



Srilanka Institute of Information Technology

Web Application Security Audit Assessment Report

IE2062 – Web Security

Submitted by:

Student Registration Number	Student Name
Sulaxshayan.N	IT20660116

Date of submission: 05/06/2022

Bug Bounty

We are instructed to perform a bug bounty program as our web security assignment at legal websites such as [Hackerone](#), [Bugcrowd](#), and [initigrity](#). Initially, to succeed in these programs, professionals used to go into the approach called ‘Hackers Methodology’. Therefore, we also decided to go like a professional; how he/she approaches the target.

1. Reconnaissance.
2. Enumeration.
3. Gaining access.
4. Privilege escalation (or privacy exploration).
5. Covering tracks.

These are the tools that we going to use for this bug bounty program.

Reconnaissance.

This is the initial phase of this Methodology; People also call it **Recon**. In this phase, we are not interacting with the target overall but, we collect information about the target.

1. Who. is
2. Crt.h
3. Maltego.
4. Knock
5. Sudomy.
6. Nmap.

Enumeration

In this phase, attackers try to interact with the target and start attempting to find vulnerabilities related to that target. Attackers started to interact with the target from all of their attack surfaces. If we simply say, this is the phase where we find the target system is vulnerable to which kind of attack.

1. Burp Suite.
2. PWN XSS.

3. XSpear.
4. Nikto.
5. OWASP ZAP.
6. Skipfish

BOUNTY PROGRAM INFORMATION

I choose the shipt program from the [Hackerone](#) platform to do my bug bounty.

The screenshot shows the Shipt Bug Bounty Program page on the HackerOne platform. At the top left is the Shipt logo. Next to it, the word "Shipt" is written in a bold, black, sans-serif font, followed by the tagline "Over-delivering delivery." Below the logo is the URL "https://shop.shipt.com" and the handle "@shipt". To the right of the logo is a large pink button labeled "Submit report". On the far right, there's a section titled "Bug Bounty Program" which includes the text "Launched on Jul 2018", "Managed by HackerOne", and "Includes retesting". In the center, there are three metrics: "Reports resolved 136", "Assets in scope 16", and "Average bounty \$200-\$300". At the bottom right are "Bookmark" and "Subscribe" buttons.

These are the program rules:

Program Rules

Do:

- Abide by these Shipt Bug Bounty Program Terms & Conditions.
- Comply with all applicable laws.
- Be respectful when interacting with Shipt Staff/Team members and HackerOne Triage members.
- Only conduct testing with accounts you own.
 - Test accounts must comply with testing conditions (Researchers must use [@wearehackerone](#) accounts.)
- Immediately stop conducting research/testing when unsure. If you think you may have caused damage with testing a vulnerability, please report your initial findings and request authorization to continue testing.

Items in the following section are also Out of Scope of our Bug Bounty Program.

Do NOT:

- Access, process or destroy Shipt employee data, customer data or Shipt intellectual property.
- Mass create accounts to perform testing against our applications or services.
- Publicly disclose vulnerabilities.
- Make any attempts to extort us.
- Run automated destructive testing.
- Leave any of our systems in a more vulnerable state than you found them.
- Upload shells, scripts, or create a backdoor of any kind.
- Conduct Brute Force attacks or guess credentials against user accounts to gain access to our Systems.
- Interact with accounts you do not own.
- Make changes to/or save any Shipt data that is not yours or that you do not have explicit written permission to manipulate.
 - *If prompted to change a password of an account not registered by yourself or provided to you, stop conducting research and report the finding immediately.*
- Conduct activities to be considered a privacy violation or destruction of data, such as exfiltrating or destroying data for any account that is not yours or that you do not have explicit written permission to manipulate.
- Conduct tests that disrupt service to Shipt users or employees.
 - *Examples include, but are not limited to: Denial of service, Distributed Denial of Service, DNS Spoofing, Buffer Overflow, Ping of Death, Syn Flood, Teardrop.*
- Make attempts in any form of social engineering.
 - *Examples include, but are not limited to: Phishing (Angler, Spear), Whaling, Diversion Theft, Baiting, Honey Trap, Pretexting, SMS Phishing, Scareware, Watering Hole, etc.*
- Engage in any form of physical penetration testing against Shipt properties or offices.

And these are the program scopes and out-of-scope,

In Scope

While we expect to expand our scope in the future, we are currently focused on our primary and critical public facing sites and applications. However, if you enumerate other Shipt assets and identify vulnerabilities against those, your reports may still be eligible for bounty, so please responsibly disclose those to us as well.

If you identify a vulnerability within a third party that Shipt uses and it is not explicitly in scope, please report it to the third party's responsible disclosure program, bug bounty program, or security team. If you report it to us, we will be happy to assist you in reporting it to the third party, however, these reports usually will not be eligible for bounty from Shipt (unless special conditions are met such as the root cause being a misconfiguration by Shipt and not under the control of the 3rd party).

Out of Scope

In addition to our "Do Not" rules section, items below are out of scope.

- API key disclosure or leak without POC of exploitability.
- Internal IP Address disclosures without any proof of exploitation.
- Stack traces or error messages in staging environments without proof of exploitation.
- Any activity that could lead to the denial of service (DoS).
- Clickjacking on pages with no sensitive actions.
- Content spoofing and text injection issues without showing an attack vector (DO NOT actively engage in social engineering or phishing attempts to our employees or support staff via chat, email, or phone for POCs).
- Invalid or missing SPF (Sender Policy Framework), DMARC, or DKIM records/configurations.
- Attacks requiring MiTM or physical access to a users device
- Missing best practices in SSL/TLS configuration.
- Password and account recovery policies, such as reset link expiration or password complexity.
- Rate limit attacks or spamming.
- Unauthenticated or logout CSRF.
- User enumeration.
- "Information Disclosure" involving auserID or email address in a query parameter on a POST request
- Vulnerabilities only affecting users of outdated or unpatched browsers and platforms.
- Previously known vulnerable libraries or software versions without a working Proof of Concept.
- Any CORS-related vulnerabilities without working POC showing sensitive actions or data being affected.
- Self XSS or XSS that affects only out-of-date browsers or devices that require the user to configure them in an insecure state (allowing installation of applications from untrusted sources).
- Attacks predicated on a user's non-Shift account being compromised (e.g. Attackers compromise the user's Google/Gmail first before the vulnerability can be exploited).

These are the domain in the scope list,

Type	Identifier	Max. Severity	In Scope	Bounty
Domain	app.shipt.com	■ Critical	Yes	Yes
Domain	api.shipt.com	■ Critical	Yes	Yes
Domain	admin.shipt.com	■ Critical	Yes	Yes
Domain	shop.shipt.com	■ Critical	Yes	Yes
Domain	staging-api.shipt.com	■ Critical	Yes	Yes
Domain	staging-shop.shipt.com	■ Critical	Yes	Yes
Domain	staging-app.shipt.com	■ Critical	Yes	Yes
Domain	staging-admin.shipt.com	■ Critical	Yes	Yes
Domain	shoppingcart.shipt.com	■ Critical	Yes	Yes
Domain	www.shipt.com	■ Critical	Yes	Yes
Domain	*.shipt.com	■ Critical	Yes	Yes
Android: Play Store	com.shipt.groceries	■ Critical	Yes	Yes
Android: Play Store	com.shipt.shopper	■ Critical	Yes	Yes
iOS: App Store	971888874	■ Critical	Yes	Yes
iOS: App Store	976353472	■ Critical	Yes	Yes

These are the vulnerability rewards that are given by the Shipt.

Critical severity bug examples	\$3000 - \$5000
--------------------------------	-----------------

Remote Code Execution

Vertical Authentication bypass

SQL Injection that leaks targeted data

High severity bug examples	\$1000 - \$3000
----------------------------	-----------------

Lateral authentication bypass

Stored XSS

Insecure handling of authentication cookies

Medium severity bug examples	\$500 - \$1000
------------------------------	----------------

Reflected XSS

Insecure Direct Object References

CSRF on sensitive actions and functions

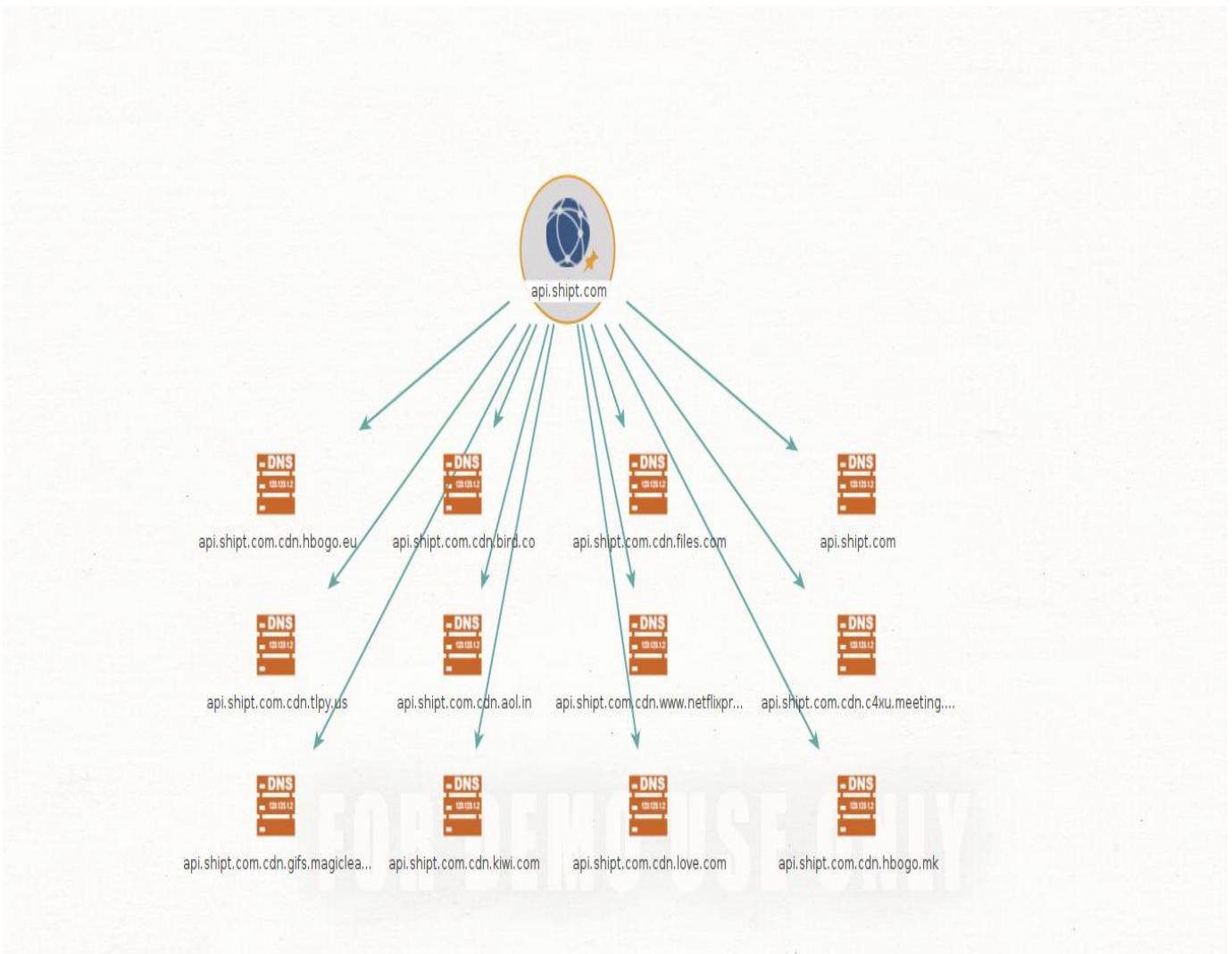
Information leaks

Subdomain Takeover

Reconnaissance.

I started my Recon phase with the **maltego** tool. This is the most effective tool to help us to think smartly. It has been used by professionals for collecting hidden information as well as, it reduces the time when they do their security assessments.

Initially, I started to gather information by searching DNS names in DNSDB, related to the domain **api.shipt.com** then I found **twelve** DNS.



Then I searched for IP addresses related to each and every DNS. It could be possible to find IP addresses of these domain with the help of the tool. Those are ,

- api.shipt.com.cdn.hbogo.eu - 185.60.68.10 – Hungary
- api.shipt.com.cdn.bird.co - 13.32.153.87
 - 13.32.153.111
 - 13.32.153.126
 - 13.32.153.20 } united states
- api.shipt.com.cdn.files.com - 34.204.145.250
 - 34.205.137.182 } united states
 - 34.204.153.236
 - 34.204.150.23

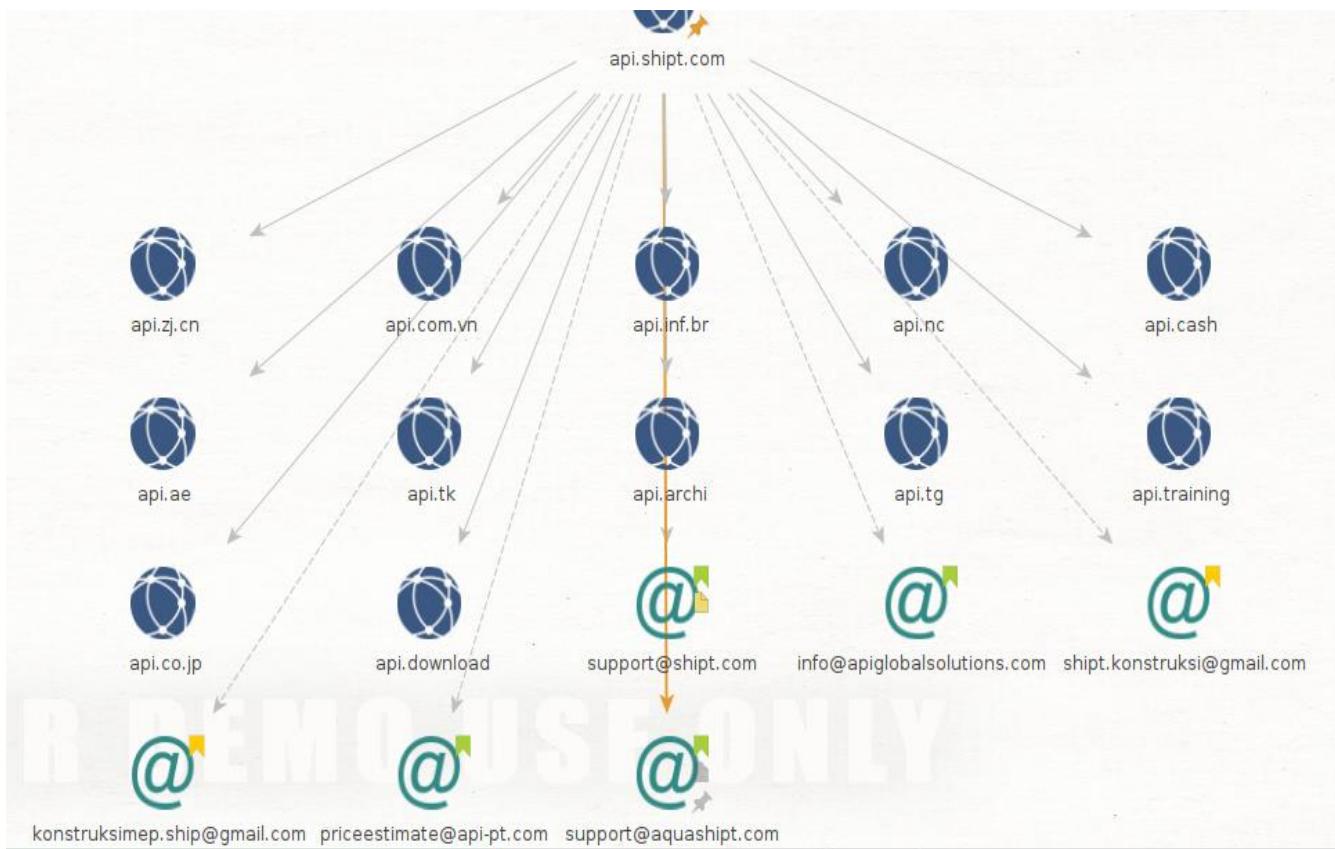
- api.shipt.com - 162.159.135.84
 - 162.159.134.84
- api.shipt.com.tipy.us - could not get IP address of DNS

- api.shipt.com.cdn.aol.in - 74.6.136.150
 - when I check this domain for IPQS, it had a fraud score of 100.

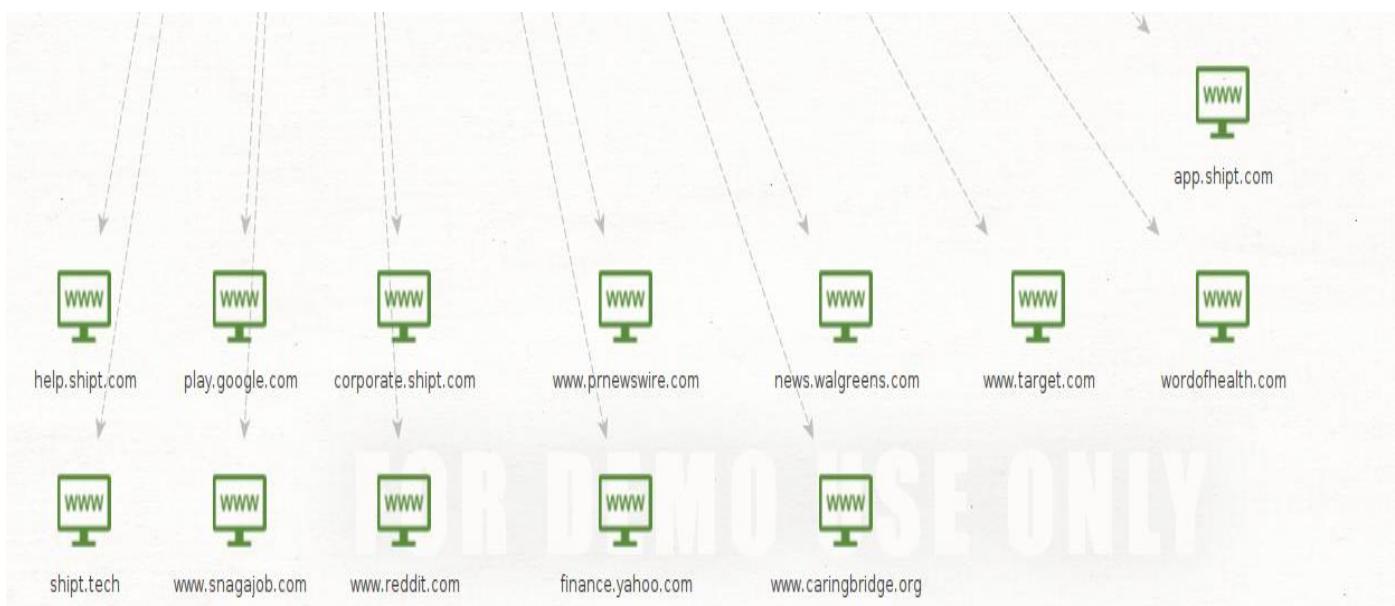
- api.shipt.com.cdn.www.netflixpremium.com - 50.17.247.9
 - 204.236.236.127
 united states
 - 107.20.175.192

- api.shipt.com.cdn.c4xu.meeting.hbogo.rs – 13.81.126.187

- api.shipt.com.cdn.gifs.magicleap.io.bridgeapp.com – 18.217.79.116
 - 18.217.209.204
 - 3.140.47.227
- api.shipt.com.cdn.hbogo.mk – 13.81.126.64
- api.shipt.com.cdn.kiwi.com – could not find the IP address of DNS
- api.shipt.com.cdn.love.com - could not get the IP address of DNS

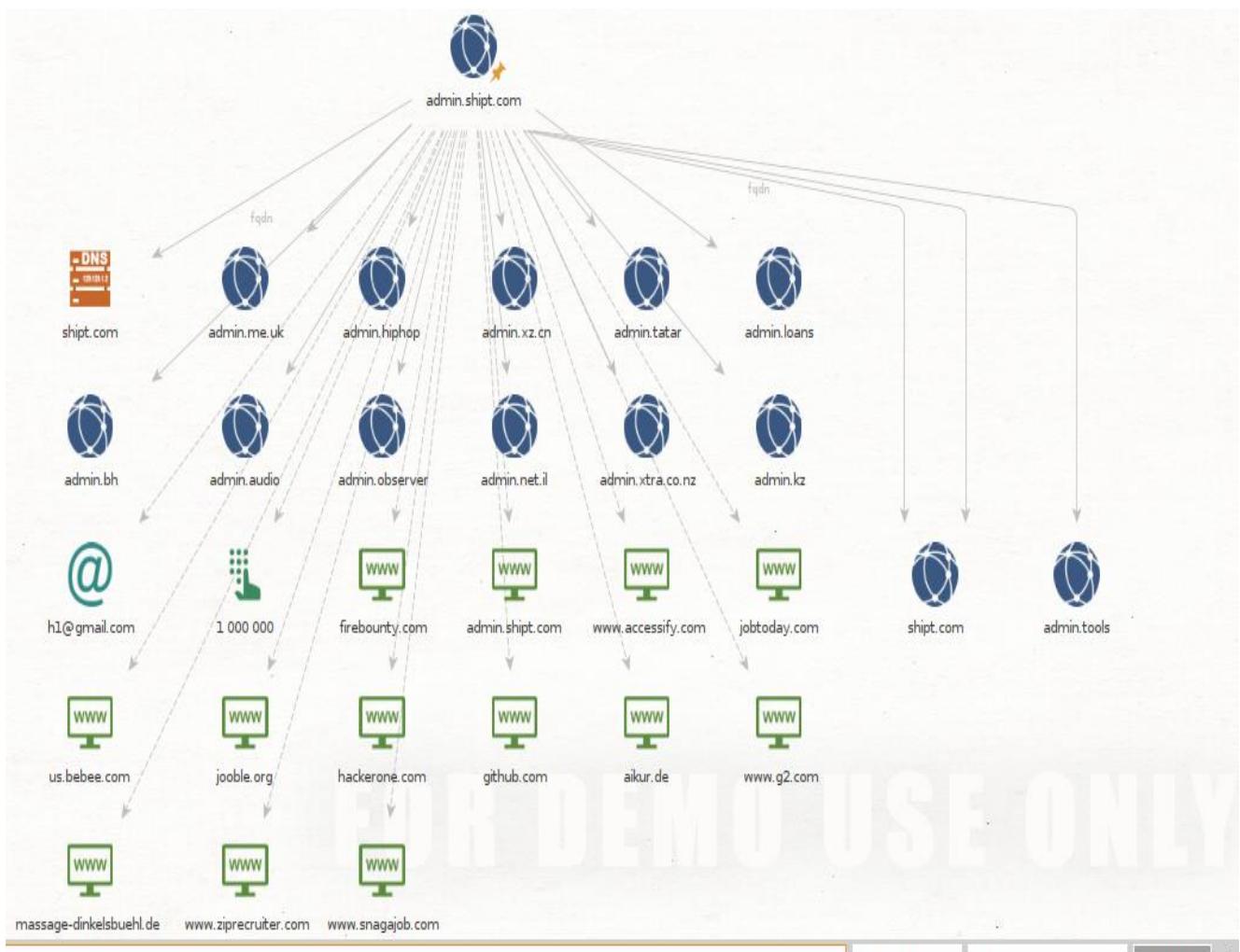


I searched for the top-level domains related to **api.shipt.com**, I found **twelve** top-level domain that is currently used by that site. Moreover, I found **six** email addresses on that site. I error-checked those email addresses and the result was relatively less than 75 (fraud score should be less than or equal to 75).



In search, I found some top-level domains, DNS, Email addresses, and websites related to **app.shipt.com**

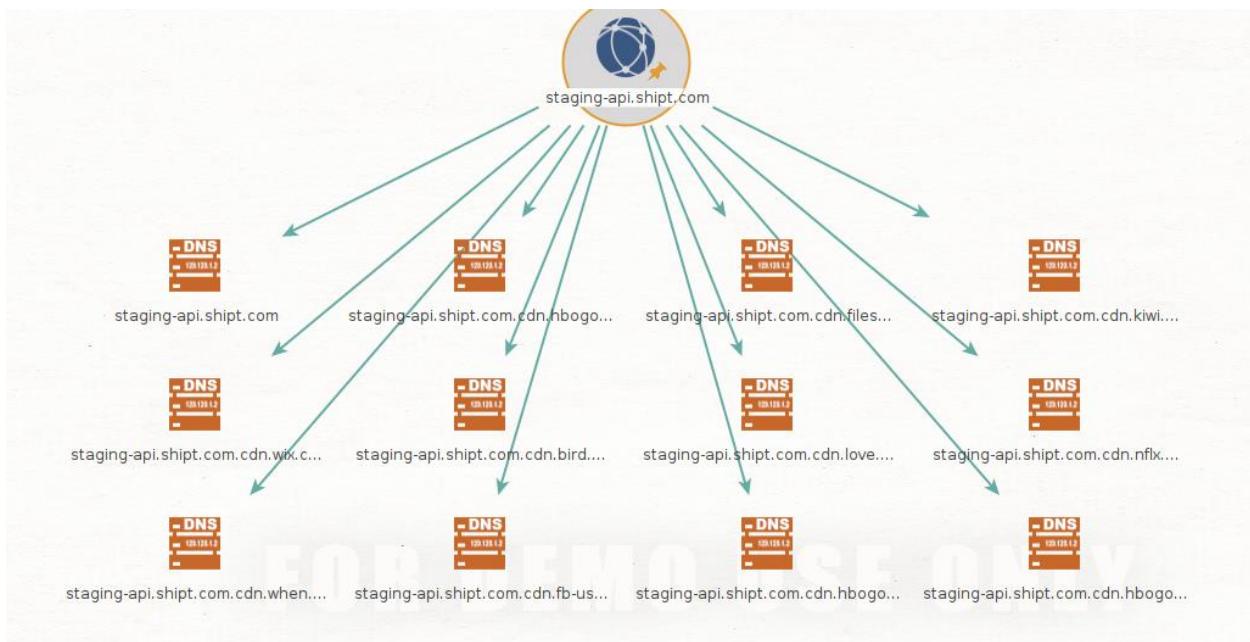
These are the records related to the **admin.shipt.com**



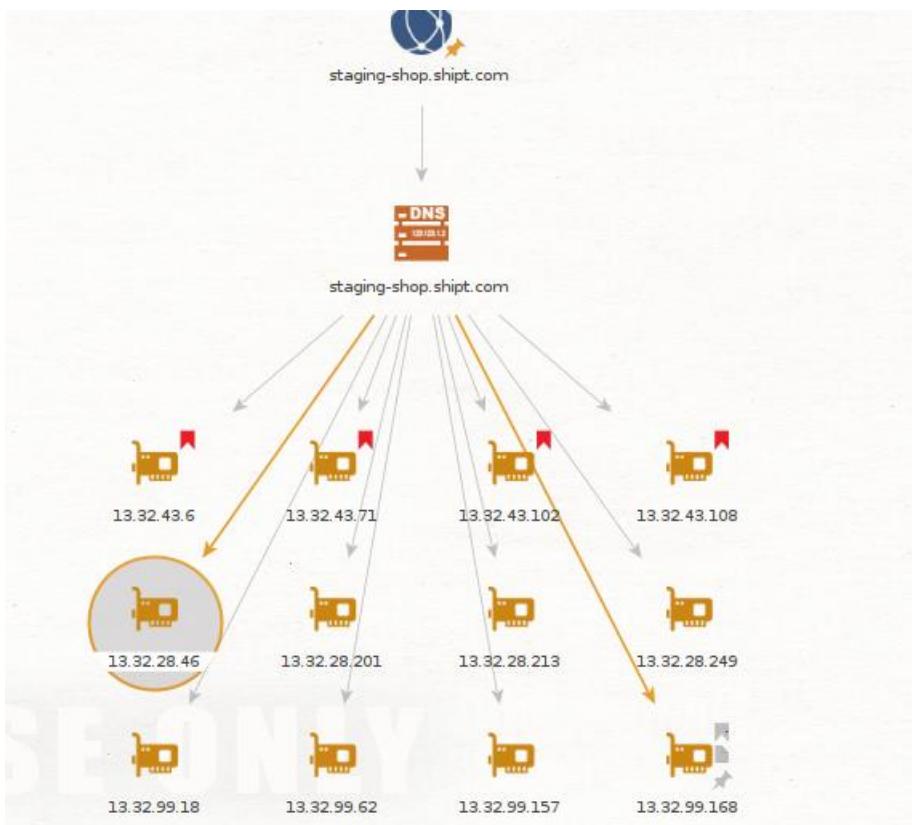
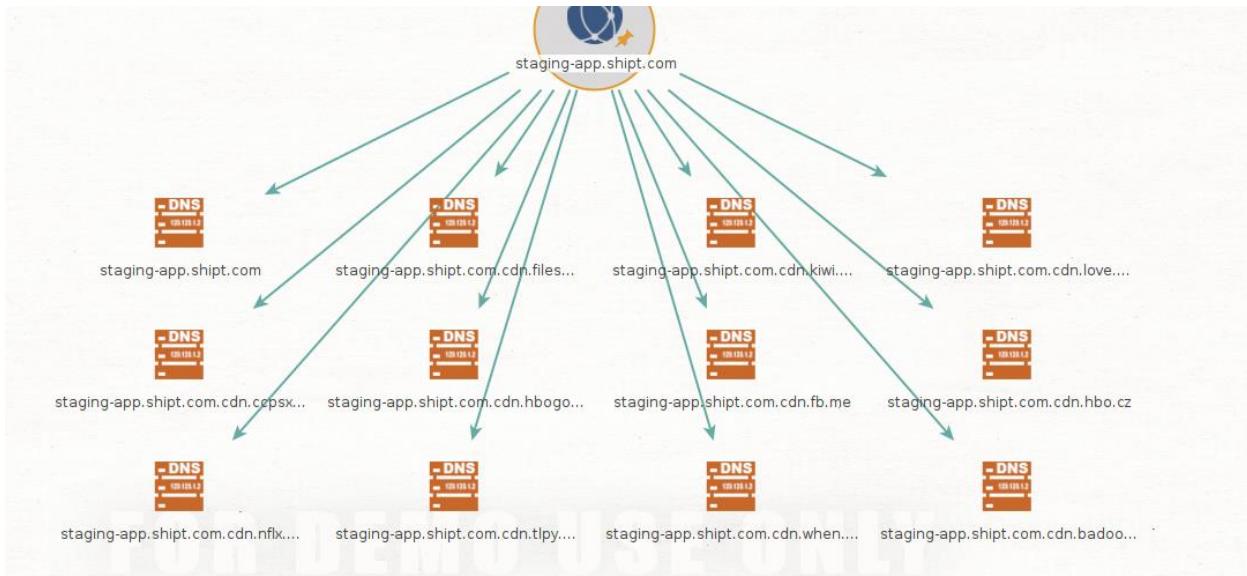
This is the DNS records information of **staging-shoppingcart.shipt.com**

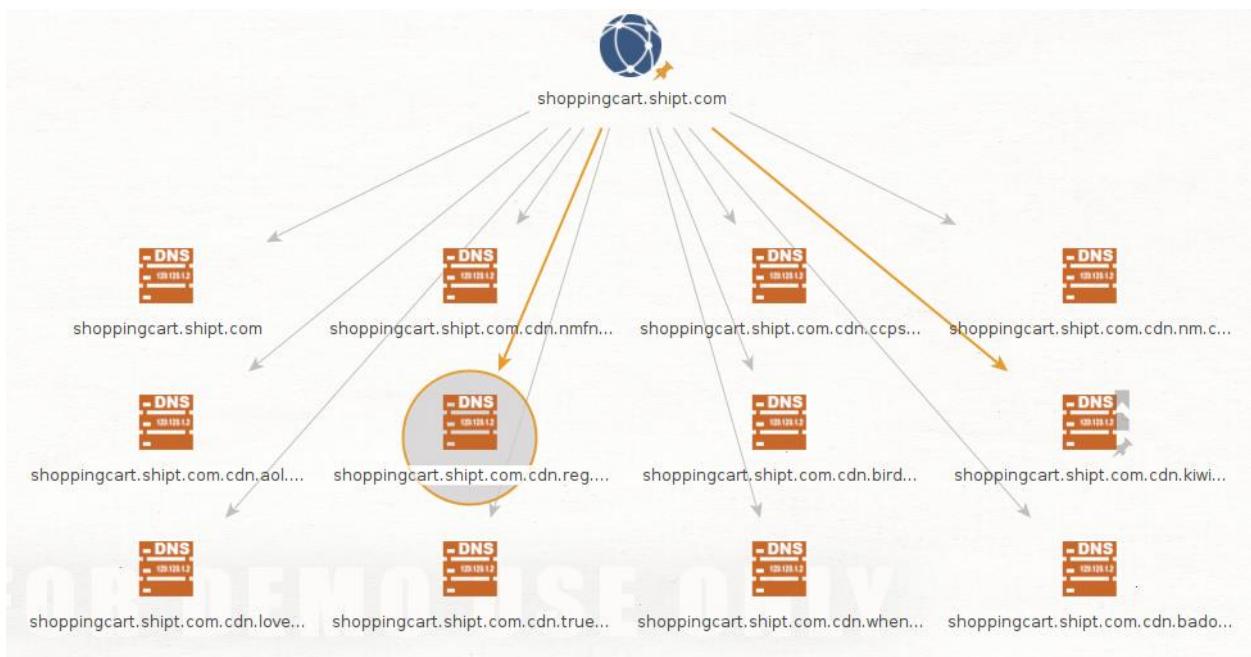
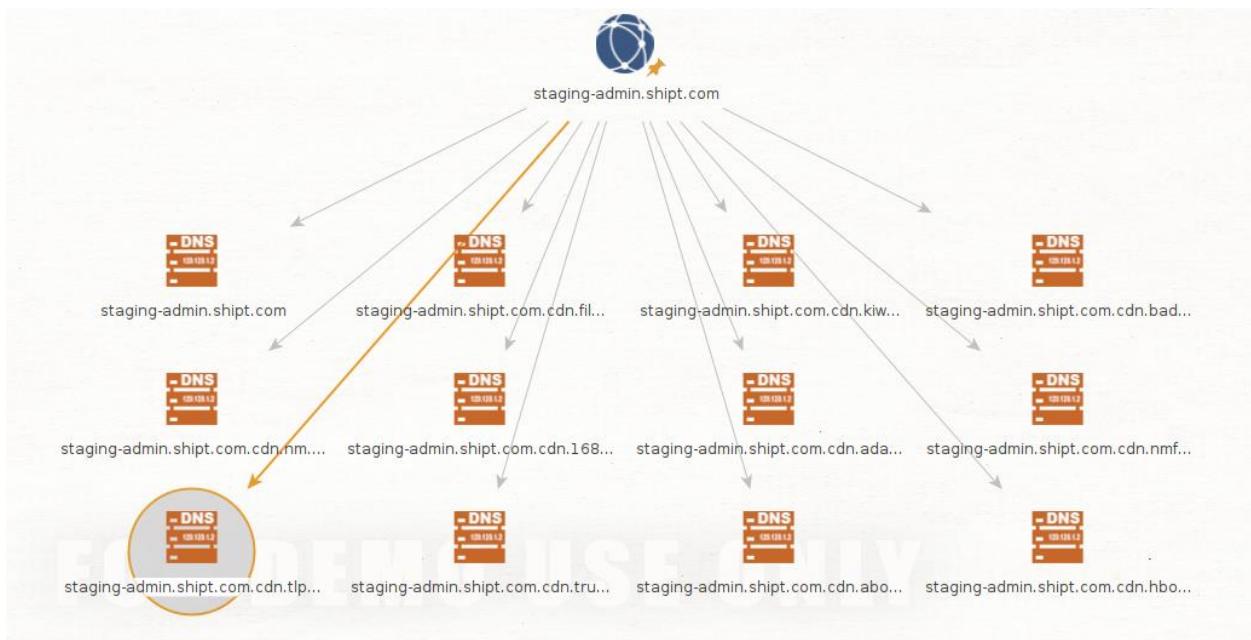


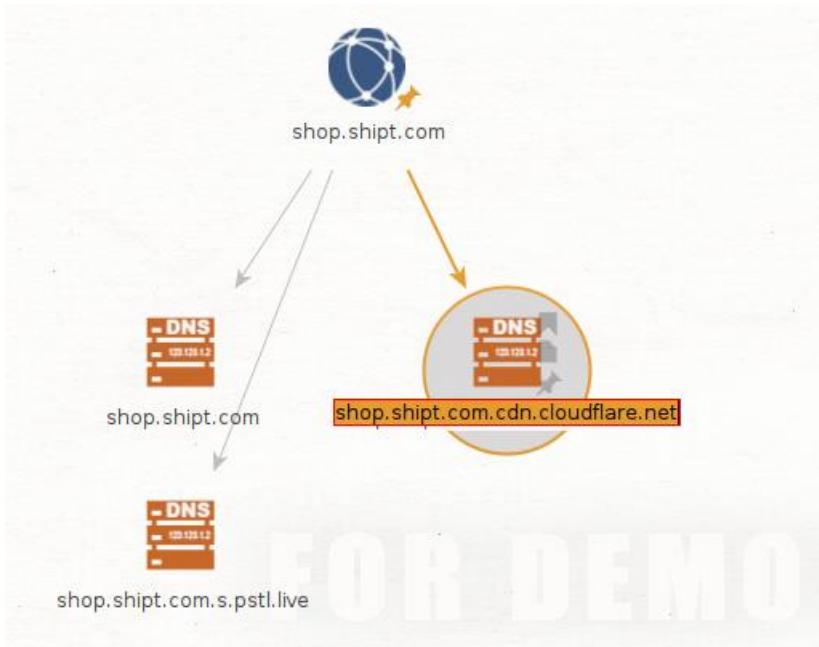
Then I found a DNS record related to **staging-api.shipt.com**



I searched for DNS name-related **staging-app.shipt.com** as well.



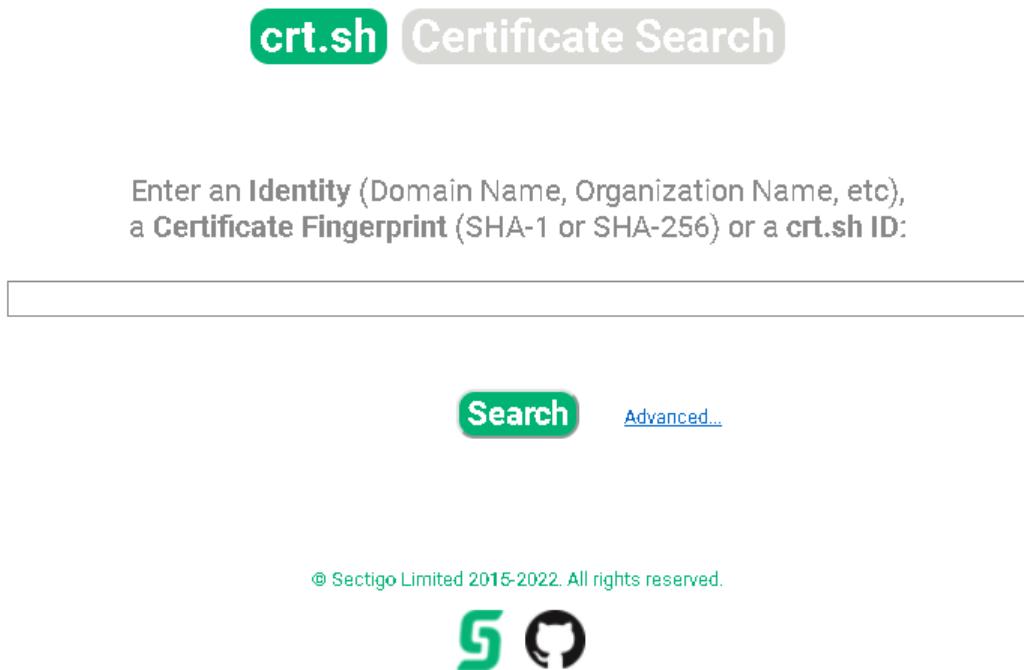




I had to take a 'ToDoList' to avoid unwanted loops in the testing process. Overall, maltego is an awesome tool that illustrated organizational structure as a rooted tree structure and it helps us to identify complex structures in one keystroke.

Crt.h

'Crt.h' is a website. That is used to find all TLS or SSL certificates of targeted domains. It is a completely free and open-source site. The idea behind this search is, that I have found that **api.shipt.com** uses **twelve** DNS names (CNAME records). So, we can inspect each one's certificates with the help of this tool whether the DNS name has been registered under the shipt.com; registered period of date is expired or not; verify the SSL is owned by the targeted domain. If there any possibilities in the DNS configuration, we can be able to do a **subdomain takeover**.



Above this image indicates the interface of the **crt.sh** site.

I searched for the CA certificate of this **api.shipt.com.cdn.hbogo.eu** subdomain,
But I could not get it.

crt.sh Identity Search  [Group by Issuer](#)

Criteria	Type: Identity Match: ILIKE Search: 'api.shipt.com.cdn.hbogo.eu'
Certificates None found	

© Sectigo Limited 2015-2022. All rights reserved.



then I checked each one of them accordingly,

•

crt.sh Identity Search  [Group by Issuer](#)

Criteria	Type: Identity Match: ILIKE Search: 'api.shipt.com.cdn.bird.co'
Certificates None found	

© Sectigo Limited 2015-2022. All rights reserved.



[Group by Issuer](#)

Criteria

Type: Identity Match: ILIKE Search: 'api.shipt.com.cdn.files.com'

Certificates

None found

© Sectigo Limited 2015-2022. All rights reserved.



At last of this process, I was not able to find any of the CA-related api.shipt.com.*DNS names.

For an alternate, I searched CA for DNS names within the property **api.shipt.com.cdn.hbogo.eu** then I found **cdn.hbogo.eu** as a sub-DNS name within that domain. So, I had to decide to find CA for that DNS. Finally, **crt.sh** shows the output like this

[Group by Issuer](#)

Criteria Type: Identity Match: ILIKE Search: 'cdn.hbogo.eu'

Certificates	crt.sh ID	Logged At	Not Before	Not After	Common Name	Matching Identities	Issuer Name
	6696456356	2022-05-09	2022-04-30	2023-04-30	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
	6638219077	2022-04-30	2022-04-30	2023-04-30	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
	6545622588	2022-04-15	2022-04-15	2023-04-19	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS Hybrid ECC SHA384 2020 CA1
	6545617462	2022-04-15	2022-04-15	2023-04-19	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
	6213285059	2022-02-20	2022-02-18	2023-03-21	sni11659gl.wpc.edecastcdn.net	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
	6199056201	2022-02-18	2022-02-18	2023-03-21	sni11659gl.wpc.edecastcdn.net	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
	5066273613	2021-08-18	2021-08-02	2022-08-02	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
	4973760354	2021-08-02	2021-08-02	2022-08-02	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
	4669387687	2021-06-08	2021-06-08	2022-06-13	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
	4667587526	2021-06-08	2021-06-08	2022-06-13	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
	4272524679	2021-03-25	2021-03-18	2022-04-18	sni11659gl.wpc.edecastcdn.net	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
	4234418199	2021-03-18	2021-03-18	2022-04-18	sni11659gl.wpc.edecastcdn.net	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
	3646264499	2020-11-14	2020-11-05	2021-11-04	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
	3606398149	2020-11-05	2020-11-05	2021-11-04	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert TLS RSA SHA256 2020 CA1
	2777696836	2020-05-07	2020-05-07	2021-08-06	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, OU=www.digicert.com,CN=DigiCert Secure Site ECC CA-1
	2777653074	2020-05-07	2020-05-07	2021-08-06	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
	2451819851	2020-02-12	2020-02-08	2021-02-08	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
	2430523854	2020-02-08	2020-02-08	2021-02-08	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
	1881344790	2019-09-13	2019-05-13	2020-05-13	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
	1881344802	2019-09-13	2019-05-14	2021-05-18	sni11659gl.wpc.edecastcdn.net	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
	1598560203	2019-06-21	2019-06-21	2020-06-20	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
	1469964964	2019-05-14	2019-05-14	2021-05-18	sni11659gl.wpc.edecastcdn.net	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
	1467485097	2019-05-13	2019-05-13	2020-05-13	cdn.hbogo.eu	cdn.hbogo.eu	C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA

According to its last activity, that domain has not expired.

crt.sh Certificate Search

Criteria ID = '6696456356'

crt.sh ID	6696456356					
Summary	Leaf certificate					
Certificate Transparency	Log entries for this certificate:					
	Timestamp	Entry #	Log Operator	Log URL		
	2022-05-09 21:58:20 UTC	35625109	Google	https://ct.googleapis.com/submariner		
Revocation	Mechanism	Provider	Status	Revocation Date	Last Observed in CRL	Last Checked (Error)
Report a problem with this certificate to the CA	OCSP	The CA	Check	?	n/a	?
	CRL	The CA	Not Revoked	n/a	n/a	2022-05-31 23:59:10 UTC
	CRLSet/Blocklist	Google	Not Revoked	n/a	n/a	n/a
	disallowedcert.stl	Microsoft	Not Revoked	n/a	n/a	n/a
	OneCRL	Mozilla	Not Revoked	n/a	n/a	n/a
Certificate Fingerprints	SHA-256	5FA1A0F7763F53E790EA84605736FB014617B9B5B485F5C74E0B80607E48D161				SHA-1
Certificate ASN.1 Graph px	Certificate: Data: Version: 3 (0x2) Serial Number: 03:a7:60:ae:bf:7a:5d:86:9c:0e:b1:85:ca:82:a2:10 Signature Algorithm: sha256WithRSAEncryption Issuer: (OID 10576) commonName = DigiCert TLS RSA SHA256 2028 CA1 organizationName = DigiCert Inc countryName = US Validity Not Before: Apr 30 00:00:00 2022 GMT Not After : Apr 30 23:59:59 2023 GMT Subject: commonName = cdn.hbogo.eu organizationName = Microsoft Corporation localityName = Redmond stateOrProvinceName = Washington countryName = US					
Download Certificate: PEM						

According to the findings, I just started to turn my thoughts to a command-line tool to enumerate the list of subdomains. Because I was frustrated by checking each site in the crt.sh web site. Finally, in search of a tool, I found a subdomain enumeration tool called “**knock**” on GitHub.

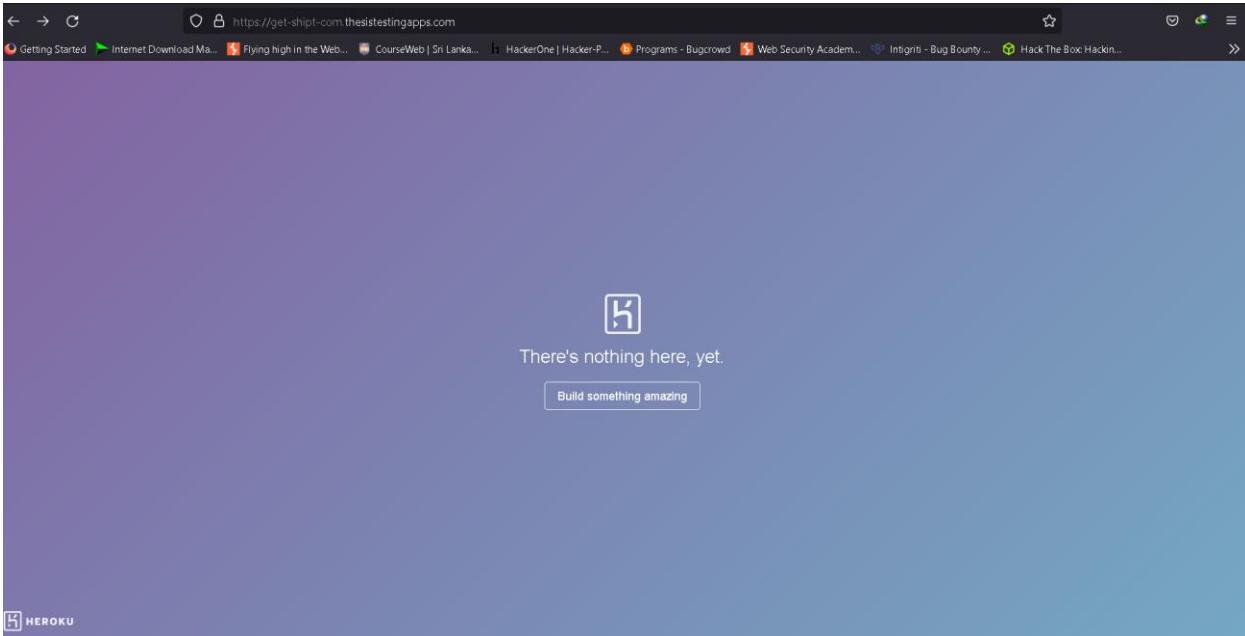
Knock

Knockpy is written in python. It is designed to enumerate subdomains on a target domain through a wordlist. It is intended to monitor for DNS zone transfer and if it is enabled, to attempt to circumvent the wildcard DNS record automatically. So, I enumerated subdomains to www.shipt.com website.

These are the outputs that I got.

```
[JustBot@kali)-[~/knock] nmap/tnk-subs
$ python3 knockpy.py shipt.com
shipt.com installed with:
sudo apt install gccgo-go
sudo apt install golang-go
v5.3.0
Reading state information ...done.
The following packages were automatically installed and are no longer required:
  fonts-roboto-slab liblommel2 python3-ipaddr python3-singledispatch python3-twisted-bin
local: 10757 | google: 5 | duckduckgo: 0 | virustotal: 0
The following additional packages will be installed:
Wordlist: 10762 | Target: shipt.com | Ip: 162.159.135.84 atomici libc++0 libgcc-12-dev libgcc-s1 libgo-12-dev libgoz1 libgomp1 libitm1 libtsan1 libublasmath0 libstdc++-12-dev libstdc++6 libtsan2 libubsan1
05:32:01 packages:
  gce-12-locales g++-12-multilib gcc-12-doc gcc-12-multilib gccgo-12-doc libstdc++-12-doc
Ip address      Code Subdomain           be installed:          Server          Real hostname
-----          --- -----
162.159.135.84 403 admin.shipt.com     ed:                 cloudflare      admin.shipt.com.cdn.cloudflare.net
162.159.135.84 403 api.shipt.com       libitm1 libbsl:        cloudflare@ath0 libstdc++6 libubsan1
162.159.135.84 403 apply.shipt.com     Remove and 246 not:   cloudflare      apply.shipt.com.cdn.cloudflare.net
162.159.134.84 403 app.shipt.com       es:                 cloudflare      app.shipt.com.cdn.cloudflare.net
146.148.55.211 403 blog.shipt.com     additional disk space:  cloudflare@ed.      shipt.wpeengine.com
162.159.134.84 403 cms.shipt.com       :                  cloudflare      cms.shipt.com.cdn.cloudflare.net
40.112.243.49  200 corporate.shipt.com :                  Microsoft-IIS/10.0  waws-prod-bay-163-2d26.westus.cloudapp.azure.com
162.159.134.84 403 design.shipt.com    :                  cloudflare      design.shipt.com.cdn.cloudflare.net
50.16.128.128  200 drivers.shipt.com   :                  nginx          shipt-driver.helpjuice.com
162.159.135.84 403 envoy.shipt.com     :                  cloudflare      envoy.shipt.com.cdn.cloudflare.net
-----          --- -----
162.159.135.84 403 envoy.shipt.com     :                  cloudflare      envoy.shipt.com.cdn.cloudflare.net
162.223.233.185 404 email.shipt.com   subs:              Apache          envoy.shipt.com.cdn.cloudflare.net
52.87.200.168  not found:             ftpserver.shipt.com:  installed with:
104.18.32.46   200 get.shipt.com      :                  cloudflare      ec2-52-87-200-168.compute-1.amazonaws.com
50.16.128.128  200 help.shipt.com    :                  nginx          get-shipt.com.thesistestingapps.com
162.159.135.84 403 img.shipt.com      :                  cloudflare      shipt.helpjuice.com
52.8.229.47   403 links.shipt.com    :                  cloudflare      img.shipt.com.cdn.cloudflare.net
162.159.134.84 403 metrics.shipt.com :                  cloudflare      thirdparty.bnc.lt
44.232.27.116  200 mi.shipt.com      :                  cloudflare      metrics.shipt.com.cdn.cloudflare.net
65.1.46.214   200 neptune.shipt.com  :                  Netlify         ink1001.com
162.159.134.84 403 partner.shipt.com :                  cloudflare      shipt.netlifyglobalcdn.com
34.229.9.55   not found:             prtg.shipt.com    :  Package already installed and are no longer required:  partner.shipt.com.cdn.cloudflare.net
54.164.13.186  not found:             rabbit.shipt.com  :  installed with:
162.159.135.84 403 secure.shipt.com  :                  cloudflare      ec2-34-229-9-55.compute-1.amazonaws.com
162.159.135.84 403 sgl.shipt.com     :  Package already installed and are no longer required:  secure.shipt.com.cdn.cloudflare.net
162.159.135.84 403 shopper.shipt.com  :  Package already installed and are no longer required:  sgl.shipt.com.cdn.cloudflare.net
50.16.128.128  200 shoppers.shipt.com:  Package already installed and are no longer required:  shopper.shipt.com.cdn.cloudflare.net
34.200.2.28   200 sftp.shipt.com    :                  nginx          libit
162.159.135.84 403 shop.shipt.com    :                  cloudflare      libtsan2 libubsan1
162.159.134.84 403 signup.shipt.com   :  Package already installed and are no longer required:  shop.shipt.com.cdn.cloudflare.net
54.204.238.15 403 share.shipt.com   :  Package already installed and are no longer required:  signup.shipt.com.cdn.cloudflare.net
13.236.8.150  200 status.shipt.com   :  Package already installed and are no longer required:  mbsy.co libstdc++-12-dev libtsan2
146.148.55.211 200 support.shipt.com  :  Package already installed and are no longer required:  hjjhgpfdmjbj.stspg-customer.com
162.159.135.84 403 talent.shipt.com  :  Package already installed and are no longer required:  shipt.wpeengine.com
44.226.119.150 200 vpn.shipt.com    :  Package already installed and are no longer required:  talent.shipt.com.cdn.cloudflare.net
162.159.134.84 403 www.shipt.com     :  Package already installed and are no longer required:  vpn-west.shipt.com
Do you want to continue? [y/n] n
05:44:19
Ip address: 25 | Subdomain: 35 | elapsed time: 00:12:18
```

Then check each subdomain on the internet which has a status code of 200. Then I search for [get.shipt.com](https://get.shipt.com.thesistestingapps.com) it redirected me to a website. So on, it had a DNS entry pointing to **get-shipt-com.thesistestingapps.com** but when I checked nothing was hosting on that page. When I redirected it shows like this,



I assumed that something not going well on that website. So decided to check this CA on **crt.sh** website.

crt.sh Certificate Search

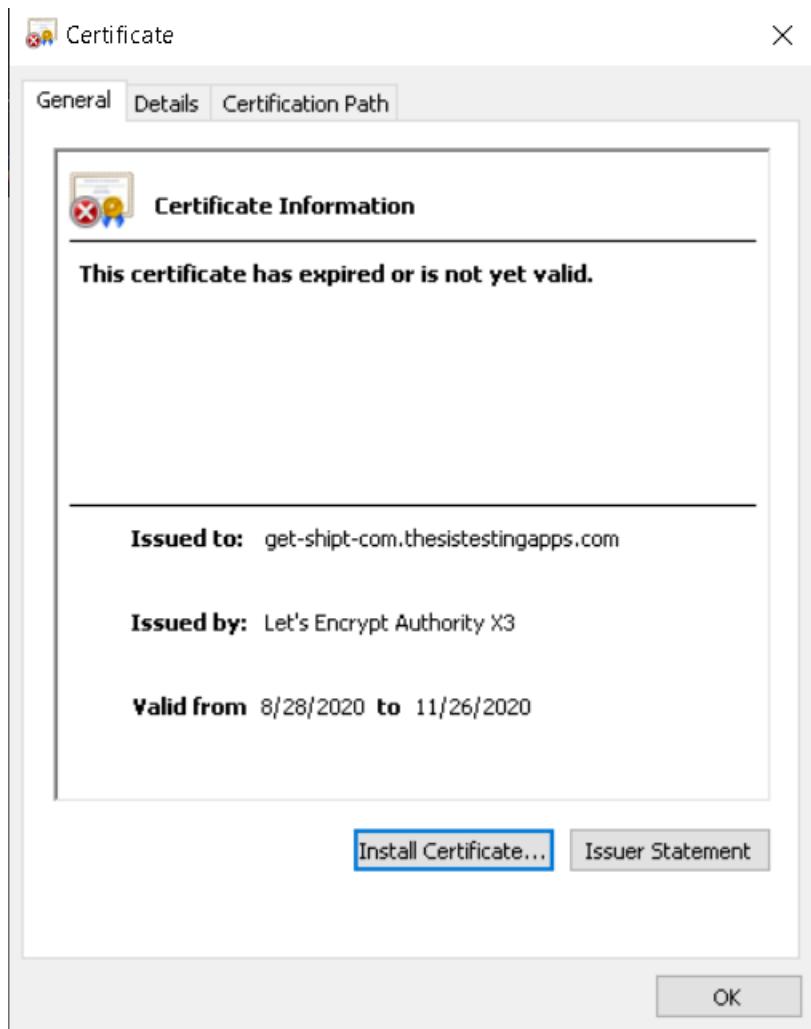
Criteria ID = '3300014662'

crt.sh ID	3300014662					
Summary	Leaf certificate					
Certificate Transparency	<i>Log entries for this certificate:</i>					
	Timestamp	Entry #	Log Operator	Log URL		
	2020-08-28 02:16:48 UTC	859395812	Google	https://ct.googleapis.com/logs/argon2020		
	2020-08-28 02:16:49 UTC	890319429	Google	https://ct.googleapis.com/logs/xenon2020		
Revocation	Mechanism	Provider	Status	Revocation Date	Last Observed in CRL	Last Checked (Error)
Report a problem with this Certificate to the CA	OCSP	The CA	Check	?	n/a	?
	CRL	The CA	Unknown (Expired)	n/a	n/a	
	CRLLSet/Blocklist	Google	Not Revoked	n/a	n/a	n/a
	disallowowedcert.stl	Microsoft	Not Revoked	n/a	n/a	n/a
	OneCRL	Mozilla	Not Revoked	n/a	n/a	n/a
Certificate Fingerprints	SHA-256 0753D188A94C43BAA028BF766BEF4CC42515F0DA651880A80BC585A2E148C6A4 SHA-1 72336C34C4A0F98493B493852076F42C907EEBFC					
Certificate ASN.1 Graph pvt	Certificate: Data: Version: 3 (0x2) Serial Number: 03:b0:33:81:7d:5f:1d:0d:f2:53:9a:24:f4:c1:cb:ca:cc:85 Signature Algorithm: sha256WithRSAEncryption Issuer: (CAID 16410) commonName = Let's Encrypt Authority X3 organizationName = Let's Encrypt countryName = US Validity (Expired) Not Before: Aug 28 01:15:43 2020 GMT Not After : Nov 26 01:15:43 2020 GMT Subject: commonName = get-shipt-com.thesistestingapps.com Subject Public Key Info:					
	<pre>Version: 3 (0x2) Serial Number: 03:b0:33:81:7d:5f:1d:0d:f2:53:9a:24:f4:c1:cb:ca:cc:85 Signature Algorithm: sha256WithRSAEncryption Issuer: (CAID 16410) commonName = Let's Encrypt Authority X3 organizationName = Let's Encrypt countryName = US Validity (Expired) Not Before: Aug 28 01:15:43 2020 GMT Not After : Nov 26 01:15:43 2020 GMT Subject: commonName = get-shipt-com.thesistestingapps.com</pre>					
	Subject Public Key Info: Public Key Algorithm: rsaEncryption RSA Public-Key: (2048 bit) Modulus: 00:9c:d3:0c:f0:5a:e5:2e:47:b7:72:5d:37:83:b3: 68:63:38:ea:d7:35:26:19:25:e1:bd:be:35:f1:70: 92:2f:b7:b8:4b:41:05:ab:a9:9e:35:88:58:ec:b1: 2a:c4:68:87:0b:a3:63:75:e4:e6:f3:a7:62:71:ba: 79:81:60:1f:d7:91:9a:9f:f3:d0:78:67:71:c8:69: 0e:95:91:cf:fe:66:99:09:60:3c:48:cc:7e:ca:4d: 77:12:24:9d:47:1b:5a:eb:b9:ec:1e:37:00:1e:9c: ac:7b:a7:05:ea:ce:4a:eb:bd:41:e5:36:98:b9:cb: fd:6d:3c:96:68:df:23:2a:42:90:0c:86:74:67:c8: 7f:a5:9a:b8:52:61:14:13:3f:65:e9:82:87:cb:db: fa:0e:56:f6:8e:89:f3:85:3f:97:86:af:b0:dc:1a: ef:6b:0d:95:16:7d:04:2b:a0:65:b2:99:04:36:75: 80:6b:ac:4a:f3:1b:90:49:78:2f:a2:96:4f:2a:20: 25:29:04:c6:74:c0:d0:31:cd:8f:31:38:95:16:ba: a8:33:b8:43:f1:b1:1f:c3:30:7f:a2:79:31:13:3d: 2d:36:f8:e3:fc:f2:33:6a:b9:39:31:c5:af:c4:8d: 0d:1d:64:16:33:aa:fa:84:29:b6:d4:0b:c0:d8:7d: c3:93 Exponent: 65537 (0x10001)					
Certificates	crt.sh ID	Not Before	Not After	Issuer Name		
	47997543	2016-10-06	2021-10-06	C=US, O=Internet Security Research Group, CN=ISRG Root X1		
	15706126	2016-03-17	2021-03-17	O=Digital Signature Trust Co., CN=DST Root CA X3		

crt.sh CA Search

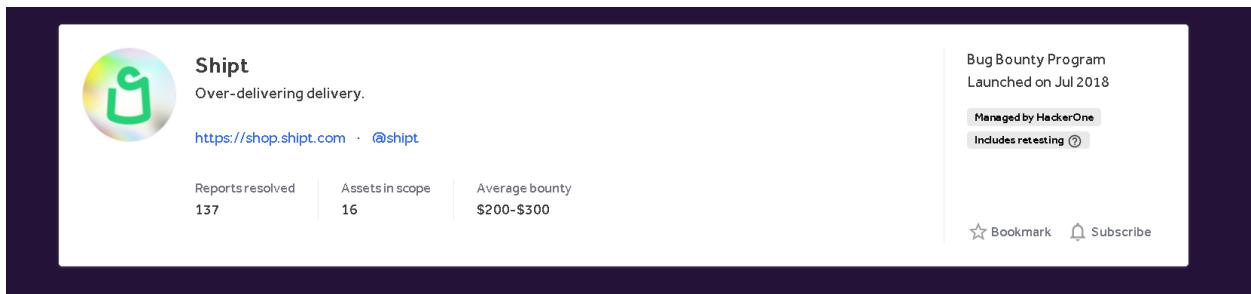
Criteria Type: CA ID Match: = Search: '16418'

crt.sh CA ID	16418												
CA Name/Key	<p>Subject:</p> <pre>commonName = Let's Encrypt Authority X3 organizationName = Let's Encrypt countryName = US</pre> <p>Subject Public Key Info:</p> <pre>Public Key Algorithm: rsaEncryption RSA Public-Key: (2048 bit) Modulus: 00:9c:d3:0c:f0:5a:e5:2e:47:b7:72:5d:37:83:b3: 68:63:38:ea:d7:35:26:19:25:e1:bd:be:35:f1:70: 92:2f:b7:b8:4b:41:05:ab:a9:9e:35:88:58:ec:b1: 2a:c4:68:87:0b:a3:63:75:e4:e6:f3:a7:62:71:ba: 79:81:60:1f:d7:91:9a:9f:f3:d0:78:67:71:c8:69: 0e:95:91:cf:fe:66:99:09:60:3c:48:cc:7e:ca:4d: 77:12:24:9d:47:1b:5a:eb:b9:ec:1e:37:00:1e:9c: ac:7b:a7:05:ea:ce:4a:eb:bd:41:e5:36:98:b9:cb: fd:6d:3c:96:68:df:23:2a:42:90:0c:86:74:67:c8: 7f:a5:9a:b8:52:61:14:13:3f:65:e9:82:87:cb:db: fa:0e:56:f6:8e:89:f3:85:3f:97:86:af:b0:dc:1a: ef:6b:0d:95:16:7d:04:2b:a0:65:b2:99:04:36:75: 80:6b:ac:4a:f3:1b:90:49:78:2f:a2:96:4f:2a:20: 25:29:04:c6:74:c0:d0:31:cd:8f:31:38:95:16:ba: a8:33:b8:43:f1:b1:1f:c3:30:7f:a2:79:31:13:3d: 2d:36:f8:e3:fc:f2:33:6a:b9:39:31:c5:af:c4:8d: 0d:1d:64:16:33:aa:fa:84:29:b6:d4:0b:c0:d8:7d: c3:93</pre> <p>Exponent: 65537 (0x10001)</p>												
Certificates	<table border="1"> <thead> <tr> <th>crt.sh ID</th> <th>Not Before</th> <th>Not After</th> <th>Issuer Name</th> </tr> </thead> <tbody> <tr> <td>47997543</td><td>2016-10-06</td><td>2021-10-06</td><td>C=US, O=Internet Security Research Group, CN=ISRG Root X1</td></tr> <tr> <td>15706126</td><td>2016-03-17</td><td>2021-03-17</td><td>O=Digital Signature Trust Co., CN=DST Root CA X3</td></tr> </tbody> </table>	crt.sh ID	Not Before	Not After	Issuer Name	47997543	2016-10-06	2021-10-06	C=US, O=Internet Security Research Group, CN=ISRG Root X1	15706126	2016-03-17	2021-03-17	O=Digital Signature Trust Co., CN=DST Root CA X3
crt.sh ID	Not Before	Not After	Issuer Name										
47997543	2016-10-06	2021-10-06	C=US, O=Internet Security Research Group, CN=ISRG Root X1										
15706126	2016-03-17	2021-03-17	O=Digital Signature Trust Co., CN=DST Root CA X3										



So I double-checked it CA. finally, I found something interesting which is the CA certificate has Expired. This means any person who intended to purchase that domain can purchase and host a service or website. This leads to a security threat to a company. Therefore, I decided to submit a report to HackerOne.

These are the steps it showed me to enter in that submit report section.



The screenshot shows the Shipt Bug Bounty Program page on the HackerOne platform. At the top, there's a logo for Shipt, a description "Over-delivering delivery.", and links to their website (<https://shop.shipt.com>) and social media (@shipt). Below this, there are three metrics: "Reports resolved" (137), "Assets in scope" (16), and "Average bounty" (\$200-\$300). To the right, there's information about the program being "Managed by HackerOne" and "Includes retesting". Buttons for "Bookmark" and "Subscribe" are also present.

Policy Hacktivity Thanks Updates (2)

Submit Vulnerability Report

Caution!
Shipt enforces a [Signal Requirement](#).

This requirement is temporarily waived for new users.

Please learn about [how Signal is calculated](#) and focus on submitting accurate reports. A Signal of less than 1.0 can result in your inability to submit additional reports.

1

Asset
Select the attack surface of this issue.

Q Select Asset Type...

com.shipt.groceries
Android: Play Store • Critical • Eligible for bounty

com.shipt.shopper
Android: Play Store • Critical • Eligible for bounty

971888874
iOS: App Store • Critical • Eligible for bounty

976353472
iOS: App Store • Critical • Eligible for bounty

Currently selected: [\(Deselect\)](#)

[www.shipt.com](#)

Please follow normal scope (no DOS, social engineering, etc.) and please refrain from assessing any other wp-engine platforms.

2

Weakness

Select the type of the potential issue you have discovered. Can't pick just one? Select the best match or submit a separate report for each distinct weakness.

🔍 Subdomain Takeover

All clusters

Currently selected: None

3

Severity (optional)

Estimate the severity of this issue.

None Low Medium High Critical ⓘ

Low

— OR —

CVSS v3.0 Calculator ⓘ

Attack Vector ⓘ	Scope ⓘ
Network	Unchanged
Adjacent	Changed
Local	
Physical	

Attack Complexity ⓘ	Confidentiality ⓘ
Low	None
High	Low
	High

Privileges Required ⓘ	Integrity ⓘ
None	None
Low	Low
High	High

User Interaction ⓘ	Availability ⓘ
None	None
Required	Low
	High

No Rating ---

4

Proof of Concept

The proof of concept is the most important part of your report submission. Clear, reproducible steps will help us validate this issue as quickly as possible.

Title*

A clear and concise title includes the type of vulnerability and the impacted asset.

Subdomain Takeover at get.shipt.com 115

Description*

What is the vulnerability? In clear steps, how do you reproduce it?

```
initially, i was able to enumerate a list of subdomains of 'www.shipt.com' with the help of the tool called 'knockpy'. After then, i redirected to every subdomains which has status code as 200. So on, i redirected to a subdomain 'get.shipt.com'. In its record, it has a DNS entry pointing to 'get.shipt.com.thesistestingapps.com' but there is nothing existing on that site. Moreover, i also checked CA for get.shipt.com in the crt.sh site. there were CA record pointing to 'get-shipt.com.thesistestingapps.com' when i checked its CA, it was paired.
```

Write Preview Parsed with Markdown

Impact *

What security impact could an attacker achieve?

A malicious person can claim ~~get-shipt-com.thesistestingapps.com~~ and host a site there.

Write Preview

Parsed with [Markdown](#)

F1756257: test1.png 42.8KB Download X

F1756261: test2.png 0.1MB Download X

F1756263: test3.png 44.7KB Download X

F1756264: test4.png 43.1KB Download X

I also attached screenshots of my finding.

The screenshot shows a web-based reporting platform. On the left, there's a navigation bar with tabs: 'Open (1)', 'Pending disclosure (0)', 'Pending retests (0)', and 'All (0)'. Below the tabs is a search bar with placeholder text 'Search all reports' and a 'Show filters' button. The main area displays a list of reports. One report is highlighted with a purple header: '#1590896 Subdomain Takeover at get.shipt.com'. The report details section includes a summary from user 'just_bot' dated 'Jun 4th (< 1 min ago)' describing a subdomain takeover attempt. It lists four attachments: F1756257: test1.png, F1756261: test2.png, F1756263: test3.png, and F1756264: test4.png. The summary text states: 'initially, i was able to enumerate a list of subdomains of 'www.shipt.com' with the help of the tool called 'knockpy'. After then, i redirected to every subdomains which has status code as 200. So on, i redirected to a subdomain 'get.shipt.com'. In its record, it has a DNS entry pointing to 'get-shipt.com.thesistestingapps.com' but there is nothing existing on that site. Moreover, i also checked CA for get.ship.com in the crt.sh site. there were CA record pointing to 'get-shipt.com.thesistestingapps.com' when i checked its CA, it was paired.'

Then, I started to check different kinds of vulnerabilities on that site to identify whether the web application is vulnerable or not.

Nikto

Nikto is a web server and web application inspector that is free and open source. Nikto can test web servers for a wide range of security concerns, including around 6700 potentially harmful files and programs. Nikto may also check for out-of-date web server software and also version-specific problems.

So, I started to gather information from get-shipt-com.thesistestingapps.com

```
(JustBot㉿kali)-[~]
$ nikto -h get-shipt-com.thesistestingapps.com
- Nikto v2.1.6

+ Target IP:      104.18.32.46
+ Target Hostname:  get-shipt-com.thesistestingapps.com
+ Target Port:    80
+ Message:        Multiple IP addresses found: 104.18.32.46, 172.64.155.210
+ Start Time:     2022-06-04 10:54:49 (GMT-4)

+ Server: cloudflare
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ All CGI directories 'found', use '-C none' to test none
found
D:Sat Jun  4 10:58:57 2022 'Request Hash' = {
  'Connection' => 'Keep-Alive',
  'Referer' => '() { ; } >[$$()] { echo 93e4r0-CVE-2014-6278: true; echo;echo; }',
  'User-Agent' => '(){ ; }; echo 93e4r0-CVE-2014-6271: true;echo;echo;',
  'whisker' => {
    'protocol' => 'HTTP',
    'ssl_certfile' => undef,
    'http_space2' => '',
    'include_host_in_uri' => 0,
    'uri_postfix' => '',
    'ssl' => 0,
}

Defined plugin macros:
@@EXTRAS = "dictionary;siebel;embedded"
@@ALL = "shellshock;docker_registry;dishwasher;dir_traversal;ssl;drupal;fileops;origin_reflection;sitefiles;report_json;negotiate;clientaccesspolicy;paths;apacheusers;embedded;report_sql;cgi;robots;multiple_index;put_del_test;struts;apache_expect_xss;report_xml;auth;tests;report_html;parked;content_search;siebel;favicon;report_text;report_nbe;report_csv;domino;cookies;headers;outdated;ms10_070;msgs;dictionary;httppoptions"
@@NONE = ""
@@DEFAULT = "@@ALL;-@@EXTRAS;tests(report:500)"
(expanded) = "ms10_070;tests(report:500);clientaccesspolicy;dir_traversal;dishwasher;report_html;outdated;sitefiles;report_xml;multiple_index;apacheusers;cgi;msgs;httppoptions;headers;struts;report_sql;domino;ssl;paths;report_json;report_text;favicon;negotiate;cookies;robots;report_csv;parked;put_del_test;report_nbe;fileops;origin_reflection;apache_expect_xss;shellshock;docker_registry;content_search;drupal;auth"
```

This is a plugin detail that is provided by the Nikto. In common, when checking for all subdomains it showed the same output as this.

OWASP ZAP

OWASAP Zed Attack Proxy is an open-source web application security scanner. With help of this tool, we can be able to identify web application issues without struggle. In pen-testing, this tool reduces the time. So, I started to check all websites with these tools.

When I check one of the sub-domains of **shipt.com**. it showed below output

The image consists of three vertically stacked screenshots of the OWASP ZAP (Zed Attack Proxy) application interface. Each screenshot shows the 'Alerts' tab selected in the top navigation bar.

- Screenshot 1:** Shows a list of 11 alerts. The first alert is highlighted: "Content Security Policy (CSP) Header Not Set".
 - URL: https://corporate.shipt.com/news
 - Risk: Medium
 - Confidence: High
 - Parameter:
 - Attack:
 - Evidence:
 - CVE ID: 693
 - WASC ID: 15
 - Source: Passive (10038 - Content Security Policy (CSP) Header Not Set)
 - Description: Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
 - Other Info:
- Screenshot 2:** Shows a list of 12 alerts. The second alert is highlighted: "HTTP/1.1 301 Moved Permanently".
 - Header: HTTP/1.1 301 Moved Permanently
 - Content-Length: 0
 - Date: Sat, 04 Jun 2022 19:48:48 GMT
 - Server: Microsoft-IIS/10.0
 - Location: https://corporate.shipt.com/news
 - Request-Header: ARRAffinity=e50c0e6ad085812de914c92ddade2528d51966499f44ed8424c2c97cc94d;Path=/;HttpOnly;Secure;Domain=corporate.shipt.com
 - Set-Cookie: ARRAffinitySameSite=e50c0e6ad085812de914c92ddade2528d51966499f44ed8424c2c97cc94d;Path=/;HttpOnly;SameSite=None;Secure;Domain=corporate.shipt.com
 - Request-Context: appId=cld-v1:7ea292f-f973-49db-a692-d66ebbd4aae89
 - X-Powered-By: ASP.NET
- Screenshot 3:** Shows a list of 12 alerts. The third alert is highlighted: "Cookie without SameSite Attribute".
 - URL: https://corporate.shipt.com
 - Risk: Low
 - Confidence: Medium
 - Parameter: ARRAffinity
 - Attack:
 - Evidence: Set-Cookie: ARRAffinity
 - CVE ID: 1275
 - WASC ID: 13
 - Source: Passive (10054 - Cookie without SameSite Attribute)
 - Description: A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks.
 - Other Info:

When I checked there were **eight** issues on that site. In those, CSP header not set, Cross-Domain misconfiguration, Missing anti-clickjacking header missing was

medium issues. So, an attacker can execute XSS, CSRF, and Clickjacking attacks on this site.

Then, I checked get.shipt.com ZAP also showed the same issues as before.

Screenshot of a web application security audit tool showing findings for <https://get.shipt.com>.

Missing Anti-clickjacking Header

- URL: https://get.shipt.com/
- Risk: Medium
- Confidence: Medium
- Parameter: X-Frame-Options
- Attack:
- Evidence: CVE ID: 1021, WASC ID: 15
- Source: Passive (10020 - Anti-clickjacking Header)
- Description: The response does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.

X-Content-Type-Options Header Missing

- URL: https://get.shipt.com/
- Risk: Low
- Confidence: Medium
- Parameter: X-Content-Type-Options
- Attack:
- Evidence: CVE ID: 893, WASC ID: 15
- Source: Passive (10021 - X-Content-Type-Options Header Missing)
- Description: The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

As an assumption, we can say that because of these missing headers, an attacker can able to perform XSS, CSRF, and, Clickjacking attacks on this site.

So on, I checked the support.shipt.com

There were **six** medium issues found by the zap. The server leaks the information via the ‘X-Powered-By’ header and the absence of anti-CSRF tokens most considerable issue on this site.

Burp Suite

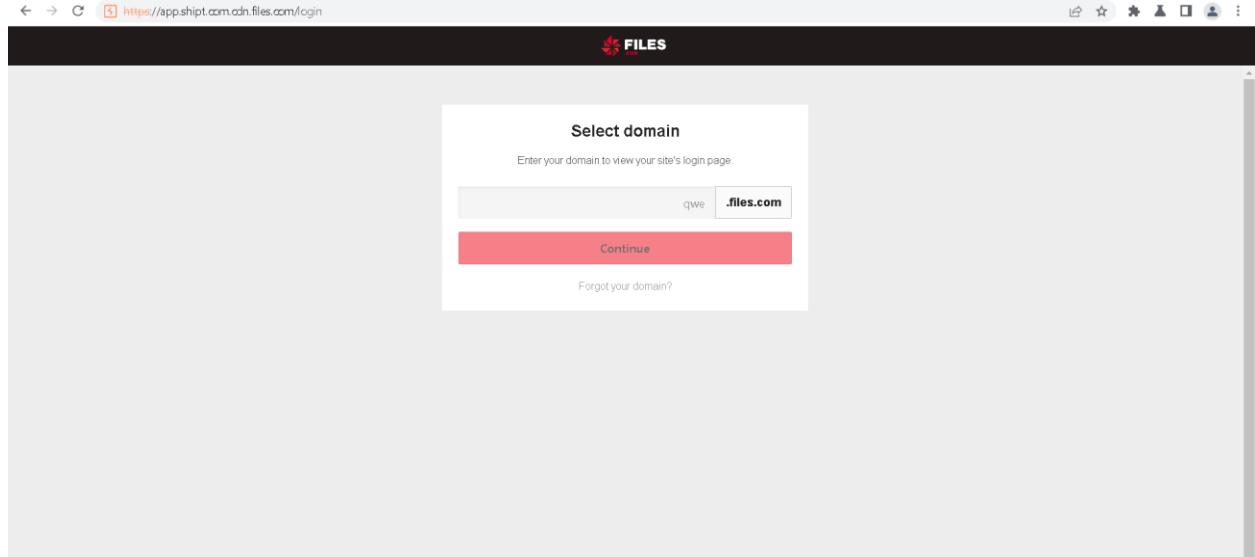
This is a well-known web application pen-testing tool that is used by professionals all around the world. This is come-up with two editions. There are enterprise edition and community edition. The community edition is free.

When I intercepted the traffic of help.shipt.com/pricing, these are the output I got a response.

So I decided to inject a basic SQL injection into this website to understand whether the website is a response or not to a SQL injection.

After SQL injection the response header responded with status code 200. From this instance, we can assume this site is vulnerable to SQL injection.

When I searched for, the vulnerable getting user input function which is not showing user input without encoding it in the URL. So on, I found **api.shipt.com** has a 3rd party service pointing to **api.shipt.com.cdn.files.com**. So, re-directed to that site. This showed to me like this,



Then I intercepted this website traffic to the burp suite repeater tab.

Request

```

1 GET /api/test/v1/settings/domain.json?domain=qwe.files.com HTTP/2
2 Host: app.shipt.com.cdn.files.com
3 Sec-CH-UA: "Not A Brand";v="0", "Chromium";v="102"
4 Accept: application/json
5 X-Files-React-Version: production-210
6 Sec-CH-UA-Mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-CH-UA-Platform: "Windows"
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: https://app.shipt.com.cdn.files.com/
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15
16

```

Response

```

1 HTTP/2 400 Bad Request
2 Server: npd
3 Date: Sun, 05 Jun 2022 08:08:26 GMT
4 Content-Type: application/json
5 Content-Length: 189
6 Cache-Control: no-cache, no-store, max-age=0, must-revalidate
7 Pragma: no-cache
8 Expires: Fri, 01 Jan 1990 00:00:00 GMT
9 X-Request-ID: 8602d0ff-a774-427f-9550-2902d2f40034
10
11 {
    "error": "Invalid request parameters: domain",
    "http-code": 400,
    "instance": "8602d0ff-a774-427f-9550-2902d2f40034",
    "title": "Request Params Invalid",
    "type": "bad-request/request-params-invalid"
}

```

Inspector

Selected text: qwe.files.com

Decoded from: Select ▾

Request Attributes: 2

Request Query Parameters: 1

Request Body Parameters: 0

Request Cookies: 0

Request Headers: 16

Response Headers: 8

Choose an attack type

Attack type: Sniper

Start attack

Payload Positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://app.shipt.com.cdn.files.com

Add \$

Clear \$

Auto \$

Refresh

```

1 GET /api/test/v1/settings/domain.json?domain=qwe.files.com HTTP/2
2 Host: app.shipt.com.cdn.files.com
3 Sec-CH-UA: "Not A Brand";v="0", "Chromium";v="102"
4 Accept: application/json
5 X-Files-React-Version: production-210
6 Sec-CH-UA-Mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
8 Sec-CH-UA-Platform: "Windows"
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: https://app.shipt.com.cdn.files.com/
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15
16

```

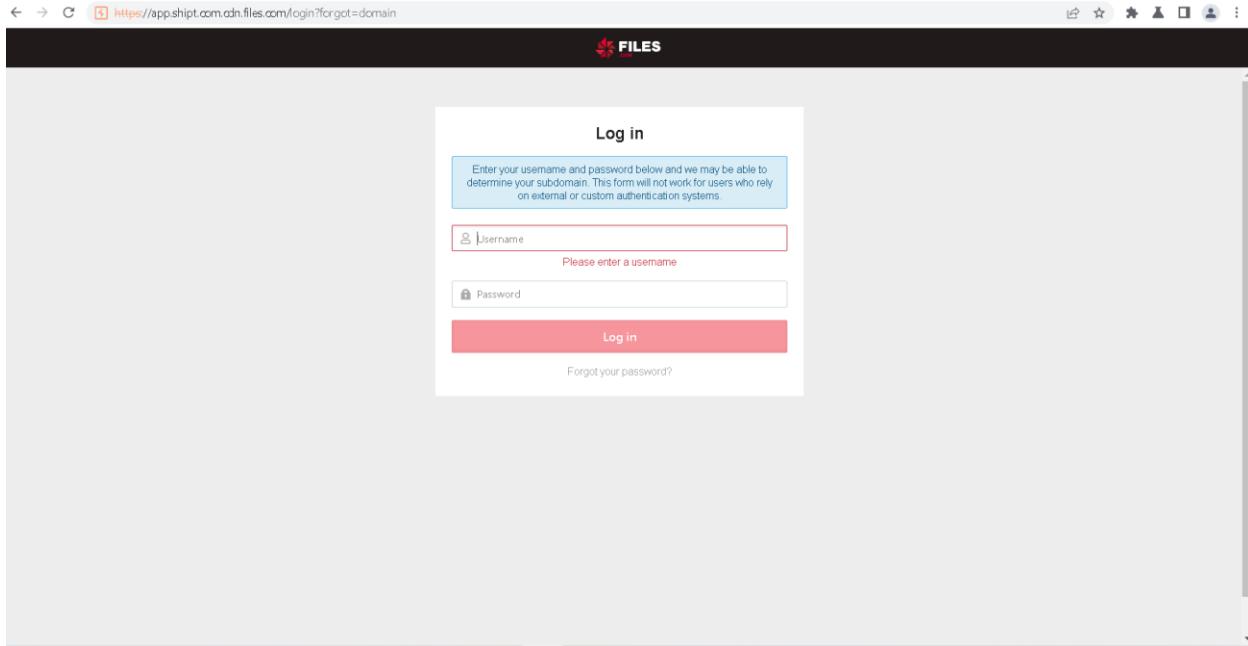
payload position

Length: 592

After that, I opened up the intruder tab then I added some XSS payloads to check whether that website is Vulnerable to the XSS.

There was no impact by my payload. But According to the header information, XSS is still possible on that website. But, My lack of knowledge in the Scripting and Payload is the reason that I could not get any output.

On that page, there was a link redirected to a login form. For an alternate, I thought to check whether it is vulnerable to a SQL injection or not.



So decided to perform SQL injection to this site as well.

Target: <https://app.shipt.com.cdn.files.com> | HTTP/2

Request

Pretty	Raw	Hex
POST /api/test/v1/sessions?interface=web HTTP/2		
Host: app.shipt.com.cdn.files.com		
Content-Length: 53		
Sec-Ch-Ua: "Not A/Brand";v="0", "Chromium";v="102"		
Accept: application/json		
Content-Type: application/json		
X-Powered-By: Express		
Sec-Ch-Ua-Mobile: <10		
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36		
Sec-Ch-Ua-Platform: "Windows"		
Origin: https://app.shipt.com.cdn.files.com		
Sec-Fetch-Site: same-origin		
Sec-Fetch-Mode: cors		
Sec-Fetch-Dest: empty		
Referer: https://app.shipt.com.cdn.files.com/		
Accept-Encoding: gzip, deflate		
Accept-Language: en-US,en;q=0.5		
{		
"password": "123456789",		
"username": "administrator--"		
}		

Response

Pretty	Raw	Hex	Render
HTTP/2 401 Unauthorized			
Server: nginx			
Date: Sun, 06 Jun 2022 08:43:53 GMT			
Content-Type: application/json			
Content-Length: 201			
Cache-Control: no-cache, no-store, max-age=0, must-revalidate			
Pragma: no-cache			
Expires: Fri, 01 Jan 1960 00:00:00 GMT			
X-Request-ID: fc748e1d-760c-4fb0-b3f0-f5ca8a83c027			
11 {			
"error": "Invalid username or password",			
"http-code": 401,			
"instance": "fc748e1d-760c-4fb0-b3f0-f5ca8a83c027",			
"title": "Invalid Username Or Password",			
"type": "not-authenticated/invalid-username-or-password"			
}			

Request Attributes: 2

Request Query Parameters: 1

Request Cookies: 0

Request Headers: 19

Response Headers: 8

0 matches | Search... 0 matches | Search... 0 matches | Search... 0 matches | Search... 506 bytes | 1,034 millis

Intercepted the traffic to burp, then sent it back to the repeater.

The screenshot shows the Burp Suite interface with the 'Payload Positions' tab selected. A POST request is captured with the following payload:

```
POST /api/rest/v1/sessions?interface=$web HTTP/2
Host: app.shipt.com.cdn.files.com
Content-Length: 53
Sec-Ch-Ua: "Not A Brand";v="0", "Chromium";v="102"
Accept: application/json
Content-Type: application/json
X-Files-React-Version: production-210
Sec-Ch-Ua-Mobile: 10
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36
Sec-Ch-Ua-Platform: "Windows"
Origin: https://app.shipt.com.cdn.files.com
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://app.shipt.com.cdn.files.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
("password": "$1$34467098", "username": "$administrator--$")
```

Below the payload, there are buttons for 'Add \$', 'Clear \$', 'Auto \$', and 'Refresh'. At the bottom, there are search and filter buttons.

Then I added some SQL injection payloads.

The screenshot shows the Burp Suite interface with the 'Payload Sets' tab selected. A simple list payload set is defined with 198 payloads and 594 requests. The payload list includes various SQL injection variants such as 'or 1=1', 'or 1=1#', etc., and a specific payload '1234' AND 1=0 UNION ALL--'. The intruder attack view shows the request for payload 56, which contains the union attack payload.

Request	Response
Pretty	Raw
1 POST /api/rest/v1/sessions?interface=\$web 1234' AND 1=0 UNION ALL-- HTTP/2 Host: app.shipt.com.cdn.files.com Content-Length: 53 Sec-Ch-Ua: "Not A Brand";v="0", "Chromium";v="102" Accept: application/json Content-Type: application/json X-Files-React-Version: production-210 Sec-Ch-Ua-Mobile: 10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36 Sec-Ch-Ua-Platform: "Windows" Origin: https://app.shipt.com.cdn.files.com Sec-Fetch-Site: same-origin Sec-Fetch-Mode: cors Sec-Fetch-Dest: empty Referer: https://app.shipt.com.cdn.files.com/ Accept-Encoding: gzip, deflate Accept-Language: en-US,en;q=0.5 ("password": "\$1\$34467098", "username": "\$administrator--\$")	64 of 594

In final, unfortunately, I could not get any positive response.

Conclusion

The main reason for this web audit is to find security vulnerabilities and report back to the organizations. At first, I started to identify the DNS records and CNAMEs of domains listed on a scope list. Then I moved forward to check each subdomain and their DNS entry pointing websites. Then I validated each DNS entry pointing to the website CA that they are expired or not. Then I submitted a Subdomain Takeover vulnerability to **HackerOne**. Then I inspected each website's Response header information. When I was doing it, I found some security issues related to service subdomains. Those issues are,

- CSP header not set.
- Cross-domain misconfiguration.
- Missing anti-clickjacking header.
- Cookie with same-site attribute.
- Server leaks information.
- X-Content-Type-Option header missing.
- Cookie without secure flag.
- Cross-domain JavaScript source file inclusion.

Expected attacks,

- XSS.
- CSRF.
- ClickJacking.
- Information Leak.
- Man in the Middle attack.