CS-491 (NETWORK SECURITY)

HOMEWORK-4

ANIMESH JAIN

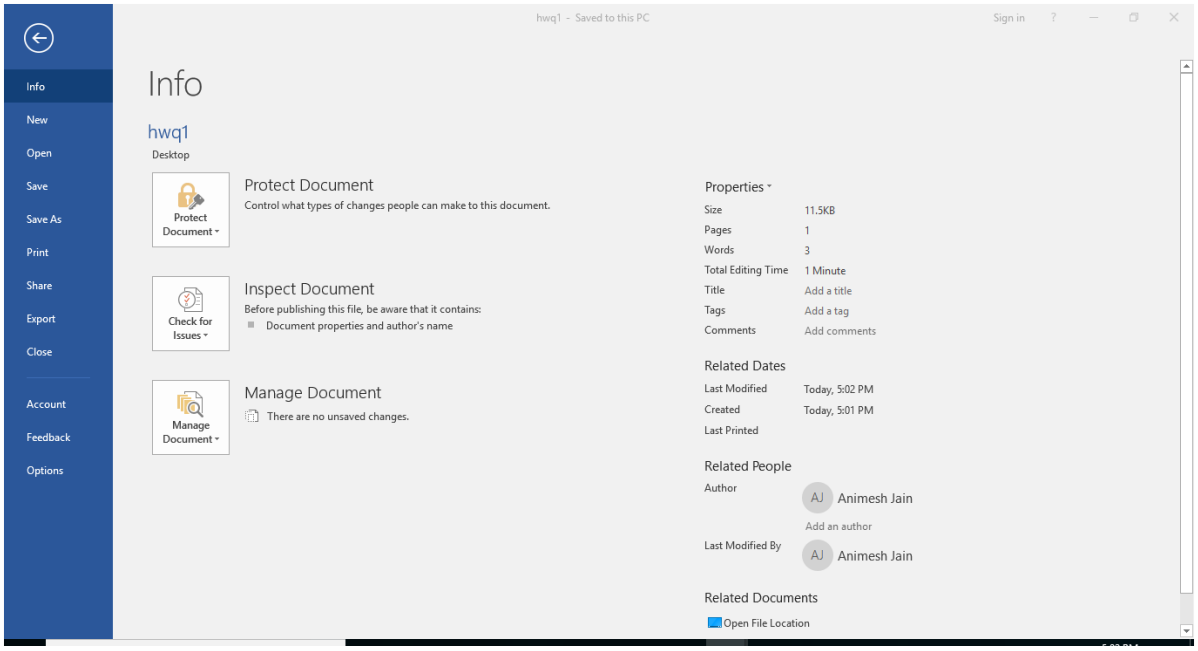
UIN-669653028

ajain65@uic.edu

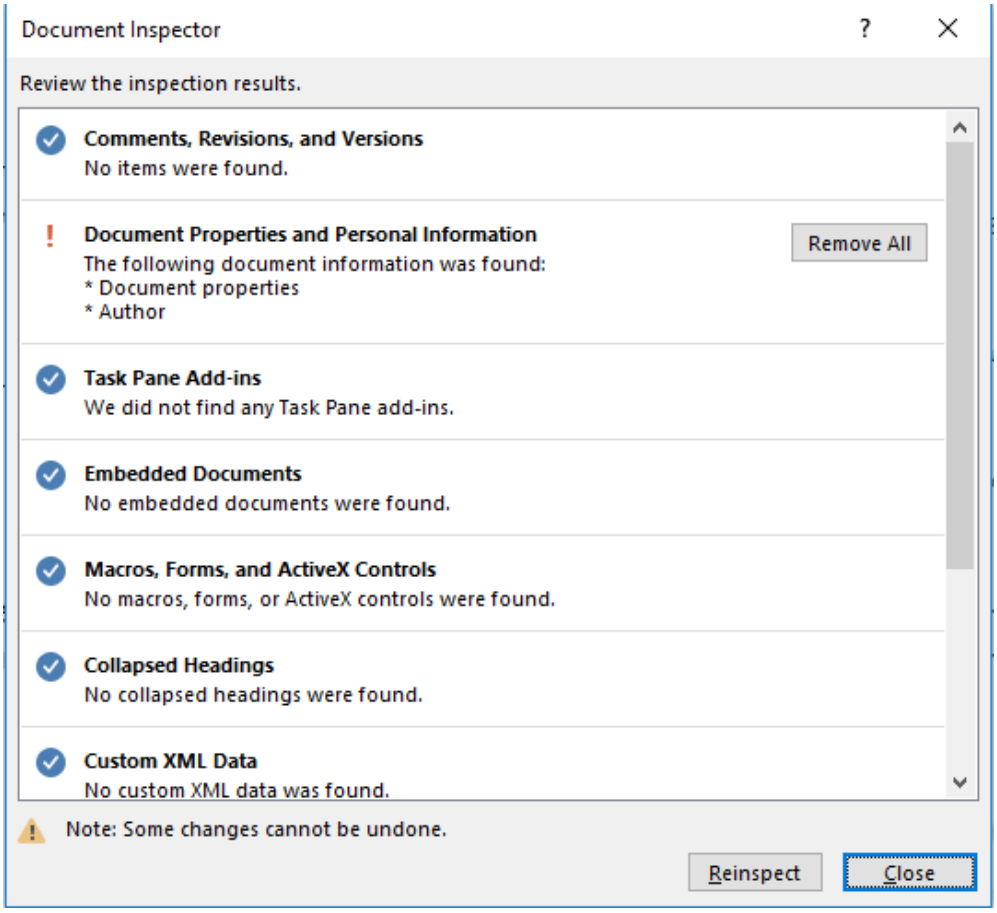
Solution 1: **ANONYMITY IN DOCUMENT SUBMISSION**

(a) WORD DOCUMENT

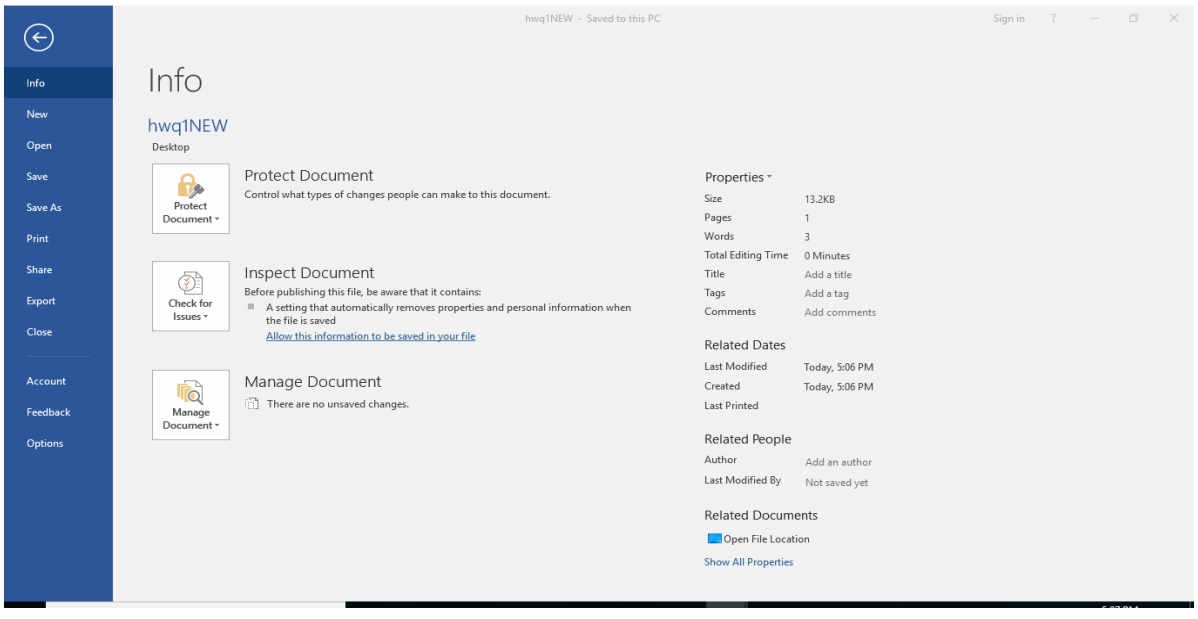
I created a word document named “hwq1”. When I investigate the file info, I can see the Author and by whom the document was last modified.



So, looking at the file info it can easily be figured out whose document is this. While submitting the document to the server we want it to be anonymous. To do this we will go to the document inspector and remove all the document properties and personal information. This step is shown in the next screenshot.



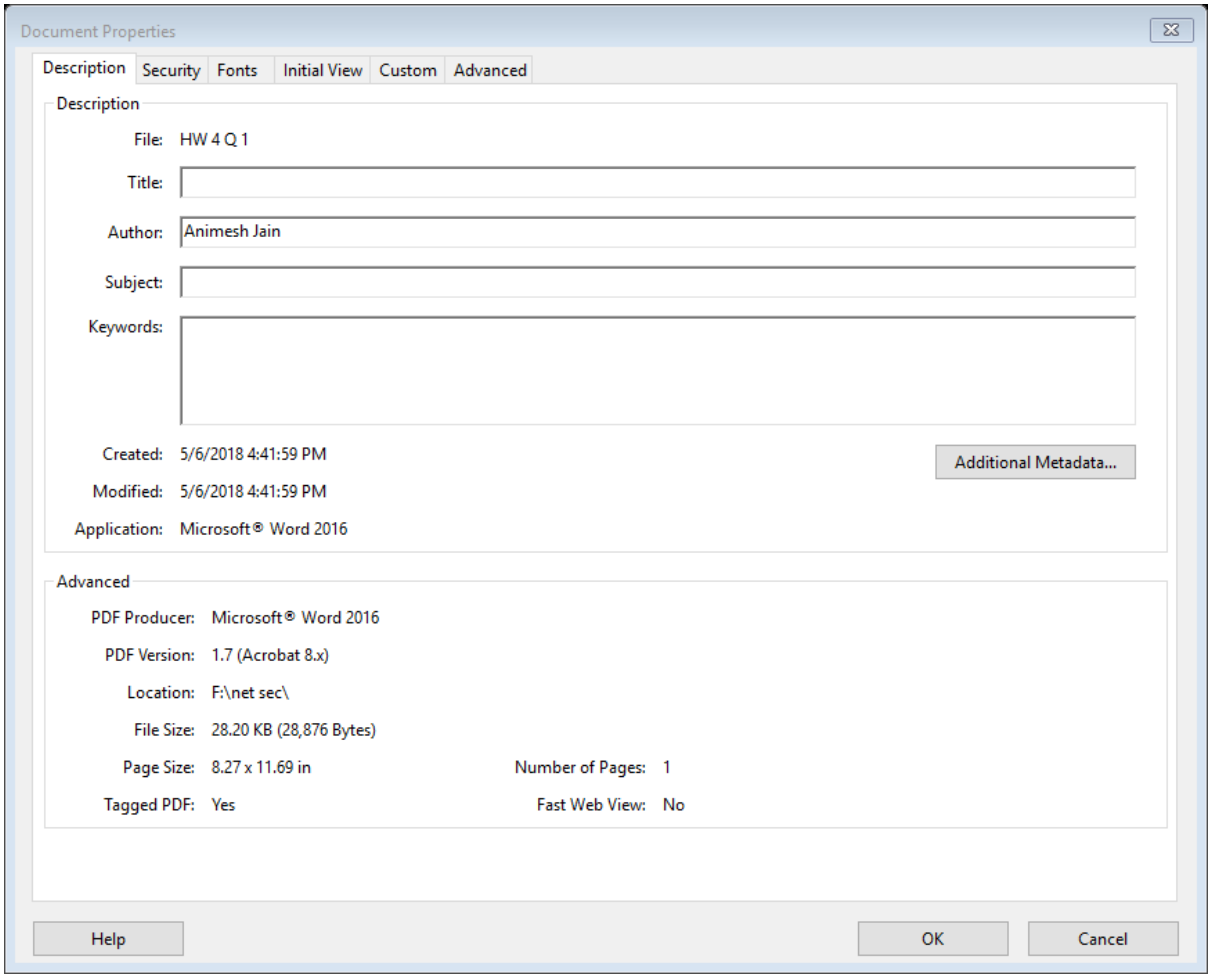
After removing all the document properties and personal information and saving the file as "hwq1NEW", when we see the file info again we can see that the author's info and who modified the document information is no longer there.



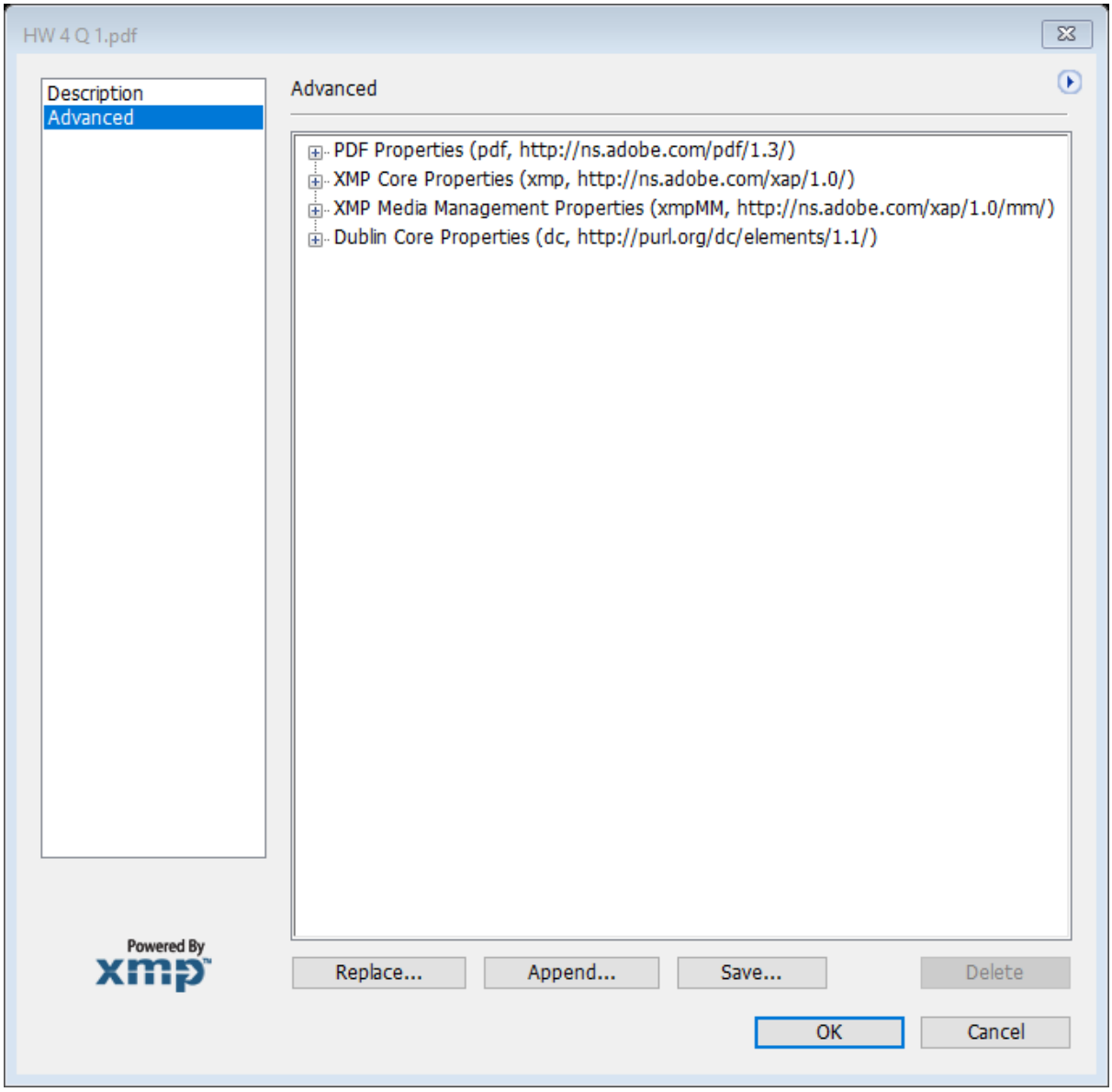
No name under the "Related People" tag.

(b) PDF DOCUMENT

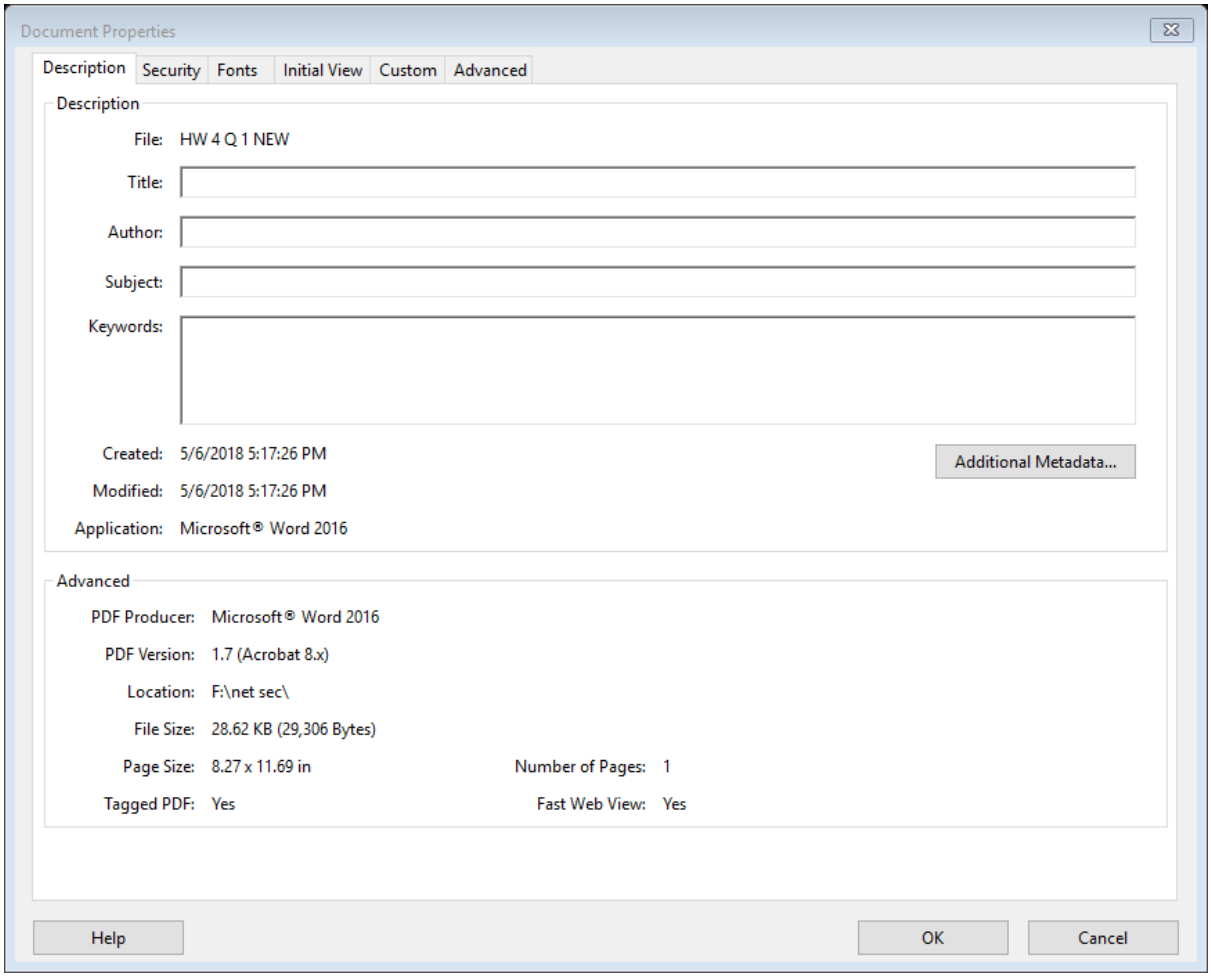
Now created a PDF document named “HW4Q1”. When we investigate the document properties we can clearly see the authors name. So, if we upload the file on the server the author of the document can be know by looking at the document properties and hence the submission won’t be regarded as anonymous. Therefore, we want to remove the authors information from the documents properties.



To remove the information, we will click on the “Additional Metadata…” tab and we will get the dialog box that’s given in the next screenshot. There we will go to the Advanced tab.



Now what all files are visible here we will delete them. After deleting all the of them we will save the file as “HW4Q1NEW”. Now we will look at the document properties of this file.



Now we can see that the author information is not being displayed. If this document is uploaded on the server then the authors identity cannot be known.

Identifying information that an embedded image in a word document can contains are the EXIF data. EXIF data is Exchangeable Image File, a format that contains interchange information in the digital file. The EXIF format stores information like: -

- Date and time the image was taken
- Image resolution
- Color type
- Compression
- Copyright information etc.

References: -

https://www.addictivetips.com/internet-tips/view-complete-exif-metadata-information-of-any-jpeg-image-online/

Solution 2: **CLOUD COMPUTING**

Although the cloud service providers encrypt the files but since they are the ones who encrypts the file then definitely they also know how to decrypt them as well. Designs we can implement to ensure that cloud service providers cannot access our files are: -

DESIGN 1: - ENCRYPTION

It is better that before exchanging the sensitive data with the third party we encrypt the data. We will use an encrypting algorithm and a key such that it changes the plaintext data to the ciphertext. So before uploading any data on the cloud, if we encrypt the data then the cloud service provider can store the data but cannot access the data as it needs to know the key to decrypt the algorithm and to access the plaintext.

The encryption could be done by either of the two methods mentioned below: -

Symmetric key encryption: - Here we use the same key for encryption as well as decryption. But if the key is compromised then it can be used to decrypt all the data.

Asymmetric key encryption: - Encryption and decryption of a file is done using two different keys. We can encrypt the data using our public key before sending data to the cloud. This way the cloud service provider cannot access the data, as to access the data, provider first needs to decrypt the data for which it needs a key (private key of organization), which is not known to anyone else apart from the organization itself.

If we implement this design, then we are adding an extra layer of security that reduces the risk of somebody else getting access to that data. But a lot depends upon the strength of your encryption algorithm. We must make sure that the crypto system mentioned here should be strong and properly implemented.

DESIGN 2: - TOKENIZATION

For storing data like credit card numbers, account numbers, social security numbers etc. we can use a method called **tokenization.**

Tokenization is a process of taking a value, obfuscating it and randomizing it to make a value from where the original value cannot be figured. The value which we get is called as token value which is a string of random number that has got no meaning if breached. Token values generated are mapped with the original values and are stored in a database called **token vault**.

This can be used by organizations like banks or credit card companies when the want to securely store details of their customers. So instead of the real data we just send the token values to the cloud service provider. This way we remove all the sensitive data from the cloud environment. Even if cloud service provider wants to access the data he cannot understand anything as the token alone has no meaning.

But organization can use these tokens to get back the original data by looking at the value corresponding to the token value in the token vault.

References: -

https://www.skyhighnetworks.com/cloud-security-university/tokenization-vs-encryption/

https://tokenex.com/securing-usernames-passwords-tokenization/#more-3040

Solution 3: **FIREWALLS**

ALLOWING FTP TRAFFIC

For data transfer FTP can use TCP port 20. And FTP control is handled on TCP port 21. All the below 4 commands will allow the traffic from external clients and from clients within the organization.

- iptables -A INPUT -p tcp - -dport 21 - - sport 1024:65535 -j ACCEPT (for data transfer)

  Here all the incoming packets that uses TCP port, 1024:65535 and 21 as their source and the destination port respectively are allowed by the server. sport 1024:65535 represents that the source port could be any port ranging between 1024 and 65535.

- iptables -A INPUT -p tcp - -dport 20 - - sport 1024:65535 -j ACCEPT (for control)

  Here all the incoming packets that uses TCP port, 1024:65535 and 20 as their source and the destination port respectively are allowed by the server. sport 1024:65535 represents that the source port could be any port ranging between 1024 and 65535.

- iptables -A OUTPUT -p tcp - -sport 20 - - dport 1024:65535 -j ACCEPT
- iptables -A OUTPUT -p tcp - -sport 21 - - dport 1024:65535 -j ACCEPT

  Above two commands allow the server to reply to the client.

ALLOWING MAIL TRAFFIC (allowing all SMTP traffic)

- Iptables -A INPUT -p tcp - -dport 25 - - sport 1024:65535 -j ACCEPT
- Iptables -A OUTPUT -p tcp - -sport 25 - - dport 1024:65535 -j ACCEPT

BLOCK ALL TELNET TRAFFIC

- iptables -A INPUT -p tcp - -dport 23 - - sport 1024:65535 -j REJECT
- iptables -A OUTPUT -p tcp - -sport 23 - -dport 1024:65535 -j REJECT

  Blocking both the outbound as well as the inbound telnet traffic.

References: -

https://unix.stackexchange.com/questions/93554/iptables-to-allow-incoming-ftp

Solution 4: **DNS CACHE POISONING**

The DNS server corresponding to cs.uic.edu uses port randomization and transaction ID checking. Port randomization means that the UDP port used for query won't be default port 53 anymore. It can be anything from the entire range of the UDP port. This makes guessing of the query parameter by the attacker even more difficult.

(a) When Bob searches for anything on the web browser, the browser forwards the query to the DNS server and asks for the address and will take the user to the desired website. The local DNS server makes this process faster by caching the addresses. So that way the request doesn't have to go to Authoritative DNS server every time. Authoritative DNS server comes to the action only when the address requested is not in the cache of local DNS server.
Steps through which Bob's request to the search engine are made available to Charlie through DNS Cache poisoning are: -

- Charlie sends a random request that is invalid or not present in the cache of local DNS server to the local DNS server.
- Since the query made is not present in the server, the query is forwarded to the Authoritative DNS server.
- Charlie then floods the local DNS with fake responses (while guessing the port and transaction id).
- Response from the authoritative DNS is not accepted by the Local DNS as it has already accepted a response from the attacker (if the port number and the transaction id guessed by the attacker matches the actual one). Response is also stored in the cache of the local DNS.
- Now if anyone makes the query for that website then they will be directed to the address (Charlie's address in it's DNS server) that is stored in the local DNS cache, which is controlled by the attacker. So, whenever bob makes query for that website, the query goes through the Charlie's DNS server and that is how bob's request to the search engine are made available to Charlie.

For this scenario to be successful few things need to work together like: -

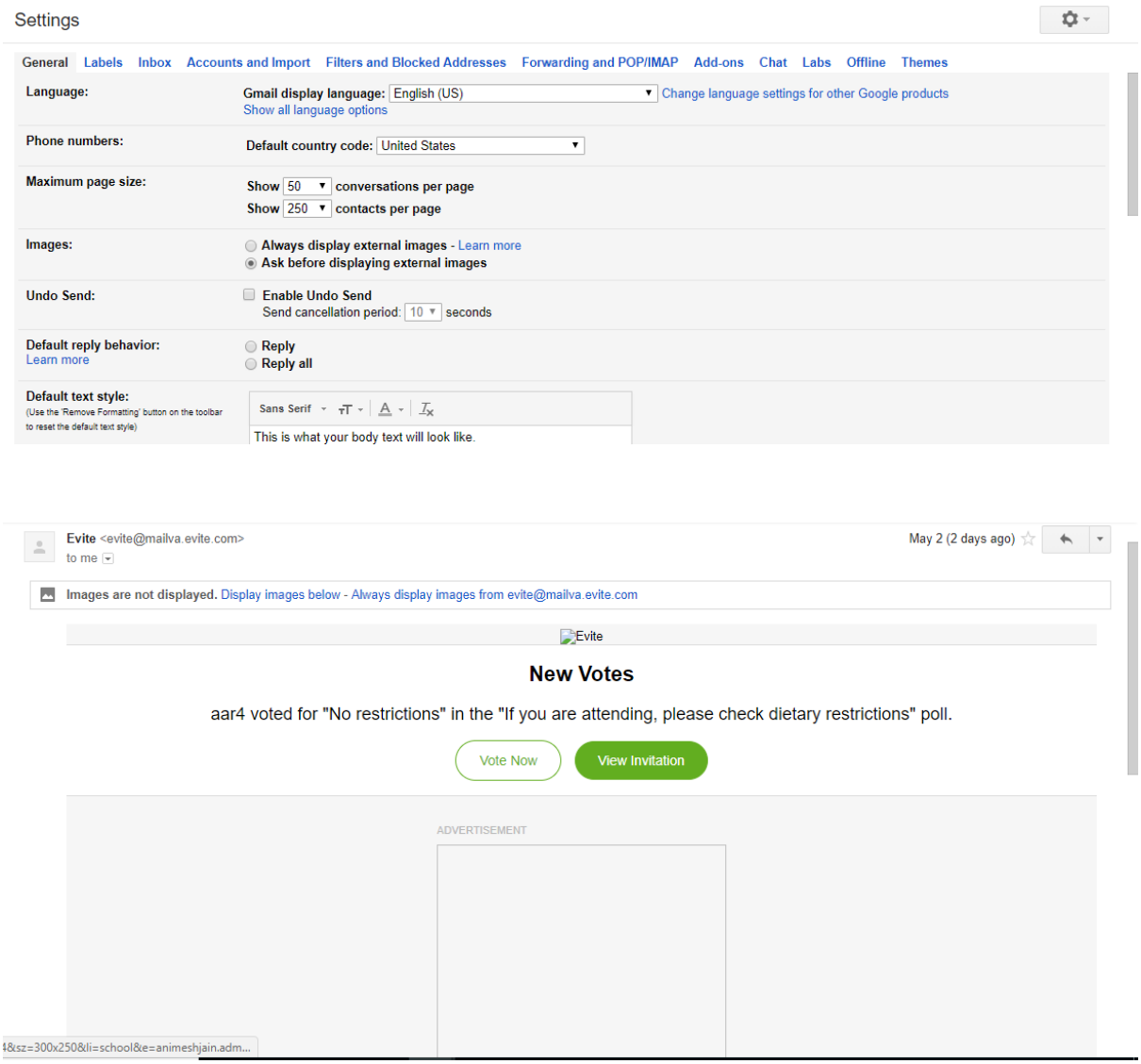
- Transaction id and the port number guessed by the attacker should match the real one otherwise that response will be rejected by the DNS server.
- Response from the attacker should reach the local DNS before the response from the Authoritative DNS.

(b) UDP ports are 32 bits
Transaction id = 16 bits.
So, the number of DNS records Charlie must forge to successfully poison the cache of the DNS server = 2^ (32+16) = 2^ (48)

References: -

https://www.youtube.com/watch?v=1d1tUefYn4U

Solution 5: **GMAIL IMAGE INLINING**



Automatic inlining of images can be threat because of the following reasons: -

- If we allow to display the images automatically that means no check on the sent message is being performed to look for the possible harmful software's.
- If these images content some malicious content, then the sender might get information about our computer or location.
- Sender might get information about conversation history.
- If the image has got malicious content, then the sender can get access to read cookies in our browser.

So, if the inlining of image is turned off then whenever Gmail receives a rich text that contains inline images, then Gmail will first scan the image for doubtful contents, and if Gmail thinks that a sender is malicious then the user is asked before displaying the image.

Circumstances under which automatic inlining could be turned on as a reasonable option

If we trust the source from where we are receiving the message, then we don't need to disable the automatic inlining of the image. So automatic inlining can be turned on whenever we are receiving mail from a trusted source. For this we can click the hyperlink saying, "Always display images from xyz.abc.com". So here onwards any messages sent from xyz.abc.com will be completely displayed without being scanned by Gmail.

References: -

https://support.google.com/mail/answer/145919?co=GENIE.Platform%3DDesktop&hl=en

Solution 6: **DENIAL OF SERVICE**

DESIGNS FOR DISTINGUISHING BETWEEN LACK OF CAPACITY AND DENIAL OF SERVICE

DESIGN 1: - One of the ways to differentiate between the lack of capacity (regular traffic) and Denial of Service (attack) is by looking at the "referrer header". Using referrer header, the server can know from where the crowd is visiting them. Basically, it tells the server from where the request from user was originated. Generally, in the case of Denial of Service attack the referrer field will be null.

DESIGN 2: - **CHALLENGING THE USER**

We will have a challenge for every user who wants to access the website. We can make use of **CAPTCHA** to challenge the users. It could be anything like identifying an image or simple mathematical calculations or identifying letters from a distorted image. These challenges are designed in a way such that it is easy to perform by an actual user but could not be done by bots. So, depending upon the responses provided by the users to these challenges it can be decided whether the source is a legitimate user or the malicious user.

DESIGN 3: - **OBSERVING THE CONNECTION STATE**

The purpose of Denial of service attack is to occupy the resources by the server so that the services by the server are not accessible by the legitimate user. So, in this case of DOS, the attacker will never complete the connection establishment but will occupy the server resources (bandwidth). Therefore, looking at the connection state we can know whether the user is a legitimate user or the malicious one. If the process of connection establishment is completed, then it is safe to say that the user is legitimate one and if the connection is not established then it is safe to assume that the user is the malicious user is trying to perform the denial of service attack.

References: -

https://blog.radware.com/security/2013/06/distinguish-between-legitimate-users-and-attackers-the-secret-sauce-of-ddos-protection/

https://security.stackexchange.com/questions/135653/how-does-sending-referrer-http-headers-protect-against-csrf-attacks?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa