Cybersecurity / CNS exam

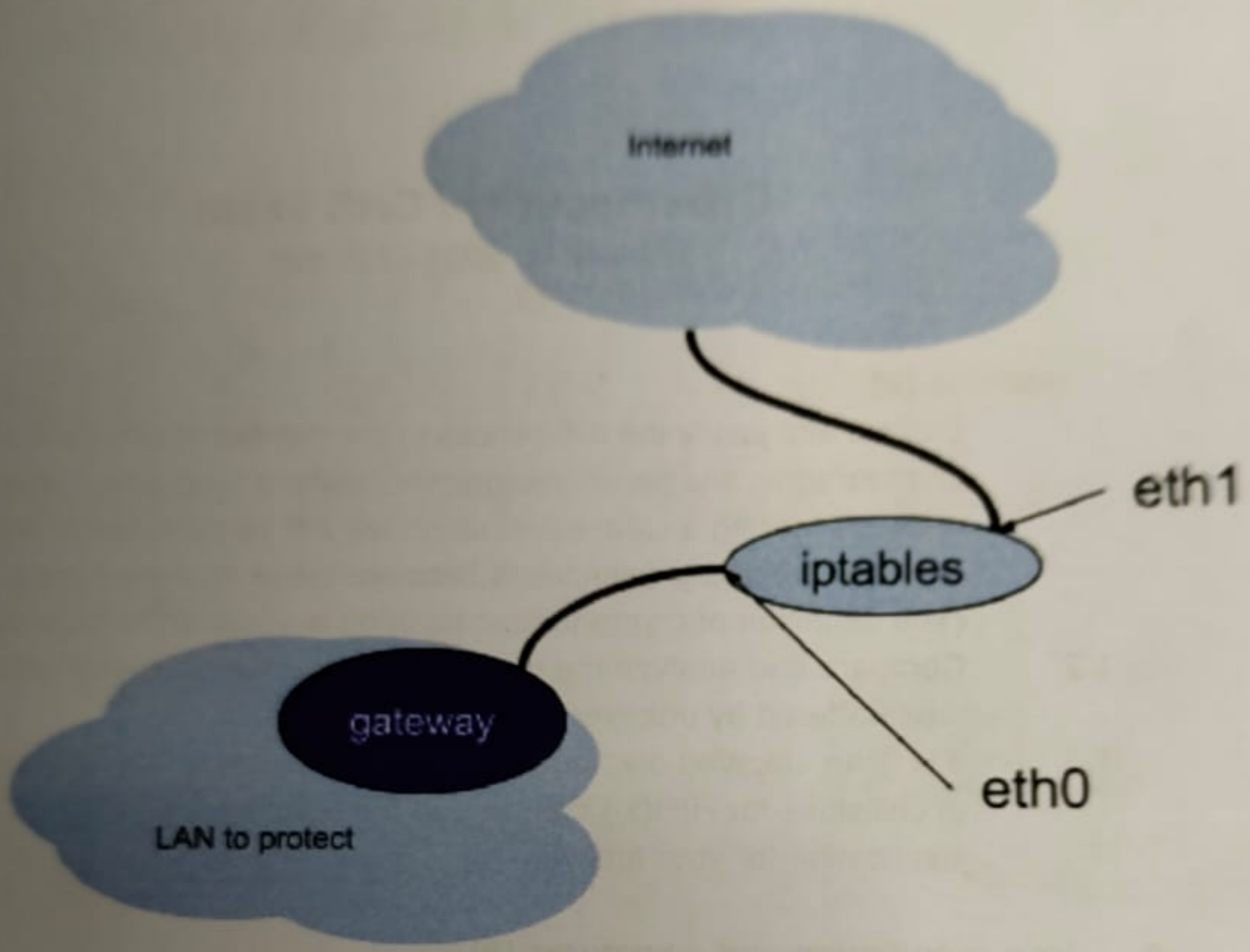
January 14, 2025 - 120 min

1. **Hashing [9]**
   1.1. Explain and justify the differences in the number of collisions between cryptographic and non-cryptographic hashing functions, noting that answers lacking a clear explanation will not be considered. Additionally, highlight other key distinctions between these hashing functions. (The definition of cryptographic hashing is assumed to be known.) [3]
   1.2. Compare and analyze the guarantees provided by keyed hashing versus those offered by unkeyed hashing. [3]
   1.3. If H is an unkeyed cryptographic hashing function, how does the number of collisions for H(H(x)) compare to those for H(x) as x varies? Provide justification for your answer. [3]

2. **Non-repudiation and signatures [8]**
   2.1. In RSA signature, if the signer's private key is compromised at time t_0, what will be the effect on the signatures at $t_1 < t_0$ and at $t_2 \geq t_0$? Discuss and motivate. [3]
   2.2. In DSS signature if the private key created by the signer at time of signing is compromised what will be the effect on past and future signatures? Discuss and motivate. [3]
   2.3. Can HMAC be used for non-repudiation? Discuss and motivate. [2]

3. **Authentication and attacks to it [6]**
   3.1. What is salting? Why use it, how and for what purpose? Discuss. [2]
   3.2. What are rainbow tables and what are they used for? [1]
   3.3. Describe carefully what an offline dictionary attack on passwords is. [3]

[iptables] Consider the following drawing, showing a LAN to be protected, a firewall external to LAN and the Internet. [9]



4.1. How to configure whitelisting? [2]

4.2. Write the rule(s) to allow LAN-firewall conversations and blocking firewall-Internet conversations. [3]

4.3. How to use iptables to (partially) protect the LAN from DDoS attacks? [4]

1.1 NON-CRYPTOGRAPHIC HASHES ARE BUILT MAINLY FOR SPEED, ERROR DETECTION, OR DATA DISTRIBUTION THEY ARE NOT OPTIMIZED TO RESIST ADVERSIAL INPUTS. AS A RESULT, COLLISIONS ARE RELATIVELY FREQUENT, AND AN ATTACKER CAN DELIBERATELY GENERATE THEM WITH MODEST EFFORT.

IN CONTRAST, CRYPTOGRAPHIC HASH FUNCTIONS ARE DESIGNED TO MAKE FINDING COLLISIONS COMPUTATIONALLY INFEASIBLE: WITH AN n-BIT OUTPUT, THE BEST POSSIBLE ATTACK NEEDS ABOUT $2^{n/2}$ OPERATIONS SO THERE ARE FEWER COLLISIONS.

CRYPTOGRAPHIC HASHES GUARANTEE PREIMAGE RESISTANCE (GIVEN h, INFEASIBLE TO FIND x WITH $h = H(x)$) AND SECOND PREIMAGE RESISTANCE (GIVEN x, INFEASIBLE TO FIND $x' \neq x$ WITH $H(x) = H(x')$).
NON CRYPTOGRAPHIC HASHES CAN OFTEN BE INVERTED OR ATTACKED WITH SIMPLE ALGEBRIC TECHNIQUES.
MOREOVER CRYPTOGRAPHIC HASHES PROVIDE STRONG AVALANCHE EFFECT (A SINGLE BIT CHANGE IN INPUT PRODUCES A COMPLETELY DIFFERENT OUTPUT), WHILE NON CRYPTOGRAPHIC HASHES MAY SHOW PREDICTABLE PATTERNS

1.2 UNKEYED HASHING PROVIDES ONLY INTEGRITY WITHOUT AUTHENTICATION, SO ANYONE CAN RECOMPUTE THE DIGEST, AND AN ATTACKER CAN MODIFY THE MESSAGE AND GENERATE A NEW HASH.
KEYED HASHING USES A SECRET KEY, SO ONLY LEGITIMATE PARTIES CAN PRODUCE OR VERIFY THE TAG. IT THEREFORE GUARANTEES BOTH INTEGRITY AND AUTHENTICITY, PREVENTING FORGERY, WHILE UNKEYED HASHING IS SUITABLE ONLY FOR NON ADVERSARIAL USES LIKE ERROR DETECTION OR FILE CHECKS.

1.3 ?

2.1 ALL SIGNATURES GENERATED AFTER THE TIME ($\geq \mathcal{I}_0$) CAN NO LONGER BE TRUSTED, SINCE AN ATTACKER WITH THE PRIVATE KEY CAN FORGE ARBITRARY VALID SIGNATURES. HOWEVER, SIGNATURES CREATED BEFORE $\mathcal{I}_0$ REMAIN VALID, BECAUSE THEY CAN STILL BE VERIFIED WITH THE PUBLIC KEY AND COULD NOT HAVE BEEN FORGED BEFORE THE PRIVATE KEY WAS LEAKED.

2.2 IN DSS, EACH SIGNATURE REQUIRES A FRESH RANDOM VALUE K, CALLED THE EPHEMERAL PRIVATE KEY. IF THIS SECRET K IS COMPROMISED, THEN THE SIGNER'S LONG TERM PRIVATE KEY CAN BE COMPUTED FROM IT TOGETHER WITH THE SIGNATURE. AS A RESULT, BOTH PAST AND FUTURE SIGNATURES BECOME INSECURE: AN ATTACKER WHO LEARNS K CAN FORGE FUTURE SIGNATURES AND CAN RECOVER THE PRIVATE KEY AND FORGE PAST SIGNATURES.

2.3 NO, HMAC IS A KEYED HASH THAT PROVIDES INTEGRITY AND AUTHENTICITY UNDER A SHARED SECRET KEY. BOTH PARTIES CAN GENERATE AND VERIFY THE SAME TAG. BECAUSE OF THIS SIMMETRY, NEITHER PARTY CAN LATER PROVE TO A THIRD PARTY WHO ACTUALLY CREATED A GIVEN HMAC, SINCE EITHER ONE COULD HAVE DONE IT. NON REPUDIATION REQUIRES ASYMMETRIC MECHANISMS LIKE DIGITAL SIGNATURES, WHERE ONLY THE HOLDER OF THE PRIVATE KEY CAN PRODUCE A VALID SIGNATURE AND ANYONE WITH THE PUBLIC KEY CAN VERIFY IT.

3.1 SALTING IS THE PROCESS OF ADDING A RANDOM VALUE (SALT) TO A PASSWORD BEFORE HASHING IT. EACH USER GETS A UNIQUE SALT, STORED ALONGSIDE THE HASH.
THE PURPOSE IS TO PREVENT ATTACKS WITH PRECOMPUTED TABLES (RAINBOW TABLES) AND TO ENSURE THAT TWO USERS WITH THE SAME PASSWORD HAVE DIFFERENT HASHES.
THE SYSTEM GENERATES A RANDOM SALT WHEN THE PASSWORD IS SET, CONCATENATES IT WITH THE PASSWORD, AND STORES SALT || H(SALT || PASSWORD).
DURING LOGIN, THE SYSTEM REPEATS THE PROCESS AND COMPARES RESULTS.

3.2 THEY ARE LARGE PRECOMPUTED TABLES THAT MAP PLAINTEXT PASSWORDS TO THEIR HASH VALUES. THEY ARE USED BY ATTACKERS TO REVERSE HASH FUNCTIONS: INSTEAD OF BRUTE FORCING EVERY POSSIBLE PASSWORD ONLINE, THE ATTACKER COMPUTES HASHES IN ADVANCE AND LATER LOOKS UP THE HASH FOUND IN A PASSWORD DB TO RECOVER THE ORIGINAL PASSWORD.
MANY SYSTEMS USED TO STORE ONLY UNSALTED HASHES, SO THE SAME PASSWORD PRODUCES THE SAME DIGEST.

3.3 THIS ATTACK CONSISTS OF TESTING CANDIDATE PASSWORDS AGAINST STORED PASSWORD HASHES WITHOUT INTERACTING WITH THE SYSTEM.

1. OBTAIN THE PASSWORD DB
2. FOR EACH CANDIDATE PASSWORD p IN A DICTIONARY OF COMMON PASSWORDS:
   a. IF A SALT IS USED, COMBINE p WITH THE CORRESPONDING SALT
   b. COMPUTE THE HASH OF THE CANDIDATE
   c. COMPARE THE RESULT WITH THE STORED HASH
3. IF A MATCH IS FOUND, p IS THE RECOVERED PASSWORD

SINCE THIS IS DONE OFFLINE, THE ATTACKER CAN TRY MANY TIMES WITHOUT BEING DETECTED.

4.1 TO CONFIGURE WHITELISTING, WE SET THE DEFAULT IPTABLES POLICIES ON INPUT, OUTPUT, FORWARD TO DROP.

```
IPTABLES -P INPUT DROP
IPTABLES -P OUTPUT DROP
IPTABLES -P FORWARD DROP
```

THEN EXPLICITLY ADD ACCEPT RULES ONLY FOR THE DESIRED TRAFFIC:

```
IPTABLES -A INPUT -i lo -j ACCEPT
IPTABLES -A OUTPUT -o lo -j ACCEPT

IPTABLES -A FORWARD -i eth0 -o eth1 -p TCP --DPORT 80 -j ACCEPT
IPTABLES -A FORWARD -i eth0 -o eth1 -p TCP --DPORT 443 -j ACCEPT
IPTABLES -A FORWARD -i eth0 -o eth1 -p UDP --DPORT 53 -j ACCEPT
IPTABLES -A FORWARD -m CONNTRACK --CTSTATE ESTABLISHED,RELATED
-j ACCEPT
```

4.2 TO ALLOW LAN-FIREWALL COMMUNICATION WE ADD ACCEPT RULES ON INPUT/OUTPUT FOR INTERFACE eth0:

```
IPTABLES -A INPUT  -i eth0 -j ACCEPT
IPTABLES -A OUTPUT -o eth0 -j ACCEPT
```

TO BLOCK FIREWALL-INTERNET TRAFFIC WE ADD DROP RULES ON INPUT/OUTPUT FOR INTERFACE eth1:

```
IPTABLES -A INPUT  -i eth1 -j DROP
IPTABLES -A OUTPUT -o eth1 -j DROP
```

4.3 THE FIREWALL SHOULD BE CONFIGURED IN WHITELIST MODE, ALLEPTING ONLY NECESSARY TRAFFIC AND DROPPING EVERYTHING ELSE.
ANTI SPOOFING RULES SHOULD BLOCK PACKETS WITH INVALID OR PRIVATE SOURCE ADDRESSES ARRIVING FROM THE INTERNET. STATEFUL FILTERING (ESTABLISHED, RELATED) ENSURES ONLY LEGITIMATE REPLIES ARE ACCEPTED. CONNECTION LIMITS CAN BE SET TO PREVENT A SINGLE ATTACKER FROM EXHAUSTING RESOURCES.