Assignment part 2 Sliding:

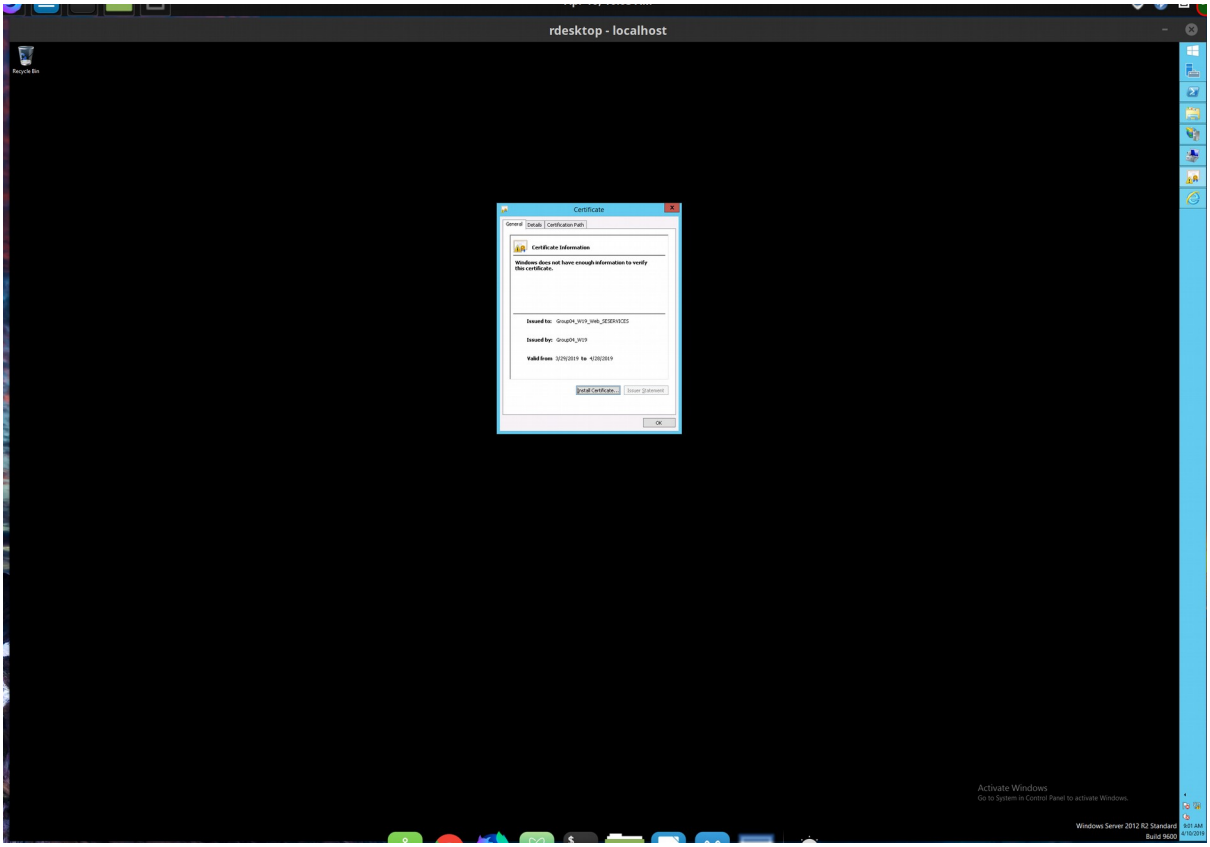
For step 1 we used the command '*openssl genrsa -out private.key 2048*' to create our private key
And the command ' *openssl rsa -in private.key -pubout -out public.key*' our private key is the Ca that we used to sign the certificate

For step 2 we used the command '*openssl genrsa -out step2.key 2048*' to create a new key. We also used the command '*openssl req -new -key step2.key -out step2.csr'* to create the certificate request. We made the organization name Group04_W19_Web_Services as the organization name when asked for it. We left all other values blank. For the private key the password was hello.

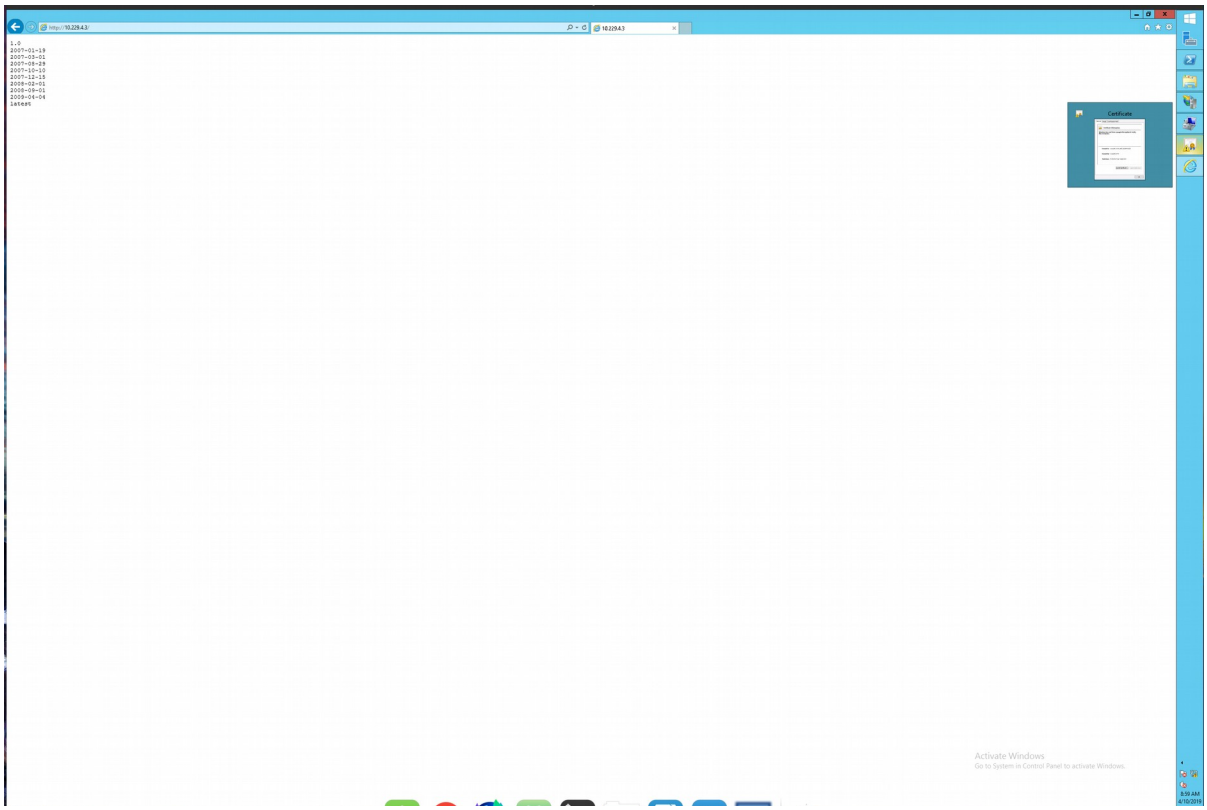
For step 3 we signed the certificate with our CA with the command ' *openssl x509 -req -in step2.csr -CA part1.crt -CAkey private.key -CAcreateserial -out step3.crt*' the reason we set policy_anything is because then only require a CN and no specific countryName, stateOrProvinceName, localityName, organizationName, organizationalUnitName, commonName or emailAddress
.

For step 4 we went to etc/apache2/sites-available/default-ssl.conf
And in the virtualhost block we changed the paths to
SSLCertificateFile /home/hyeon/A2P4/step3.crt
SSLCertificateKeyFile /home/hyeon/A2P4/step2.key

For step 5 we gave the windows server the certificate we made in part 3 and when opening it we installed it using the default settings.



We then went to internet explorer and put in the ip of the https server and it gave us this result



which i believe means that it recognized the certificate.

step 6:
We can use the generated CA to sign code to ensure it has not been tampered with. This way the author of the code can sign it and it can be guaranteed that the code is the same as it was at the time of the signing. Jar files are often used in Java to store data like classes, images, and other data structures. Signing these jar files could be useful to maintain integrity in the codebase. This would work by having the author sign their pushed code using their private key, then anybody could use the authors public key to gain access to the untampered code.

These steps are necessary because the author is the only one with access to their private key, and therefore they can encrypt their code in a way nobody else could. After doing this, anybody can use the public key to access this code, and there's no way the code could be tampered with while it is in an encrypted state without making it completely distorted during decryption.

This is a very similar process then using web server certificates but in this case we are signing users to restrict access to the https server, while with code signing we are restricting access to who can change the code, while still allowing everyone access to it via the public key.

7.For the https to have the exact same applicable firewall rules, we added "-m multiport" and port 443 with the port 80. For example, -A INPUT -s 10.229.100.51/32 -p tcp -m multiport tcp --dport 80,443 -m conntrack --ctstate NEW,ESTABLISHED -j DROP