HW1

Answer the following questions. For questions asking for short answers, there may not necessarily be a "right" answer, although some answers may be more compelling and/or much easier to justify. But I am interested in your explanation (the "why") as much as the answer itself. Also, do not use shorthand: write your answers using complete sentences.

A fundamental aspect of protection in operating systems is rights amplification. Rights amplification enables a more privileged protection domain to perform an operation on behalf of a less privileged protection domain in a controlled fashion without violating protection in the system. For each of the following operating systems, state

a. the protection domain that they support

b. the mechanism for crossing protection domains

c. how rights are represented

d. how rights are amplified crossing domains

e. how the OS determines whether to allow the domain crossing.

Support your answers with a bit of explanation, such as a concise summary explanation in your own words (a quote of a phrase or sentence from the papers is fine as well). For instance, two possible answers to part (a) for Hydra are:

> 1) A protection domain in Hydra is the "local name space" (LNS). An LNS represents the current set of objects and rights to which a process has access, and those objects and rights change when a process moves from one LNS to another.
> 2) A protection domain in Hydra is the "local name space" (LNS): "At any instant, the execution environment (domain) of a program is defined by an LNS object associated with it…the rights lists in each capability define the permissible access rights of this program at this instant." (Hydra p. 341).

In other words, we're looking for more than just "local name space" – but at the same time your answers don't have to be lengthy discussions. The balance in the example above is fine.

Hydra

a. Hydra's protection domain is "local namespace" (LNS). Each LNS has its list of objects, local variables and trasitive closure of all capabilites. It defines the environment the process is running and its access rights.

b. Normally, each precedure executes in their own domain, but it is possible to call precedures with provided parameters, and Hydra will construct a new LNS to execute and return back to the caller.

c. Capability: it contains a reference to the object and a list of access rights. In theory capabilities cannot be forge since only kernel can modify it, and capabailities can be pass from one process to another.

d. When the caller called the callee, Hydra will generate a new capabilities based on the capabilities passed by caller and the callee for the new constructed LNS. The precedure running within the LNS will have more power than the caller to finish the action based on the parameters passed in.

e. Template. Each predecure object has a template, and it check whether the caller has sufficient privileges and Hydra will generate a LNS with corresponded capabilities for the precedures to execute and return to the caller. However, caller during this process cannot obtain the capabilities as the operation is under a different LNS, and precedures can make use of both the callers' rights and its own right.

Multics

a. Multics's protection domain is "Protected Subsystem". Each subsystem contains its procedures and data, and has corresponded rings. Arbitrary number of subsystem may be created.

b. A process may call another subsystem via its entry point, known as gate.

c. The rights are represented as access control list and segment descriptor. the descriptor is sync up with ACL, and descriptor is mostly for speed up checking process.

d. When a higher ring process called into lower ring segment, it will be switched to the ring of the segment and execute.

e. Corresponded descriptor will be checked by hardware, and see if the caller's subsystem entry point is in the list.

Unix

a. Each user will be assigned an unique ID. Each file will be marked with the user id when created, and each process will have its own virtal memory space.

b. A user may call a program with proper setting of set-user-ID feature, so that a user have no access to a specific file, for example, may access it via some programs with bit being set.

c. It is represented by the access control scheme via the set-user-identification bit. The actual program can still get the real user ID for its own verification.

d. Each user may have the set-user-identification bit of their own programs so that other user may run that program as other's identification.

e. Before running the program, Unix will check scheme (6bits for read, write, execute) and the set-user-identification bit.

Pilot

a. It does not have any protection domain concepts as it is designed as personal computer for single user. It is also using single memory space.
b. since a), N/A
c. The rights here is represented as capabilties. However, it is stated that they are "intended for defensive protection against errors" (Multics, P.82).
d. since a), N/A
e. since a), N/A

Some of the systems we have read about and discussed use specialized hardware to facilitate their implementation. Choose one such instance, describe the hardware that was used, and what advantage it gave the system implementors and designers. What is one drawback of relying upon specialized hardware? Do we still use hardware of this form today?

Advantage In Multics, the ring system requires special hardware support. To be specific, Multics has 8 rings, and among rings, memory are isolated. The communication/data transfer can only be made using "gate". The ring implementation is done in hardware level most of the time for maximum level of security.

Disadvantage One of the hardware pieces supported 8 rings is Honeywell 6180 (Multics P395). The hardware ring implementation makes access checks efficient and secure, ease the pressure of security concerns. On the otherhand, the ring requirement rarises the bar of building capable hardware, which hinders the popularity of Multics Operating System.

Today Nowadays, still, our CPUs still support rings, like x86 support 4 rings. However, most of the operating systems only use two of them: 0 for kernel mode and 3 for user mode. An operating system requires less rings mean it could run on more hardwares. It is believed to be one of the reason why Unix become more popular later on with its portability.

Pilot made a strong and persuasive argument for tailoring the design and implementation of operating systems to personal computers. We have also seen commercial operating systems like MS-DOS, Windows before NT, and “classic” MacOS tailored towards personal computers as well. Why do you think we still run multi-user timesharing systems like Unix on our PCs? (Consider, for example, the requirements we have of the systems that we use today.)

Even though most of the people nowadays have their own devices where most of the time will not be used by others, the multi-user timesharing is still pretty useful for isolating applications – we assign different user roles and groups when running different applications on our devices, even though from the users’ perspective, it is still a “single user” experience

The prevalence of Internet connection made it possible for hacker to gain access without physical access to the machine, which breaks the assumptions made by Pilot at the time. From both the security and privacy perspective, user would like to run programs with minimum privilege and limited information access if possible. This is mostly done by setting programs under different non-privilege users in moderm operating system such as linux and windows. I believe it is a perfect solution for lightweight isolation requirement under single user situation.

Also part of the reason may be for the sharing. Big data analysis requires unprecedented amount of compute power and require pretty expensive hardware for individuals. The reasearch lab, for example, may purchase a single machine with big harddrives and high-end graphics card to share among the lab. It is more or less like the situation years ago but it is still an requirement for the OS today.