HY457 - Recitation
(Chapters 7-11)
Chapter 7
Question 1

- What types of resources are targeted by DoS attacks?
Question 1

- What types of resources are targeted by DoS attacks?
  - Network bandwidth
  - System resources
  - Application resources
Question 2

- Why do many DoS attacks use packets with spoofed source addresses?
Question 2

Why do many DoS attacks use packets with spoofed source addresses?

Many DoS attacks use packets with spoofed source addresses so any response packets that result are no longer reflected back to the original source system, but rather are scattered across the Internet to all the various forged source addresses.

Some of these addresses might correspond to real systems, others may not be used, or not be reachable. Any response packets returned as a result only add to the flood of traffic directed at the target system.
Question 3

- What is “backscatter traffic”?
- Which types of DoS attacks can it provide information on?
- Which types of attacks does it not provide any information on?
Question 3

What is “backscatter traffic”?

“Backscatter traffic” are packets generated in response to a DoS attack packet with a forged random source address, e.g. the ICMP echo response from an ICMP echo request being used to flood a link.

Monitoring these packets, which are randomly distributed over the Internet, gives valuable information on the type and scale of attacks.
Question 3

- Which types of DoS attacks can it provide information on?
- Which types of attacks does it not provide any information on?

- This backscatter traffic provides information on any DoS attacks that use a forged random source address with the destination address being the target, including various single and distributed flooding attacks, and syn spoofing attacks.

- It does not provide information on attacks that don't use randomly forged source addresses, or reflection or amplification attacks where the forged source address is that of the desired target.
Question 4

- What defenses are possible against nonspoofed flooding attacks?
- Can such attacks be entirely prevented?
Question 4

- Non-spoofed flooding attacks are best defended against by:
  - The provision of significant excess network bandwidth and replicated distributed servers, particularly when the overload is anticipated. This does have a significant implementation cost though.
  - Rate limits of various types on traffic can also be imposed.
  - However such attacks cannot be entirely prevented, and may occur “accidentally” as a result of very high legitimate traffic loads.
What defenses are possible against TCP SYN spoofing attacks?
It is possible to specifically defend against the SYN spoofing attack by using a modified version of the TCP connection handling code, which instead of saving the connection details on the server, encodes critical information in a “cookie” sent as the server’s initial sequence number.

Cookie is a small piece of data that a server sends to the user's web browser. The browser/client may store it and send it back with the next request to the same server. Typically, it's used to tell if two requests came from the same browser/client — keeping a user logged-in.

When a legitimate client responds with an ACK packet, the server is able to reconstruct this information.
Question 5

- Typically this technique is only used when the table overflows, as it does take computation resources on the server, and also blocks the use of certain TCP extensions.
Question 1

- What is the difference between a packet filtering firewall and a stateful inspection firewall?
What is the difference between a packet filtering firewall and a stateful inspection firewall?

- **Traditional packet filter** makes filtering decisions on an individual packet basis and does not take into consideration any higher layer context.

- **Stateful inspection packet filter** tightens up the rules for TCP traffic by creating a directory of outbound TCP connections. There is an entry for each currently established connection. The packet filter will now allow incoming traffic to high-numbered ports only for those packets that fit the profile of one of the entries in this directory.
Question 2

- Why is it useful to have host-based firewalls?
Filtering rules can be tailored to the host environment. Specific corporate security policies for servers can be implemented, with different filters for servers used for different applications.

Protection is provided independent of topology. Thus both internal and external attacks must pass through the firewall.

Used in conjunction with stand-alone firewalls, the host-based firewall provides an additional layer of protection. A new type of server can be added to the network, with its own firewall, without the necessity of altering the network firewall configuration.
Question 3

- How does an IPS differ from a firewall?
How does an IPS differ from a firewall?

- An IPS blocks traffic, as a firewall does, but makes use of the types of algorithms developed for IDSs.

For example:
- Firewall block all traffic of port 8080 without looking a payload
- IPS will block packet’s at any port if the payload contain malicious content
Chapter 10
Question 1

- Describe how a stack buffer overflow attack is implemented.
Describe how a stack buffer overflow attack is implemented.

- A stack buffer overflow occurs when the targeted buffer is located on the stack, usually as a local variable in a function’s stack frame. If the function code that copies externally sourced data into such a buffer fails to correctly limit the amount of data written, then the values of adjacent variables, and even control fields such as the saved frame pointer and return address, can be overwritten.

This can result in the program crashing, or in execution being transferred to (shell) code the attacker provides.
Question 2

- Describe what a NOP sled is and how it is used in a buffer overflow attack.
A NOP sled is a run of NOP (no operation, do nothing) instructions, which are included before the desired shellcode to help overcome the lack of knowledge by the attacker of its precise location. In a buffer overflow attack, the attacker arranges for the transfer of control (via overwritten return address) to occur somewhere in the NOP Sled (guessing around the middle of the most likely location).

When control transfers, no matter where in this run it occurs, the CPU executes NOPs until it reaches the actual desired shellcode.
Question 3

- List and briefly describe some of the defenses against buffer overflows that can be used when compiling new programs.
Question 3

List and briefly describe some of the defenses against buffer overflows that can be used when compiling new programs.

- writing programs using a modern high-level programming language that is not vulnerable to buffer overflow attacks
- using safe coding techniques to validate buffer use
- using language safety extensions and/or safe library implementations
- using stack protection mechanisms (for example canaries)
List and briefly describe some of the defenses against buffer overflows that can be implemented when running existing, vulnerable programs.
Question 4

- using “Executable Address Space Protection” that blocks execution of code on the stack, heap, or in global data

- using “Address Space Randomization” to manipulate the location of key data structures such as the stack and heap in the processes address space

- by placing guard pages between critical regions of memory in a processes address space
Chapter 11
Define an injection attack. List some examples of injection attacks. What are the general circumstances in which injection attacks are found?
Question 1

Define an injection attack. List some examples of injection attacks. What are the general circumstances in which injection attacks are found?

- **Injection attacks** refer to a wide variety of program flaws related to invalid handling of input data, particularly when such input data can accidentally or deliberately influence the flow of execution of the program.

- Examples: a) command injection, b) SQL injection, c) code injection, d) remote code injection
Question 1

There is a wide variety of mechanisms that can result in injection attacks. These include:

- When input data is passed as a parameter to another helper program (command) or to a database system (SQL), whose output is then processed and used by the original program.

- When the input includes either machine or script code that is then executed/interpreted by the attacked system (code).
Question 2

State the similarities and differences between command injection and SQL injection attacks.
In a command injection attack, the unchecked input is used in the construction of a command that is subsequently executed by the system with the privileges of the attacked program.

In an SQL injection attack, the user-supplied input is used to construct a SQL request to retrieve information from a database.

In both cases the unchecked input allows the execution of arbitrary programs/SQL queries rather than the program/query specified by the program designer. They differ in the syntax of the respective shell/SQL metacharacters used that allow this to occur.
Define *input fuzzing*. State where this technique should be used.
Define *input fuzzing*. State where this technique should be used.

- Input fuzzing is a software testing technique that uses very large amounts of randomly generated data as inputs to a program, to determine whether the program or function correctly handles all such abnormal inputs, or whether it crashes or otherwise fails to respond appropriately.

- The major advantage of fuzzing is its simplicity, low cost, and its freedom from assumptions about the “expected” input to any program, service or function. It ought to be deployed as a component of any reasonably comprehensive testing strategy, especially in relation to commonly deployed software.