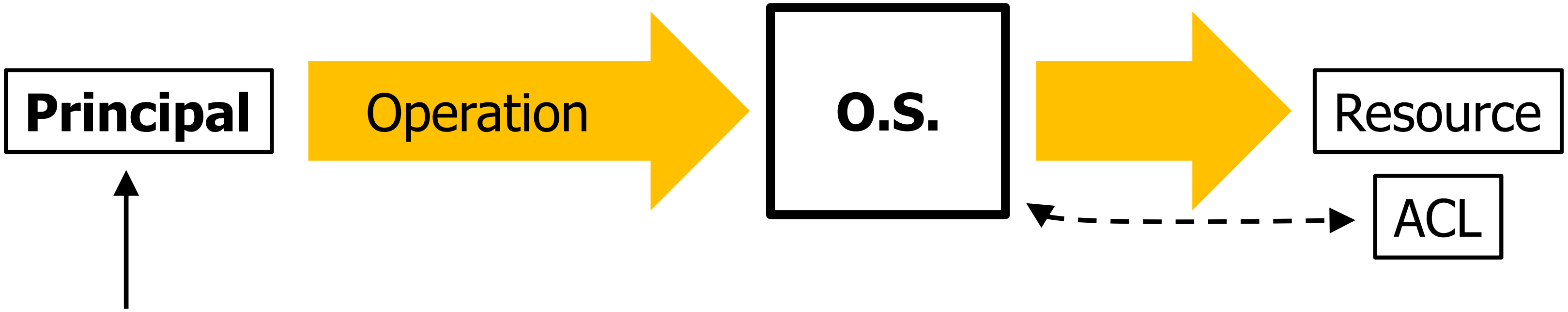
# Principle of Complete Mediation

# Access Control (REMIND)



- Account
- Which executable
- How it was authenticated
- Local / Network
- ...

The O.S. can take **different** decisions for the **same** (Account, Operation, Resource)

# Important question (IV) (REMIND)

- User U executes GUI / Shell
- How can you make sure that the GUI / Shell can **only** execute operations **allowed to U**?

- Resource access is **mediated** by the O.S.
- O.S. grants/denies based on Resource.ACL
- Resource.ACL describes what U can and cannot do

# Important question (VI) (REMIND)

- Web server
- User U logged on a **webapp** (e.g., Banking)
- How can you make sure that U can **only** access **"his/her" data**?

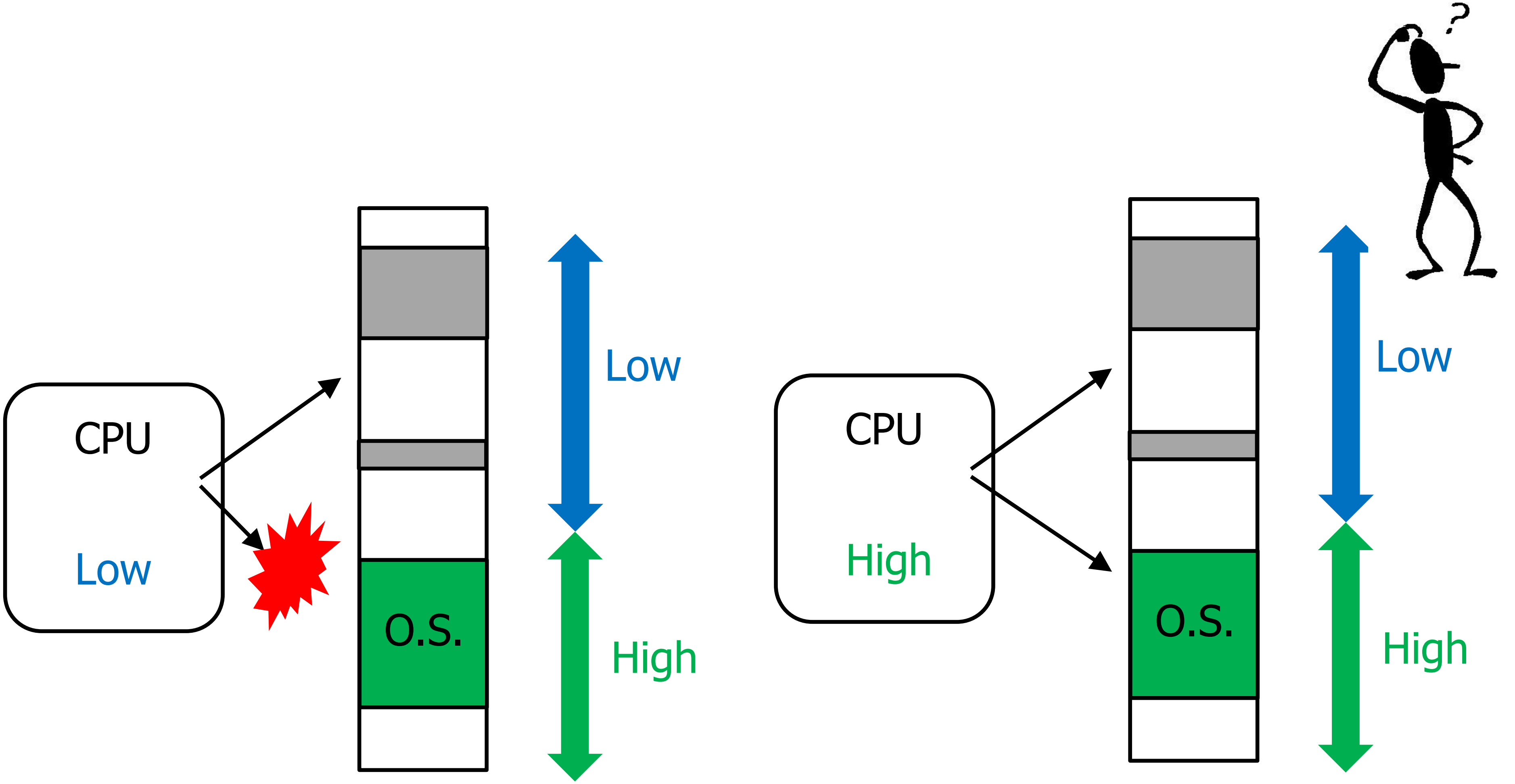
- Resource access is **mediated** by the application server
- Application server grants/denies based on Resource.ACL
- Resource.ACL describes what U can and cannot do

# Important question (V) (REMIND)

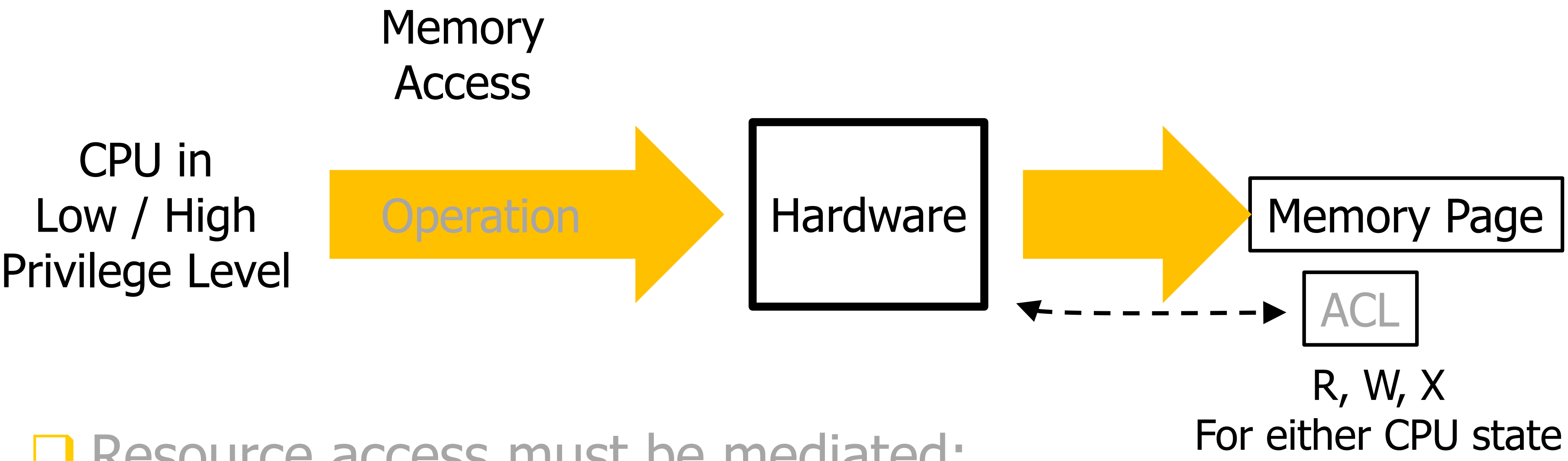
- User U executes some program P
- How can you make sure that P cannot **modify** the internal code/data **of the o.s.**?

- CPU privilege level
- Memory access rights
- ...but how can **the memory** know which CPU privilege level can access it?
- ...and how **enforced**?

# Hhmmm...



# Access Control: Hardware


Memory Access

CPU in Low / High Privilege Level

Operation

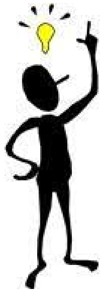
Hardware

Memory Page

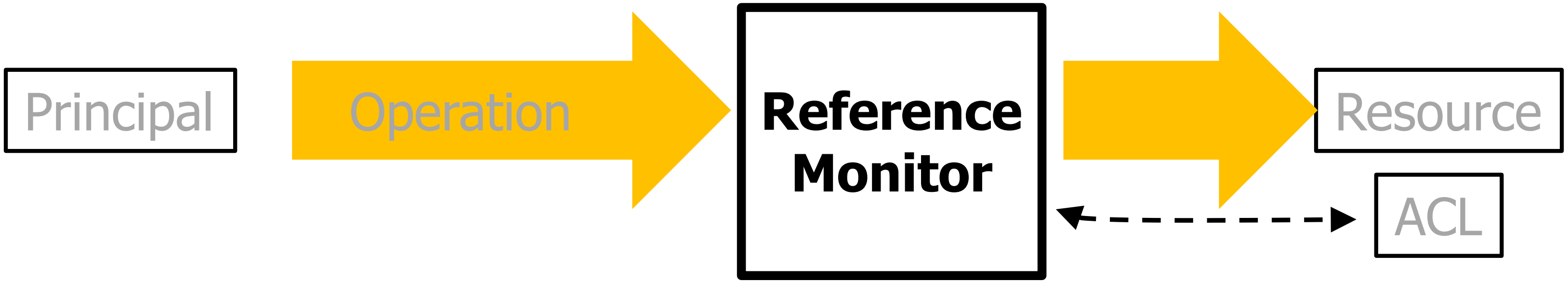
ACL

R, W, X
For either CPU state

- Resource access must be mediated:
  - Hardware level
  - Operating system level
  - Application level
- Mechanisms **independent of each other**



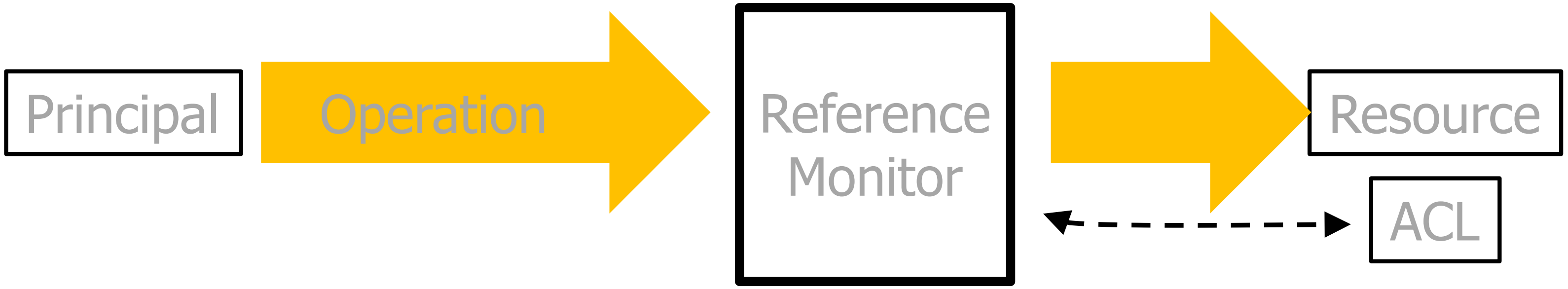
# Access Control: Abstract (=GENERAL) Model (I)



- Every access to **resources** is mediated (**guarded)** by the Reference Monitor
- Every resource has an **ACL**
- Reference Monitor decides whether to execute the operation:
  - Principal, Operation, Resource.ACL

# Access Control: Abstract (=GENERAL) Model (II)

| | | | |
|---|---|---|---|
| Username Authenticated Session | HTTP Request | Web Server | URL |
| Authenticated Username | SMTP / POP Request | Mail Server | Mailbox |
| Account | System call | O.S. | O.S. Resource |
| CPU state | Memory access | Hardware | Memory page |



Principal

Operation

Reference Monitor

Resource

ACL

# Access Control

- **FUNDAMENTAL** feature of computer systems
- **ENFORCES** the **security policy**: "who can do what"

- Occurs at **multiple** and **different** levels:
  - Application
  - Operating system
  - Hardware

- Each level:
  - Is **independent** of the other levels
  - Has **its own** mechanisms

# Saltzer and Schroeder (1974)

- **Complete mediation: Every access to every object must be checked for authority.**

- This principle, when systematically applied, **is the primary underpinning** of the protection system...
- It implies that **a foolproof method of identifying the source of every request** must be devised.

- Please take a moment to reflect and admire its depth and generality
- We will find more examples of its relevance

# Keep in mind

- Different **operational scenarios**
  - One machine
  - Many machines in a single organization
  - Many machines in many organizations
  - Web apps
  - Web apps with delegated authentication / authorization
  - Cloud services (AWS, Azure, GCP,...)
  - ...
- **Every access** to **every object**. Period.

# But this is obvious…!

❑ Complete mediation: Every access to every object must be checked for authority.

# Hhhmmm...really?

GPDP | GARANTE PER LA PROTEZIONE DEI DATI PERSONALI

**Provvedimento del 23 marzo 2023 [9883731]**

- ...anyone, after having gone through the computer authentication procedures within the portal, could view, select and open one or more documents in the ESF **of another specific assisted person**, simply **by entering the tax code of that assisted person** in the patient_id parameter.

- https://....it/fse/webapi/xds/getuserdocuments?**patient_id=XXXXXXXXXXXXXX**&date_from=YYYYYY&date_to=YYYYYYY&language=it&_=1617638439347

# A few words about Discretionary vs Mandatory

# Access Control: Specification (REMIND)

- MANY models with different **expressiveness**

- Every concrete scenario:
  - **Hybrid** of several models
  - Many complex details

  - Windows / Linux / Android / ...
  - AWS / Azure / GCP / ...
  - Tomcat / Postfix / MySQL / ...

# Security Policy Example (I)

- ***An intern** cannot have any access to files related to project A*

- Should be defined on the **ACL of each** *"file related to project A"*

- Inconvenient
- How to make sure **file owners** collaborate?

# Security Policy Example (II)

- *HR people can only modify files* ***from certain devices***

- *No account can have access right A and access right B* ***on the same resource*** *(separation of duties / two-person rule)*

- Should be defined on the **ACL of each** relevant resource

- Inconvenient
- How to make sure **resource owners** collaborate?

# Global Constraints

- ***An intern*** *cannot have any access to files related to project A*
- *HR people can only modify files* ***from certain devices***
- *No account can have access right A and access right B* ***on the same resource*** *(separation of duties / two-person rule)*

- Security policy requirements: **Global** constraints specified at a **central** level
- Access control mechanisms: specified at a **local** level (each single resource)

- Not a good fit

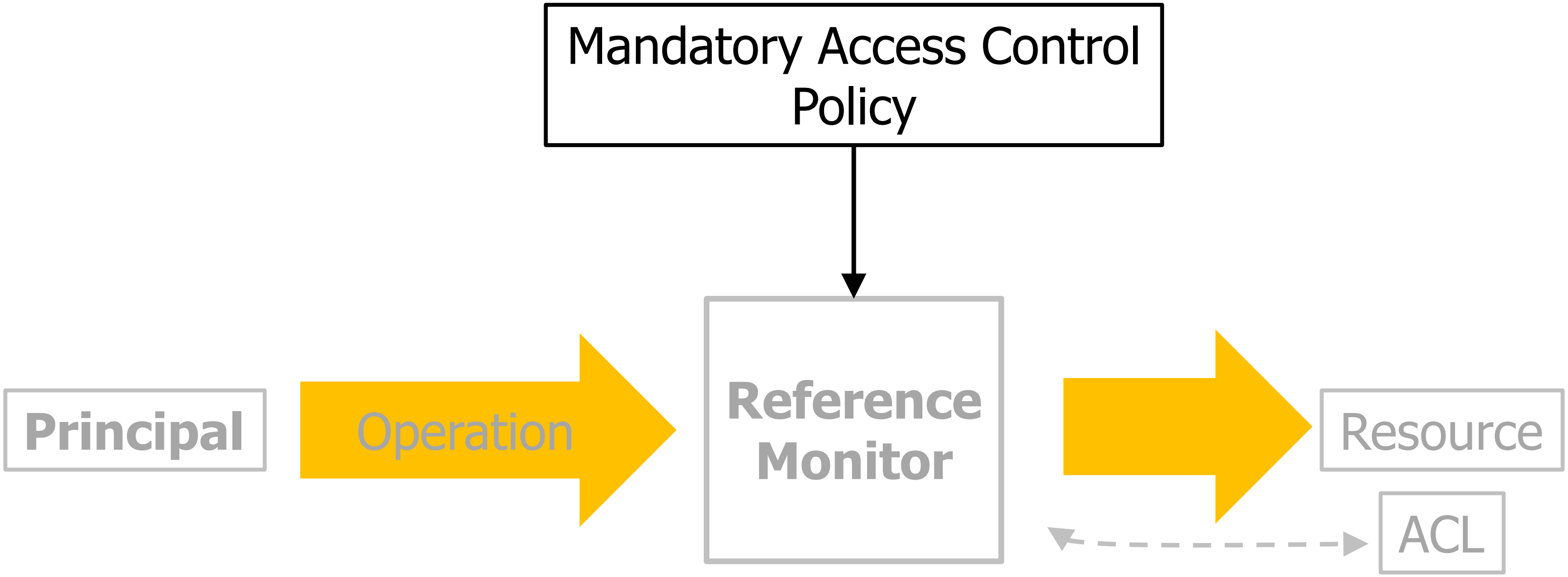
# Discretionary vs Mandatory

- **Discretionary** Access Control (DAC)
  - Resource owners manage resource ACLs
  - What we have seen so far

- **Mandatory** Access Control (MAC)
  - Allow **defining** and **enforcing global** requirements
  - Take precedence over DAC

- Real o.s. and cloud services support a mix of them
- Out of scope

# Access Control



Mandatory Access Control Policy

Principal

Operation

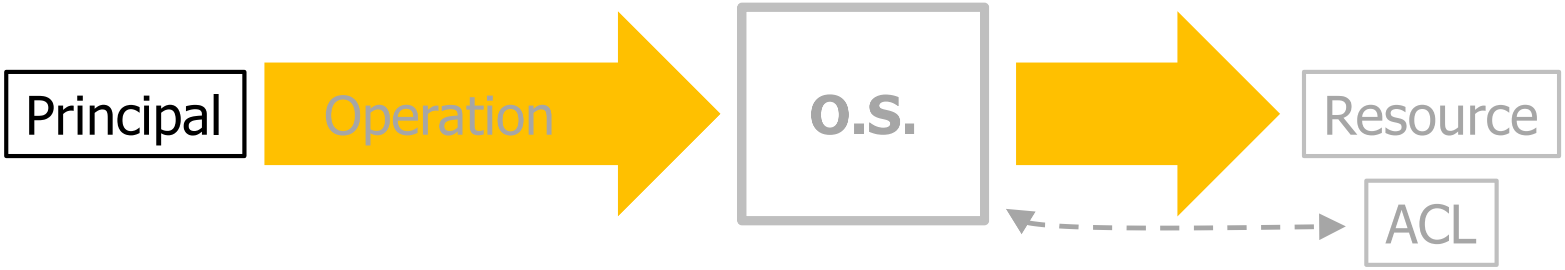
Reference Monitor

Resource

ACL

# Understanding Access Control in Cybersecurity

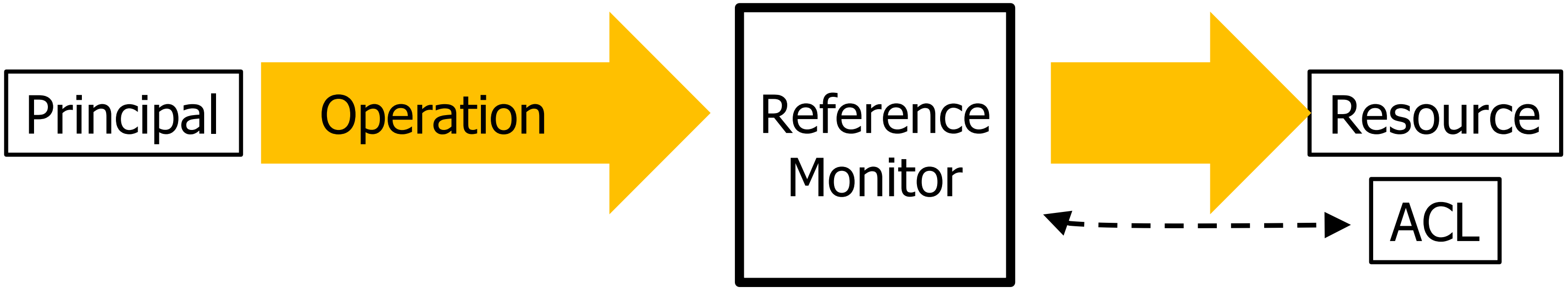
# Access Control = Authorization (≠ Authentication)



- Principal is an **input** data (it is "certain"):
  it is determined **prior** to issuing the OpRequest

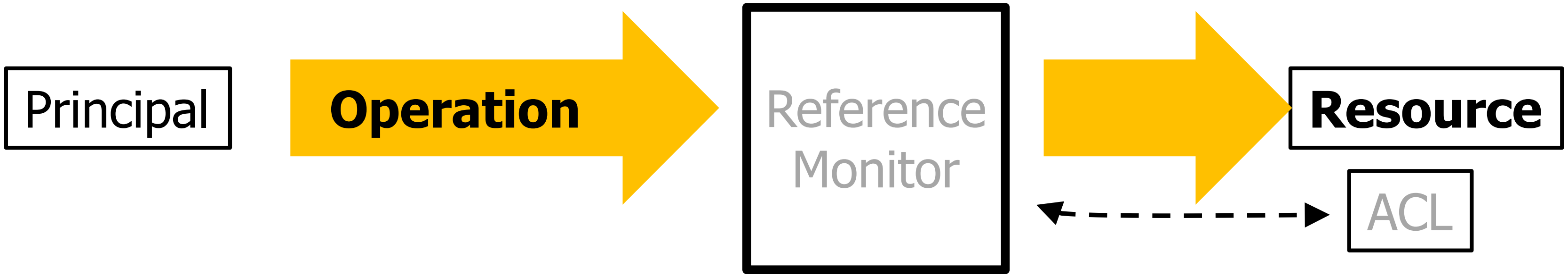
- How it is determined is a **different** problem
  - **Authentication** is usually required

# Everything is perfect (I)



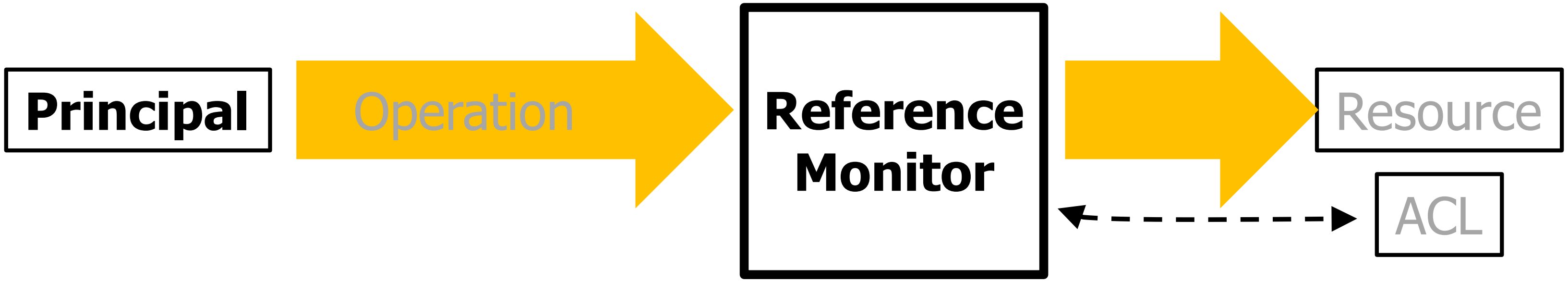
- The security policy described in the ACLs of all resources is the **intended** one:
  - No principal is allowed to do what it should **not** be allowed to do.

# Everything is perfect (II)



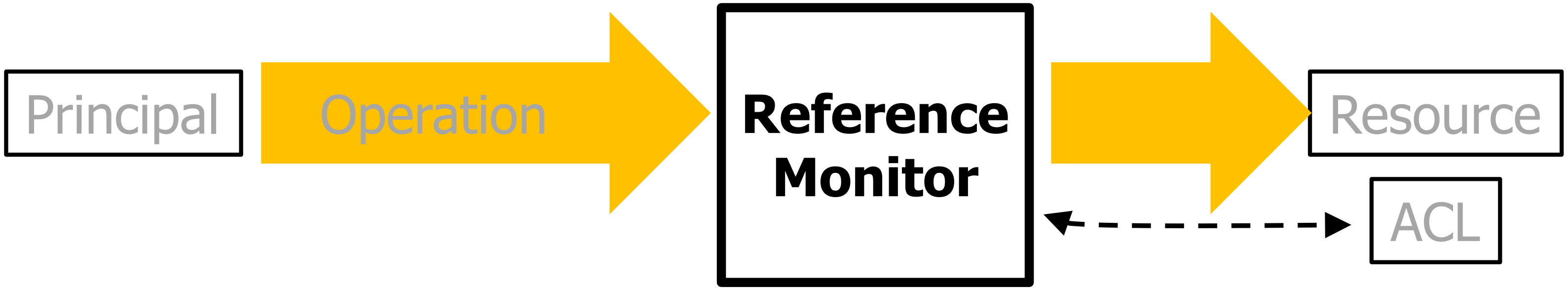
- Principals do not **abuse** their access rights to do **bad** things (they always behave as we expect them to):
  - You do not expect your browser to steal your data (e.g., read it and send it somewhere)

# Everything is perfect (III)



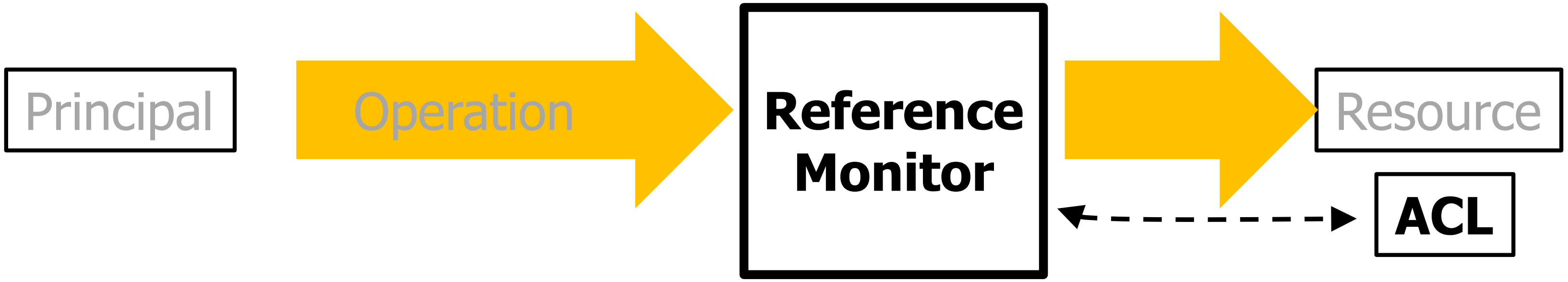
- A given Principal cannot appear to the Reference Monitor as a **different** Principal

# Everything is perfect (IV)



- Reference Monitor:
  - No way of **bypassing** it
  - No **mistakes**

# Everything is perfect (V)



- Principals are **not** able to **modify**:
  - Reference Monitor
  - ACLs
    (unless through authorized operations)

# Why Cybersecurity is an issue? (I)

- IF Everything was perfect
- THEN Cybersecurity would **not** be an issue

- The problem is that something is not perfect
  - Often **a lot** of things

# Example (I)

- "Midnight Blizzard attack to Microsoft" on the companion website:
  - Test application → Senior leadership Cybersec people email and docs

- **Actual** Security policy **different** from the intended one
- Principals **abused** their access rights

# Example (II)

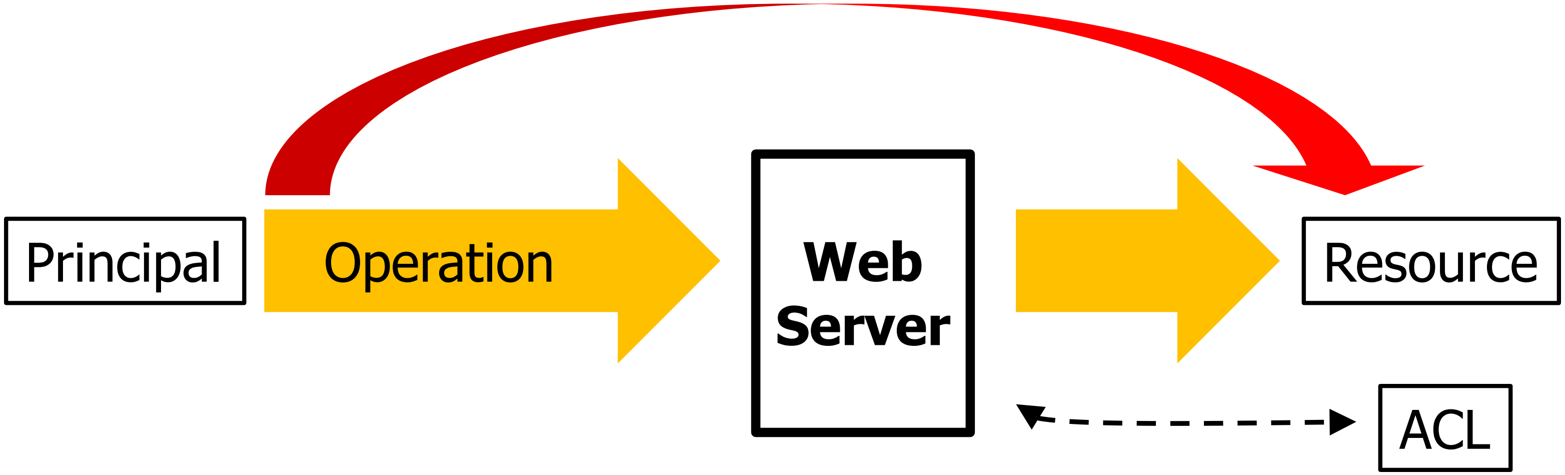
- Incident at a company in Trieste (27K ransom paid)
  - Secretary receives pdf invoice with malware from (unsuspecting) commercial partner
  - Malware encrypts all files in all folders of the company filesystem

- **Actual** Security policy **different** from the intended **intended** one
- Principals **abused** their access rights

# Example (III)

GPDP | GARANTE PER LA PROTEZIONE DEI DATI PERSONALI

Provvedimento del 23 marzo 2023 [9883731]



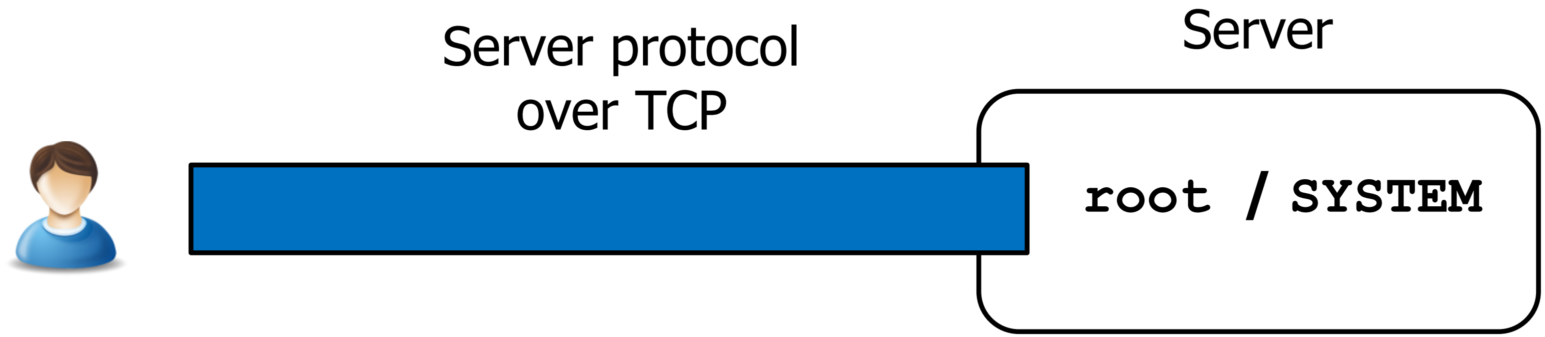
- Reference Monitor:
  - ~~No way of **bypassing** it~~

# Keep in mind

- Cybersecurity is mostly about **mistakes**

# Principle of Least Privilege

# Common Server Config. (up to a few years ago)



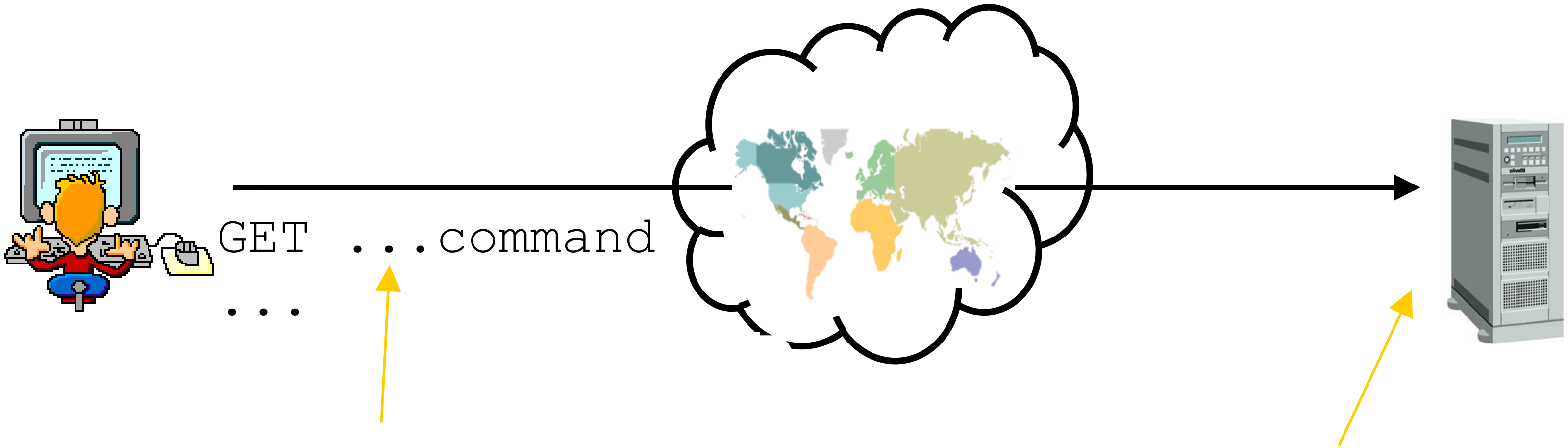
Server protocol over TCP

Server

`root / SYSTEM`

Remote Shell
Web Server
File Server
Mail Server
...

# Example (Old but interesting) (I)

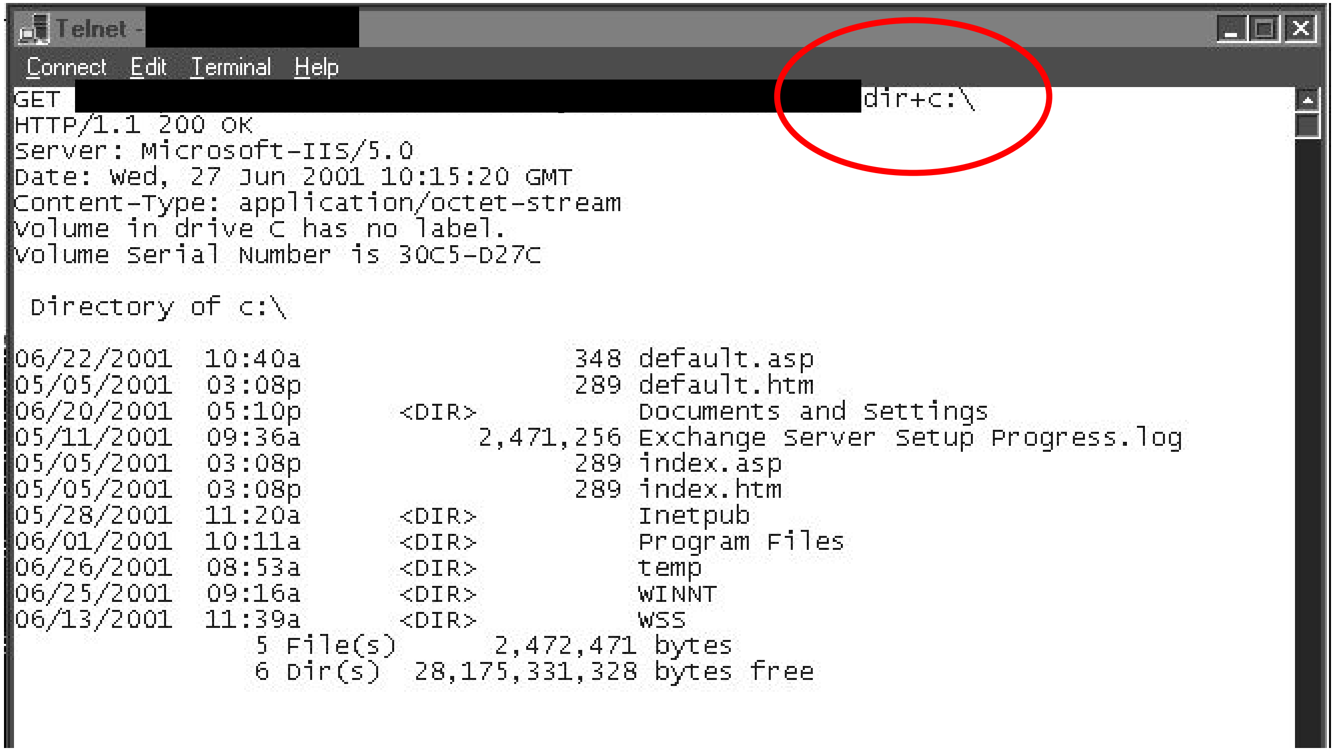


```
GET ...command
...
```

HTTP Request
with "long and wrong URL"
ending with command

Execute command

# Example
# (Old but interesting) (II)

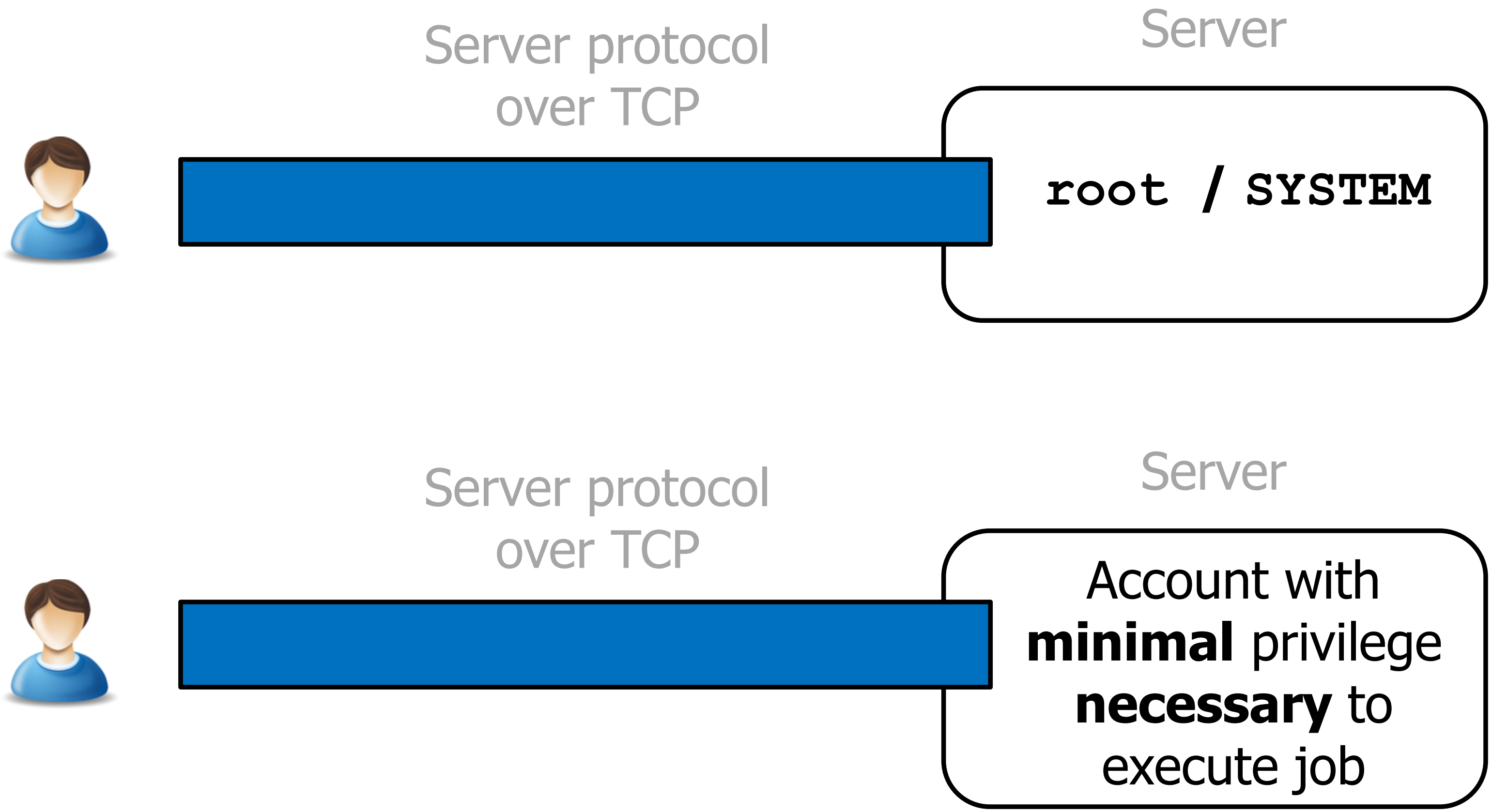


```
GET [redacted] dir+c:\
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 27 Jun 2001 10:15:20 GMT
Content-Type: application/octet-stream
Volume in drive C has no label.
Volume Serial Number is 30C5-D27C

 Directory of c:\

06/22/2001  10:40a                 348 default.asp
05/05/2001  03:08p                 289 default.htm
06/20/2001  05:10p      <DIR>          Documents and Settings
05/11/2001  09:36a           2,471,256 Exchange Server Setup Progress.log
05/05/2001  03:08p                 289 index.asp
05/05/2001  03:08p                 289 index.htm
05/28/2001  11:20a      <DIR>          Inetpub
06/01/2001  10:11a      <DIR>          Program Files
06/26/2001  08:53a      <DIR>          temp
06/25/2001  09:16a      <DIR>          WINNT
06/13/2001  11:39a      <DIR>          WSS
               5 File(s)      2,472,471 bytes
               6 Dir(s)  28,175,331,328 bytes free
```

# Which approach is wiser?



Server protocol over TCP

Server

`root / SYSTEM`

Server protocol over TCP

Server

Account with **minimal** privilege **necessary** to execute job

# Principle of Least Privilege

- **Every** program and every user of the system should operate using the **least** set of privileges **necessary** to complete the job…
- It also reduces the number of potential interactions among privileged programs to **the minimum for correct operation**, so that **unintentional**, **unwanted**, or **improper** uses of privilege are **less likely** to occur…

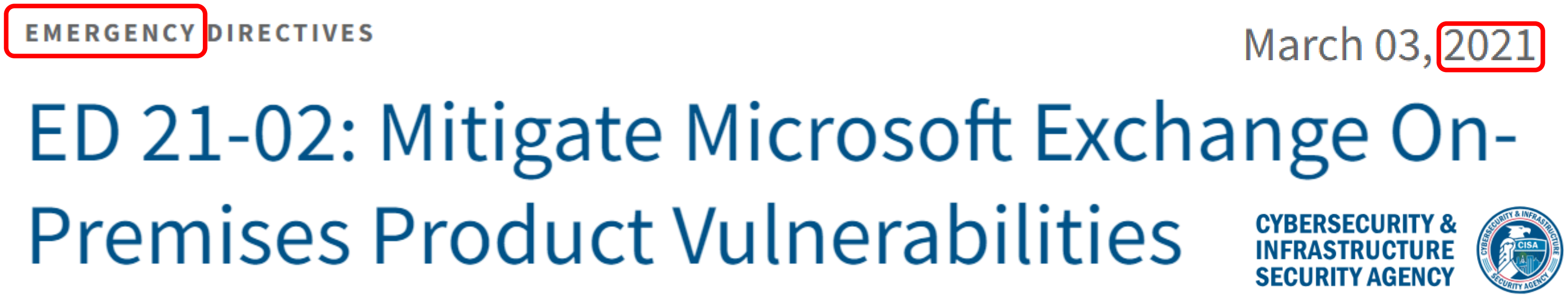
- *Saltzer and Schroeder* ***1974 (!)***

- Please take a moment to reflect and admire its depth and generality
- We will find more examples of its relevance

# But this is obvious…!

- **Least privilege: Every** program and every user of the system should operate using the **least** set of privileges **necessary** to complete the job

# Hhhmmm...



EMERGENCY DIRECTIVES

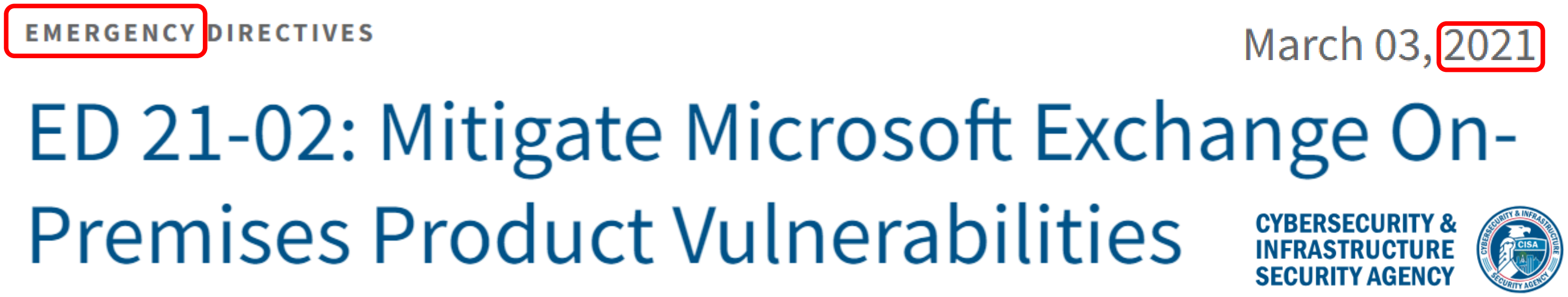
March 03, 2021

ED 21-02: Mitigate Microsoft Exchange On-Premises Product Vulnerabilities

- Mail Server used by **a myriad of organizations**
- **Necessarily exposed to the Internet**

- An **unauthenticated** attacker can **execute arbitrary commands** on Microsoft Exchange Server ("ProxyLogon")

# Hhhmmm...really?



EMERGENCY DIRECTIVES

March 03, 2021

ED 21-02: Mitigate Microsoft Exchange On-Premises Product Vulnerabilities

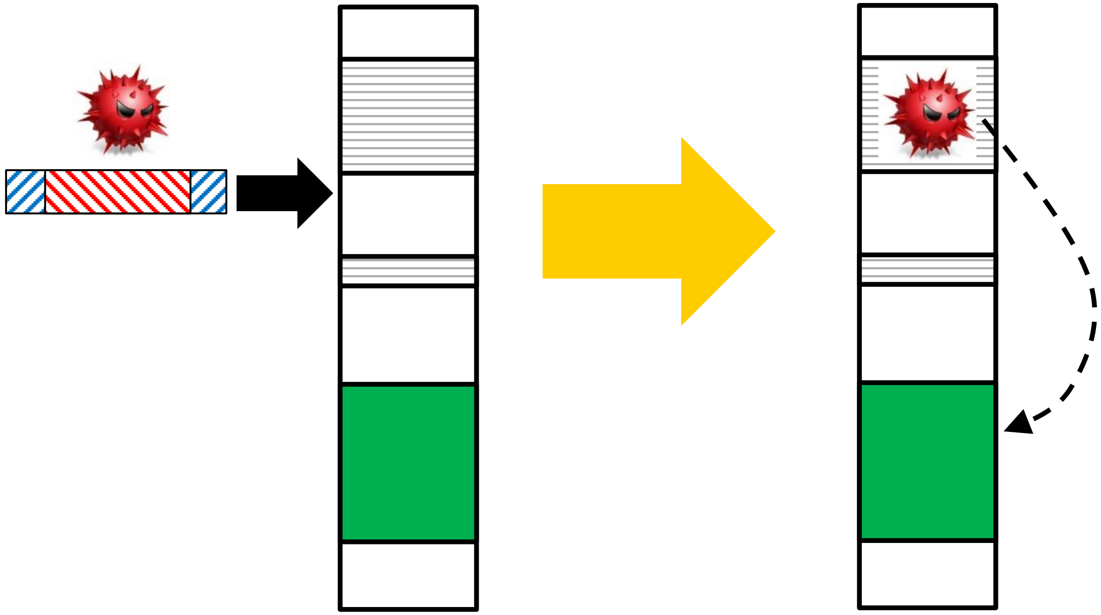
CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

- Mail Server used by **a myriad of organizations**
- **Necessarily exposed to the Internet**

- An **unauthenticated** attacker can **execute arbitrary commands** on Microsoft Exchange Server ("ProxyLogon")

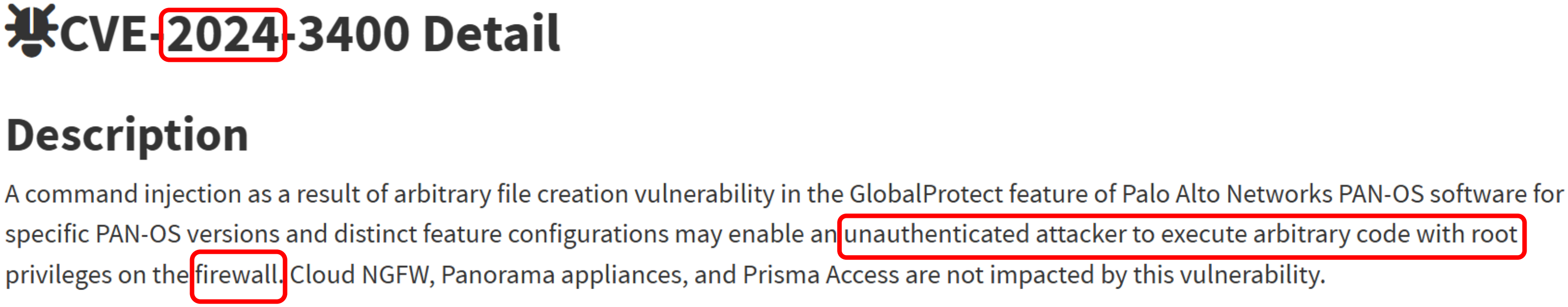
- "Exchange is, **by default**, installed with **some of the most powerful privileges** in Active Directory" (`SYSTEM`)

# Remark: RCE vulnerability



Malware executes actions with
the **identity of the vulnerable process**

# Hhhmmm...REALLY?



## CVE-2024-3400 Detail

### Description

A command injection as a result of arbitrary file creation vulnerability in the GlobalProtect feature of Palo Alto Networks PAN-OS software for specific PAN-OS versions and distinct feature configurations may enable an unauthenticated attacker to execute arbitrary code with root privileges on the firewall. Cloud NGFW, Panorama appliances, and Prisma Access are not impacted by this vulnerability.

# Cybersecurity & Economics

# Hhmmm…

- *Principle of Least Privilege:* ***1974***
- *Why in many practical scenarios it is still* ***not*** *enforced,* ***50 years later?***

# Security is NEVER the ONLY objective (I)

- **Every** choice must be a tradeoff among:
  1. Security
  2. Cost
  3. Functionality
- Design, Development, Deployment, Usage, Maintenance

- In many practical cases, Security is sacrificed

# Security is NEVER the ONLY objective (II)

- ❑ In many practical cases, Security is sacrificed

- ❑ The chosen tradeoff might be wrong (perhaps retrospectively)

- ❑ ...but it often is **economically rational**
  - ❑ More Security ⇒ More short term costs
  - ❑ Long term savings uncertain
  - ❑ Market forces could penalize short term costs

# Think in Economical Terms

- To understand cybersecurity **never** think only in **technical** terms
    - Or, worse, in "moral" terms
- **Always** think in **economical** terms

- What is the cost?
    - Attack, Defense, Incident
- Who pays?

- **Money is what drives the world**
    - It may sound cynical...but thinking in these terms is very helpful

# Keep in mind: Much easier said than done

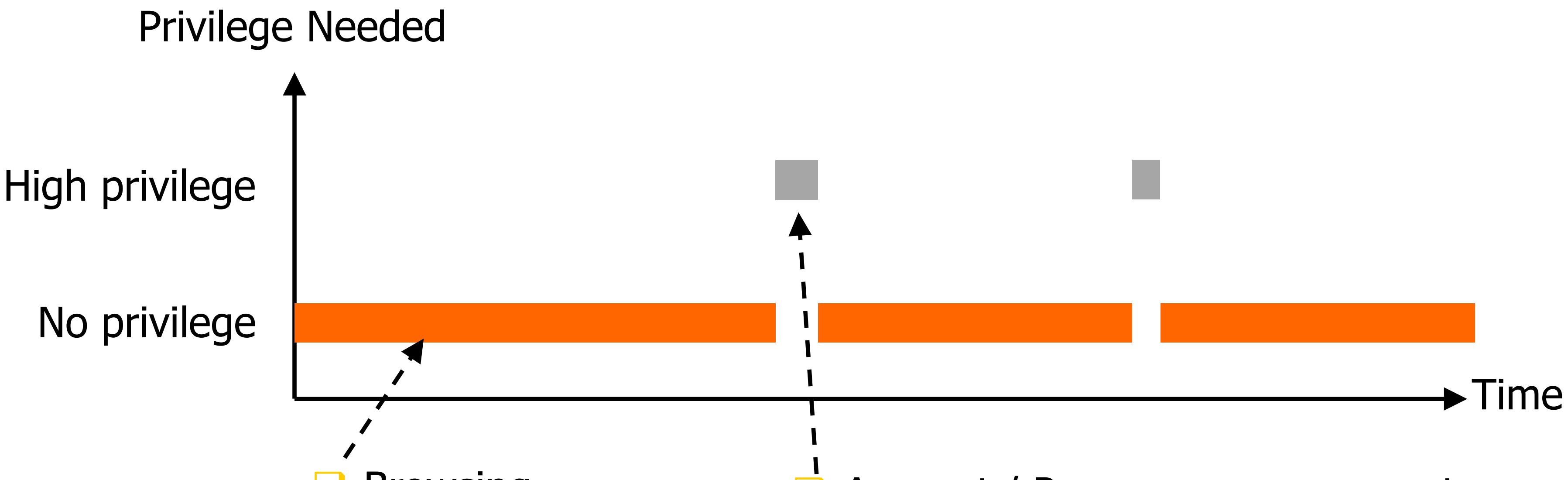
- **Every** program ... should operate using the **least** set of privileges **necessary** to complete the job...

- I have written a Python script to automatically generate random sets of exam questions

- Why should this script have the privilege to **write to any directory** that I can write to?

# Temporary Privilege Elevation

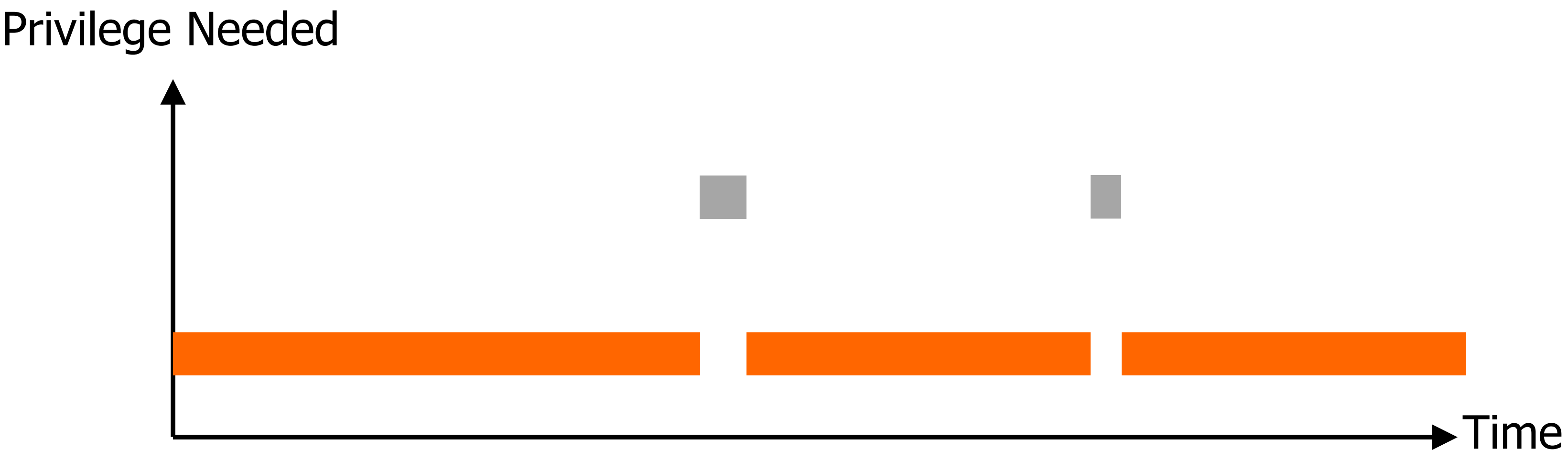
# Key practical scenario



- Browsing
- Email
- Programming
- ...

- Account / Resource management
- Program installation / removal
- Configuration management
- ...

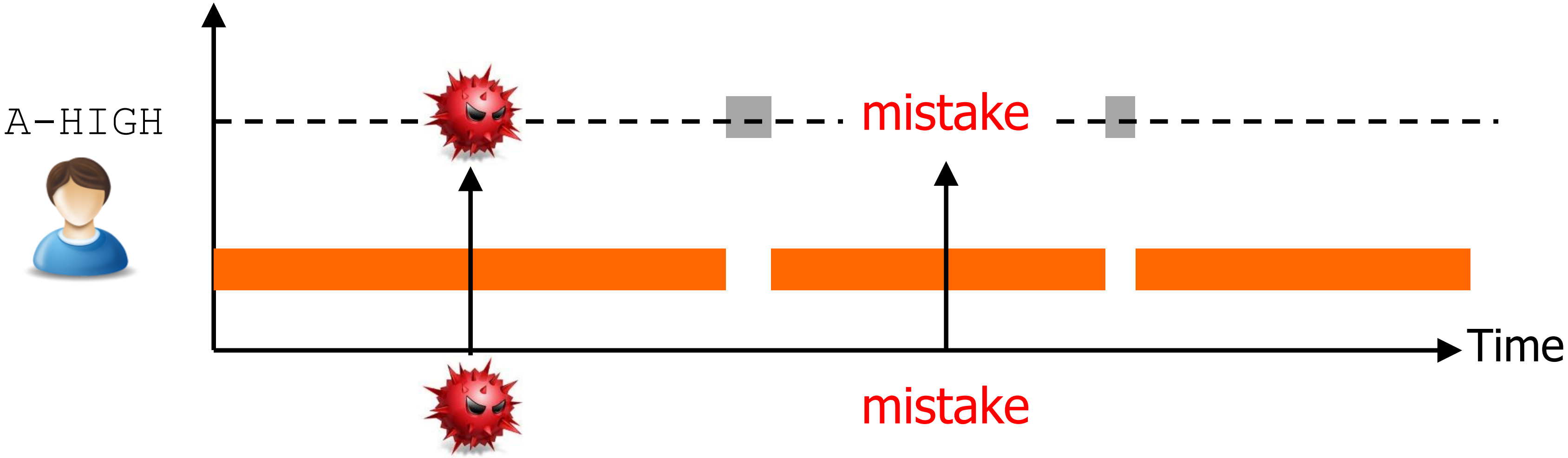
# Which account?



- ❑ Two accounts available: `A-LOW`, `A-HIGH`
- ❑ `A-LOW` is not enough: sometimes `A-HIGH` is needed

- ❑ Is it wise to **always** use `A-HIGH`?

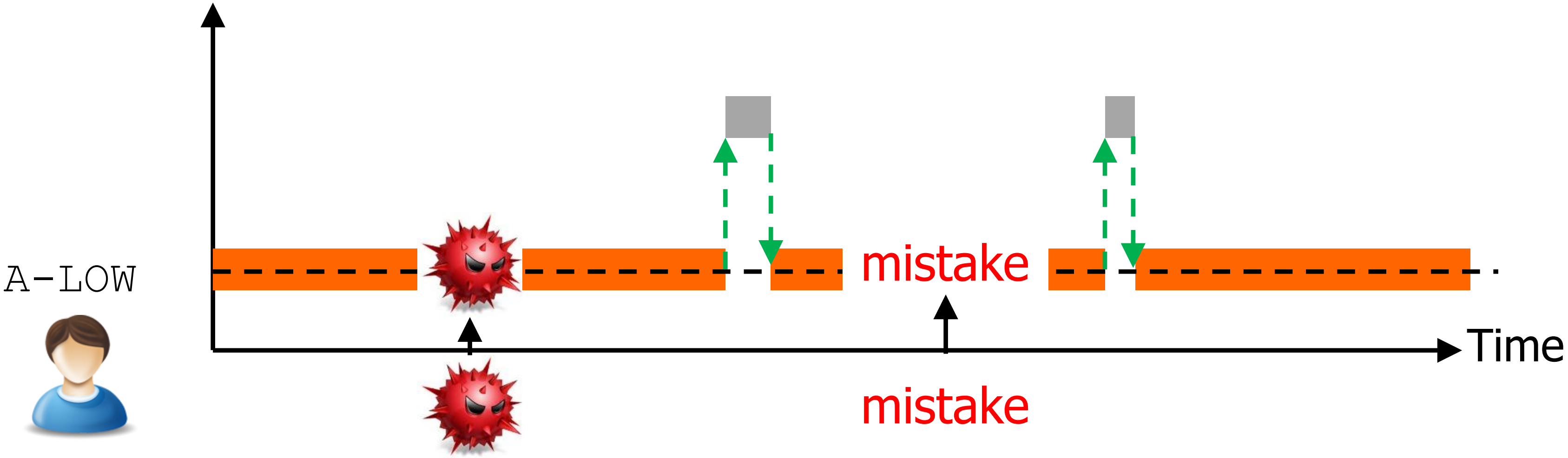
# No, no, no!



- **Unwanted** or **Unintentional** actions
- ...with identity `A-HIGH`

- Huge violation of principle of least privilege

# What we need



- ❑ Unwanted or Unintentional actions
- ❑ ...with identity `A-LOW`

+

- ❑ **Temporary privilege elevation**

# Basic idea

- Executable F:
  - Created by an account `A-H` with high privileges
  - Marked with "**elevation**"

- Process P owned by an account **different** from `A-H` :
  1. Creates P1 that executes F
  2. P1 has "elevated privileges"

- Rationale:
  - `A-H` has encoded certain actions in F
  - `A-H` trusts that F can be executed safely with high privileges

# Fundamental Risk

- Rationale:
  - `A-H` has encoded certain actions in F
  - `A-H` trusts that F can be executed safely with high privileges

- F might not behave as intended
  - Mistakes (vulnerabilities)

# Temporary Privilege Elevation: Linux

# Linux `suid`

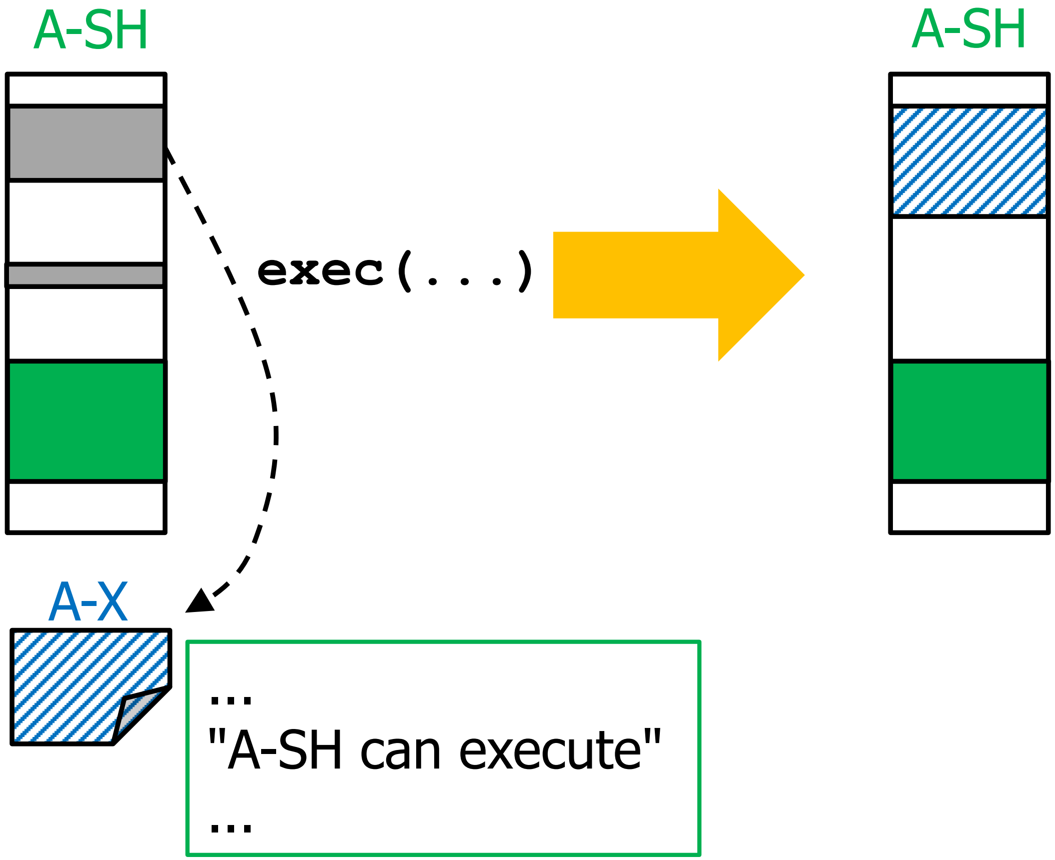
- Executable F:
  - Owned by `A-F`
  - With **set user id** attribute (`suid`)

- Process P owned by an account **different** from `A-F` :
  1. Creates P1 that executes F
  2. P1 owned by `A-F`

- Fully automatic: `A-F` credentials **not** needed

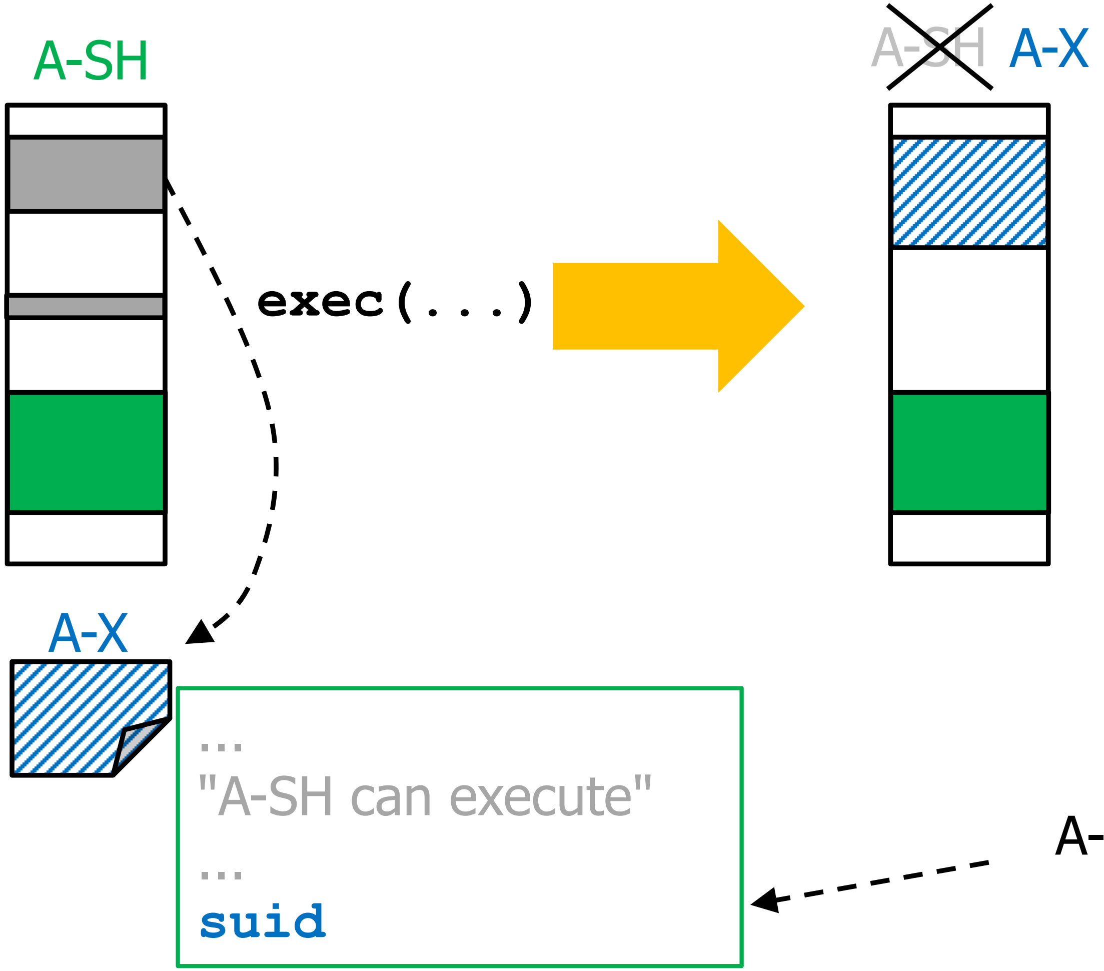
- `A-F=root` ⇒ privilege elevation

# Linux `exec` (file WITHOUT `suid`)



…
"A-SH can execute"
…

# Linux `exec` (file WITH `suid`)

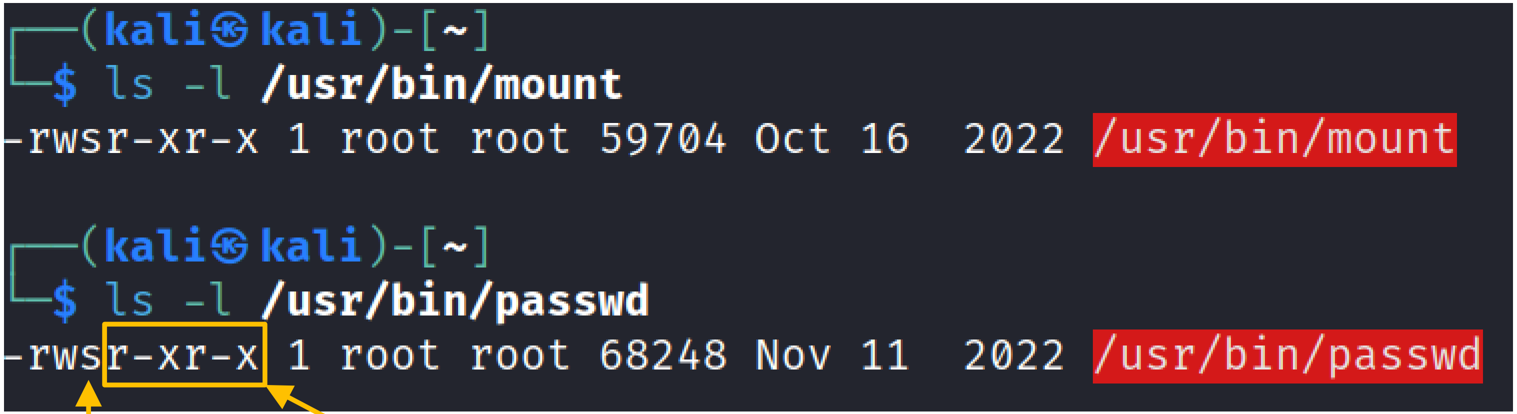


...
"A-SH can execute"
...
**suid**

A-X allows executing this file
with its own identity

# Common Use Case

- `A-X` is **high privilege** (`root`)
  - Impersonation = Elevation



```
┌──(kali㉿kali)-[~]
└─$ ls -l /usr/bin/mount
-rwsr-xr-x 1 root root 59704 Oct 16  2022 /usr/bin/mount

┌──(kali㉿kali)-[~]
└─$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 68248 Nov 11  2022 /usr/bin/passwd
```

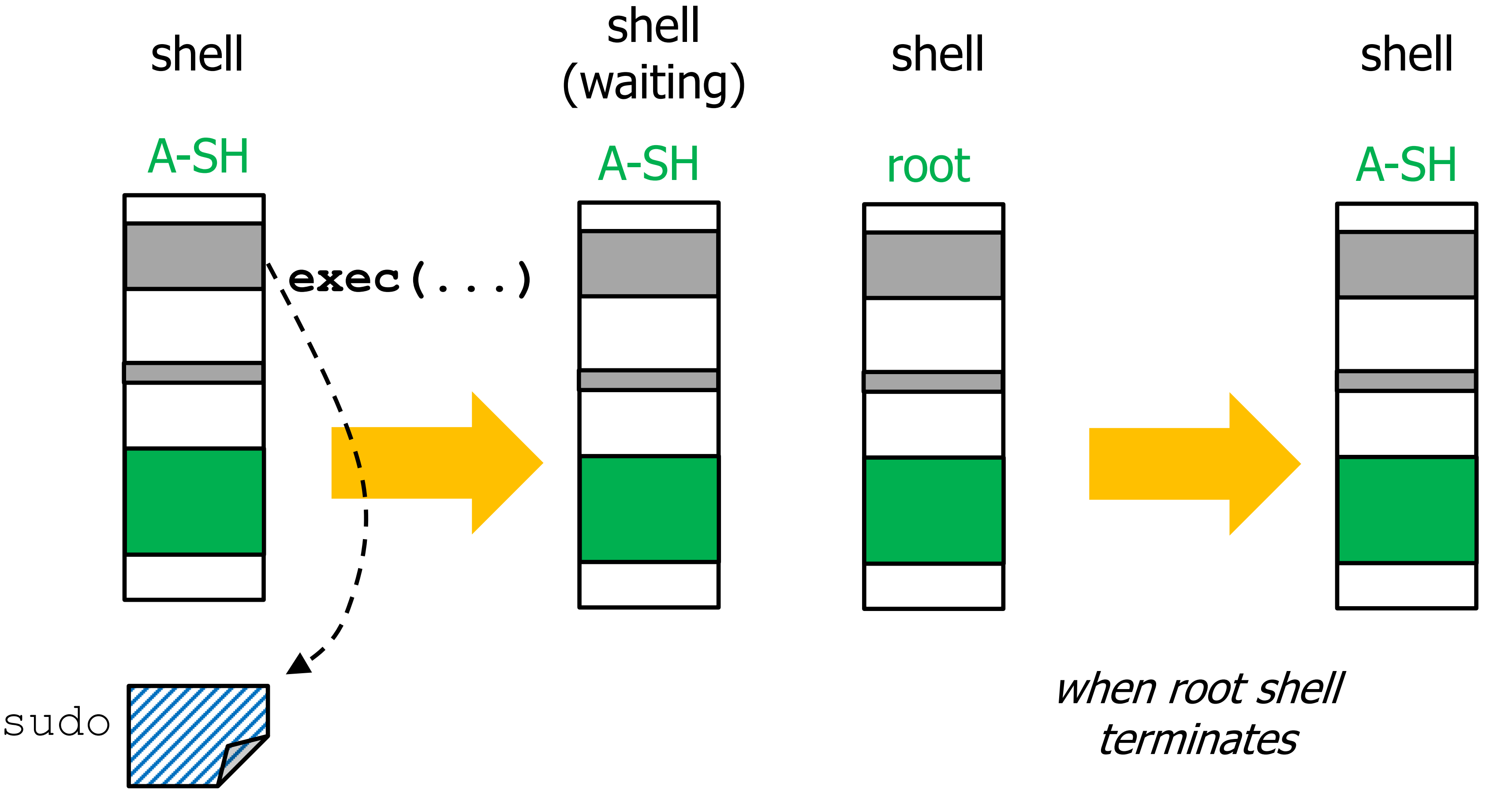
Executable file **with `suid`**

Can be read and executed (but **not modified**) by any account

# Linux `sudo` (I)

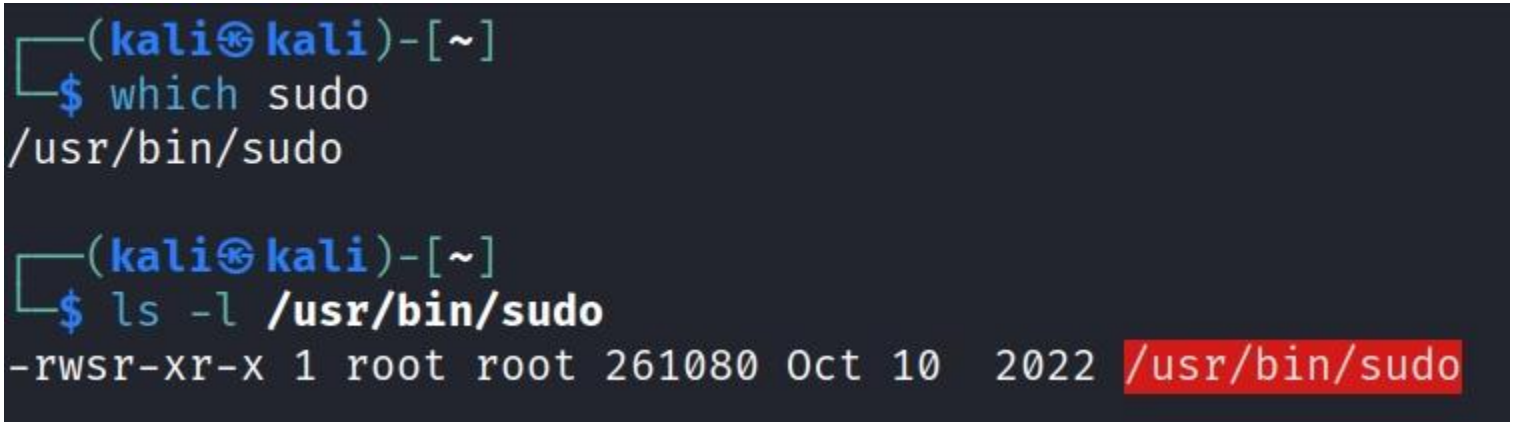
- Executable file `sudo`
- Behavior depends on **invocation arguments** and **configuration**
- Common invocation: No arguments
  - Spawns a **shell** owned by `root`
  - Password of the **invoking account** `A-SH` **required**
- Configuration: `A-SH` must belong to **`sudoers`** group
  - Normal users: **not** inserted in `sudoers`
  - Administrators: inserted in `sudoers`

# Linux `sudo` (II)



*when root shell terminates*

# How sudo works (outline) (I)

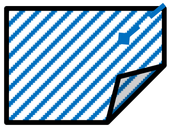


```
┌──(kali㉿kali)-[~]
└─$ which sudo
/usr/bin/sudo

┌──(kali㉿kali)-[~]
└─$ ls -l /usr/bin/sudo
-rwsr-xr-x 1 root root 261080 Oct 10  2022 /usr/bin/sudo
```

# How `sudo` works (outline) (II)

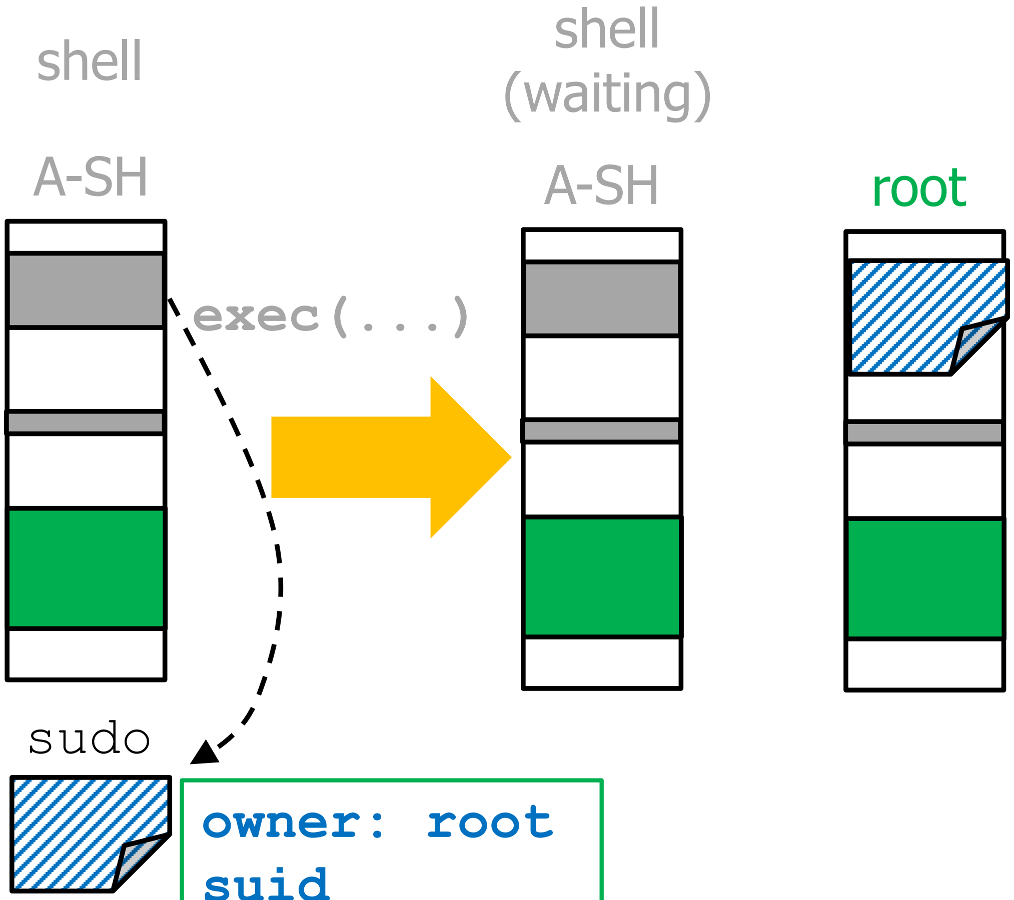
1. Verify if process account is in `sudoers`
2. Ask account credentials
3. Verify credentials
4. exec("`/bin/sh`")



**sudo**

```
owner: root
suid
...
```

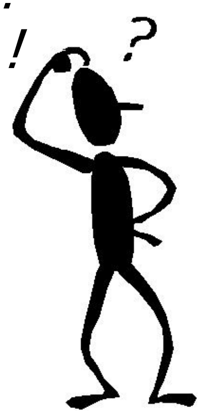
# How `sudo` works (outline) (III)



```
owner: root
suid
...
```

1. Verify if process account is in `sudoers`
2. ...

*How???*
*Account is `root`!*



# Linux `uid` (in a nutshell)

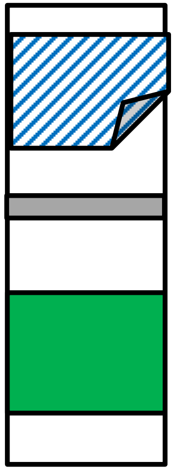
- Each process is associated with **two** account identifiers (`uid`)
  - Effective `uid` Access control
  - Real `uid` Account that created the process

- Always identical...**except** when created from `suid` file
- Example: `sudo`
  - Effective `uid` `root`
  - Real `uid` Account that invoked `sudo`

# How `sudo` works (outline) (IV-a)

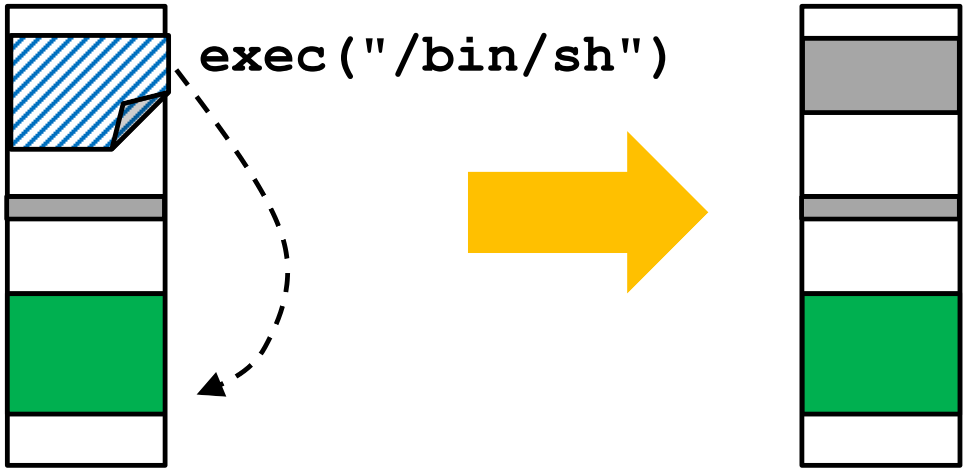
real UID = `A-SH`
effective UID = `root`

1. Verify if **real UID** is in `sudoers`
2. …



# How `sudo` works (outline) (IV-b)

1. Verify if real UID is in `sudoers`
2. Ask account credentials
3. Verify credentials
4. exec("`/bin/sh`")



shell

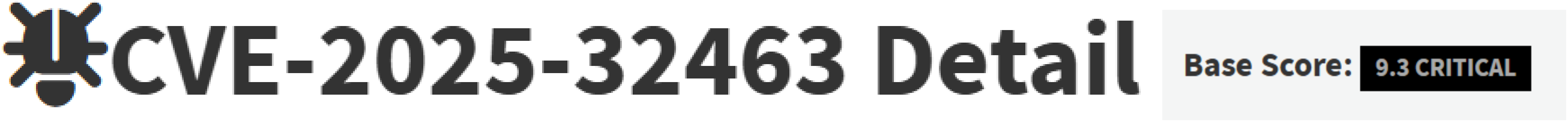
real UID = `A-SH`
effective UID = `root`

real UID = `A-SH`
effective UID = `root`

**`exec("/bin/sh")`**

# Keep in mind

- F might not behave as intended
  - Mistakes (vulnerabilities)



CVE-2025-32463 Detail

Base Score: 9.3 CRITICAL

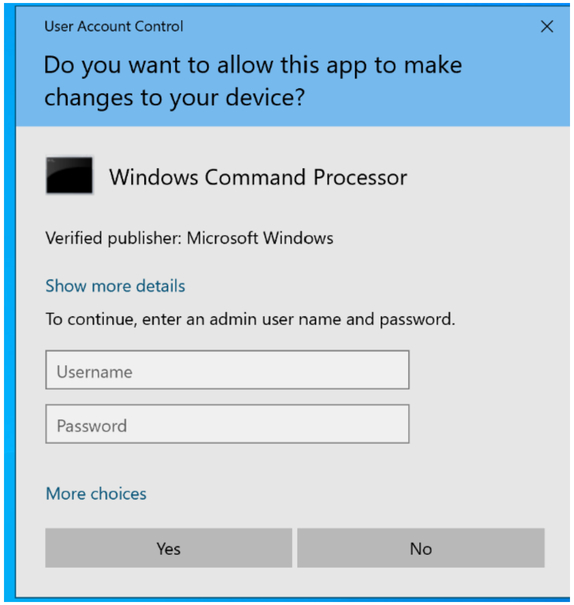
A flaw was found in `sudo`. A local attacker may trick `sudo` into executing arbitrary commands as `root` even if not included in `sudoers`.

# Temporary Privilege Elevation: Windows

# Windows `sudo`

- Very recent addition
- Must be enabled

- `sudo` *`command`*
  - Only for `Administrator`
  - Password **required**



User Account Control

Do you want to allow this app to make changes to your device?

Windows Command Processor

Verified publisher: Microsoft Windows

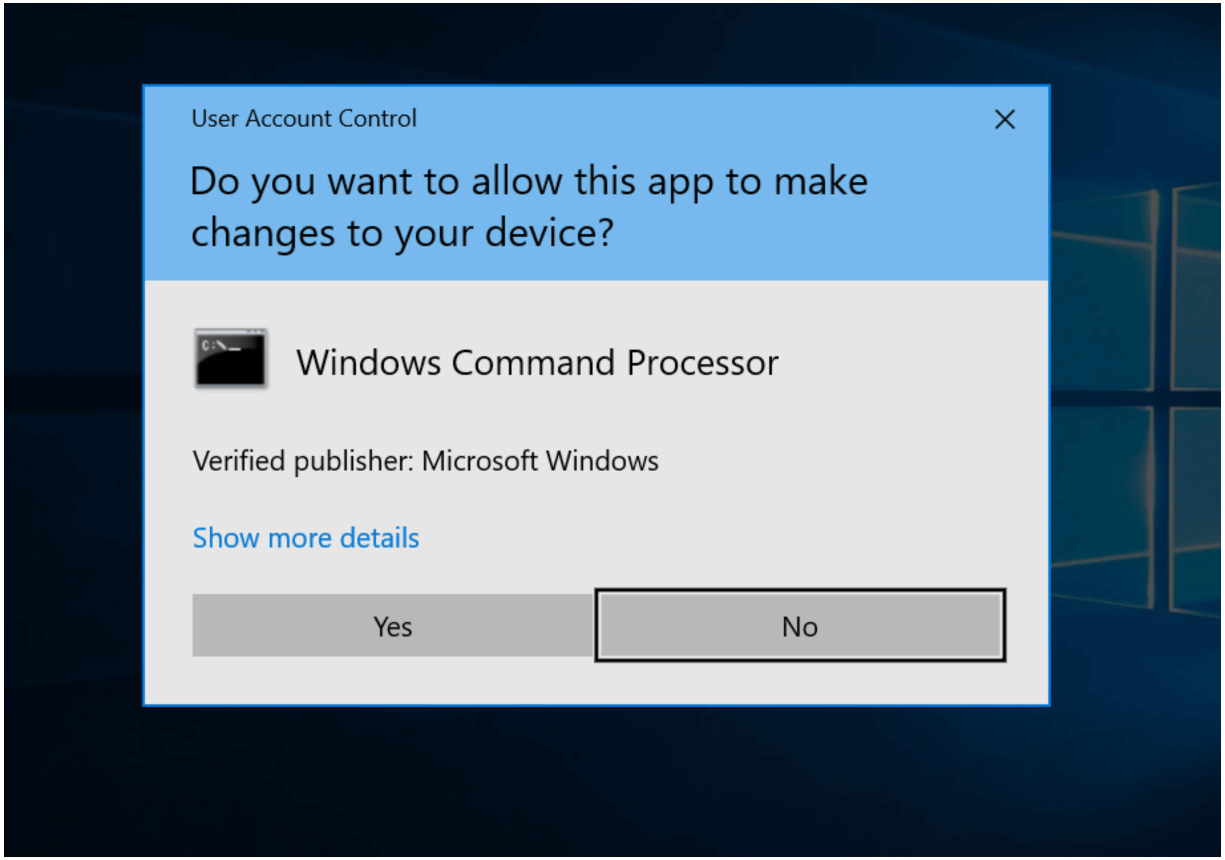
Show more details

To continue, enter an admin user name and password.

Username

Password

# UAC:
# User Account Control (I)



User Account Control

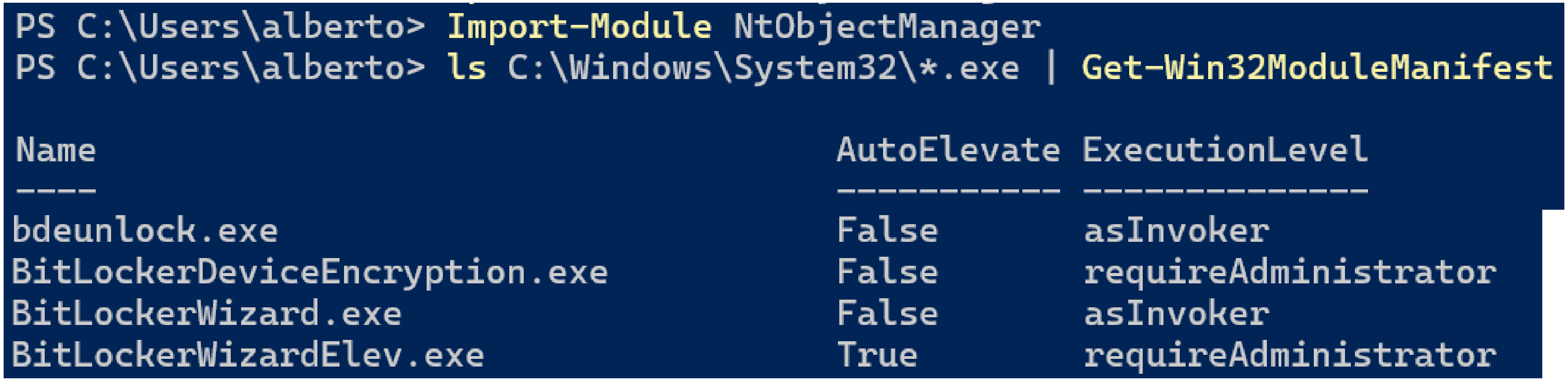
Do you want to allow this app to make changes to your device?

Windows Command Processor

Verified publisher: Microsoft Windows

# UAC: User Account Control (II)

- Executable F:
  - Owned by `Administrator`
  - Marked with `AsInvoker` / **`RequireAdmin`**
- When **`RequireAdmin`**:
  - `AutoElevate` True: No explicit consent, no credentials
  - `AutoElevate` False: Explicit consent, credentials

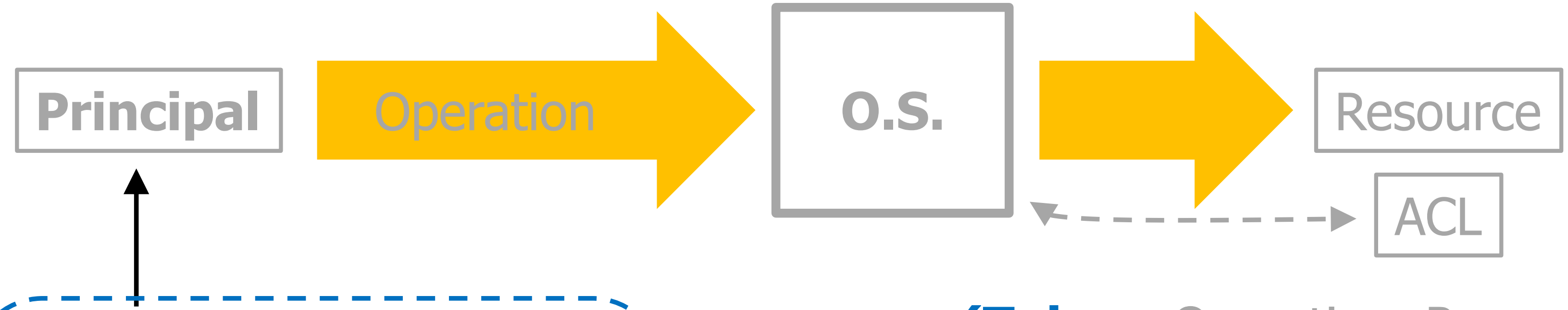


```
PS C:\Users\alberto> Import-Module NtObjectManager
PS C:\Users\alberto> ls C:\Windows\System32\*.exe | Get-Win32ModuleManifest

Name                                        AutoElevate ExecutionLevel
----                                        ----------- --------------
bdeunlock.exe                               False       asInvoker
BitLockerDeviceEncryption.exe               False       requireAdministrator
BitLockerWizard.exe                         False       asInvoker
BitLockerWizardElev.exe                     True        requireAdministrator
```

# UAC Implementation Outline (I)

Temporary Privilege Elevation does **not** change **account**



**Token**

- ❑ Account
- ❑ Which executable
- ❑ How it was authenticated
- ❑ Local / Network
- ❑ …

**(Token,** Operation, Resource)

The O.S. can take **different** decisions
for the **same**
(Account, Operation, Resource)

# UAC Implementation Outline (II)

- **GUI / shell** owned by Administrator

- Every process has **two** tokens
  - **Limited** Administrator groups and most privileges **removed**
  - **Full** All groups and privileges

- Every process uses the **limited** token
- A process uses the **full** token **only** when executing files with `requireAdministrator`

# Temporary
# Privilege Elevation Summary

- Linux
  - Change to **account** with more privileges
- Windows
  - Change **privileges** of the account

- Elevation triggered by an executable
  - It must be checked very carefully

- Windows
  - Autoelevation (no user consent / no credentials) **only** on Microsoft-approved executables