Spring 2021 Cryptography and Network Security

Homework 2

Release Date: 2021/4/13

Due Date: 2021/5/10, 23:59

Instruction

- **Submission Guide:** Please submit all your codes and report to NTU COOL. You need to put all of them in a folder named by your student id, compress it to hw2_{student_id}.zip. For example, hw2_r09922456.zip. The report must be in **PDF** format, and named report.pdf.
- You may encounter new concepts that haven't been taught in class, and thus you're encouraged to discuss with your classmates, search online, ask TAs, etc. However, you must write your own answer and code. Violation of this policy leads to serious consequence.
- You may need to write programs in the Capture The Flag (CTF) problems. Since you can use any programming language you like, we will use a pseudo extension code**.ext** (e.g., code.py, code.c) when referring to the file name in the problem descriptions.
- This homework set are worthy of 120 points, including bonus.
- You are recommended to provide a brief usage of your code in **readme.txt** (e.g., how to compile, if needed, and execute). You may lose points if TAs can't run your code.
- In each of the Capture The Flag (CTF) problems, you need to find out a flag, which is in `CNS{...}` format, to prove that you have succeeded in solving the problem.
- Besides the flag, you also need to submit the code you used and a short write-up in the report to get full points. The code should be named **code{problem_number}.ext**. For example, code3.py.
- In some CTF problems, you need to connect to a given service to get the flag. These services only allow connections from 140.112.0.0/16, 140.118.0.0/16 and 140.122.0.0/16.

Handwriting

1. BGP (20%)

In this problem, we would like you to explain and analyze some attacks against the BGP routing protocol. In both attacks, the attacker has control over AS999 and wants to attack AS1000. Figure 1 shows the routing paths in the normal state after AS1000 has announced 10.10.220.0/22. Each circle represents an Autonomous System (AS). A solid line indicates a link over which two neighboring ASes can exchange control messages such as BGP update messages. A dashed line indicates an established AS path to 10.10.220.0/22.

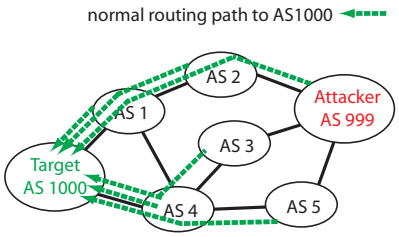

Figure 1: Normal scenario.

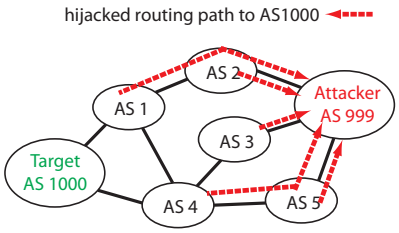

Figure 2: BGP hijacking.

1) (5%) Describe the most likely scenario that could explain the result of Figure 2. Specifically, what did AS999 announce?

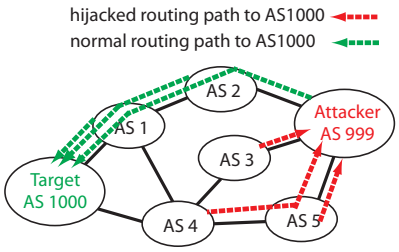

Figure 3: BGP man-in-the-middle.

2) In the second type of attack illustrated in Figure 3, the attacker can silently redirect the hijacked traffic back to the victim along the path indicated by green lines. This attack exploits **AS Path Prepending**, where an AS inserts AS numbers at the beginning of an AS path to make this path less preferable for traffic engineering purpose, and **Loop Prevention**, where AS x drops any BGP update with itself (i.e., AS x) in the AS-Path attribute to prevent routing loops.

   a) (5%) Instead of announcing the ownership of an address block, AS999 announces a fake BGP update. Specify a BGP update message (in the form of {IP prefix, [AS x, AS y, $\cdots$]}) that could cause the result of Figure 3.

   b) (5%) Briefly explain how the attacker misuses path prepending and loop prevention for malicious purpose.

3) (5%)To mitigate BGP prefix hijacking, some propose the idea of BGP Maximum

Prefix Limit (MPL). If a prefix length of an advertisement is longer than the MPL, the advertisement will be discarded. For example, if MPL is set to 24 in a BGP speaker, the announcement of 10.10.220.0/25 will simply be ignored. List one advantage and one disadvantage of MPL.

2. School's Normal Riddle (20%)

One day, a riddle appears on the school's bulletin board:

> "I have a cyclic group $\mathbb{G}$ of order q and a generator g. Here is a group element $y \in \mathbb{G}$. Can you give me $x \in \mathbb{Z}_q$ such that $y = g^x$?"

To encourage the students, the teacher declares that the first student who solves the riddle can get an A+ in CNS. Alice is so smart that she has already got x. Being such a smart student, she knows that she would still get an A+ in CNS even if she did not solve the riddle, so she decides to sell the answer x to Bob. To make Bob buy her answer, she has to convince Bob that she really knows x. However, she cannot directly reveal x, or Bob can just get the answer and run away.

Here is how Alice proves to Bob that she knows x such that $y = g^x$:

1. Alice: choose random r and send $a = g^r$ to Bob.
2. Bob: choose a random challenge c and send c to Alice.
3. Alice: compute $z = cx + r$ and send z to Bob.
4. Bob: verify that $g^z = y^c a$. If it is correct, Bob believes that Alice knows x.

a) (2%) If Alice does not know x and send random z to Bob, what is the probability that she can convince Bob that she knows the answer?

b) (3%) Randomness generation is important in cryptography protocols. Suppose that Alice does not know x, but she can predict Bob's challenge c. How can she convince Bob that she knows the answer?

c) (3%) Now assume that Alice really knows the answer, but Bob is so skeptical that he asks Alice to prove for several times. What can Bob do to reveal x if Alice uses the same r?

d) (4%) The protocol above is interactive. Now Alice wants to make it non-interactive. Briefly explain how **Fiat-Shamir Heuristic** works and apply it on the above protocol to make it non-interactive. You may need a hash function H and the symbols in the above protocol, but you should not use any other symbols.

Note: You may notice that the attack in c) no longer works.

e) (3%) However, the protocol in d) may suffer a replay attack. Bob can use Alice's message to cheat others that he knows the answer. Modify the protocol in d) against the replay attack.

Note: any possible answer is acceptable and you can use any symbols. But you should describe your symbols clearly.

f) (5%) There is another group $\mathbb{H}$ and a generator h. Alice computes a group element $w = h^x$, and she wants to prove that $\log_g(y) = \log_h(w)$ without directly revealing the answer x. Can you slightly modify the protocol above to provide the proof? You can use either the interactive or the non-interactive version.

3. SYN Cookies (10%)

a) (3%) Explain why SYN cookies can mitigate SYN flooding attacks.

b) (4%) Explain why the cookie needs to contain a timestamp and the client's IP address.

c) (3%) Explain why using a Message Authentication Codes (MAC) is better than using a hash function to calculate SYN cookies.

Capture The Flag

4. TLS (15%)

Someone has been listening on a target at `cns.csie.org:10077`, which belongs to former NTU CNS members. Even though the system is protected by TLS, they think something is unusual but could not figure it out. So, they send you a **tls_sessions.pcapng** file for investigation. All you know is that they love unique prime numbers. Go ahead and see if you can find any secrets in the system.

Note: The server will check your identity in the TLS handshake process. (Ncat does not support TLS.) You can open pcapng files with `Wireshark`.

5. Tor (10% + Bonus 10%)

Hidden services are so cool! You don't need an expensive domain name or a public IP to create a service. Even better, your service is anonymous. This means that no one will know who created the service, right?

a) **Warm-up** (5%) Unfortunately, it is possible to break anonymity if the server is configured incorrectly. I created a hidden service that stealthily teaches people something about DNS, a vicious protocol that leaks the information about domain names. A flag is separated into two pieces and embedded inside this hidden service. Try to dig out the flag from it. The address of the hidden service is: http://cns22iywy2ybrtboo77dr77dsgto5vusgrmy57bpi7f66ny6qqr6zsid.onion.

Note: No code is needed. Just explain how you find the flag.

b) **Hidden hidden-challenge** (5%) I created a hidden-challenge, which is very simple, but the address of this challenge is hidden. You can find the source code of this challenge at `tor/chal.py`, and the public key of the service is at `tor/pubkey`.

Note: The challenge is listening on port 3001.

c) **Let's Tor!** (Bonus 10%) Do you know Let's Encrypt? It is such a great place to get free TLS certificates. I created a similar service called *Let's Tor*, which gives you free flags. But there is a catch: This service only helps those who are working on CNS homework. That is, it only sends the flag to an address that begins with `cnshw`. Try your best to get the flag.

Let's Tor is at cnsggnhxxfujmdtqp4hddfd3xp2rjclz7yfiuguekdcvwar7f72yrtad.onion, listening on port 3002. Since *Let's Tor* has a user-friendly command-line interface, the source code of this service is not provided.

Note: You should provide your code that generates the onion address, and the public and private key pair you generated.

6. Mix Network (10%)

A mix nework is a technique designed to prevent attackers who are able to observe all traffic between nodes from knowing the identity of a packet's sender and receiver. However, in some cases, attackers can somehow retrieve users' relationship. Assume a simple scenario, where S is a group of senders and R is a group of receivers, and there is a mix between senders and receivers. The mix always buffers b packets from b distinct senders ($b \leq |S|$), padding packets into the same size, and then sends to receivers in random orders. In other words, a subset S' of all senders S sends b packets to a subset R' of all receivers R in each round, where $|S'| = b$.
You are an attacker who can observe all flows between end devices and the mix. Assume that $b = 3, S = \{s_1, s_2, ..., s_5\}, R = \{s_1, s_2, ..., s_5\}$. You see all flows of network, and you need to send a packet which contains "print the flag" message. The packet will be transmitted to the next user if it contains the next receiver.

Your goal is to construct a packet that traverses all users before arriving at s_5, and you will get the flag when s_5 receives the message. You should explain how you get the flag in details in your report. You can access the system by `nc cns.csie.org 8507`. The server code is included in `hw2/mix`.

7. NTLM Authentication (15% + Bonus 10%)

NTLM is a well-known challenge-response authentication mechanism in the Windows system. However, because NTLM has many security issues, Windows then migrates to Kerberos. (The implementation of Kerberos in the Windows system is still faulty though, for more information, please refer to the incredible work of gentilkiwi.) In this challenge, we require you to communicate with an NTLMv2 authentication server. The server is implemented according to https://curl.se/rfc/ntlm.html. You can connect to the server via `nc cns.csie.org 30005`.

a) Please explain how NTLMv2 works (3%) (You do not need to provide the technical details, e.g., how exactly the message is generated.)

b) Please login to any account, and provide the NTLM hash of admin as proof. (7%)

c) Please try to login as admin and give me the flag as proof (bonus 3%). Briefly explain how you can do this without knowing the admin's password (bonus 4%)

d) Try to obtain the admin's password! (bonus 3%)

Important: you should upload your code, which will also be a grading criterion.

Note: To be precise, the protocol is NTLMv2, not NTLM.

Note: We will not provide the server's complete code as that is what we want you to implement.

Note: There are some hint in the README.md provided under ntlm/