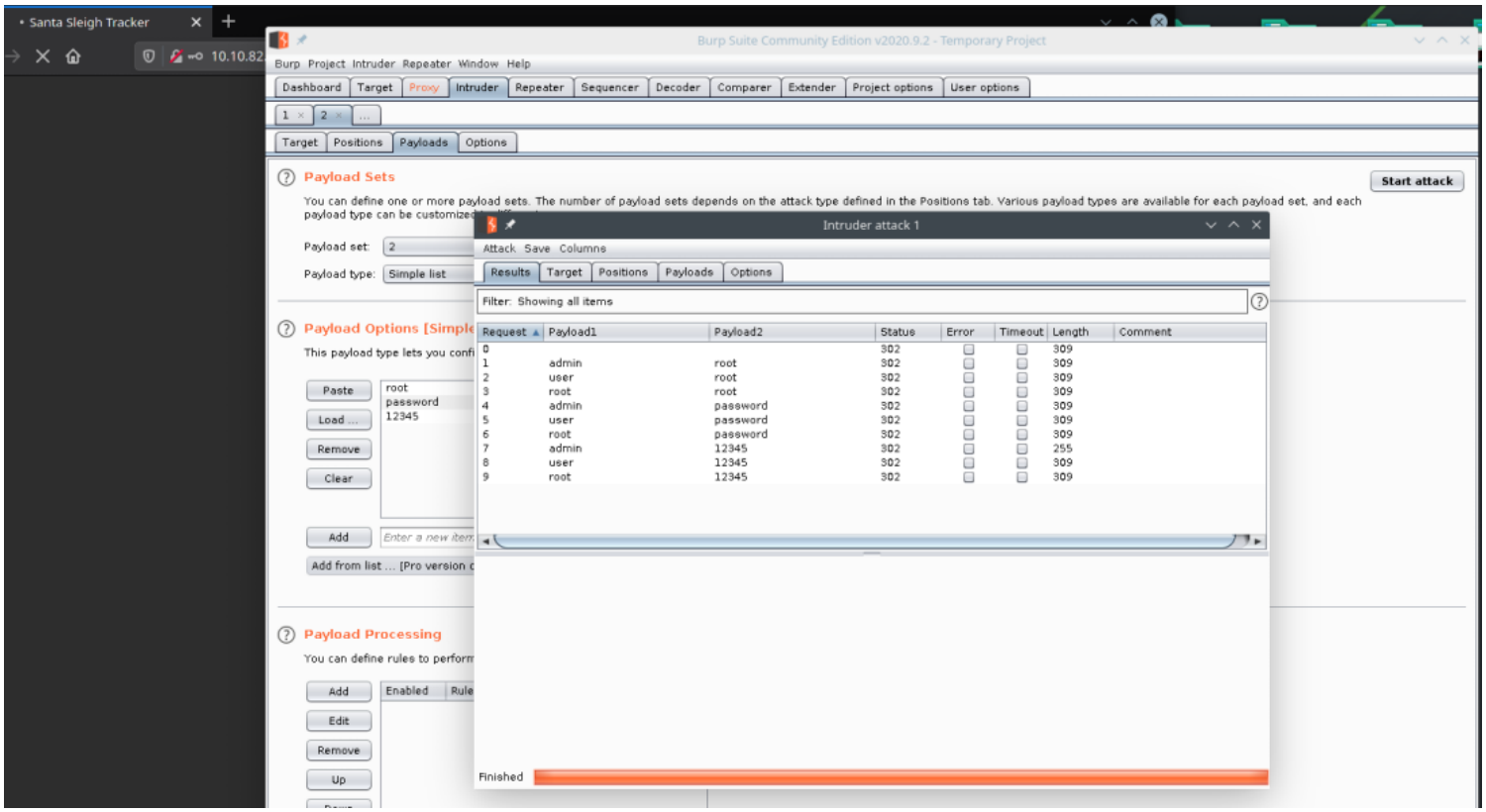
For the combination with the response size of 60 bytes,



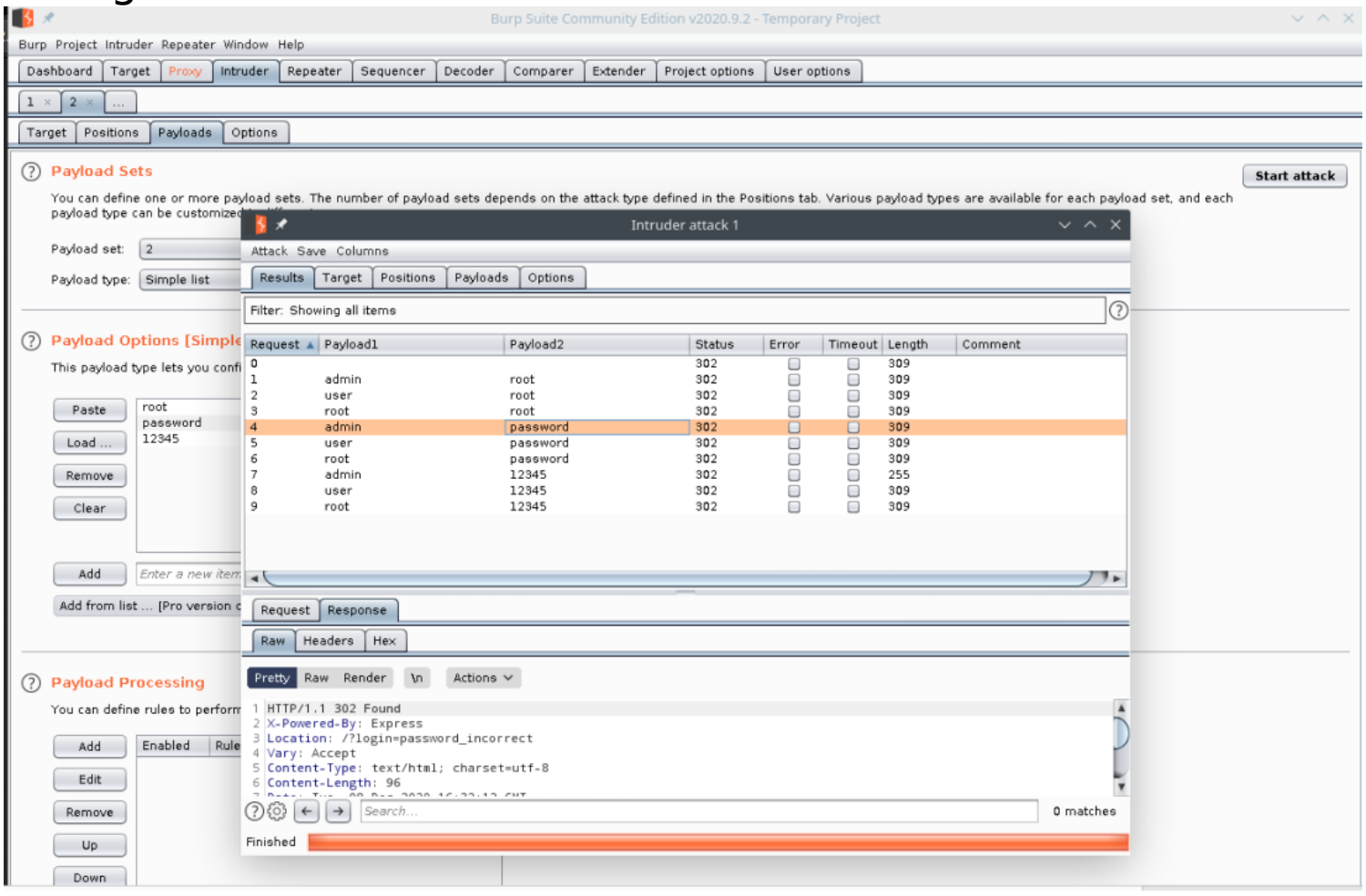


so , we have a correct combination as we can see in response .

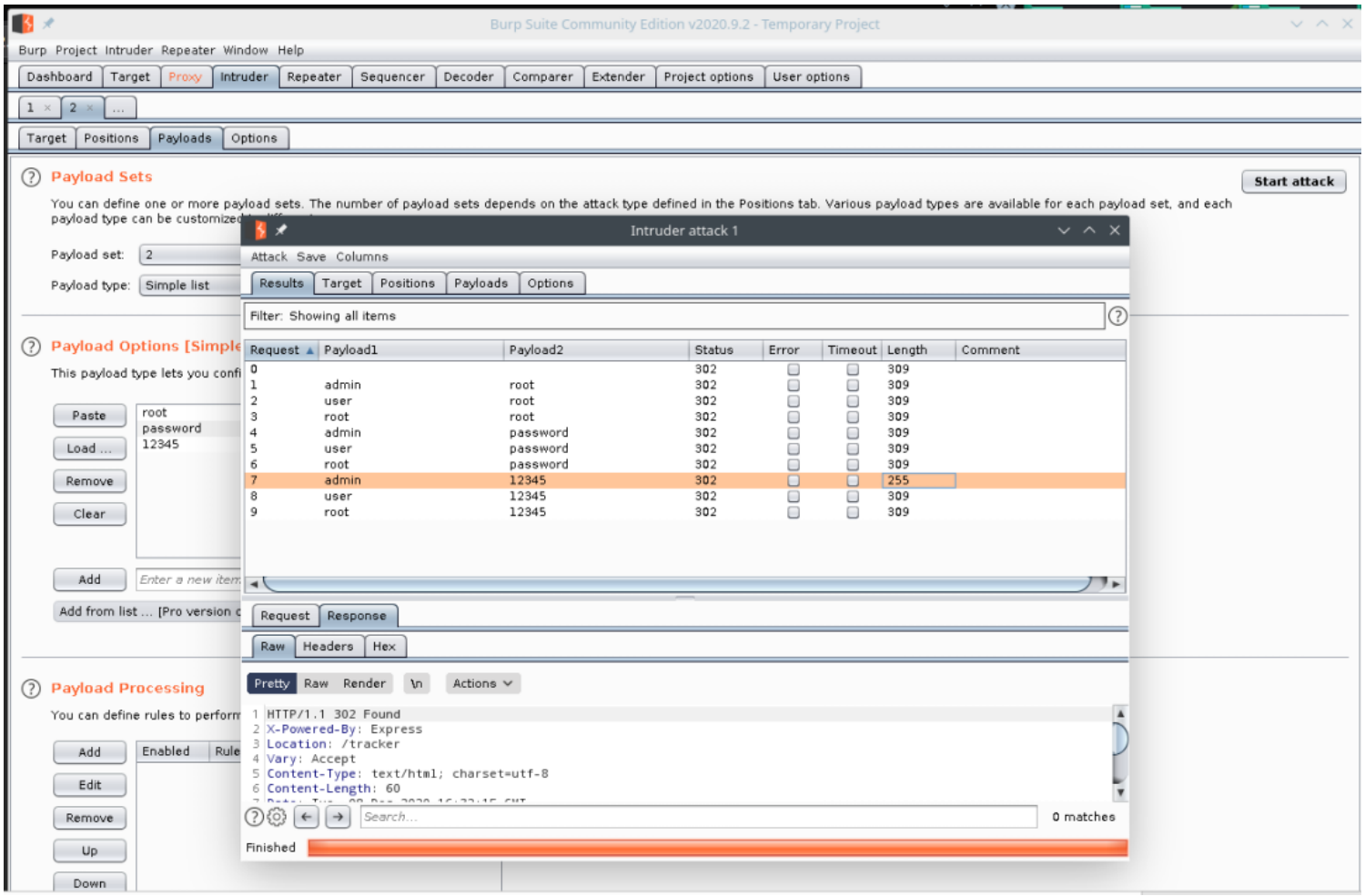
Doing the same thing we did above(using burpsuite):



wrong combination:



right combination:


Burp Suite Community Edition v2020.9.2 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Target Positions Payloads Options
Payload Sets
Start attack
Payload set: 2
Payload type: Simple list
Payload Options [Simple
Paste
Load ...
Remove
Clear
Add
Add from list ... [Pro version
root
password
12345
Payload Processing
Add
Edit
Remove
Up
Down
Enabled
Intruder attack 1
Attack Save Columns
Results Target Positions Payloads Options
Filter: Showing all items
Request Payload1 Payload2 Status Error Timeout Length Comment
0 302 309
1 admin root 302 309
2 user root 302 309
3 root root 302 309
4 admin password 302 309
5 user password 302 309
6 root password 302 309
7 admin 12345 302 255
8 user 12345 302 309
9 root 12345 302 309
Request Response
Raw Headers Hex
Pretty Raw Render \n Actions
1 HTTP/1.1 302 Found
2 X-Powered-By: Express
3 Location: /tracker
4 Vary: Accept
5 Content-Type: text/html; charset=utf-8
6 Content-Length: 60
Search...
0 matches
Finished