Privacy and Censorship using ESNI

(Final Report)

Dhruv Rauthan

(2019A7PS0095G)

Project Instructor: Dr. Vinayak Naik

Problem Statement:

1. To understand the recent advancements in network security and analyse the different approaches towards censorship circumventions
2. To build software to circumvent censorship agencies' firewalls by using ESNI and the concept of domain hiding

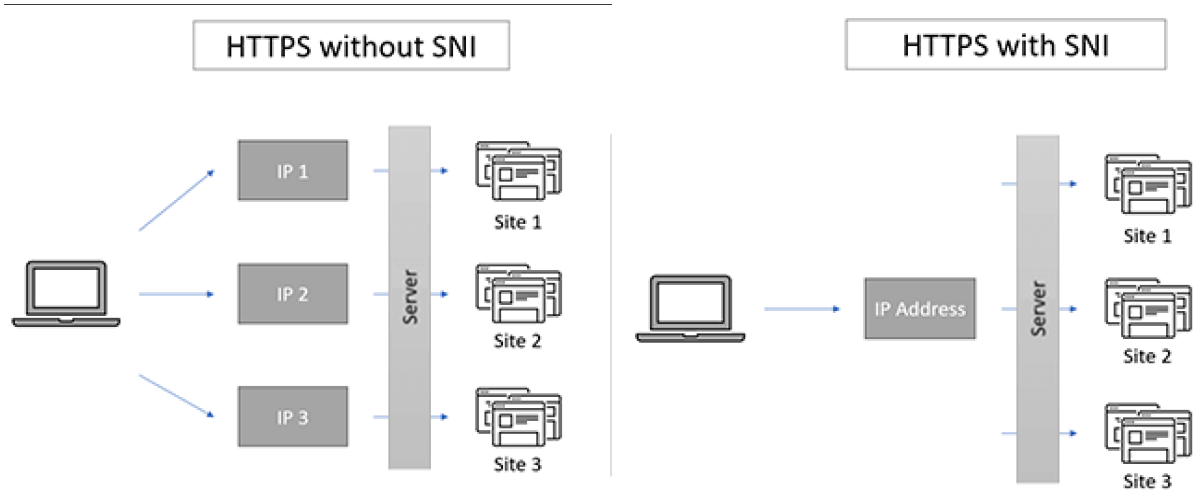
Related Work

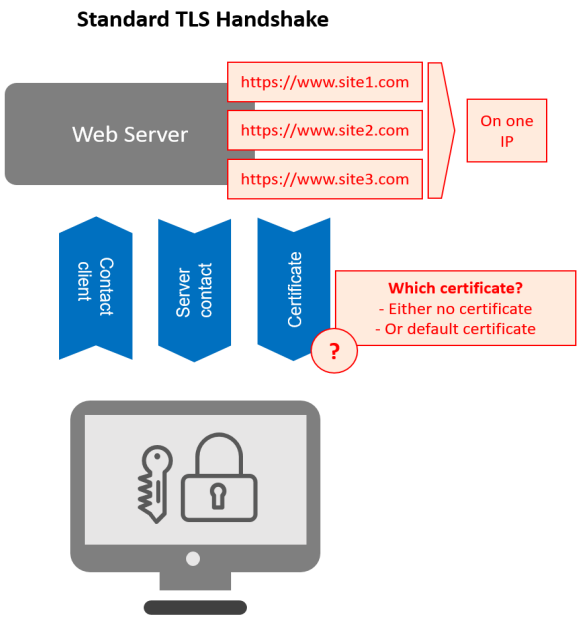
1. SiegeBreaker: An SDN Based Practical Decoy Routing System https://petsymposium.org/2020/files/papers/issue3/popets-2020-0051.pdf
2. esni-rev-proxy https://github.com/devopsext/esni-rev-proxy

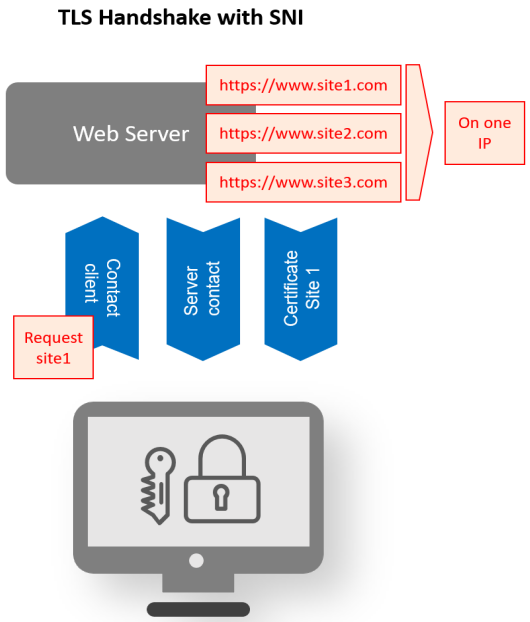
What is SNI?

- Server Name Indication
- TLS protocol extension
- Solves the issue of shortage of IPv4 addresses
- Allows for hosting multiple domains on the same IP address. The server knows which website's SSL certificate to present when a client sends a request to the IP address

- Server returns the required SSL certificate to verify authenticity of the website
- Included in TLS handshake ClientHello message
- Unencrypted and hence, attacker/firewall can observe domains visited by the user

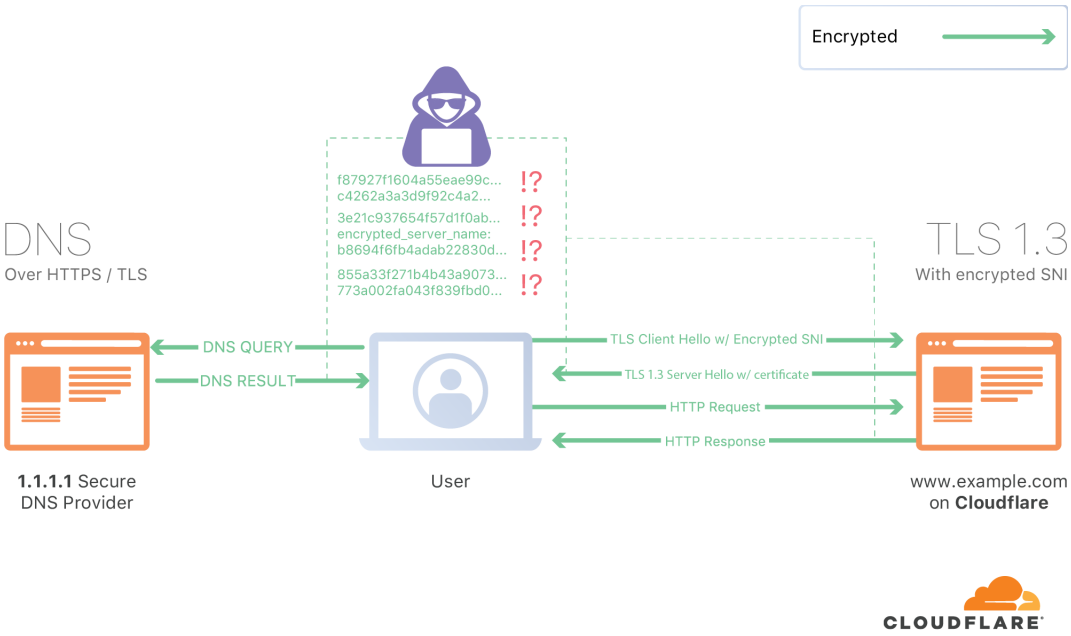




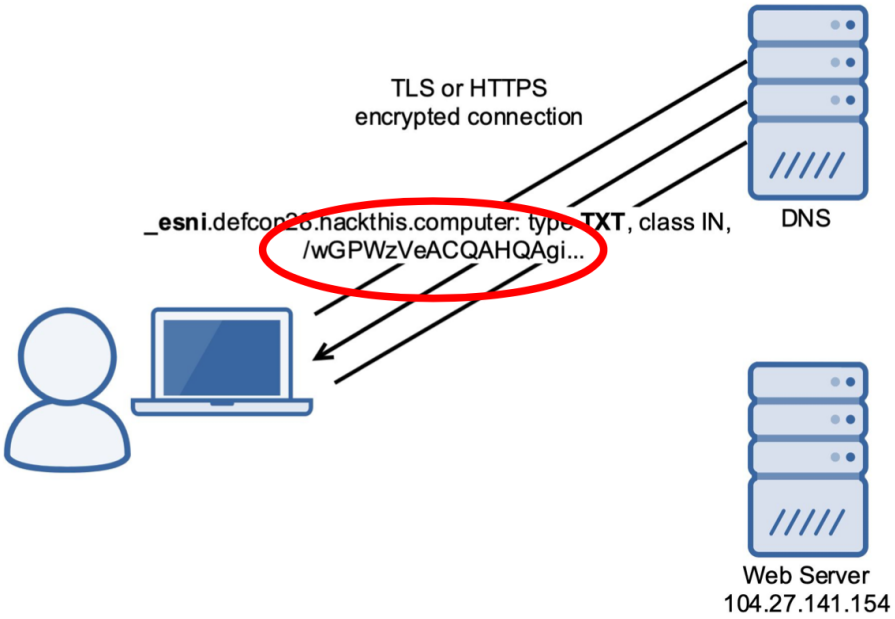


What is ESNI?

- Encrypted Server Name Indication
- Utilized to maintain privacy of users
- Encrypts the SNI field in the header
- Uses public key cryptography- client encrypts the SNI field using the web server's public key added in the DNS record itself (DoH)
- If we use only ESNI then simply Cloudflare can be used instead of hosting a proxy web server



- Attacker may poison or intercept DNS communication so it is essential to use DoH (DNS over HTTPS) to secure that traffic

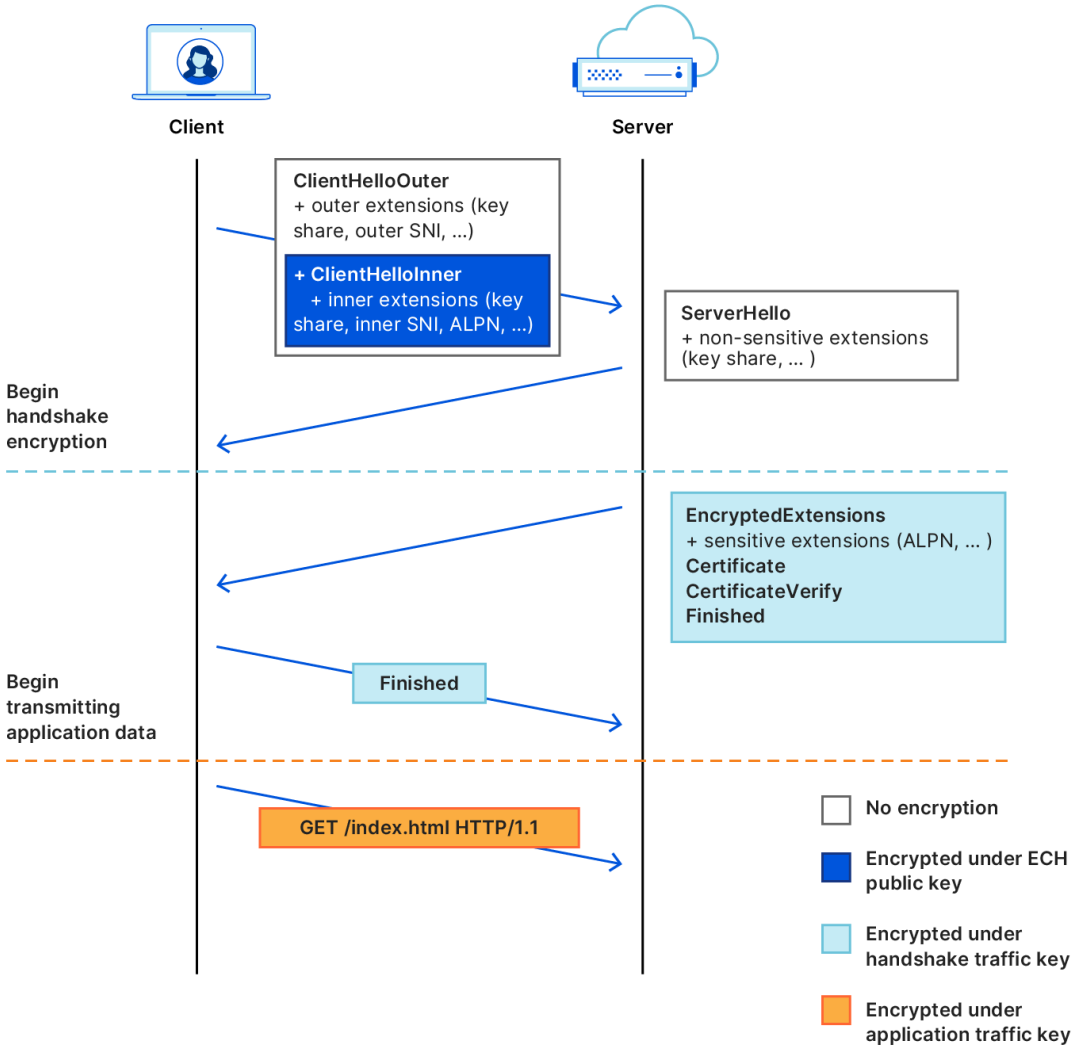


What is ECH?

- Encrypted Client Hello
- TLS protocol extension
- Evolution of the ESNI draft
- Encrypts the entire ClientHello, including ESNI, ALPN and other metadata

How does it work?

- Involves 2 ClientHello messages- ClientHelloOuter (unencrypted) and ClientHelloInner (encrypted)
- ClientHelloInner is composed of the handshake parameters the client wants to use for the connection, including the SNI of the origin server, the ALPN list etc.
- It is encrypted using the public key in the DNS record of the website
- ClientHelloOuter is not used for the intended connection, but is instead used to connect to the ECH front server.
- If the connection is established using ClientHelloOuter (due to decryption failure of the ClientHelloInner etc), the connection is aborted and the front server sends back a fresh public key for the client to retry to establish the connection



Advantages over ESNI

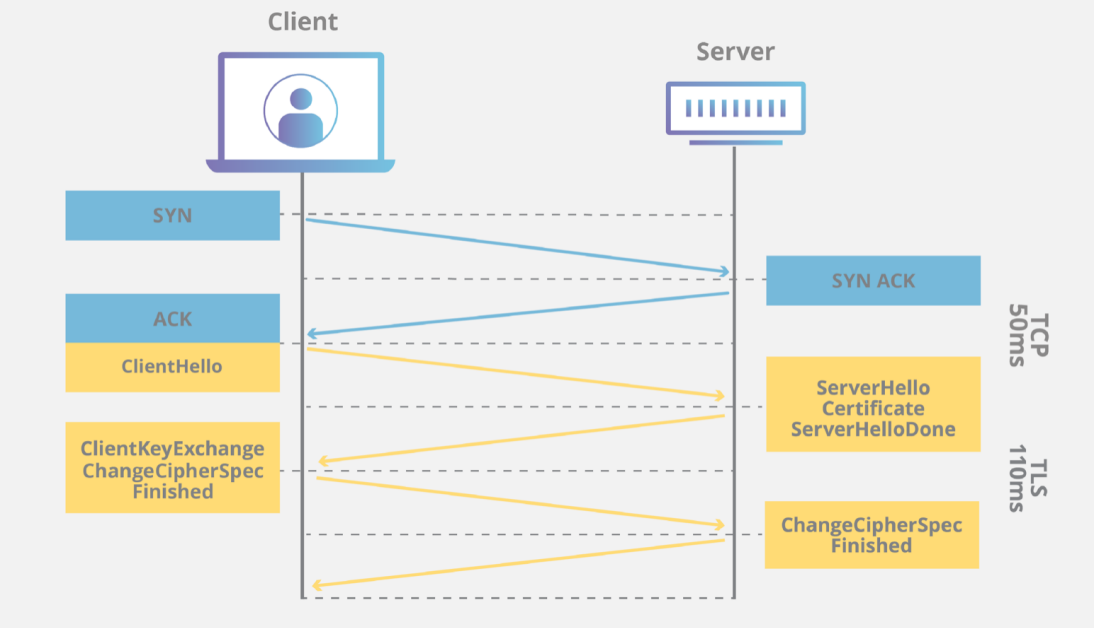
- While the ESNI server aborts the connection if decryption of the packet fails, the ECH server attempts to complete the handshake and supplies the client with a fresh public key
- Encrypts metadata other than ESNI, improving overall security

Aim with ESNI

- Our aim with ESNI was to make possible the circumvention of firewalls and third-party agencies snooping on the client's packets
- The idea consisted of a front web server which acts as a proxy for requests made by the client to a blacklisted destination
- The key was that this server could switch IP addresses, so as to not be blacklisted by the firewalls itself
- The proxy requests are made by manipulating the ESNI value to store the blacklisted destination's SNI which could be decrypted by the web server

How does it work?

1. Client initiates TCP handshake with the server. The request is made to an overt website (the front server) so the firewall allows the connection to pass through
2. After completing the TCP connection, the TLS handshake is initiated. (This handshake is unencrypted, a problem which is solved with ECH)
3. The cipherspecs are exchanged to allow for encryption of SNI data

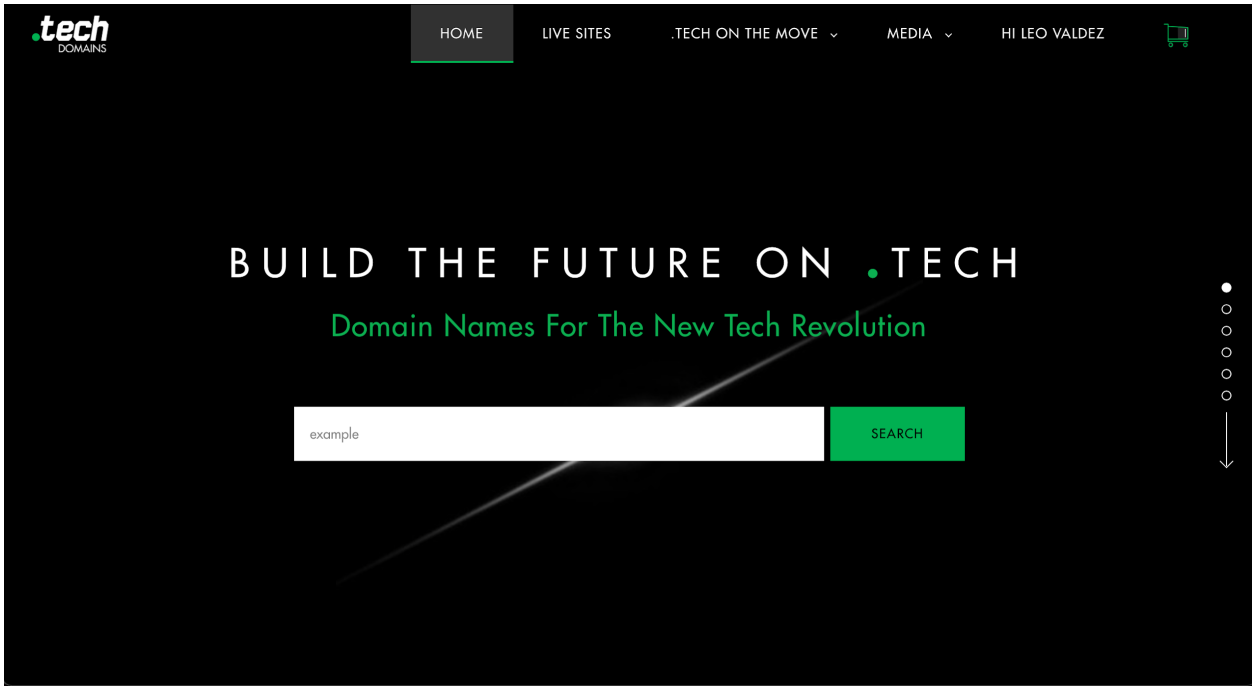


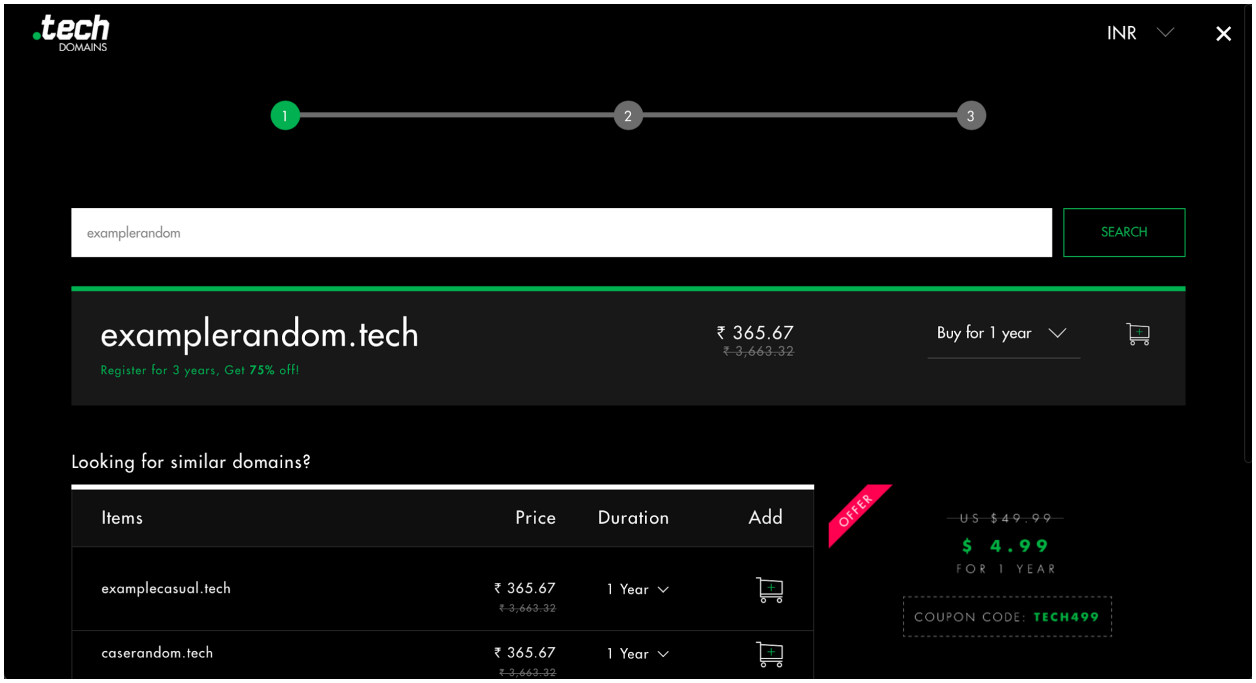
4. User sends a HTTPS packet to the IP address containing both the SNI and the ESNI fields. The first encrypted packet must contain the IP address and the SNI of the covert destination in the ESNI field
5. The SNI field has the server_name extension of the web server which is whitelisted by the organization's firewall. The ESNI is encrypted using the key exchanged in the TLS handshake and contains the server_name extension of the covert destination
6. The web server which has its own private key decrypts the ESNI header field and extracts the covert destination's server_name and the IP address of the covert destination
7. Now, this server simply acts as a proxy. It relays all packets between the covert destination and the user
8. The user modifies their HTTPS GET request according to the covert destination's website
9. When the user wants to terminate the connection, it sends a TCP FIN packet and the TCP connection is terminated normally

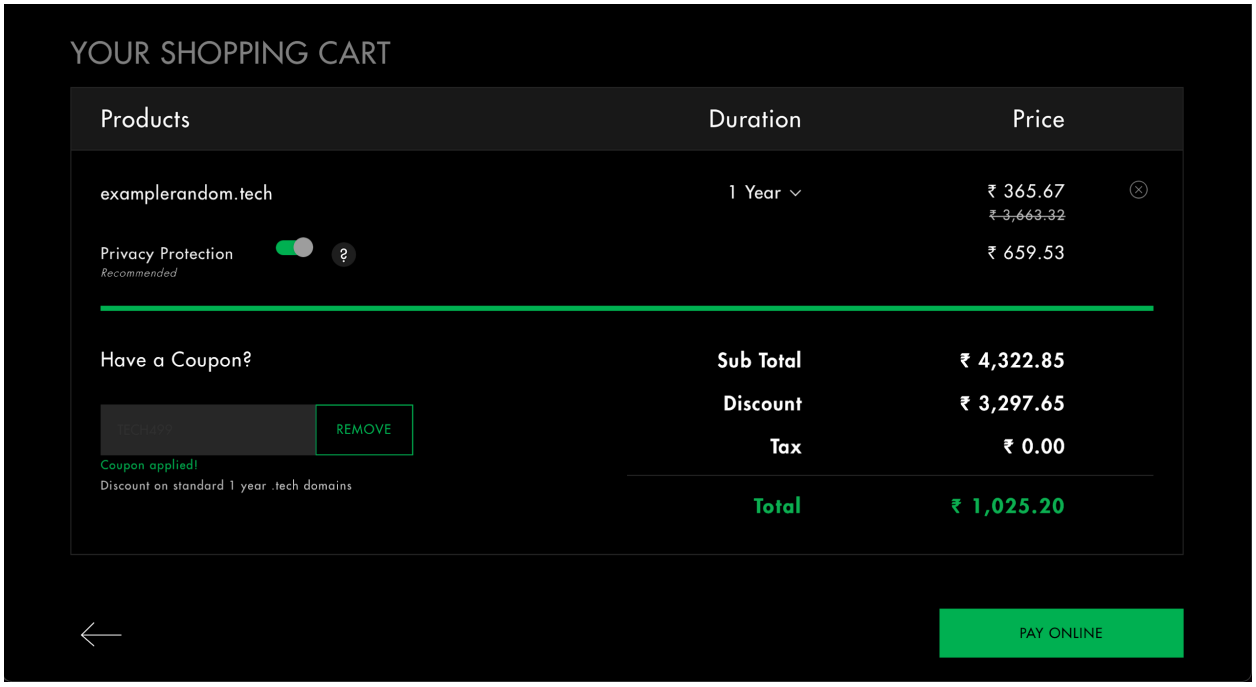
Implementation

Get domains for 2 websites

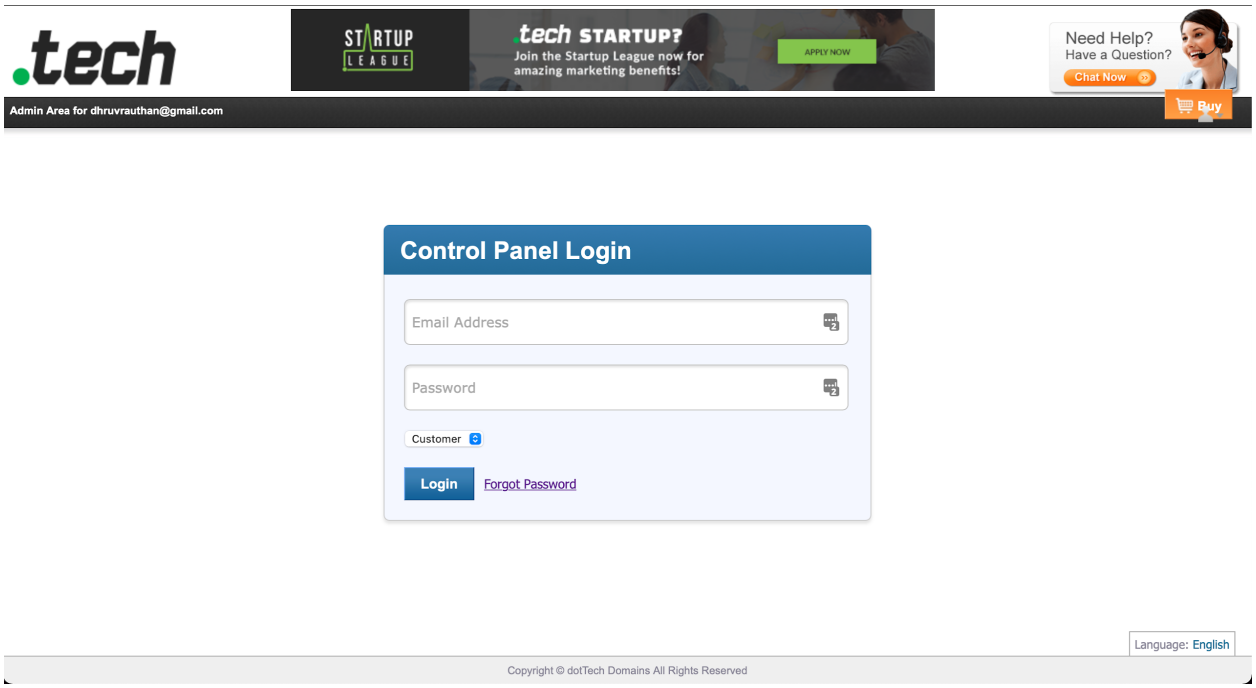
Visit https://get.tech/#tech-domain and enter the domain name you want to acquire

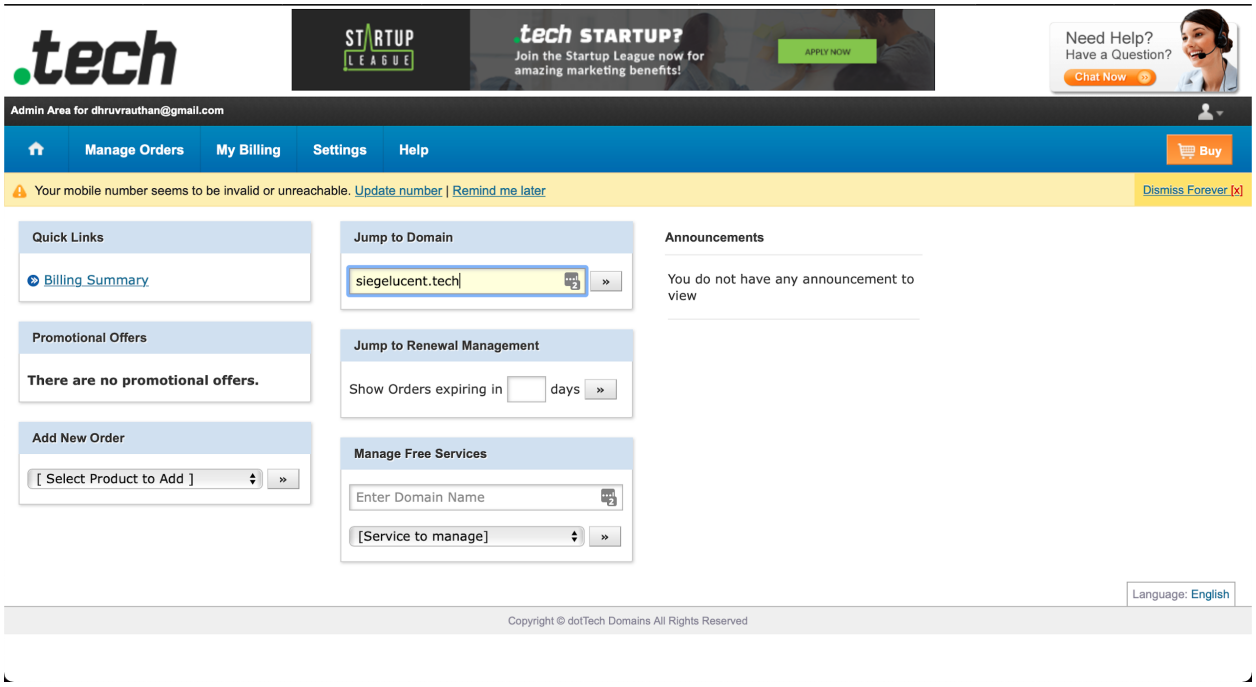

.tech
DOMAINS
HOME
LIVE SITES
.TECH ON THE MOVE
MEDIA
HI LEO VALDEZ
BUILD THE FUTURE ON .TECH
Domain Names For The New Tech Revolution
example
SEARCH


.tech
DOMAINS
INR
1
2
3
examplerandom
SEARCH
examplerandom.tech
Register for 3 years, Get 75% off!
₹ 365.67
₹ 3,663.32
Buy for 1 year
Looking for similar domains?
Items
Price
Duration
Add
examplecasual.tech
₹ 365.67
₹ 3,663.32
1 Year
caserandom.tech
₹ 365.67
₹ 3,663.32
1 Year
OFFER
US $49.99
$ 4.99
FOR 1 YEAR
COUPON CODE: TECH499



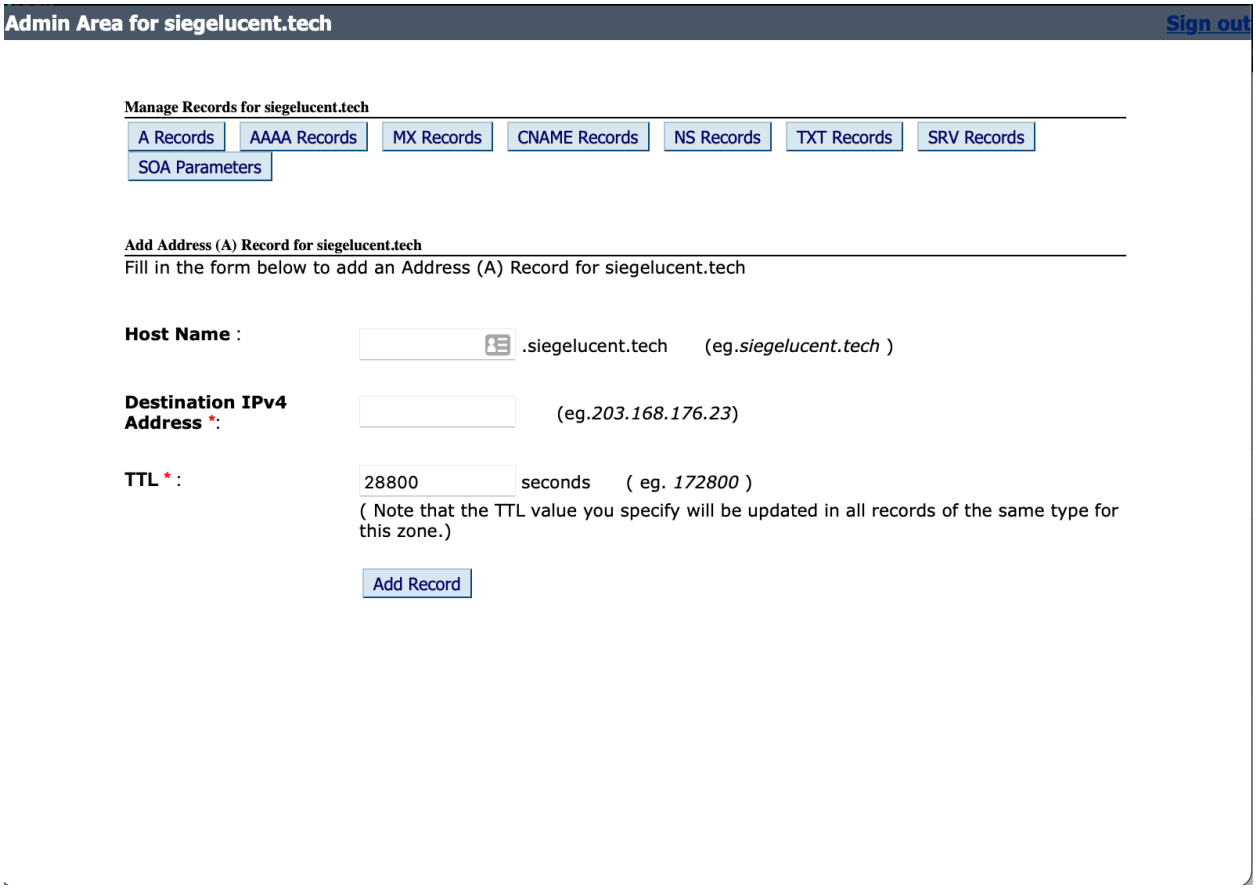
Visit https://controlpanel.tech to add subdomains for the 2 websites





Scroll down to the “DNS Management Section” and select “Manage DNS”

Select “Add Record”

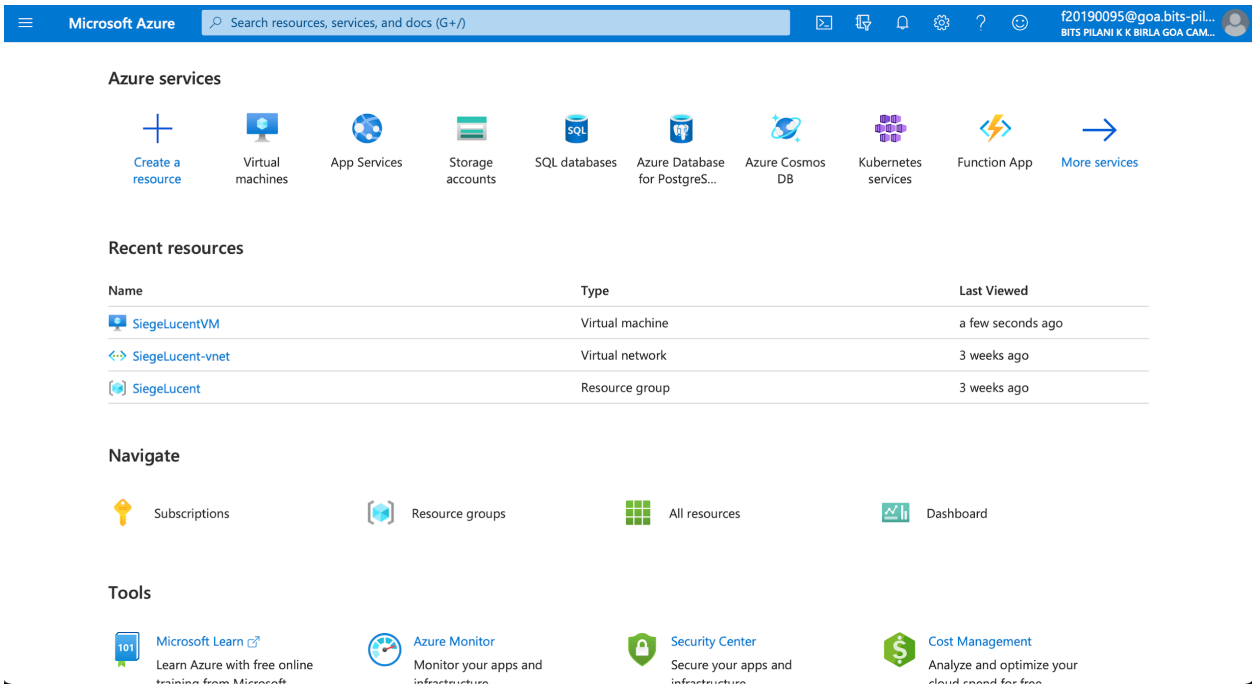


(We will acquire the destination IPv4 address in the next step)

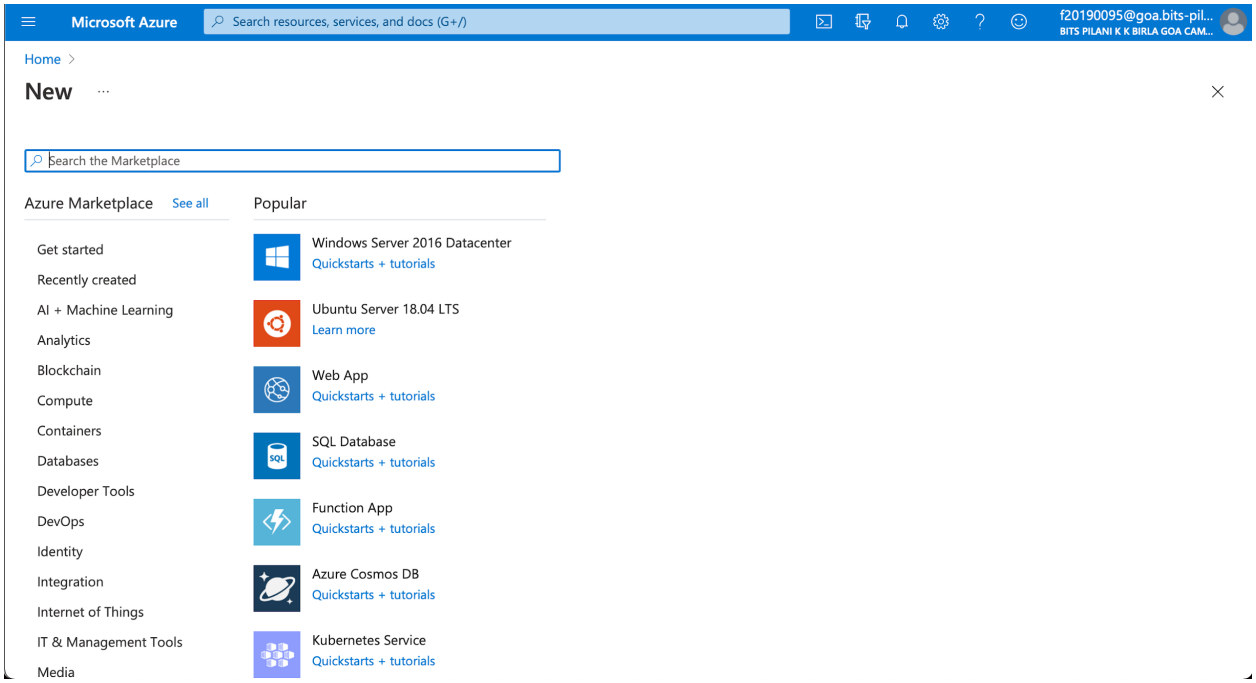
Add the 2 subdomains you would like to use

Create web server

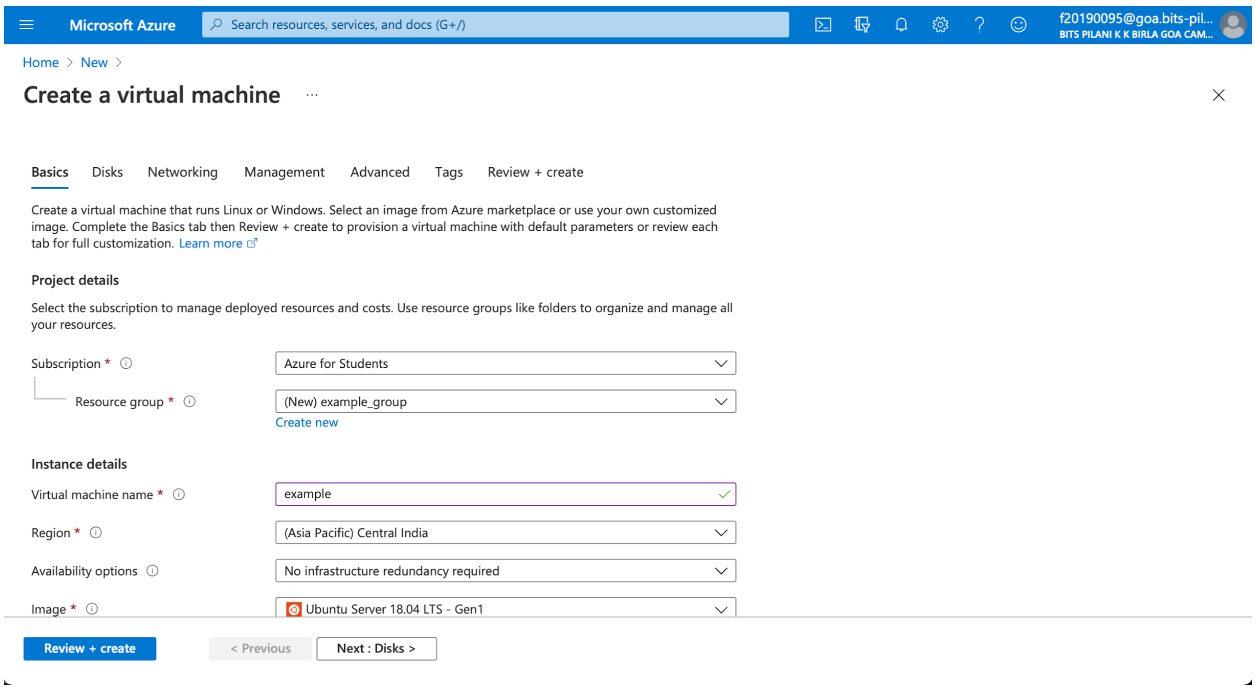
Visit https://portal.azure.com/#home to create a web server



Create a new resource



Select Ubuntu Server 18.04 LTS



Adjust details as required and click on “Review + create”

An SSH key will be downloaded to your computer which is used to connect to your server:

```
ssh -i SiegeLucentVM_key.pem azureuser@104.211.118.26
```

The Azure VM will have an IP address which will be entered in the .tech control panel for the subdomains

Create 2 websites on the Apache server

Follow the procedure as given here:
https://www.atlantic.net/vps-hosting/host-multiple-websites-on-a-single-server-with-apache-on-ubuntu-18-04/

Use the IP address of the Azure VM and the subdomain names that you entered in the .tech control panel

Welcome to test1.siegelucent.tech Website

This is my second website hosted with name-based virtual hosting

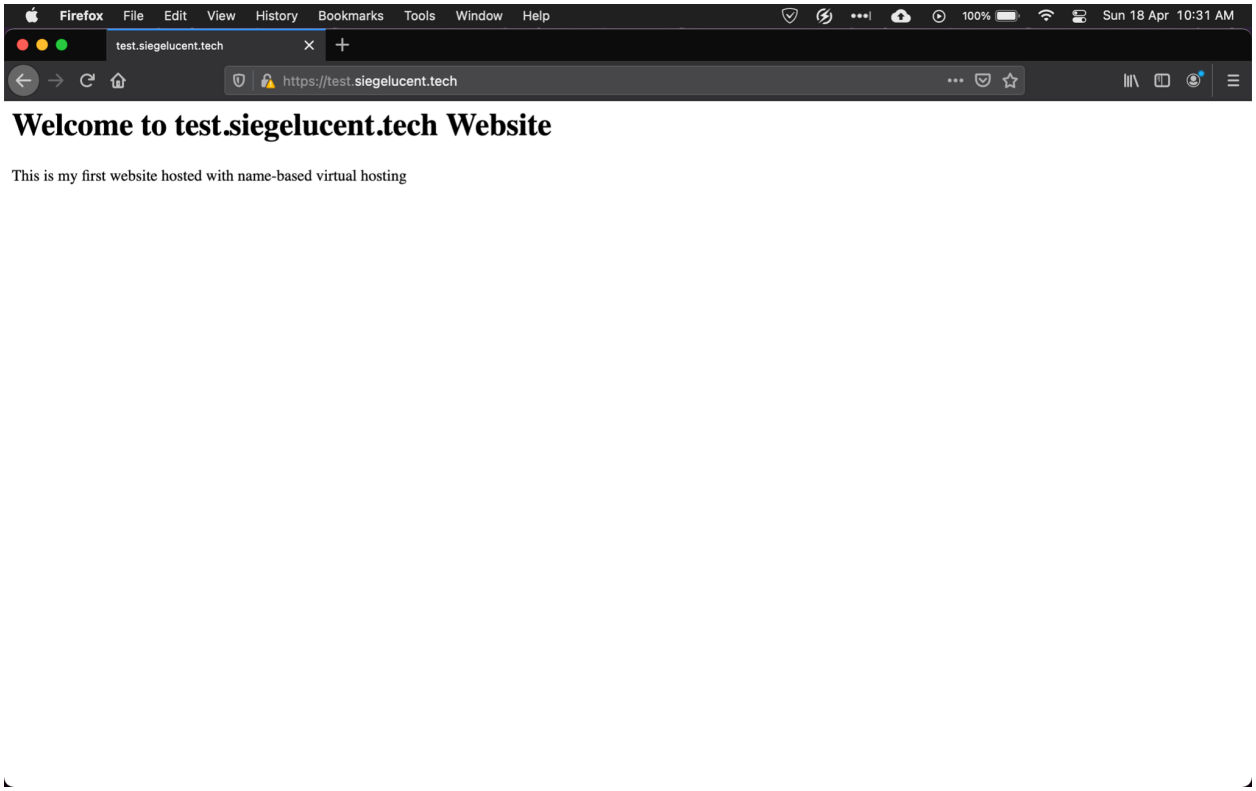
Success! The siegelucent 2.0 virtual host is working!

Enable HTTPS on the websites

You can set up multiple SSL certificates on 1 IP addresses with the help of the following articles:

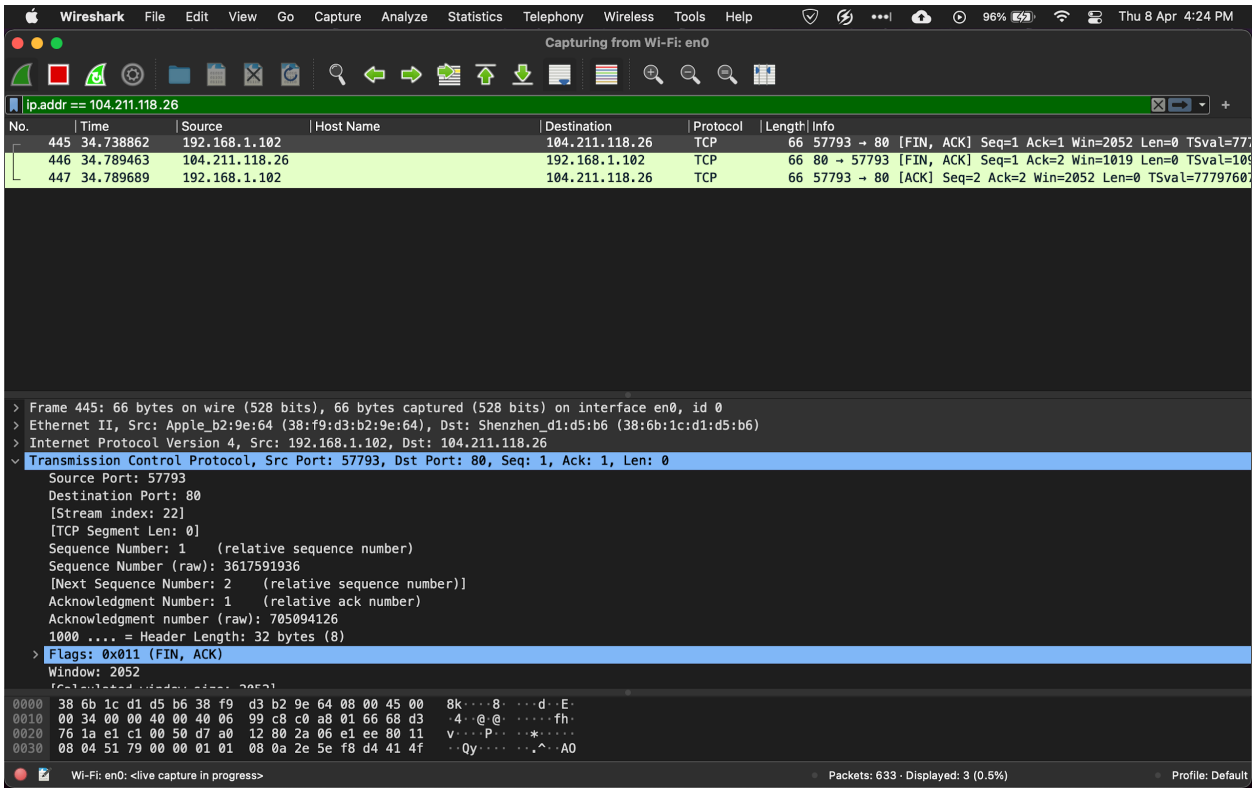
https://www.digitalocean.com/community/tutorials/how-to-set-up-multiple-ssl-certificates-on-one-ip-with-apache-on-ubuntu-12-04

https://www.digicert.com/kb/ssl-support/apache-multiple-ssl-certificates-using-sni.htm

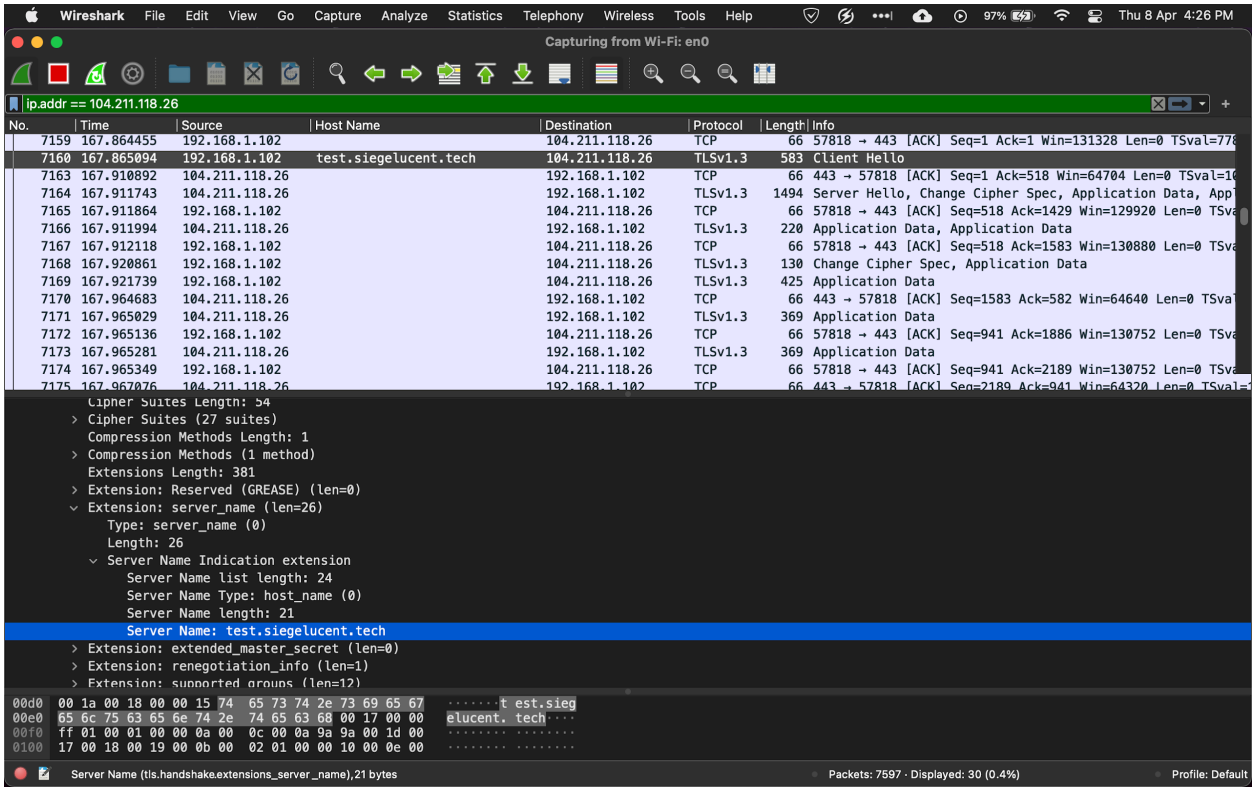


Testing using Wireshark

1. Checking the HTTP unencrypted website



2. Checking the HTTPS secure website which has the SSL certificate



We can see that the server_name value has the name test.siegelucent.tech which we were expecting as that website has an SSL certificate

Enable ESNI using a reverse proxy

ESNI was still a draft at the time of this project, and there was no set industry standard yet. Web servers such as Apache and nginx did not support ESNI, since they work with TLS using OpenSSL. And OpenSSL did not support ESNI, since it was still a draft (https://github.com/openssl/openssl/issues/7482)

An ESNI reverse proxy was the first option. A reverse proxy server is a server which usually sits behind a firewall and directs requests to the appropriate backend server, which in our case would be the Apache server on the AzureVM. https://github.com/devopsext/esni-rev-proxy is a Github repository which can build a reverse proxy server to help enable ESNI on our server.

The process for setting up the reverse proxy is elaborated in the README of the GIthub repository. However, there are a few changes we need to make in the Apache2 configuration:

1. Changing the ports.conf

```
sudo nano /etc/apache2/ports.conf
```

   Now comment out the Listen 443 line

```
#Listen 443
```

2. Change the website's .conf file

```
sudo nano /etc/apache2/sites-available/test.siegelucent.tech.conf
```

   Change <VirtualHost *:443> to

```
<VirtualHost *:8080>
```

3. Restart Apache

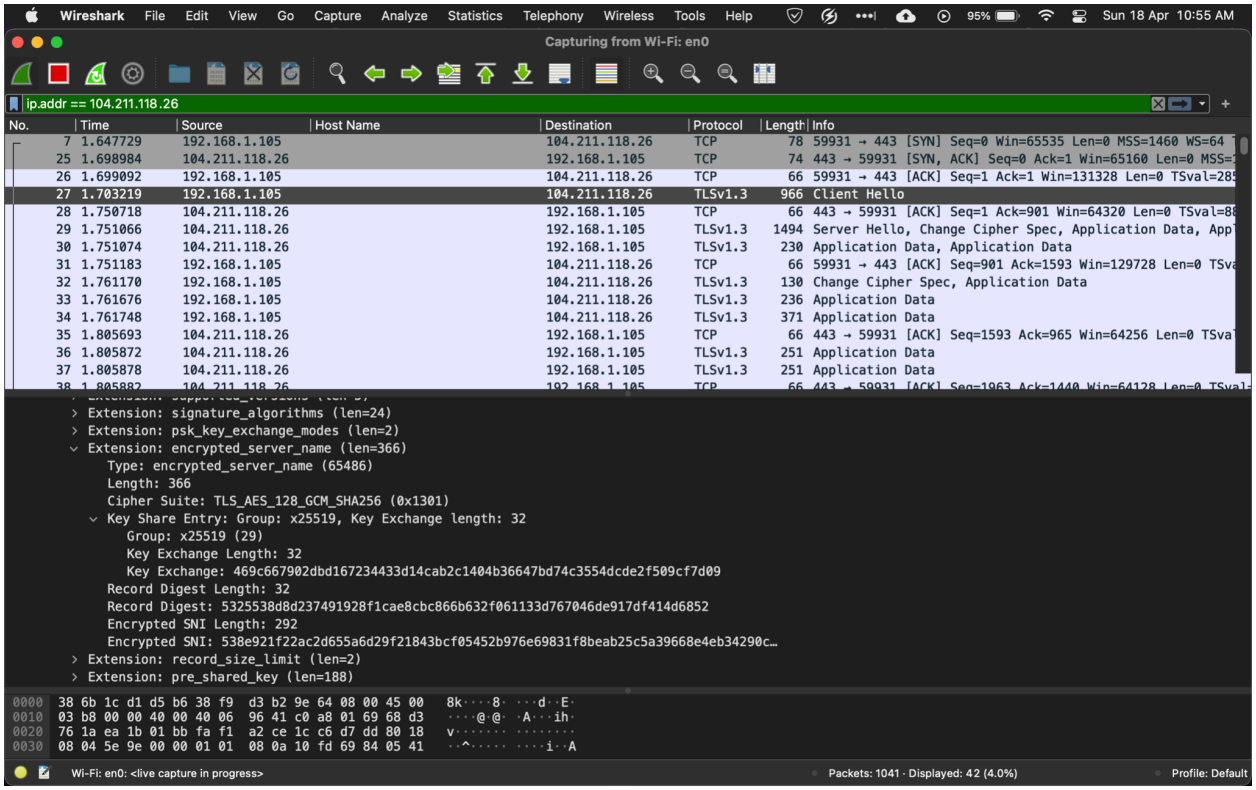
```
sudo service apache2 restart
```

After following the procedure to set up the reverse proxy from the repository, the final command to start the proxy should look something like this:

```
sudo ./esni-rev-proxy -b 0.0.0.0:443 -esni-keys ~/test/test2/esnitool/esni.pub
-esni-private ~/test/test2/esnitool/esni -cert
"test.siegelucent.tech:/etc/apache2/testsiegelucenttech.key:/etc/apache2/testsiegelucen
ttech.crt" -upstream https://test.siegelucent.tech
```

Testing ESNI using Wireshark and Firefox

First we need to enable ESNI in Firefox. Follow the procedure given in the link
https://techorbiter.com/how-to-enable-encrypted-sni-in-firefox-esni/3959/
(This requires an older version of Firefox, since the newer ones have stopped their support for ESNI)



Clearly, the ClientHello's server_name extension is now changed to encrypted_server_name and is fully encrypted

The evolution of ESNI

One more possibility was an unofficial fork of OpenSSL in which the author had included support for ESNI in Apache and nginx web servers.
https://github.com/sftcd/openssl
However, there were problems running the code since the ESNI folder had not been updated regularly with newer versions of Go and OpenSSL. An issue was raised with the author (who was a member of the Internet Architecture Board, which is a part of the IETF). They informed us that ESNI had evolved to ECH over the past few months and that ECH, rather than ESNI, was set to be the industry standard in the future

Integrating ECH into SiegeBreaker

- https://petsymposium.org/2020/files/papers/issue3/popets-2020-0051.pdf
- SiegeBreaker is an approach which uses decoy routing in networks as routers
- Users in the networks send specially crafted packets to an uncensored website. Once out of the network, these packets encounter a “Decoy Router” which identifies the user using a handshake and then proxies traffic from the user to the censored domain
- To incorporate ECH into the SiegeBreaker design, we need to change a few details in the process

How does it work?

(Bootstrapping)

1. The client initiates DR by sending an email to the controller’s email address. The payload of this email contains the DR request which is encrypted using the public key of the router
2. The email has 4 fields **(1)** The word “SIEGE”, signifying DR request. **(2)** A TCP ISN that the client will eventually use while handshaking with the overt destination (OD). **(3)** IP addresses of client and the OD as well as the SNI value of the OD’s domain **(4)** A DH exponent public value (g^x)
3. On successfully identifying a DR request, the controller installs a rule on the SDN switch, so that all client–OD packets are redirected to it
4. The client initiates a fresh TCP handshake with the OD. The SYN packet of the handshake bears the *same* ISN, as the one sent in the initial email. This packet is redirected to the controller who checks that its ISN matches the ISN specified in the initial email and forwards it to OD. Thereafter the client completes the TCP connection, and initiates a TLS handshake with the OD, negotiating the session

key.

Here, the ClientHelloInner is encrypted and contains the SNI, ALPN etc. of the OD. ClientHelloOuter contains the unencrypted SNI of the front server of the OD, which should also be uncensored.

(Hijacking)

5. After completing the TLS handshake, the client sends a TLS data packet, carrying the GET request. This packet goes to the controller which *replaces the payload* of the TLS data packet with the client's DH exponent g^x and the OD IP address and SNI. It then forwards this packet to the SP.
   The controller diverts *all subsequent packets of the flow* to the SP.
6. The SP, on receiving the first TLS data packet, assumes that the client intends to use DR and terminates the existing client–OD connection. Further, the SP derives a pre-master key (PMK) (g^{xy}) using client's g^x and its own DH private number, y. The SP (spoofing as OD) then crafts a TLS data packet carrying a random nonce N, along with its HMAC computed using PMK. To the censor, this appears to be a regular TLS data packet, carrying random bits. The client, however, treats these bits as a nonce N and calculates its HMAC. It derives the PMK (g^{xy}) using the private DH number x, and the publicly known g^y. Successful verification of the HMAC allows the client to confirm that the DR request was successful.

(Proxying)

7. Thereafter the client crafts a TLS data packet carrying Nc, the CD IP address and SNI (encrypted with key Kc using 256-bit AES in CBC mode), and a HMAC of the URL and Nc (using HMAC key Hc). The client sends this packet, addressed to the OD. En-route the packet encounters the switch that redirects it to the SP.
8. SP, on successful reception of the aforementioned TLS data packet, extracts the nonce Nc . Using Nc , N and the PMK, SP also computes the same 6- tuple as

the client. Upon successful HMAC validation, the SP decrypts the IP and SNI (using Kc). Finally, SP connects to the CD via ECH and requests data.

9. CD serves the requested content to SP.
10. The SP encrypts these responses with key Ksp and signs them with HMAC key Hsp. It then sends them back to the client, spoofing the source IP address of the OD (maintaining the state of Client–OD connection). This keeps up the pretense, to the client's censor, that the client is communicating with the OD. The same session can also be used for requesting content from various other CDs (before the idle timeout expires).

Conclusion

ESNI is a step forward in maintaining privacy and anonymity on the internet. This combined with DoH can ensure that no third party can observe network traffic and pinpoint the domains visited by the user. We achieved our aim of enabling ESNI on our own server which could further be improved so that the user can browse any internet domain without being restricted in any way.

As the defence strengthens, so does the attack. Hence, security drafts are continuously updated and improved to keep up with the industry standards. Over time, ESNI evolved into a much better ECH, which builds upon the flaws of the former. However, the work done on ESNI is not in vain, since ECH will most likely involve the same principles and ideas as the ones implemented in this project. Hopefully ECH will become the standard soon, to ensure that the experience of true anonymous browsing can become a reality.