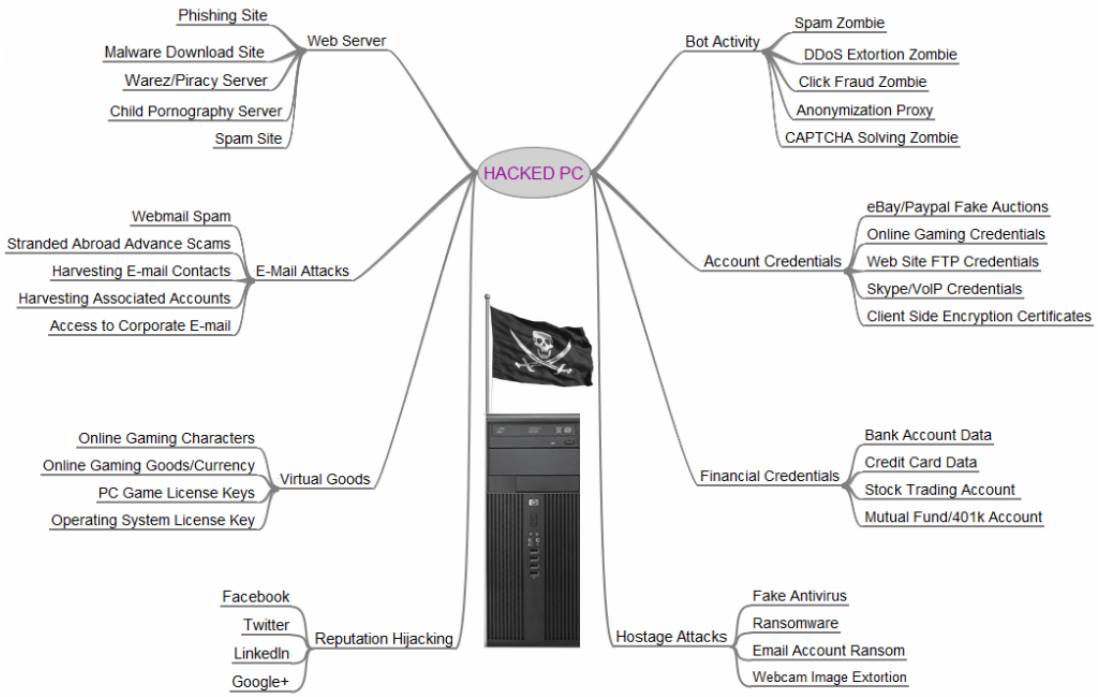
Local privilege escalation

November 18, 2021

Value of hacked PC



Secure operating system

A secure operating system provides security mechanisms that ensure that the system's security goals are enforced despite the threats faced by the system.

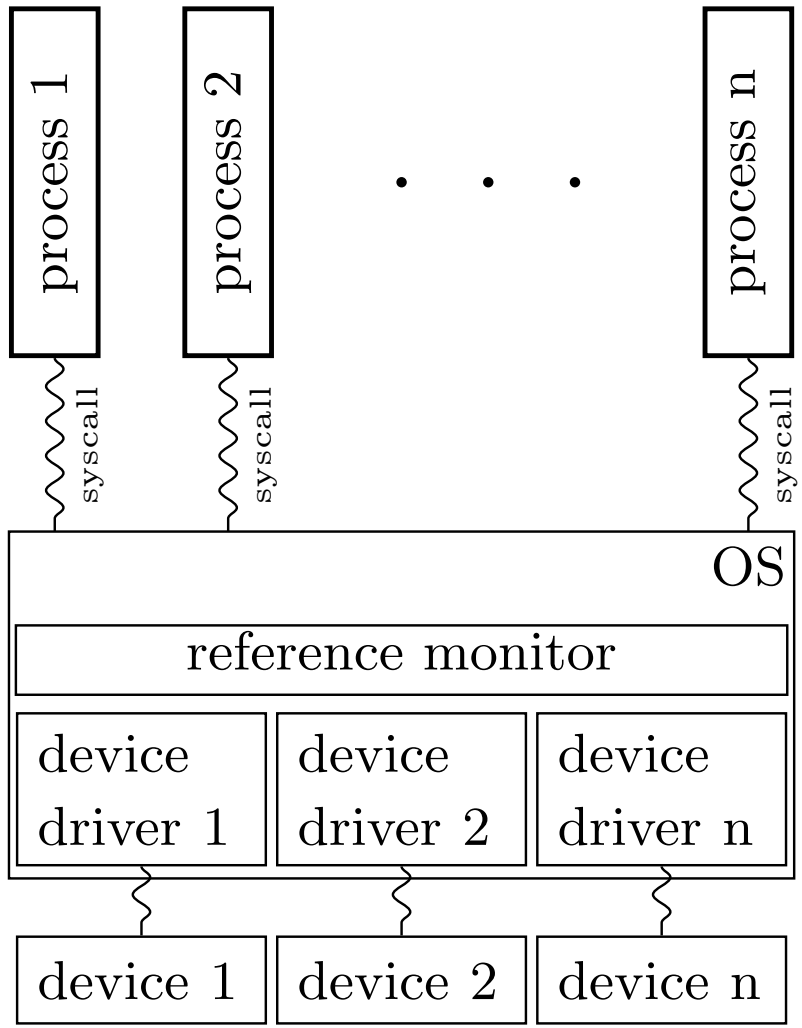
Minimal functional requirements on OS

- Scheduling
- Process isolation (memory management)
- Inter-process communication

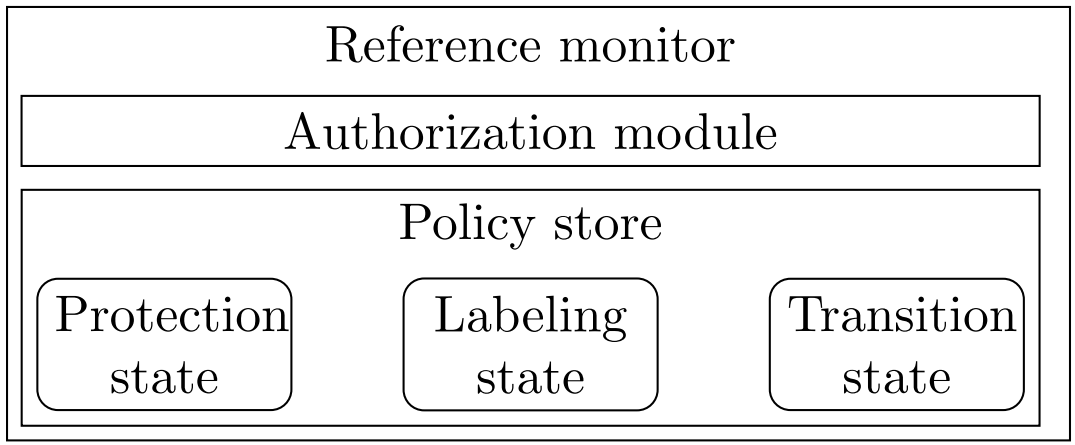
Requirements on secure OS

- Complete mediation
- Tamperproof
- Verifiable

Ensuring complete mediation



Scheme of the reference monitor



Example of TOCTTOU

Victim

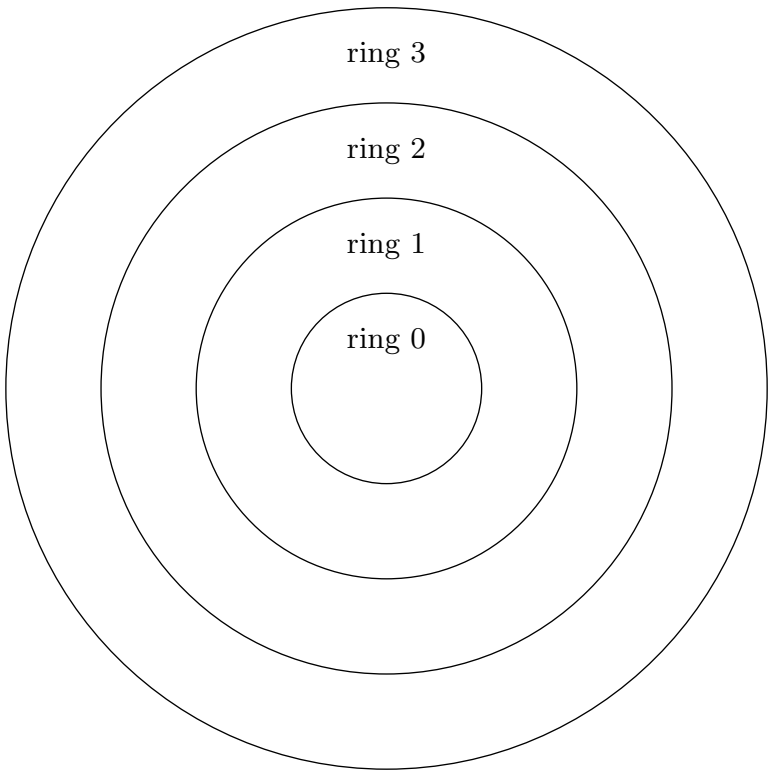
```
if (access("file", W_OK) != 0) {
 exit(1);
}

fd = open("file", O_WRONLY);
// Actually writing over
// /etc/passwd
write(fd, buffer, sizeof(buffer));
```

Attacker

```
//
//
// After the access check
symlink("/etc/passwd", "file")
// Before the open, "file"
// points to the password
// database
//
```

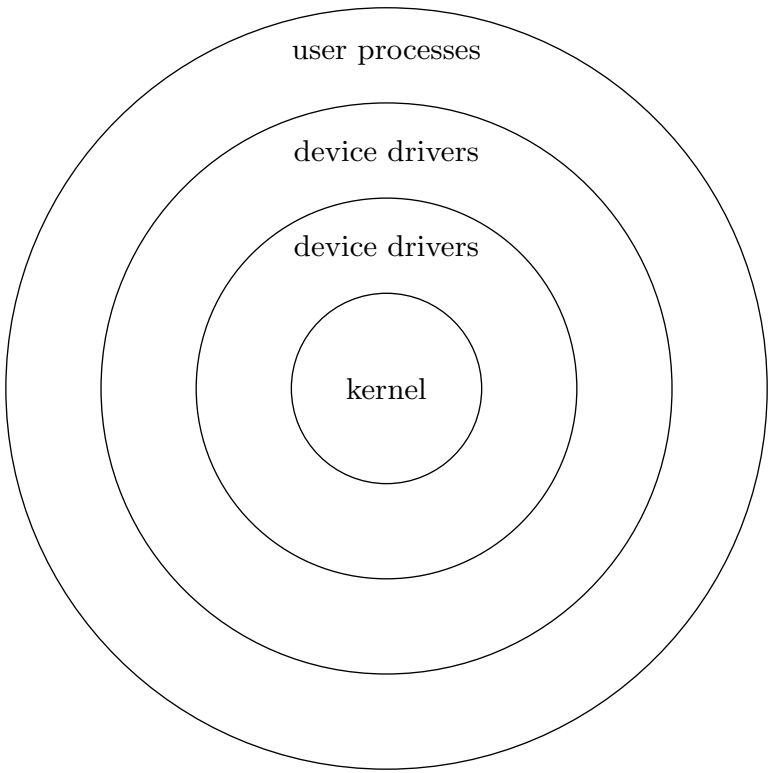
Plan

Isolating kernel — protection rings



- Requires privileged instructions.
- Each ring can access data in further out ring.
- Each ring can execute only its own instructions.
- Crossing rings is allowed through *gates*.

Isolating kernel — protection rings



- Ideally the ring structure reflects the importance of the code.

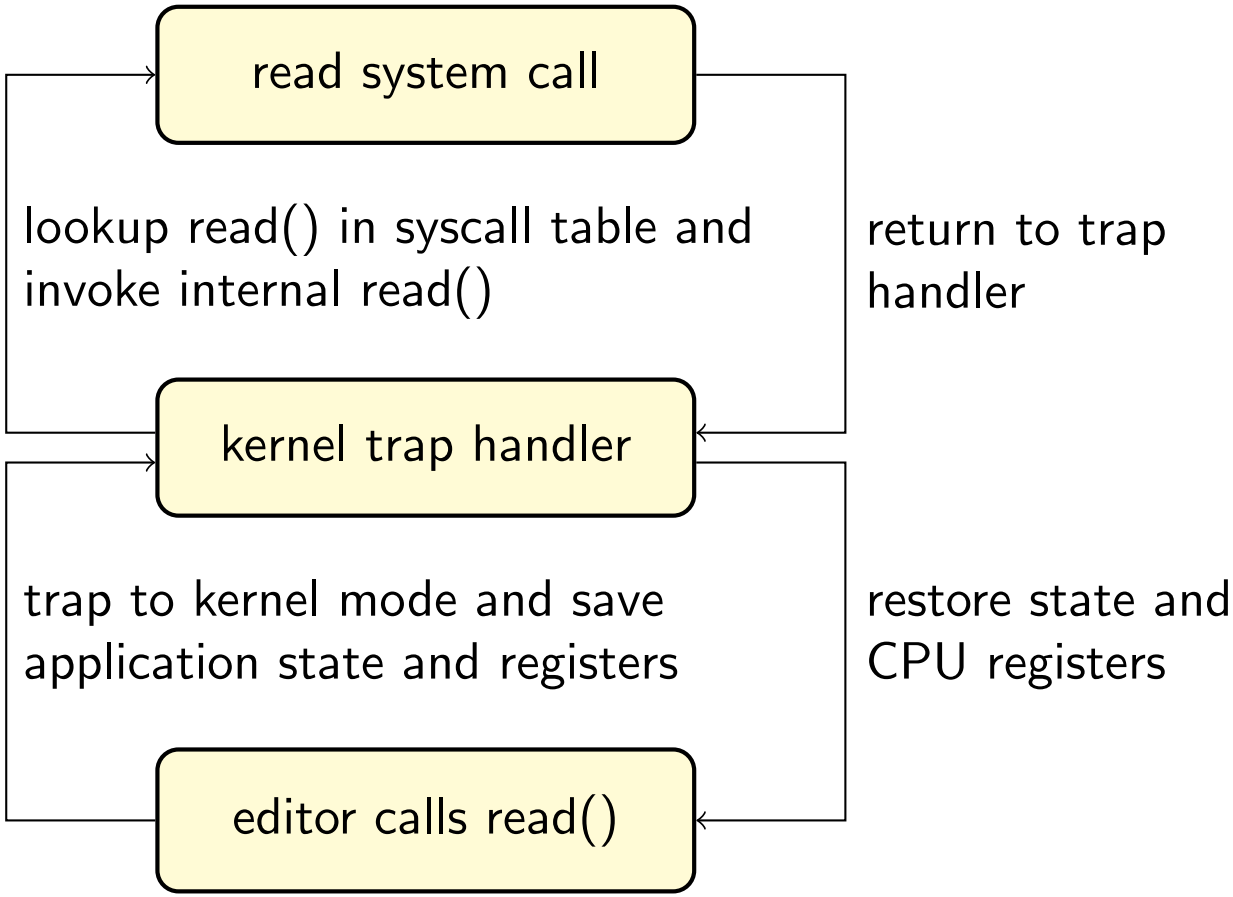
Question?

Which access model ring structure resembles?

When crossing ring is needed?

- request resources from the OS;
- establishes communication with other processes;
- request other services from the OS.

How gates and syscalls are implemented?



Isolation of processes

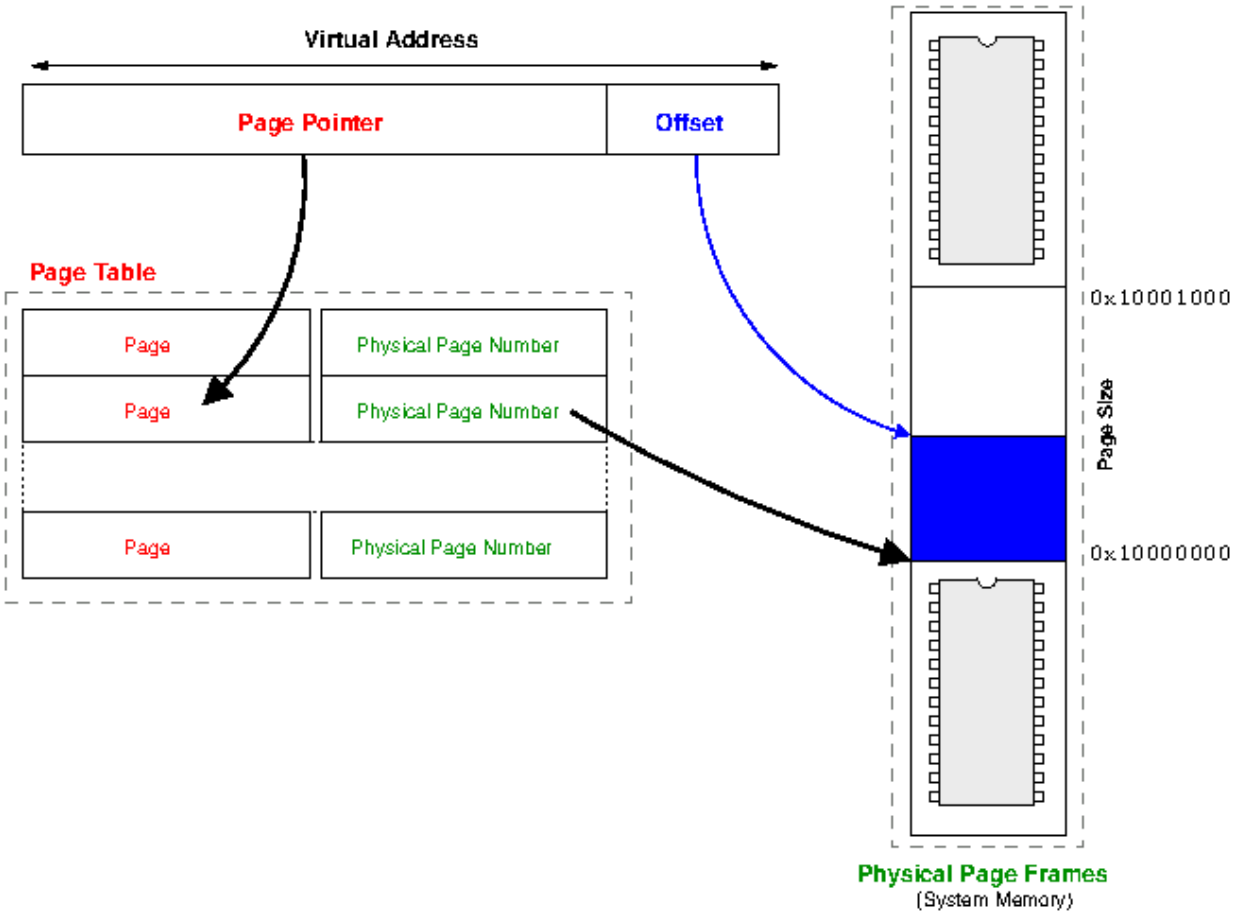
The process should feel like running alone.
Therefore it needs to have separate

- registers,
- all kernel structures (file descriptors, network connections etc.),
- memory.

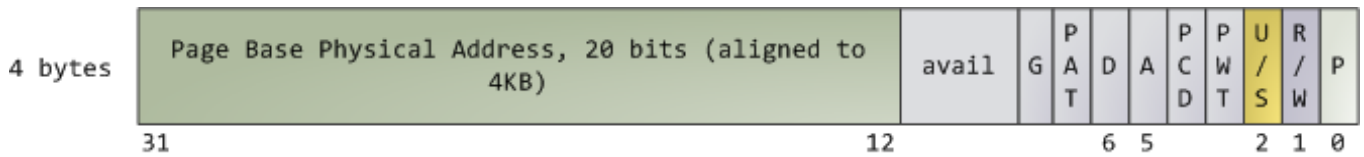
Virtual memory management

- The process sees a flat memory of size 2^{48} bits.
- Memory is divided into *pages* / *frames* and allocated page by page.
- To access the memory, virtual address is transformed to the physical address.
- The process only knows the virtual address, the translation to physical address can be handled by
  - kernel, or
  - hardware memory management unit (special instructions reserved for the ring 0)

Virtual memory management



Page table entry (32 bits)



Added benefits of virtual addresses memory management

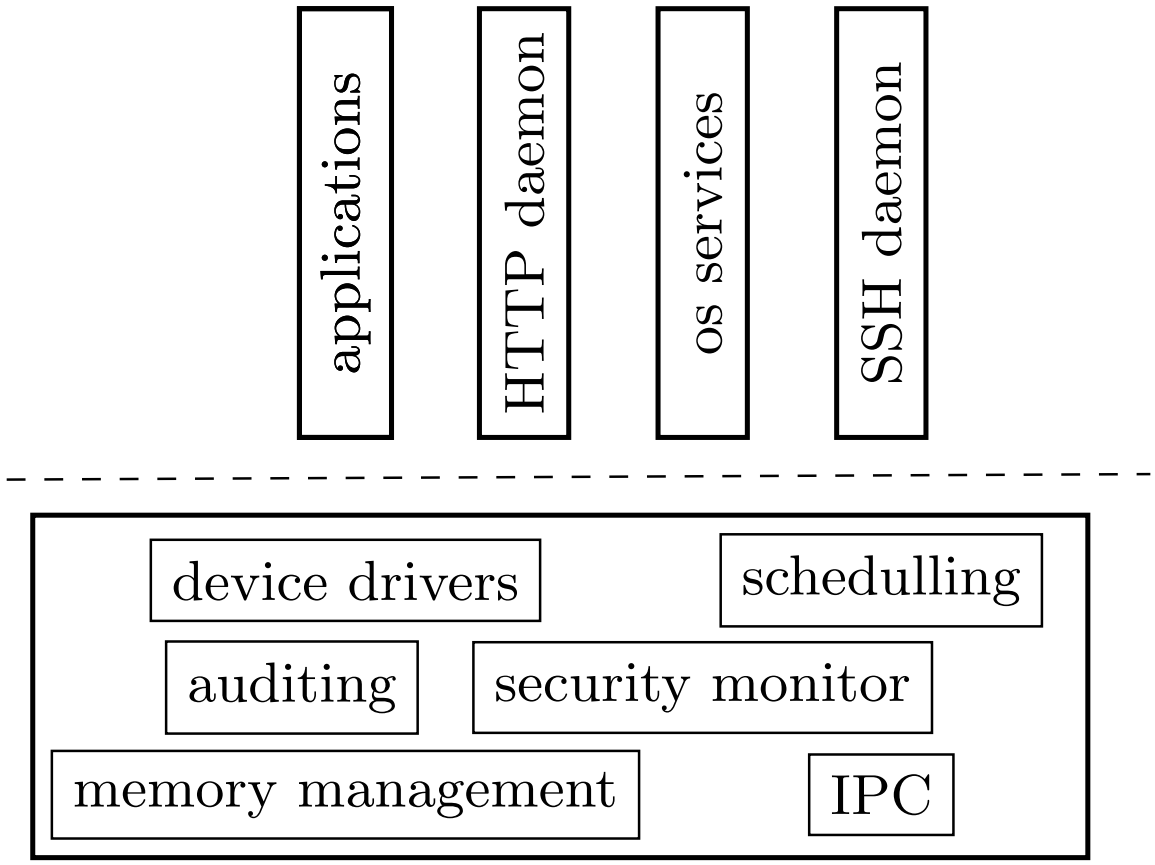
- swapping
- shared code between processes
- shared memory between processes (modulated by the kernel)
- Multi-level virtual address
- no execution bits

Plan

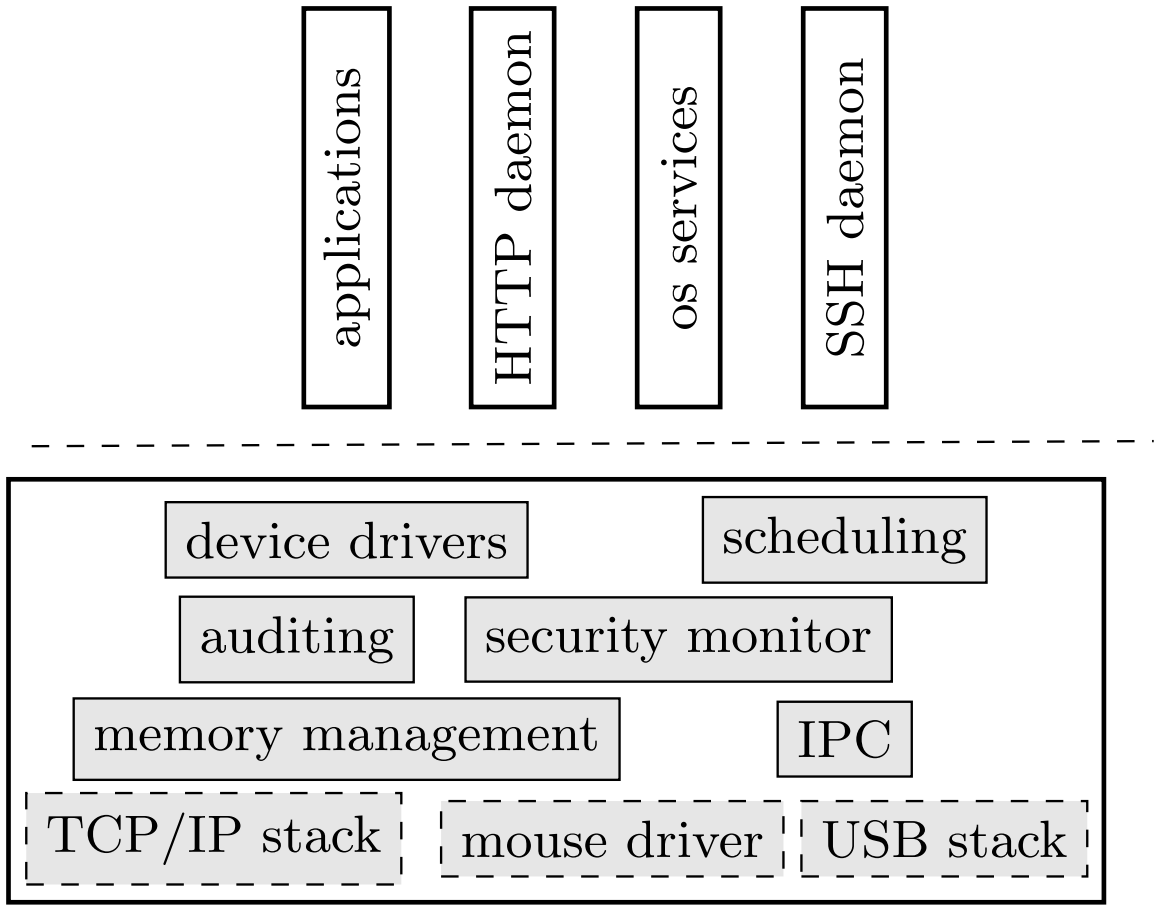
What are parts of trusted computer base on real OS?

- Kernel and all its modules (device drivers)
- Window management systems
- Systems verifying authenticity (SSH, login).
- all root processes `ps -ax -u root`

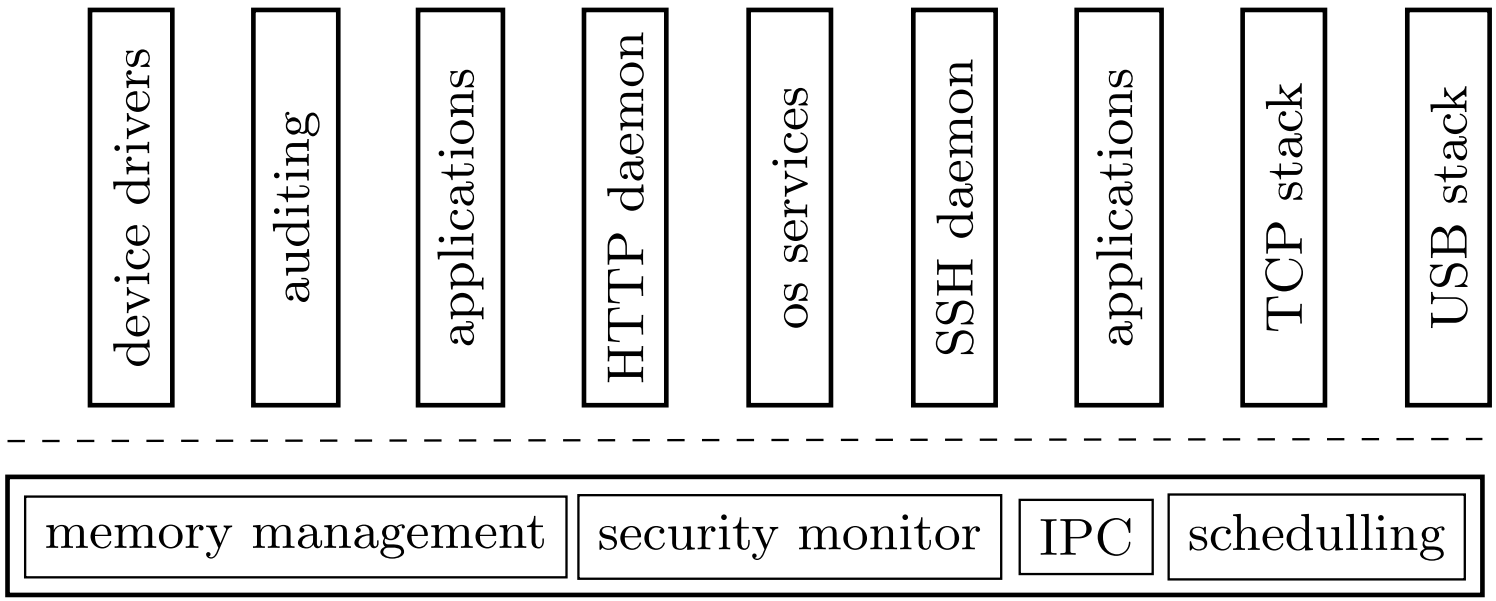
Monolithic design



Modular design



Micro kernel (seL4)



Plan

"Orange Book" — Trusted Computer System Evaluation Criteria

Each category imposes requirements of four categories

1. the security policy model, including the administration of policies described in the model and the labelling of system resources;
2. the level of accountability for system actions, including authentication of individual subjects and audit of system actions;
3. the degree of operational assurance that the system behaves as expected, including the implementation and maintenance of the system;
4. the documentation provided to support the design, implementation, assurance, and maintenance of the system.

Common Criteria Evaluation Assurance Levels

Level and TCSEC Map	Requirements
EAL1	functionally tested
EAL2 (C1: low)	structurally tested
EAL3 (C2/B1: moderate)	methodically tested and checked
EAL4 (C2/B1: medium)	methodically designed, tested and reviewed
EAL5 (B2: high)	semiformally designed and tested
EAL6 (B3: high)	semiformally verified design and tested
EAL7 (A1: high)	formally verified design and tested

C1 — discretionary security protection

- discretionary access control of named users and objects;
- all users have to be authenticated;
- hardware supports control mediation;
- requires basic testing for obvious flaws;
- only basic documentation is needed.

C2 — controlled access protection

- the granularity of access rights are on the level of single user;
- authentication is based on secret and protected from other users;
- auditing of specific set of events into log;
- reusing objects means that previous content is not accessible;
- testing for obvious flaws and design;
- documentation for user, facilities, design, and testing.
- Windows NT 4.0, most UNIXes

B1 — labeled security protection

- DAC as C1&C2 and mandatory access control to each subject is associated a label with multi-layer policy;
- labels are integrity protected and are *persistently* attached to the object;
- authentication identifies user and its security level;
- assurance requires security mechanisms to work as claimed in documentation;
- the documentation supports testing of the system through detailed description of the security model, protection mechanisms, and how the model is satisfied.
- Example: SE Linux, Trusted Solaris V1.1 Compartmented Mode Workstation

B2 — structured protection

- requiring enforcement on access to all subjects and objects (i.e., not just named ones);
- covert channel protections;
- protection-critical part of the TCB must be identified, and its interface must be well-defined
- TCB must be shown to be "relatively resistant to penetration."
- Example: Trusted Xenix 3.0 and 4.0

B3 — security domains

- TCB satisfy the reference monitor concept;
- TCB design and implementation are directed toward minimal size and minimal complexity
- system is expected to be "highly resistant to penetration."
- audit subsystem must be able to record all security-sensitive events.
- Example: BAE Systems XTS 400

A1 — verified design

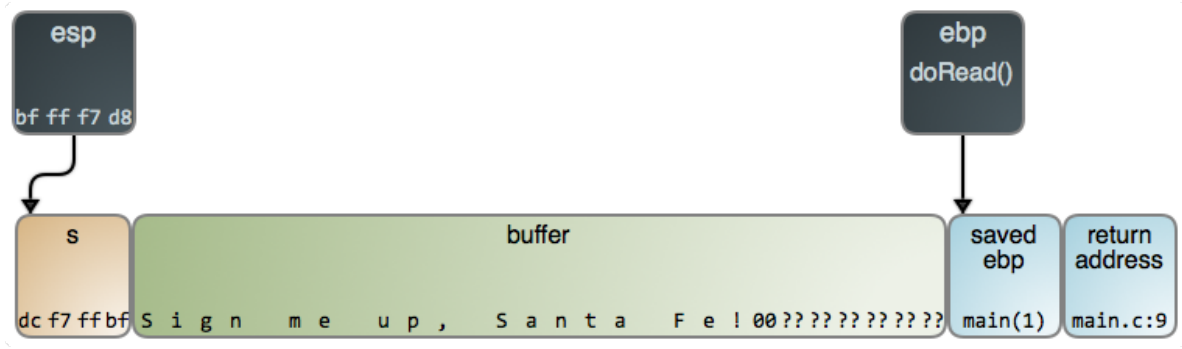
- A formal model of the security policy must be documented and include a mathematical proof that the model is consistent with the policy;
- An formal top-level specification (FTLS) must specify the functions that the TCB performs and the hardware/firmware support for separate execution domains;
- The FTLS of the TCB must be shown to be consistent with the formal model of the security policy;
- The TCB implementation must be consistent with the FTLS;
- Formal analysis techniques must be used to identify and analyse covert channels. The continued existence of covert channels in the system must be justified.
- Examples: Honeywell's SCOMP, Aesec's GEMSOS, and Boeing's SNS Server, canceled DEC VAX Security kernel

Vulnerabilities — Exploits — Remedies

Vulnerabilities	Exploits	Remedies
Buffer overflow	Injecting code to stack	Canaries
Heap overflow	Return 2 libc	DEP
Use after free	ROP	ASLR
Structured Exception Handler overwrite		SEHOP

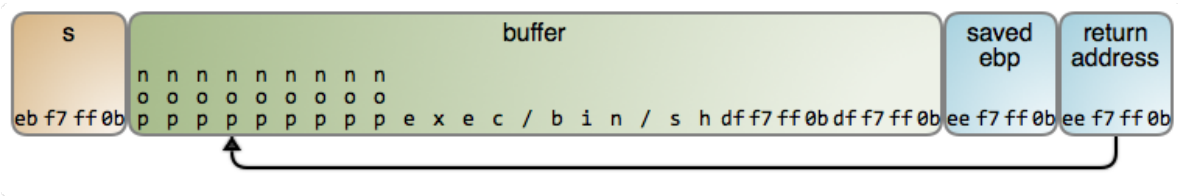
Buffer overflow

```
void doRead(){
  char buffer[28];
  gets(buffer);
}

int main(int argc){
  doRead();
}
```



Buffer overflow

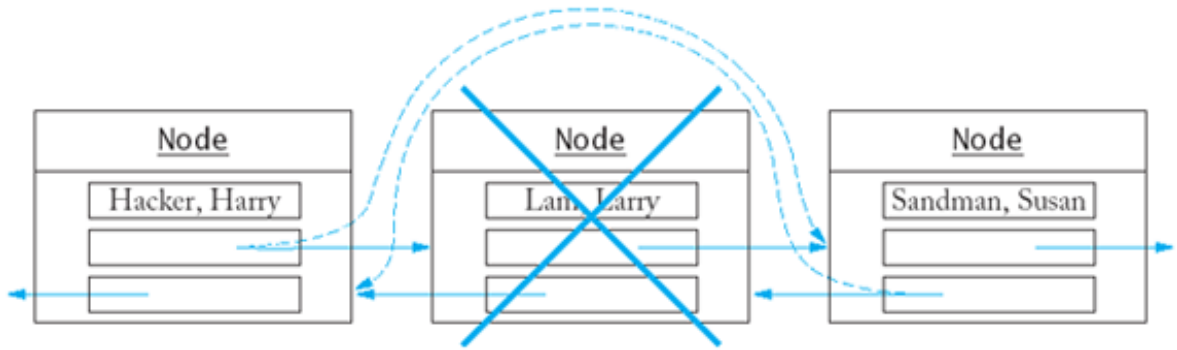
```
void doRead(){
  char buffer[28];
  gets(buffer);
}

int main(int argc){
  doRead();
}
```



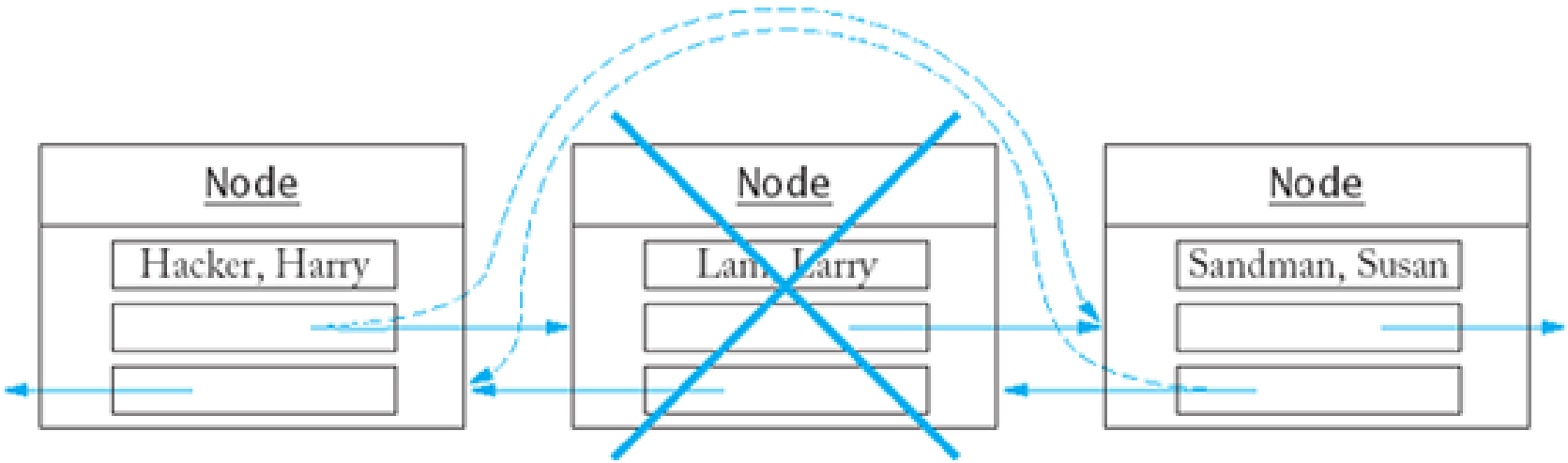
Canaries

Heap overflow

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
  char *buf1 = malloc(128);
  char *buf2 = malloc(128);

  read(fileno(stdin), buf1, 200);

  free(buf2);
  free(buf1);
}
```

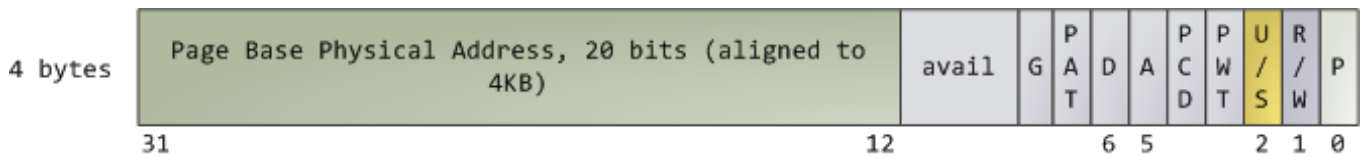


Heap overflow — Removing the chunk

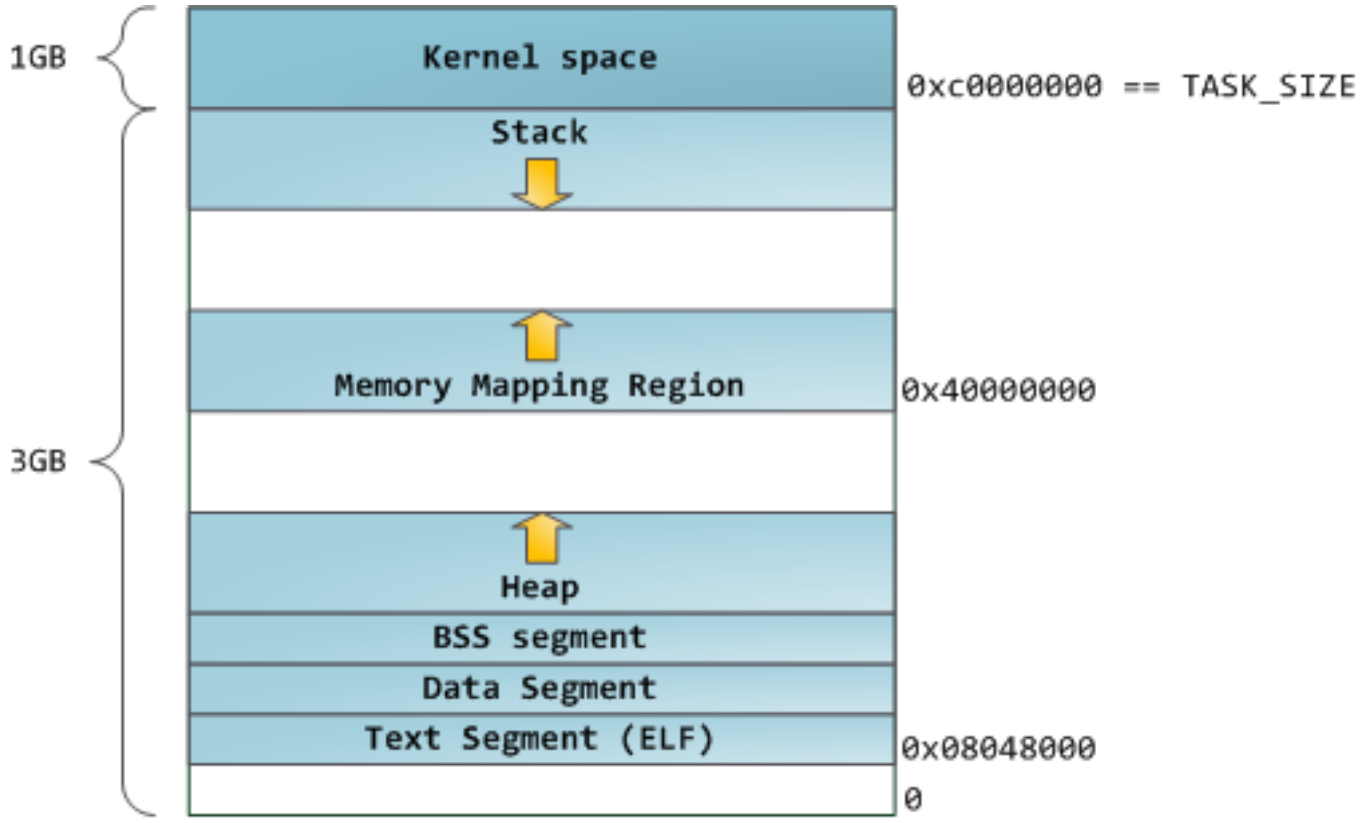



```
Larry -> BLINK -> FLINK = Larry -> FLINK
Larry -> FLINK -> BLINK = Larry -> BLINK
```

Data execution prevention



Address space randomization



SEHO protection