Operating Systems Security

The Boot Sequence

• The action of loading an operating system into memory from a powered-off state is known as booting or bootstrapping.

• When a computer is turned on, it first executes code stored in a firmware component known as the BIOS (basic input/output system).

• On modern systems, the BIOS loads into memory the second-stage boot loader, which handles loading the rest of the operating system into memory and then passes control of execution to the operating system.
BIOS Passwords

- A malicious user could potentially seize execution of a computer at several points in the boot process.
- To prevent an attacker from initiating the first stages of booting, many computers feature a BIOS password that does not allow a second-stage boot loader to be executed without proper authentication.

Hibernation

- Modern machines have the ability to go into a powered-off state known as hibernation.
- While going into hibernation, the OS stores the contents of machine’s memory into a hibernation file (such as hiberfil.sys) on disk so the computer can be quickly restored later.
- But... without additional security precautions, hibernation exposes a machine to potentially invasive forensic investigation.

1. User closes a laptop computer, putting it into hibernation.

2. Attacker copies the hiberfil.sys file to discover any unencrypted passwords that were stored in memory when the computer was put into hibernation.
Event Logging

- Keeping track of what processes are running, what other machines have interacted with the system via the Internet, and if the operating system has experienced any unexpected or suspicious behavior can often leave important clues not only for troubleshooting ordinary problems, but also for determining the cause of a security breach.

Process Explorer
Memory and Filesystem Security

• The contents of a computer are encapsulated in its memory and filesystem.
• Thus, protection of a computer’s content has to start with the protection of its memory and its filesystem.

Password Security

• The basic approach to guessing passwords from the password file is to conduct a dictionary attack, where each word in a dictionary is hashed and the resulting value is compared with the hashed passwords stored in the password file.
• A dictionary of 500,000 “words” is often enough to discover most passwords.
Password Salt

- One way to make the dictionary attack more difficult to launch is to use **salt**.
- Associate a random number with each userid.
- Rather than comparing the hash of an entered password with a stored hash of a password, the system compares the hash of an entered password and the salt for the associated userid with a stored hash of the password and salt.

How Password Salt Works

**Without salt:**

1. User types userid, X, and password, P.
2. System looks up H, the stored hash of X’s password.
3. System tests whether h(P) = H.

**With salt:**

1. User types userid, X, and password, P.
2. System looks up S and H, where S is the random salt for userid X and H is stored hash of S and X’s password.
3. System tests whether h(S || P) = H.
How Salt Increases Search Space Size

- Assuming that an attacker cannot find the salt associated with a userid he is trying to compromise, then the search space for a dictionary attack on a salted password is of size
  \[2^B \times D,\]
  where \(B\) is the number of bits of the random salt and \(D\) is the size of the list of words for the dictionary attack.

- For example, if a system uses a 32-bit salt for each userid and its users pick passwords in a 500,000 word dictionary, then the search space for attacking salted passwords would be
  \[2^{32} \times 500,000 = 2,147,483,648,000,000,\]
  which is over 2 quadrillion.

- Also, even if an attacker can find a salt password for a userid, he only learns one password.