Cryptography and Network Security HW2 Report

B07902116 資工三 陳富春

1 BGP

1) All traffic is directed to AS999, including AS closer to AS1000. Thus, it is the most likely that AS999 announced it has the prefix ”10.10.220.0/23” to all BGP routers to redirect the traffic.

2) a) The BGP update message can be {10.10.220.0/23, [AS 999, AS 2, AS 1, AS 1000]}.

   b) The attacker misuses the loop prevention. AS1 and AS2 will not update their routing rules because this BGP update includes themselves but other AS will accept the update. Thus, all traffic except AS1 and AS2 will be directed to AS999 first, then be sent to AS2, AS1 then AS1000.

3)
- Advantage: If a BGP router has already announced its prefix with maximum length, then traffic to the prefix will not suffer from such BGP prefix hijacking.
- Disadvantage: There should be a standard MPL value among BGP routers. Besids, deciding such value, coordination among routers and deployment are costly, which may be unattractive to ISP.

2 School's Normal Riddle

a) The probability that the random z satisfies $g^z = y^c a$ is $\frac{1}{q}$.

b) If Alice can predict Bob's challenge c, then she can pick arbitrary x and compute $z = cx + r$ and send it to Bob. Because Alice knows the values of $c,\ x,\ r$, she can generate the corresponding z that satisfies $z = cx + r$. Bob, who does not know that Alice can predict c, will believe that Alice know the correct x.

c) If Alice uses the same r, then Bob can send two challenge values $c_1,\ c_2$ where $c_2 = c_1 + 1$. Bob will get $z_1 = c_1 x + r$ and $z_2 = (c_1 + 1)x + r$, then $x = z_2 - z_1$ can be computed by Bob.

d) The non-interactive version of the protocol works as follow:

1. Alice choose random r and compute $a = g^r$
2. Alice compute $c = H(g, y, a)$ and $z = cx + r$
3. Alice send (z, a) to Bob the verifier

Bob knows public $g,\ y$ values and receive (z, a) from Alice, so hecan verify without knowing x value by checking whether $g^z = y^c a$.

e) The protocol can be modified by adding some information when generating c. For example, let the generation of c be $c_A = H(g, y, a, \text{"}Alice\text{"})$. If Bob want to cheat other using replay attack, though he can compute the value of $c_B = H(g, y, a, \text{"}Bob\text{"})$, he cannot compute $z_B = c_B x + r$ without knowing x. Thus Bob's replay attack will fail.

f) I modify the interactive version and it works as follow:

1. Alice choose random r and compute $a_1 = g^r,\ a_2 = h^r$ and sends (a_1, a_2) to Bob
2. Bob sends challenge c to Alice
3. Alice compute $z = cx + r$ and send z to Bob
4. Bob verifies $g^z = y^c a_1$ and $h^z = w^c a_2$. If both equations hold, then $y = g^x$ and $w = h^x$, thus $log_g(y) = log_h(w)$.

3 SYN Cookies

a) Using SYN cookie free server from keeping state of each SYN connection, so the server can keep resource when it encounter SYN flooding attacks. These saved resource allow server to provide service to normal users.

b) SYN cookie needs to contain timestamp because it prevent server from replay attack. SYN cookie needs to contain the client's IP address because it allow server to distinguish different users and it can also prevent MitM attack.

c) Because a hash function is public to everyone, including the attacker, it is possible that the attacker generate a valid cookie in accident. Using MAC prevent such situation because only the server can generate a valid cookie using its private key.

4 TLS

1. Information from tls_sessions.pcapng
   Given the hint that they love unique prime numbers, server's modulus can be factored into two Mersenne primes and the private key can be computed. Because TLS_RSA has no forward secrecy, the encrypted application message can be decrypted. From the decrypted message, we know that we can use `Alice407` and `TkBewt2jXhg2fJLv` to login and get the flag by command `prIntTheFlag!!!` .

2. Connect to the server
   I get stuck in generating a valid certificate for authentication and cannot establish a TLS connectino with server.

5 Tor

a)
1. Using "Inspect Element" of the website, then at "Network" option, in the "Response Header", there is First-Flag: **CNS{Web_s3ver_and_**
2. The onion website will fetch a .bib file from `cns.joeywang4.tw`. By executing `dig cns.joeywang4.tw txt`, it gives out the latter part of the flag: **mix3d_content_may_leak_in4o}**

Flag: CNS{Web_s3ver_and_mix3d_content_may_leak_in4o}

b)
1. Get v3 onion address from public key:
   Removing the 32 bytes header of `tor/pubkey` to get the public key value. Then generate the address by doing hash and base32 encode.
2. Challnge in the hidden service:
   The challenge can be soved by binary search. After answering the correct number, a big CNS flag is sent.

Flag: CNS{___A_hidden_hidden-flag___}

c)
1. Set up tor hidden service:
   By setting the `torrc` file, we can set up our own hidden service.
2. Generating keypair:
   Randomly select a private key and compute its corresponding public key. If the public key can generate an address with prefix `cnshw`, then we find the desired keypair. Then we can run our hidden service using the keypair and the hostname.
   The generated keypair (in hex) and hostname is in `tor3_key.txt`.
   The ed25519 reference: `https://github.com/orlp/ed25519`.
3. Echo the random message:
   After setting up hidden service using the derived keypair and hostname, we simply listen to port 80 and echo the message to get the flag.

Flag: CNS{l3t$_an0nymize!}

6 Mix Network

I try to reconstruce the 5x5 connectivity table T between these five nodes by the following steps:

1) Each node cannot send a packet to itself:
   According to the given sample code , we know that a node cannot send a packet to itself. So $T[i][i] = 0,\ for\ i = 1\ to\ 5$ can be filled.

2) According to single mix traffic:

   - $s_a\ s_b\ s_c\ >>\ s_d$:
     Only one receiver. The traffic shows that all $s_a\ s_b\ s_c$ can go to s_d. That is, $T[a][d] = 1,\ T[b][d] = 1,\ T[c][d] = 1$.
   - $s_a\ s_b\ s_c\ >>\ s_a\ s_d$:
     One of receivers is in sender set. The traffic shows that s_a can go to s_d because s_a cannot send a packet to itself. That is, $T[a][d] = 1$.
   - $s_a\ s_b\ s_c\ >>\ s_a\ s_b$:
     Both of receivers are in sender set. The traffic shows that s_a can go to s_b and s_b can go to s_a. That is, $T[a][b] = 1,\ T[b][a] = 1$

   After the step, we can fill some 1's in the connectivity table.

3) According to out degree d^o of each node:
   Because we can observe all traffic on the mix network, we can count the number of traffic that go out from any three senders $s_a,\ s_b,\ s_c$, and the number is the product of out degree of the three senders $d^o_a \cdot d^o_b \cdot d^o_c$. Because each sender has at least one out degree, all the C^5_3 products are positive and the out degree of each node can be computed.
   Then we can check if a sender s_a already has d^o_a 1's in its row $T[a]$. If so, the other entries of that row can be filled with 0's.

4) According to the fact that in degree of each node > 1:
   The step checks if there is any column has four 0's and one unknown entry. If so, that entry must be 1.

After these steps, a connectivity table can be reconstructed and used to find a traversing path.
I use a relative simple method to find a path that traverses all nodes and destinies to s_5. It starts from s_5 and recursively find a possible predecessor that has not been reached. Due to the limitation of the simple method, it can only find out a path that passes through each node exactly once. But it is enough to find a path most time.
Flag: CNS{m1x_netw0rk_iS_F0n!}

7 NTLM Authentication

a) The client send a Type1 message to negotiate with the server, which contains the features that the client support and request. The server respond with a Type2 message, which contains authentication target and challenge. The client respond with a Type3 message to finish the authentication, which contains the authentication target and challenge in Type2 message, the username, and the hash of the user's password.
By the messages change above, the server can authenticate a user without directly sending the user'spassword.

b) NTLM hash of admin (in hex) : e811c30579af2ed12a7b2211060c4892

c) After I log in as a normal user, the server sends not only the user's NTLM hash but also the admin's, so I can use the admin's NTLM hash to generate admin's LMv2 and NTLMv2 response and login as admin.
CNS{Some_techniques_exist_just_for_backward_compatibility...}

d) After getting the admin's LM hash, we can brute-force search the possible password.
The first half part I find is: **CNS{!GU**.
The second half part I find is: **3S4ME!}**.
Then again brute-force search the possible combination of upper and lower letters and find that **CNS{!Gu3s4Me!}** can generate the same NTLMhash in (b). So the admin's password should be **CNS{!Gu3s4Me!}**.