INFORMATION DISCLOSURE

(Owasp ref.: Security Misconfiguration)

Disclaimer:

This is just for an educational purpose and as a method of cyber security awareness. I highly recommend not to harm any institution/Organization without proper permissions and ROE.

Author:

Hey guys, I am vijay reddy. An enthusiastic guy who loves to explore and learn more about cyber security and other fields. I have worked on both INFRA and application security.

Inspiration:

We have many things to learn and knowledge to share.

Art of Thanks:

Thanks to my friends and family.

Bibliography:

Port swigger (https://portswigger.net/)

Hacksplanning (https://www.hacksplaining.com/prevention/information-leakage)

Tool: Burp suite

Information disclosure vulnerabilities

When a website unintentionally reveals sensitive information to its users. Depending on the context, websites may leak all kinds of information to a potential attacker, including:

- Data about other users, such as usernames or financial information.
- Sensitive commercial or business data.
- Technical details about the website and its infrastructure.

Examples of information disclosure?

- Revealing the names of hidden directories, their structure, and their contents via a robots.txt file or directory listing.
- Providing access to source code files via temporary backups
- Explicitly mentioning database table or column names in error messages
- Unnecessarily exposing highly sensitive information, such as credit card details
- Hard-coding API keys, IP addresses, database credentials, and so on in the source code
- Hinting at the existence or absence of resources, usernames, and so on via subtle differences in application behaviour

What leads to information disclosure?

- Failure to remove internal content from public content. – Dev Comments
- Insecure configuration of the website and related technologies. – Debug / verbose
- Flawed design and behaviour of the application. - Error

Impact:

- Impact increases when it is available to person who can exploit this.
- When a zero-day vulnerability is discovered, hackers will immediately try to find a way to exploit it. If your site leaks information about the technology you use, you could well become subject to automated attacks.

Severity: Depend on type of data been disclosed.

Prevention:

- Developer must be aware of this
- Audit code as a part of QA
- Use generic error message instead of more descriptive or informational.
- Double-check if debug and diagnostic is disabled.
- Study of used 3rd party apps.
- Sanitize Data Passed to the Client

Identification:

High-level techniques and tools that you can use to help identify information disclosure vulnerabilities during testing.

- Fuzzing
- Using Burp Scanner
- Using Burp's engagement tools
- Engineering informative responses

Fuzzing

Try submitting unexpected data types and specially crafted fuzz strings

Automate much of this process using tools such as Burp Intruder.

- Add payload positions to parameters and use pre-built wordlists of fuzz strings to test a high volume of different inputs in quick succession.
- Easily identify differences in responses by comparing HTTP status codes, response times, lengths, and so on.
- Use grep matching rules to quickly identify occurrences of keywords, such as error, invalid, SELECT, SQL, and so on.
- Apply grep extraction rules to extract and compare the content of interesting items within responses.

You can also use the Logger++ extension, available from the BApp store. In addition to logging requests and responses from all of Burp's tools, it allows you to define advanced filters for highlighting interesting entries. This is just one of the many Burp extensions that can help you find any sensitive data that is leaked by the website.

- Burp Scanner (Alert for pvt key, email, CC no in response)
- Burp engagement tool (Simply right click the request/proxy)
- Search (expression for fine-tune with adv search option)
- Find comments
- Discover content (Add content & functionality which is not linked from website visible content)
- Engineering informative responses (Study error msg)

Common sources of information disclosure

- Files for web crawlers (/robots.txt & /sitemap.xml)
- Directory listings (list directories that do not have an index page present)
- Developer comments (hidden dir. or app logic)
- Error messages (Ver disclosure)
- Debugging data (Key, Hostname, File, encryption method)
- User account pages (Accessing other user data through IDOR e.g.?user=carlos)
- Backup files (Source code, logic, Hard coded key. Check ~ or .bak)
- Insecure configuration (TRACE)
- Version control history (/.git)

==========

LAB 1: Information disclosure in error messages

==========

There was a functionality in website to view full data of a product by simply clicking on it.

When reviewing the URL found that the product is mapped/retrieved with product id(integer).

Then I tried some special character, which then encountered an internal server error revealing the information about the technology and version used.

As seen above information about the technology and version are disclosed.

This helps an attacker to find and try known exploit.

They are more dangerous if older or vulnerable versions are running on.

Cause: Improper error handling.

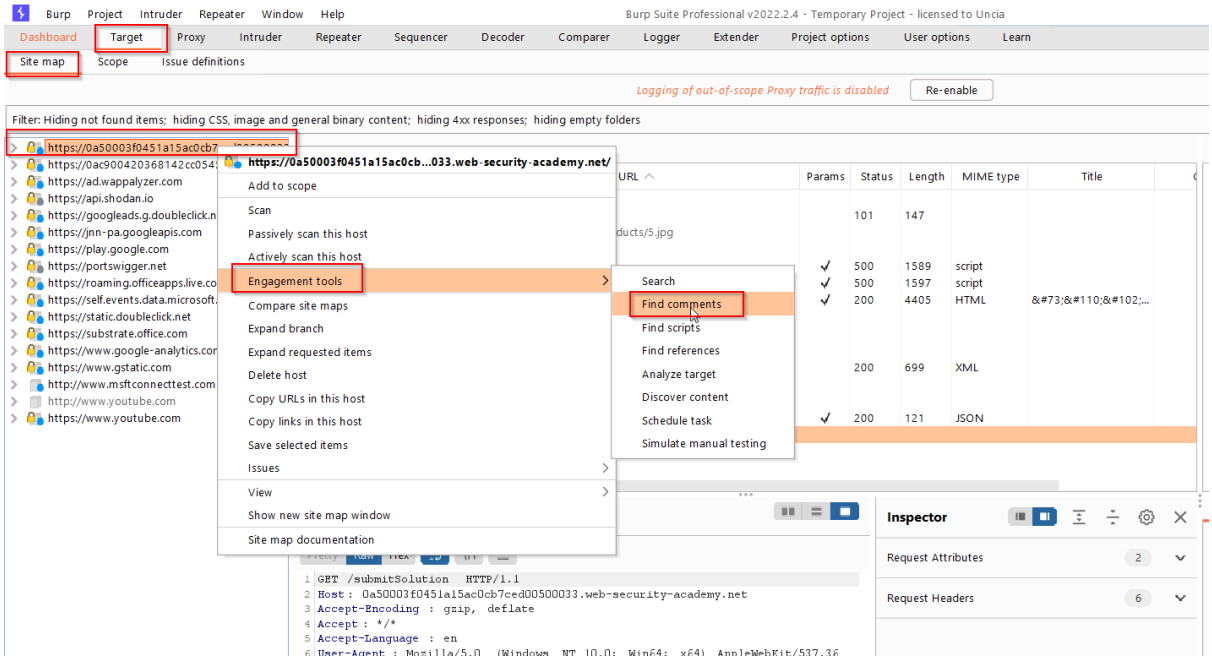
==========

LAB 2: Information disclosure on debug page

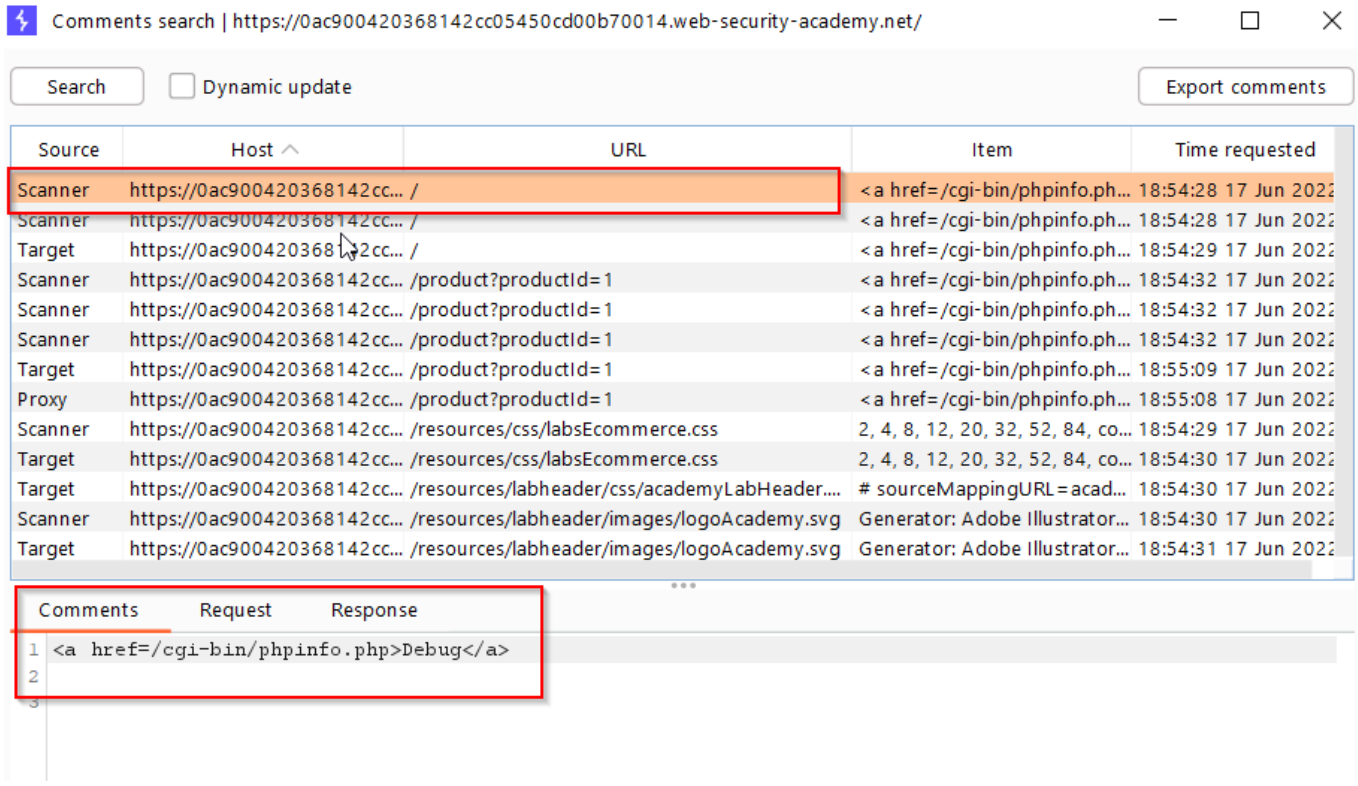
==========

At the time of development and testing many of the developers add comments to understand and mark the functionality also when troubleshooting or debugging a code.

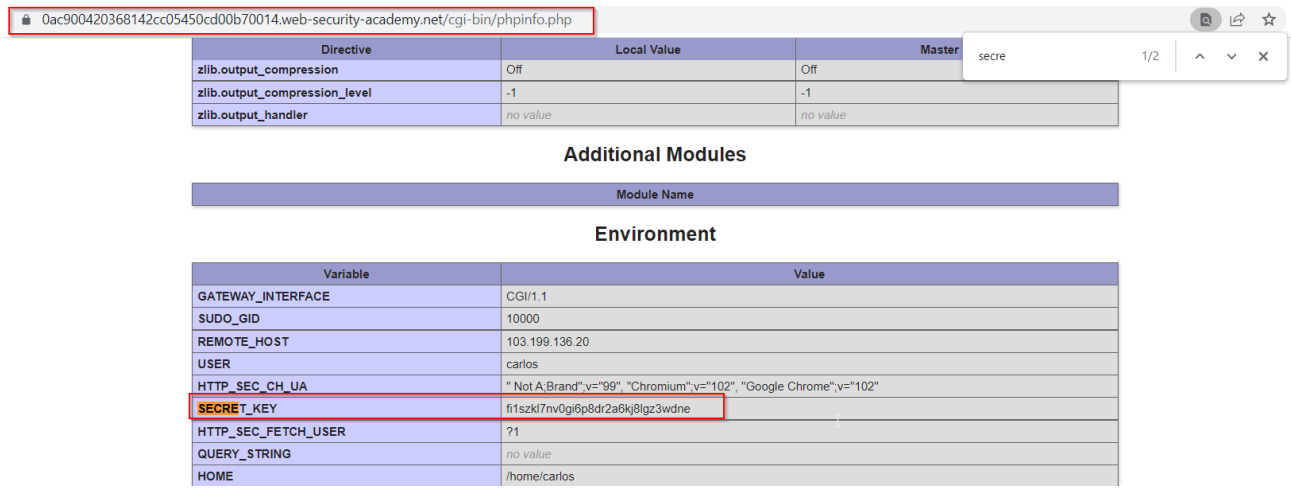
So, to find such comments. I have used “find comments” option available under section “Engagement tools” of burpsuite.



Post search found a debug path with phpinfo.php file location.



Output of /phpinfo.php



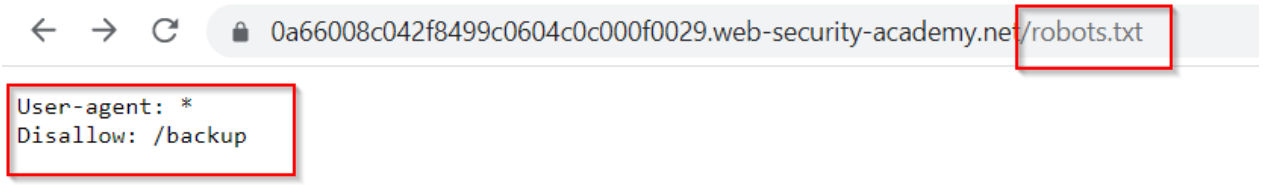
It contains sensitive information of servers on which this website is hosted also reveals the crucial data such as secret_key and others.

==========

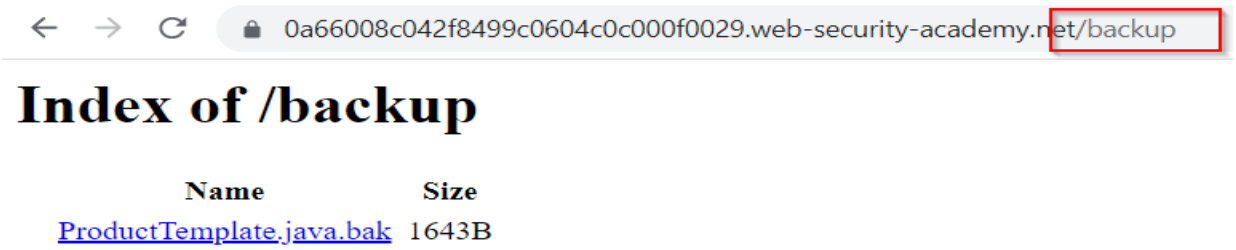
LAB 3: Source code disclosure via backup files

==========

As a test case tried /robots.txt to find the allowed path.
Found /backup as shown below.



Show simply I changed the URL to /backup. Which leads to other vulonerability directory listing and found one backup file (.bak)



When clicked on it redirects to another URL. Which was revealing the whole source code which can be used to analyse the code and found the loop wholes.

0a66008c042f8499c0604c0c000f0029.web-security-academy.net/backup/ProductTemplate.java.bak

```
    private transient Product product;

    public ProductTemplate(String id)
    {
        this.id = id;
    }

    private void readObject(ObjectInputStream inputStream) throws IOException, ClassNotFoundException
    {
        inputStream.defaultReadObject();

        ConnectionBuilder connectionBuilder = ConnectionBuilder.from(
                "org.postgresql.Driver",
                "postgresql",
                "localhost",
                5432,
                "postgres",
                "postgres",
                "t89bwymk1ot1aiztofmmtqdmb90w9scw"
        ).withAutoCommit();
        try
        {
            Connection connect = connectionBuilder.connect(30);
            String sql = String.format("SELECT * FROM products WHERE id = '%s' LIMIT 1", id);
            Statement statement = connect.createStatement();
            ResultSet resultSet = statement.executeQuery(sql);
            if (!resultSet.next())
            {
                return;
            }
            product = Product.from(resultSet);
        }
        catch (SQLException e)
        {
            throw new IOException(e);
        }
    }

    public String getId()
    {
        return id;
    }

    public Product getProduct()
    {
        return product;
    }
}
```

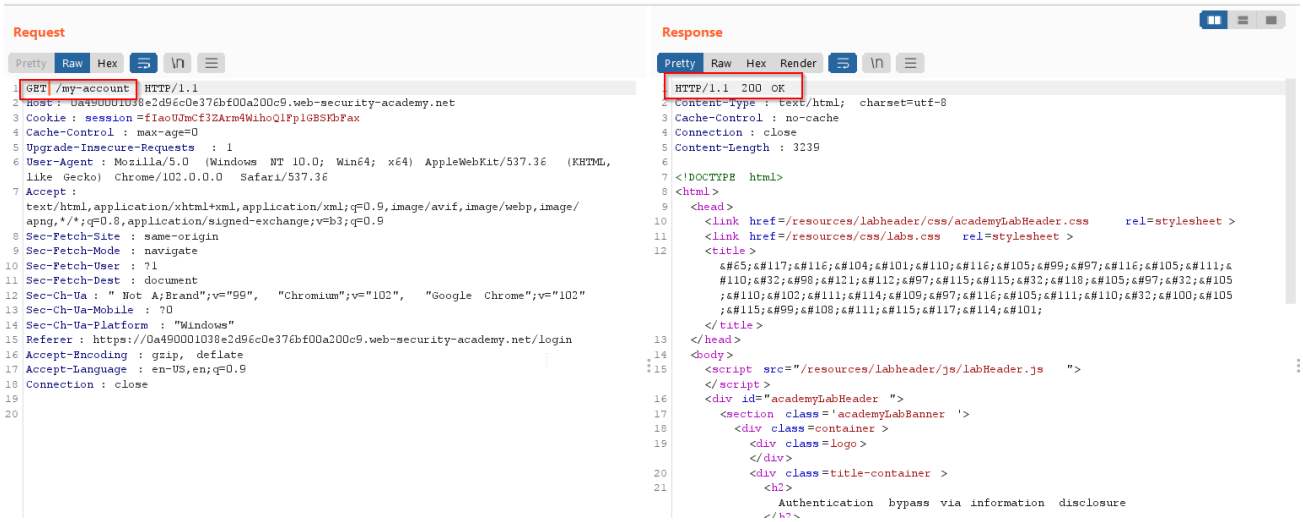
Found hard coded secret key and other parameter used for building a SQL connection also able to find inline SQL query used to fetch product details using product id from table named “products”.

==========

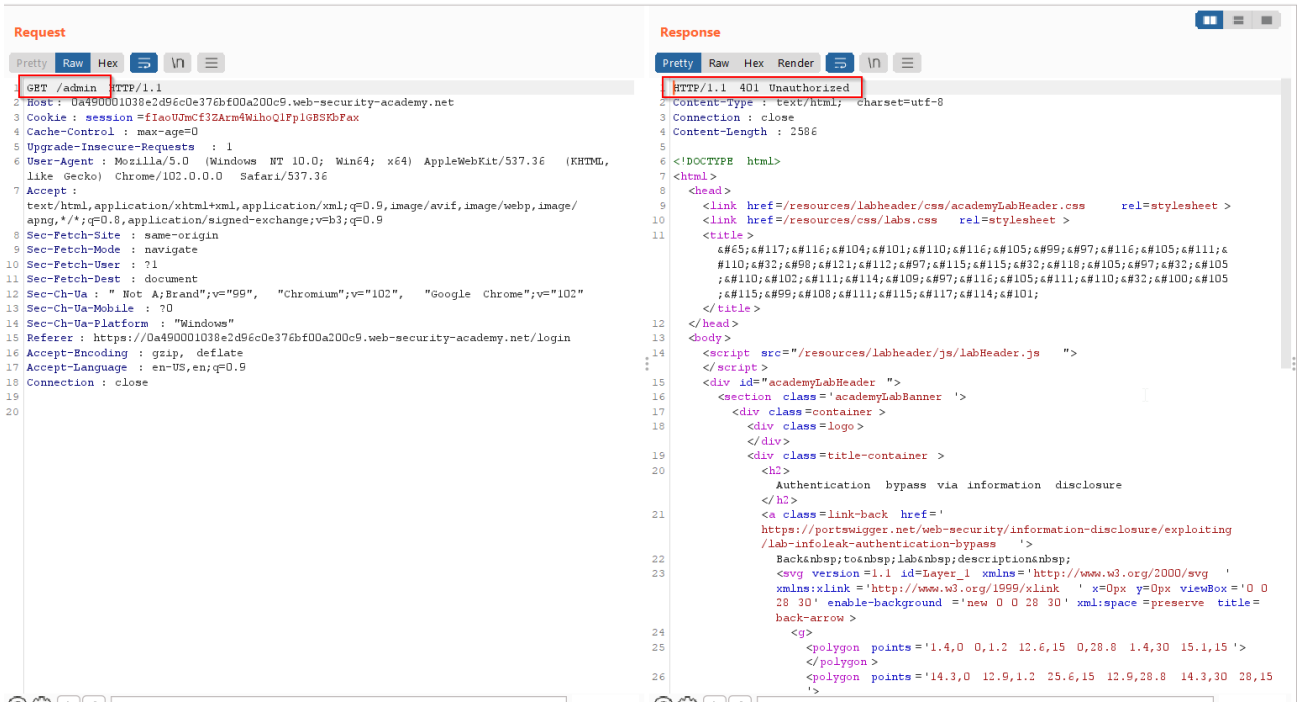
LAB 4: Authentication bypass via information disclosure

==========

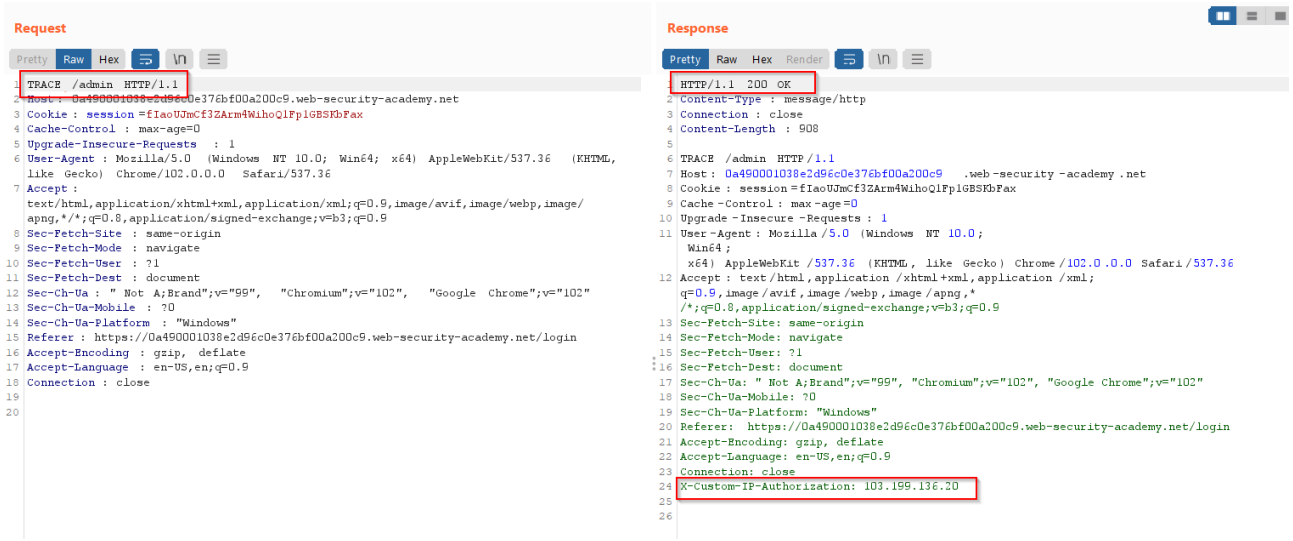
When logged in with given credential /my-account URL was accessed.



Let’s try to access /admin URL. When tried encountered an unauthorized error “401 unauthorized” instead of an error “page not found”.



Which confirms the presence of page and there might be some factor which is checking for authorization lets deep drive to find the same using TRACE method.

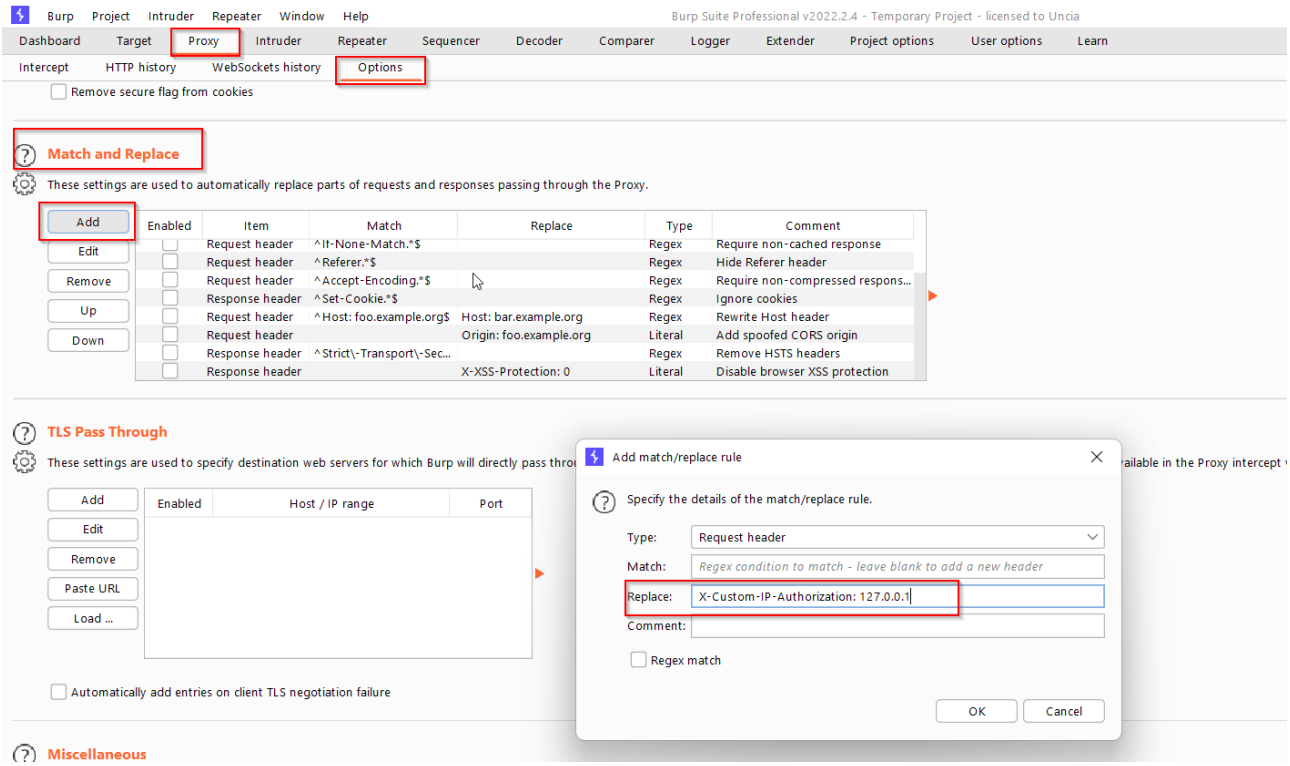


I found a parameter X-Custom-IP-Authorization: 103.199.136.20 which shows an internal IP is authorized for same.

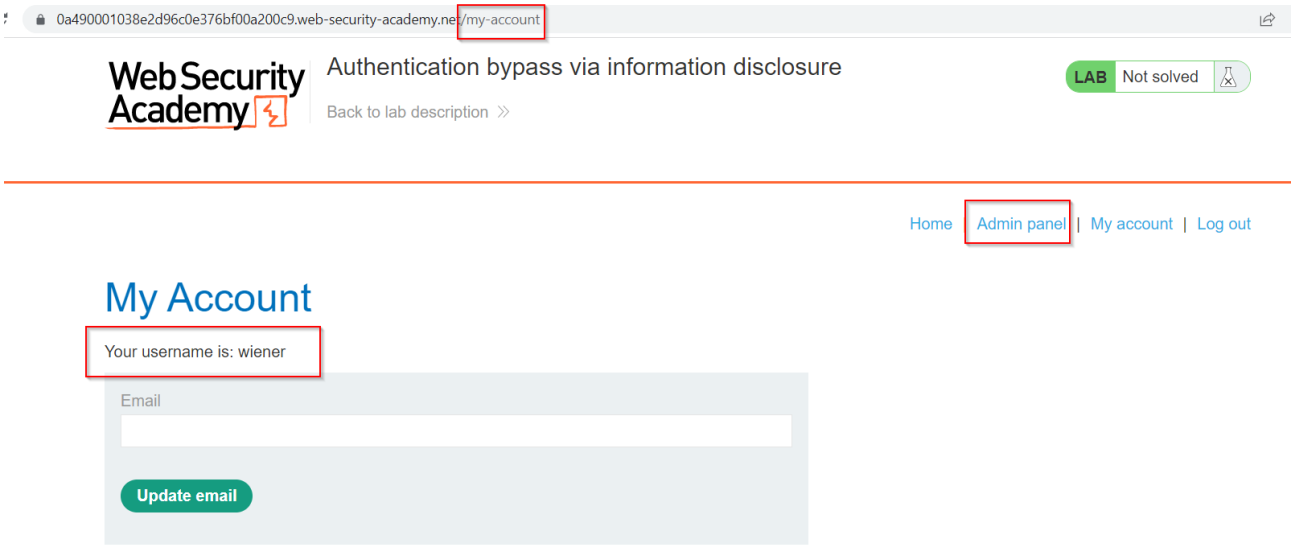
So, In Burp proxy I have replaced blank with given internal IP and when tried to access /my-account over the browser found admin panel option was enabled for me.

Let’s move ahead and delete user carlos id to complete the LAB.

Burp proxy option to replace any phrase.



Found admin panel.



==========

LAB 5: Information disclosure in version control history

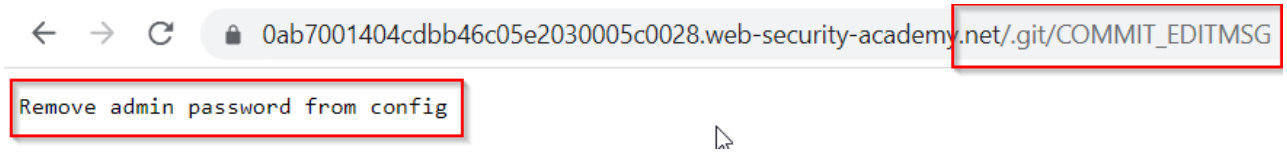
==========

To access git file, we used .git and we found some file listed as below.

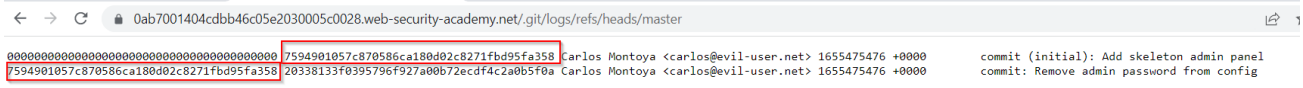
Ref: https://www.tutorialspoint.com/what-is-git-folder-and-why-is-it-hidden

By randomly accessing all file. I found a COMMIT_EDITMSG file.

In which it was written as remove admin password from config file.



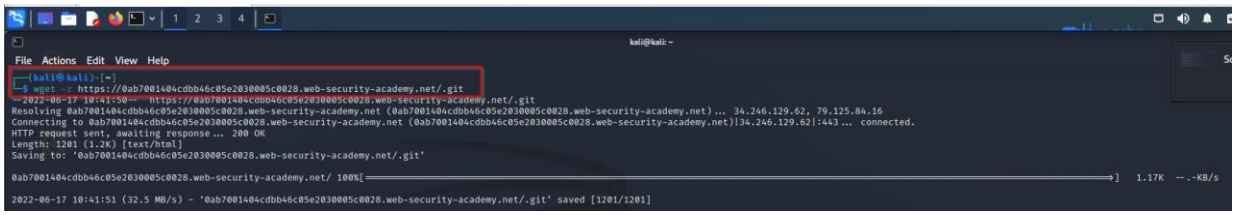
So, I tried searching that file eventually found it but it was encoded.



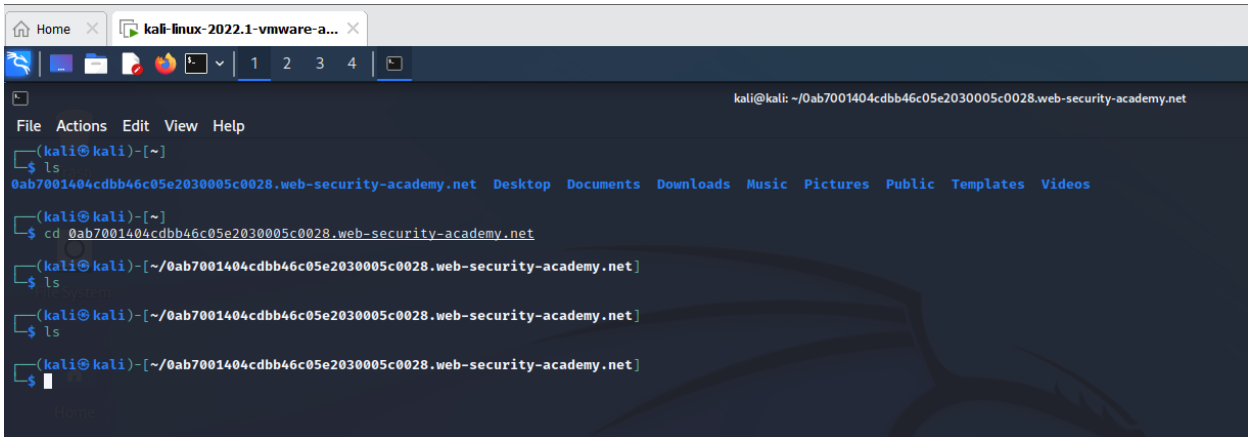
Let's move to kali and try to decode or undo the task.

Command:

wget -r https://website.com/.git



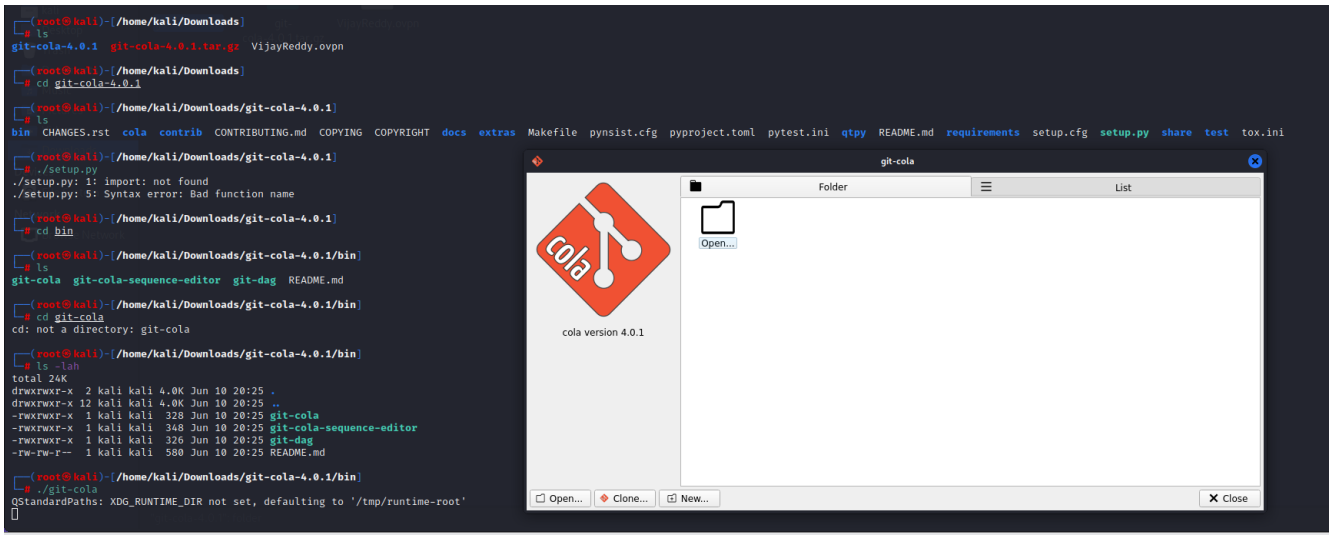
Command:ls



Tried above steps but I was unable to view files from console so need to download git-cola.

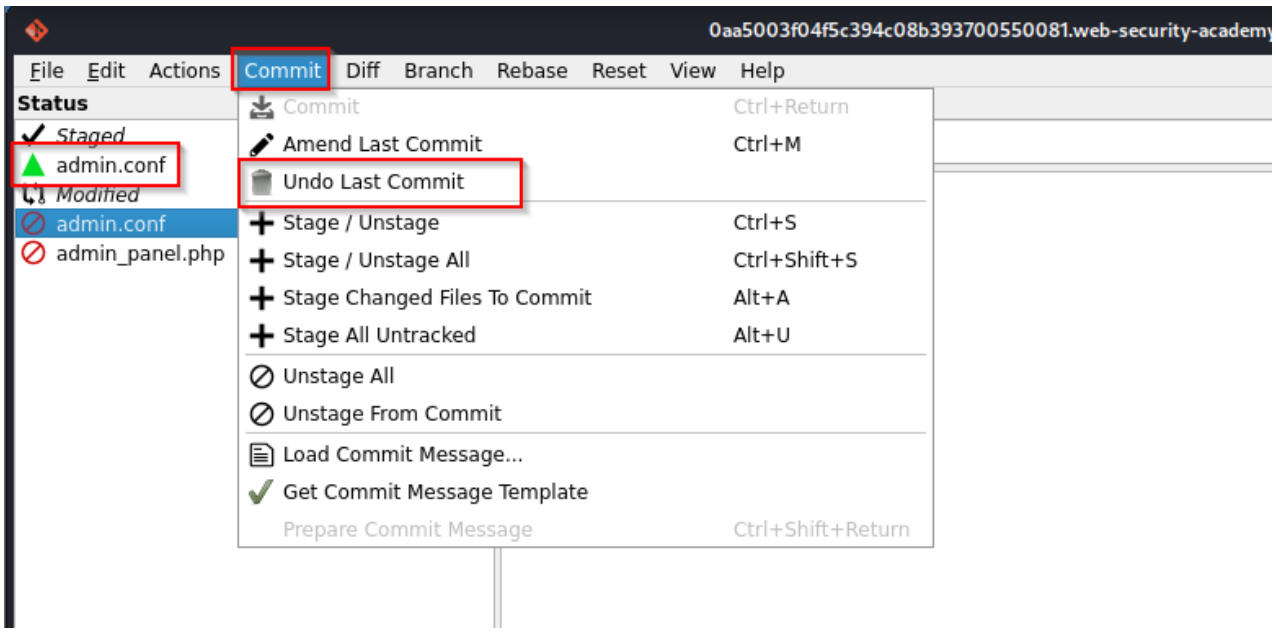
Command: sudo apt-get install git-cola

Has this was not working I have installed a tar file online and then unzip and executed the same from terminal.

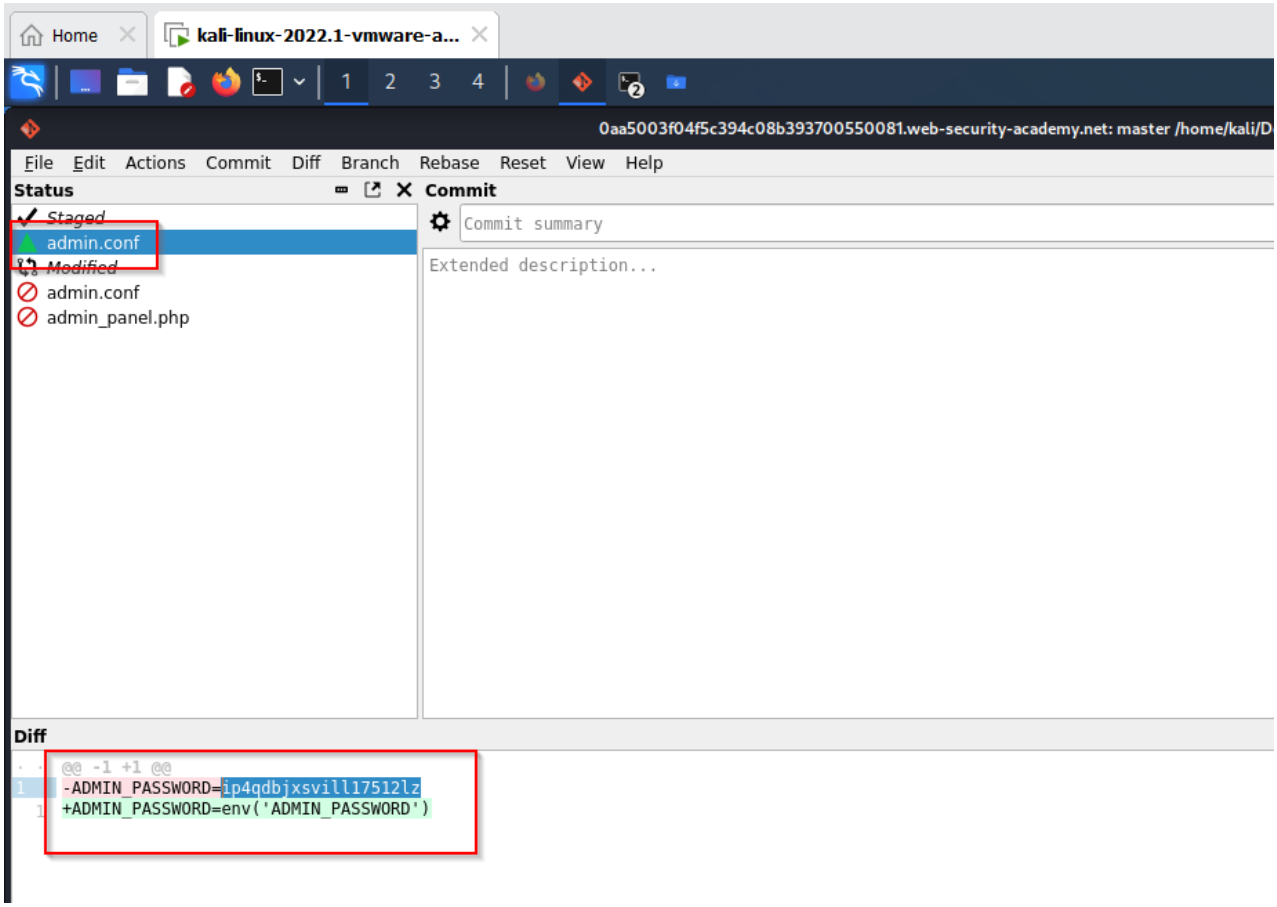


Using commit options let’s undo last commit.

Observed green file admin in below attached artifact.



Upon undo found a file "admin.conf" for which we were looking for.



Found the admin credential.

THE END