# HBO

## *Windows Security Introduction*

Gwendal Patat
Univ Rennes, CNRS, IRISA
2025/2026

# On today’s schedule

**Windows Security Overview:**

- Local Authentication and a start of network authentication.
- Access Control.
- Privilege Escalation.

# Authentication & Storage

# Main Concept

**Authentication**:

- Provide the identify of an entity (user, device, software, etc.)
- Binds the identity to the entity.

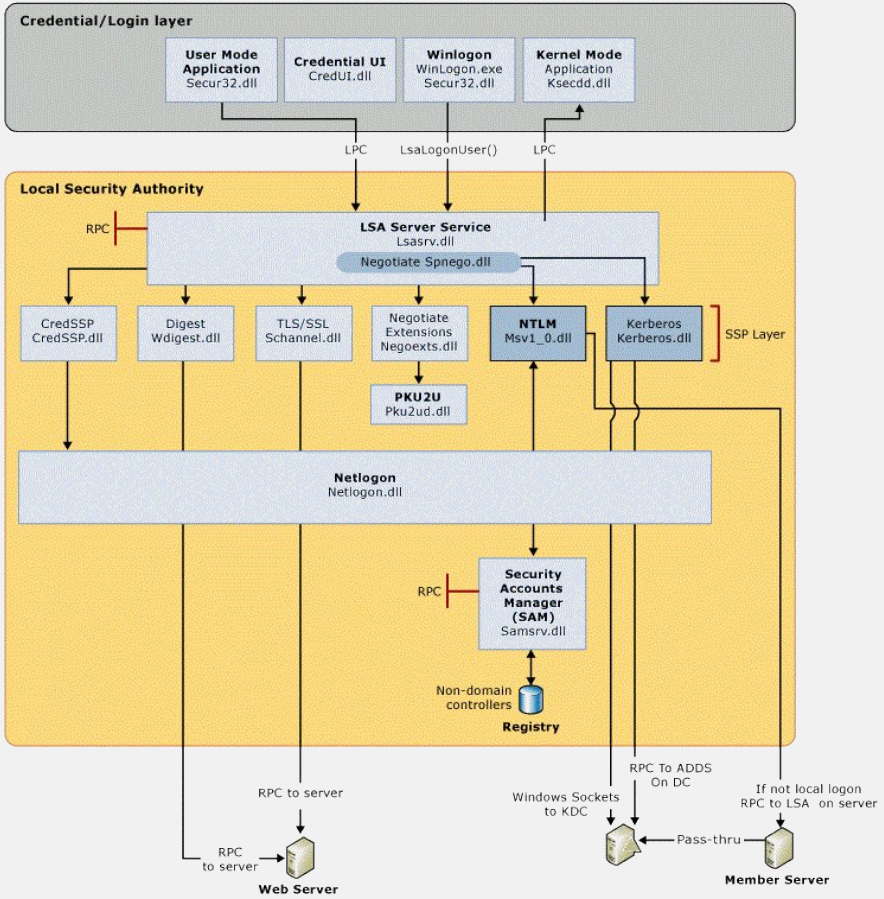
An authenticator asks the entity to prove its identity with evidences: **factors**.

- Something it knows (password)
- Something it has (smart card, physical token, phone)
- Something it is (biometrics)
- etc.

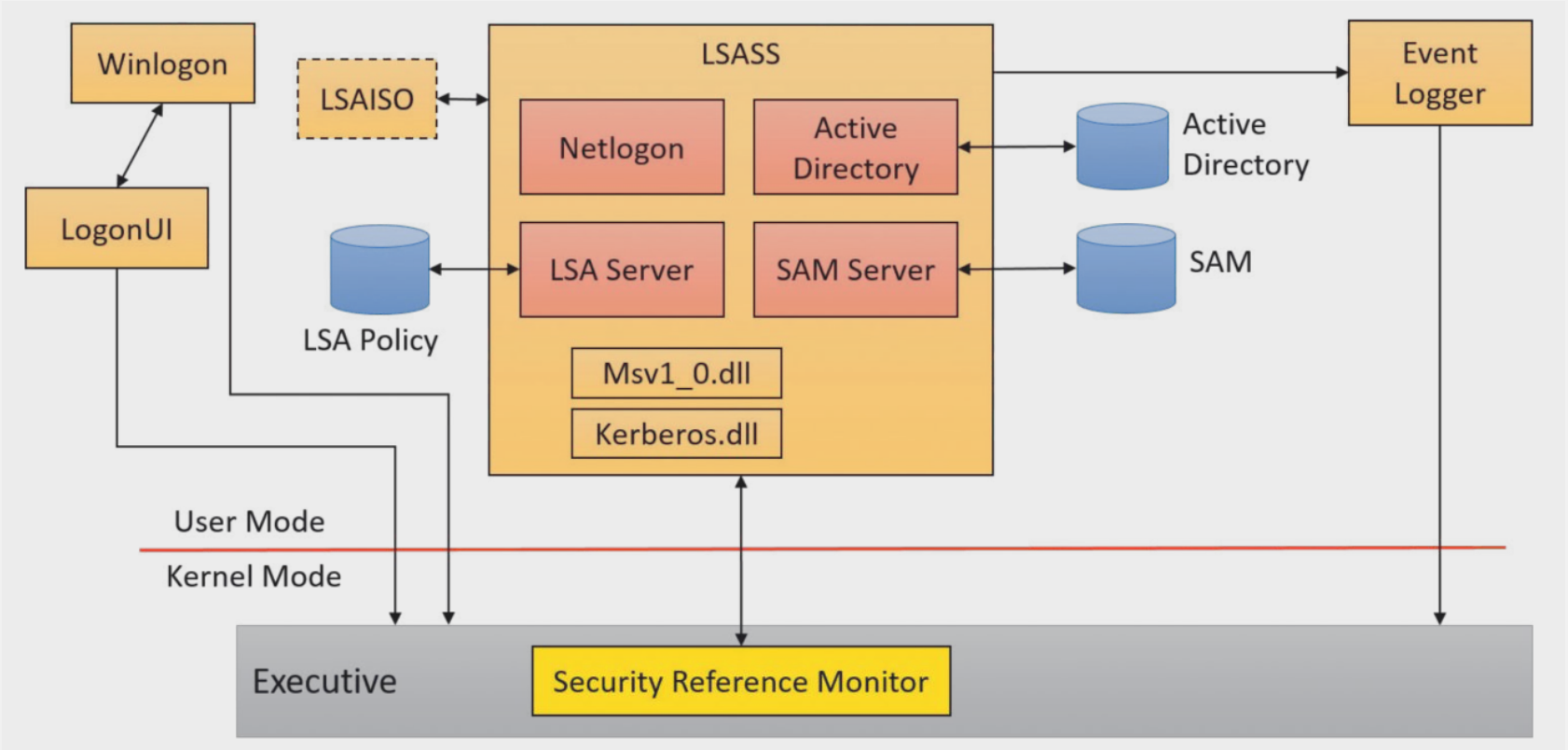
# Authentication on Windows 1/2

- User are identified by a Token including a **Security Identifier (SID)**.
- These tokens are checked by the **Security Reference Monitor (SRM)** upon access control verification.
- The **Local Security Authority (LSA)** checks the authentication policy.
  - The **LSA Subsystem Service (LSASS)** query the corresponding security package.
  - Once authenticated, the LSASS create a token for the user.
- Once authenticated, a process is spawned for the user (Explorer.exe, …)

# Authentication on Windows 2/2



# Authentication Components



# Authentication Components

- **Winlogon**: Interactive user authentication through LSASS
- **Credential Provider**: In-process objects in LogonUI used to get authentication factors.
- **Security Reference Monitor (SRM)**: Defines the access token data structure to represent a security context, performing security access checks on objects, manipulating privileges (user rights), and generating any resulting security audit messages. (More on this later)
- **Local Security Authority Subsystem Service (LSASS)**: Process run by the LSA to provide modular authentication.

# LSASS

- **LSA:** User-mode process responsible for the local system security policy.
- **Security Account Manager (SAM)**: Database that stores local user accounts and security descriptors (including hashed passwords).
- **LSA Policy**: Security policy parameters.
- **Credential Guard**: Don’t store the user’s token directly in the lsass’ memory (LSAISO)
- **Netlogon**: Service used to set up a secure channel with a domain controller.
- **Active Directory**: Service that contains information about objects in domains.

# Security Account Manager (SAM)

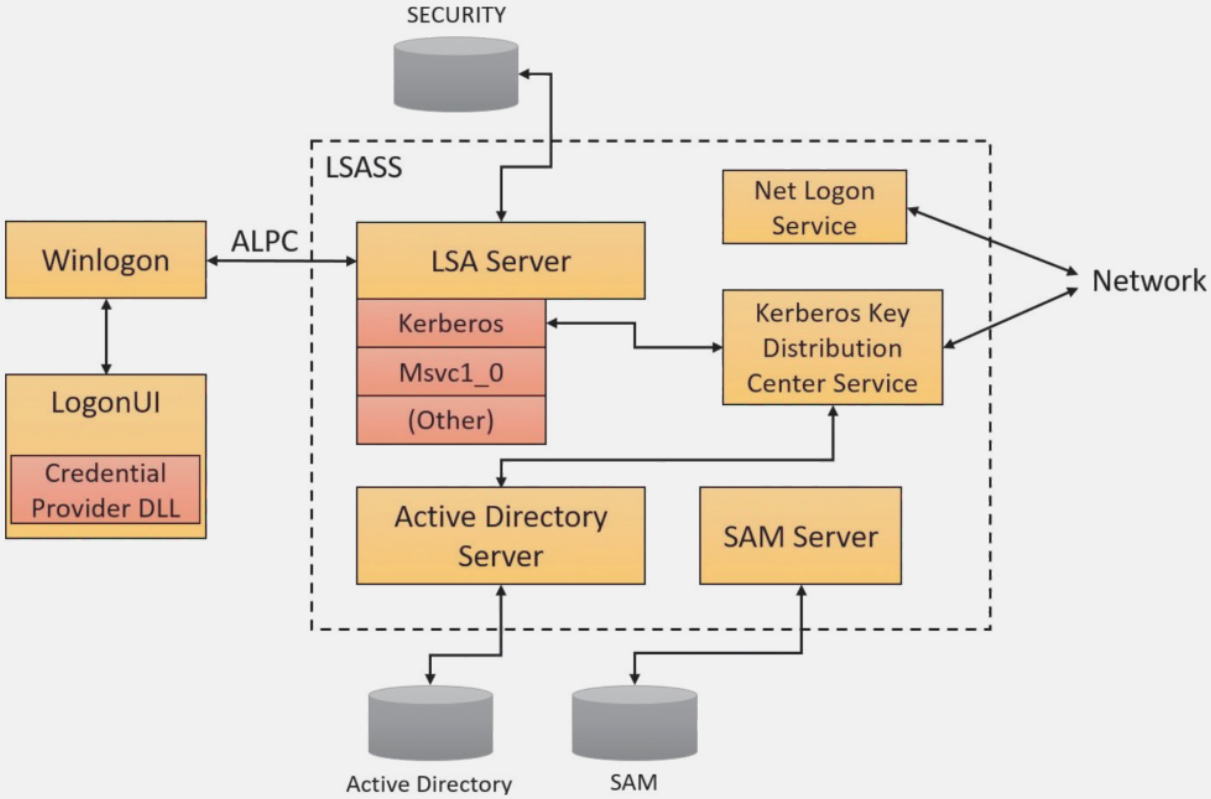
- The SAM server is linked to its own SAM database:
  - Can be seen as the /etc/shadow of Windows.
- It contains the local information for each account, settings, and password hashes.
  - Stored in HKLM\SAM (hash encrypted with a key stored in another register).

# Authentication Packages

- **Authentication Packages:**
  - Dynamic-Link Libraries (DLLs) implementing Windows authentication policies.
  - Authenticate users and give to the LSASS the required information to create a token.

The OS may contain many authentication packages, the most commonly found are msv1_0.dll and Kerberos.dll.

# Local Authentication Workflow



# Local Password-based Authentication

**Local authentication**: Uses the **MSV1_0** authentication packages

- Default authentication package on a standalone Windows.
- Takes the username and a **hashed version of the password**.
- Query the local SAM to retrieve the account information.
- Can be used in old (before Windows 2000) domains.

**Password are stored using NT / LM hashing:**

- *No salt, and very weak hash function.*
- LAN Manager (LM) hash are deprecated (since Vista).
  - Only using password of 14 characters maximum, full uppercase only.
- Hash NT is based on MD4. NTLM is the authentication protocol using these hashes.
- No anti-replay mechanism.
- Most local authentication use this.

Vulnerable to pass-the-hash attacks

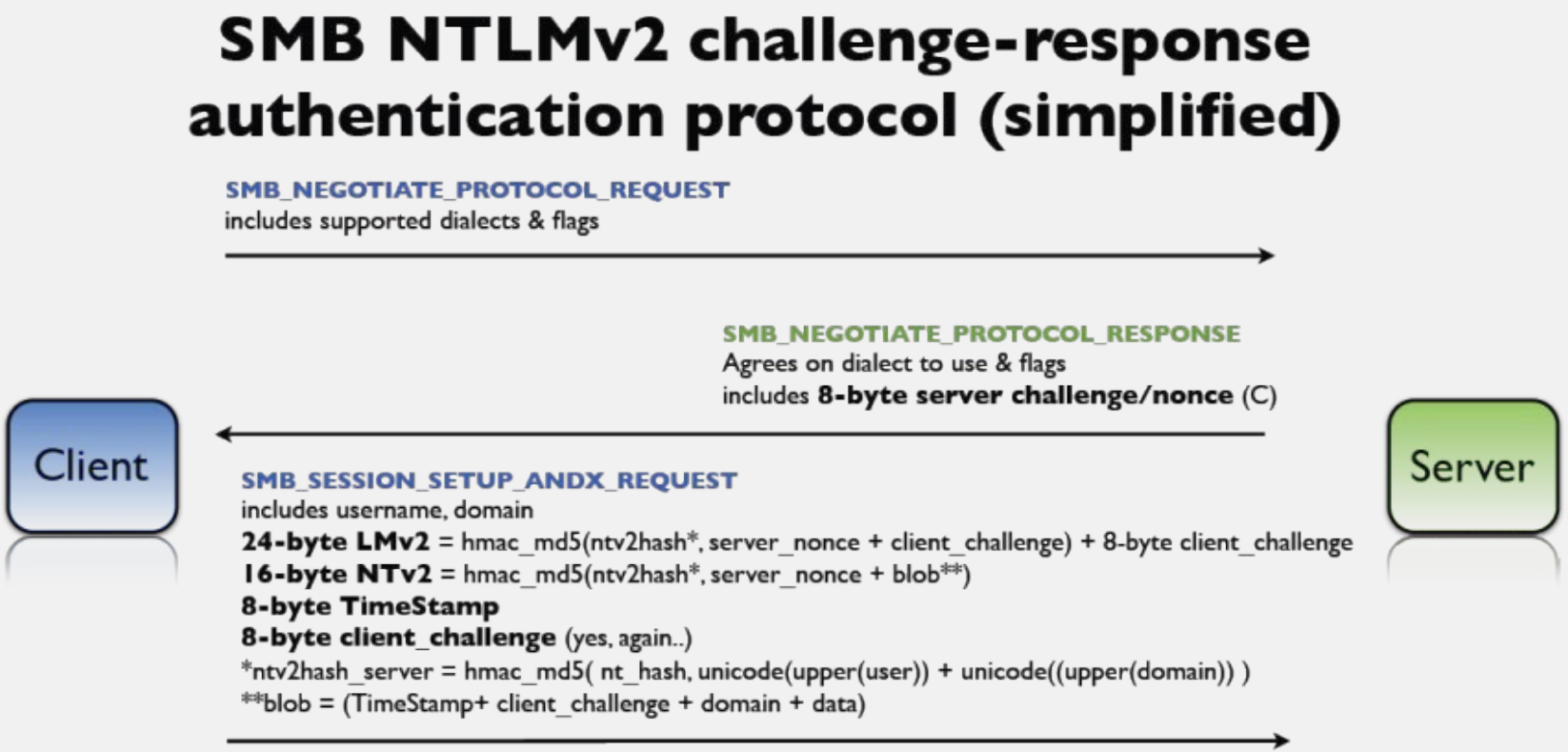
# Network Authentication - NTLMv2 Protocol

**Hash NT / LM ≠ NTLMv2**

Also provided by the MSV1_0 authentication package.

- Challenge-response between a client and Domain Controller/server.
- Still supported, but Kerberos is preferred (more on this next week).
- Different security based on LM compatibility level (0 - 5).

# NTLMv2 Protocol



# NTLMv2 Protocol

The LM compatibility level is defined as a group policy within Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options.

0 Send LM & NTLM responses

1 Send LM & NTLM - use NTLMv2 session security if negotiated

2 Send NTLM response only

3 Send NTLMv2 response only (default)

4 Send NTLMv2 response only. Refuse LM

5 Send NTLMv2 response only. Refuse LM & NTLM

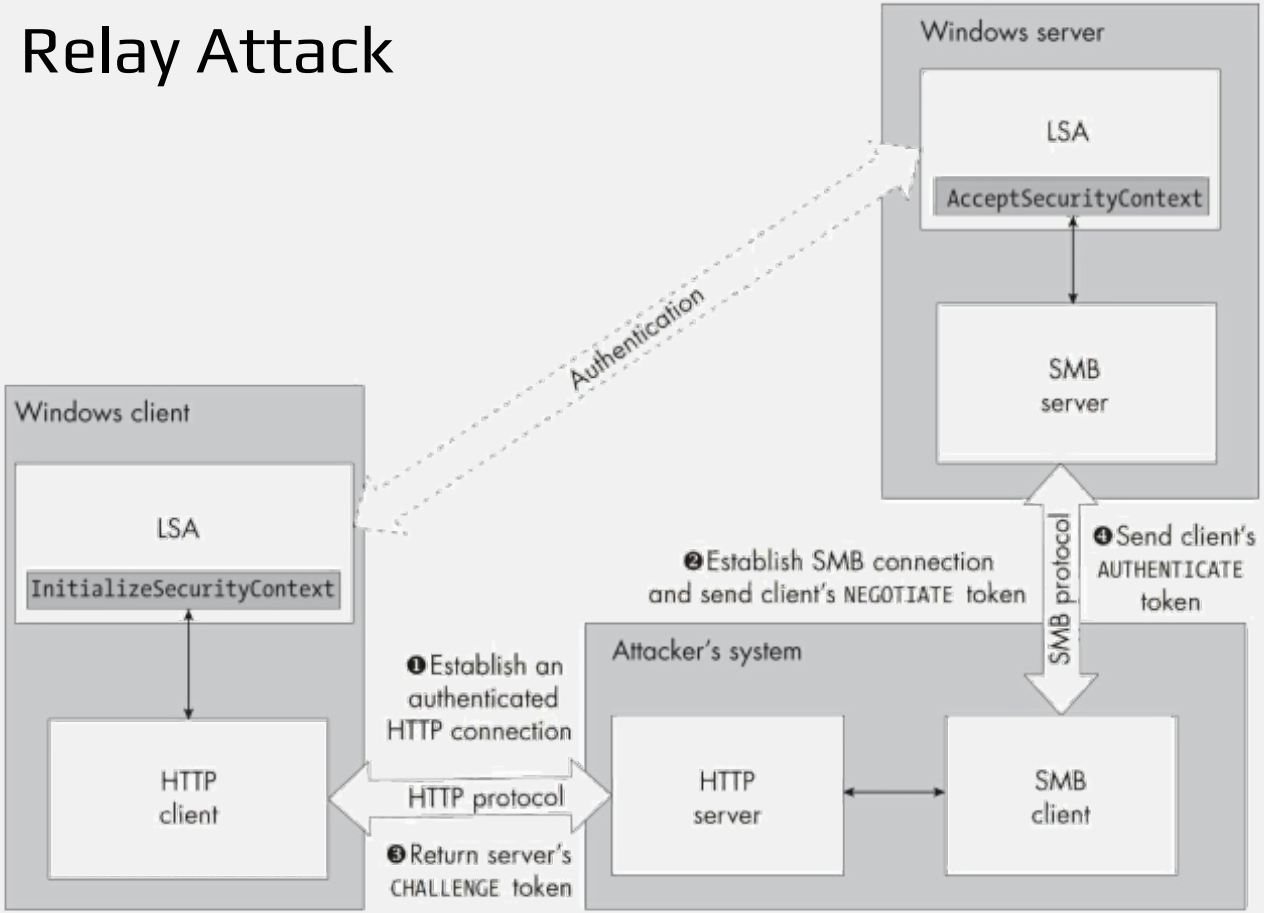
# NTLMv2 Protocol Security

- **2012**: We can break every possible 8-character NTLM hash in under 6h with 25 GPUs[1].
- **2019**: we can do it in less than 2.5h on a single NVIDIA 2080Ti.
- **Rainbow Tables** are available for 8 and nin-character passwords[2]
- Also vulnerable to *pass-the-hash* attacks.
- *Sensitive to relay attacks.*

NTLMv1 and NTLMv2 Session are disable by default since Vista. . . don't enable them

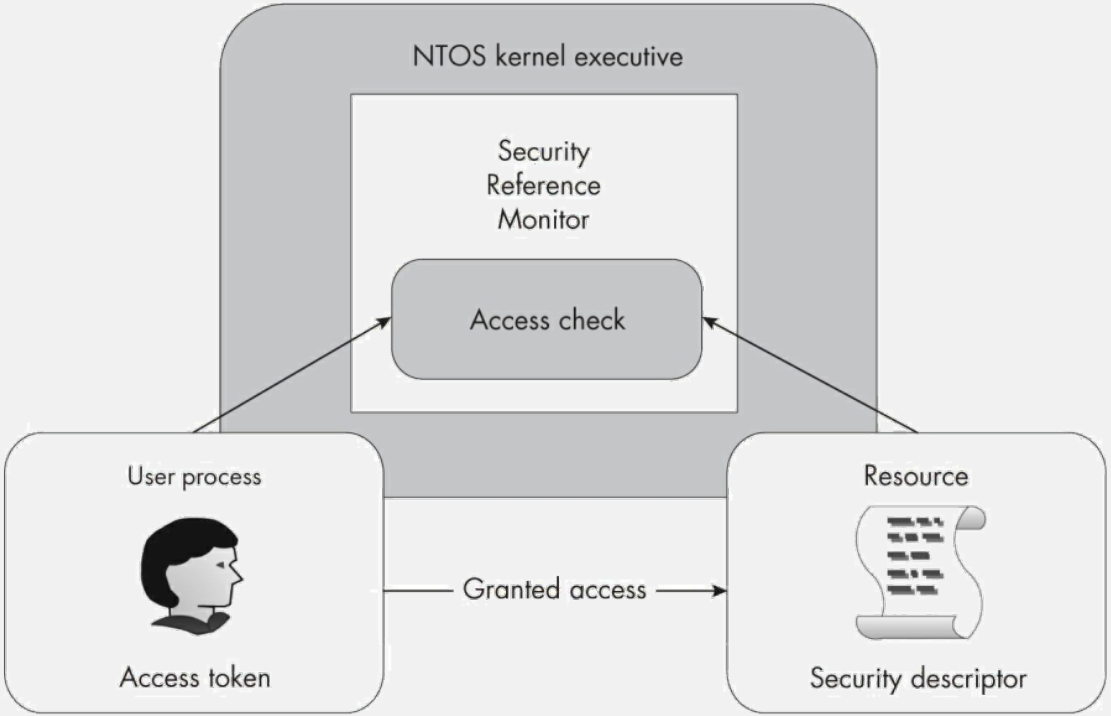
[1] https://arstechnica.com/information-technology/2012/12/25-gpu-cluster-cracks-every-standard-windows-password-in-6-hours/
[2] https://www.rainbowcrackalack.com/

# NTLMv2 Relay Attack



# Access Control and Privileges

# Security Reference Monitor (SRM)



# Mandatory Access Control (MAC) 1/2

**Oh look mom, another MAC acronym in computer science!**

DAC is the Discretionary Access Control and is based on per user permissions.

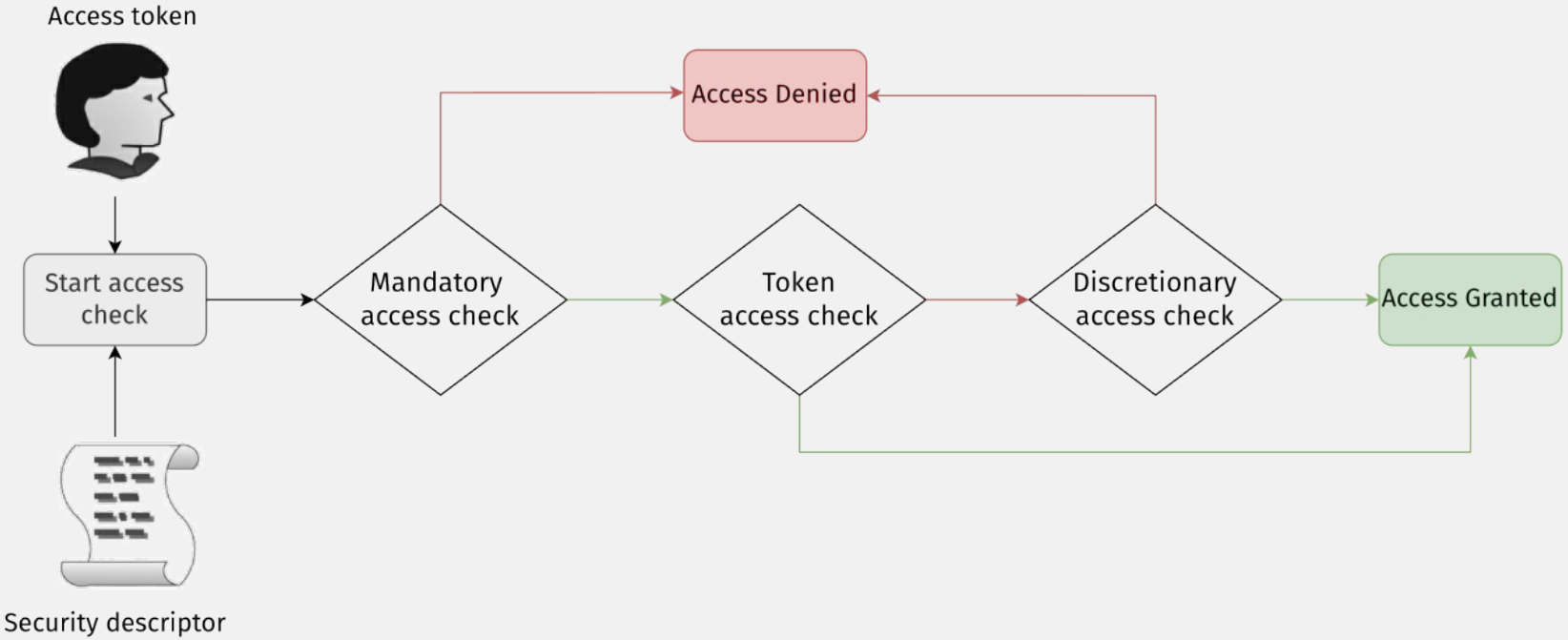
MAC represent **security levels** enforced **by the system**.

# Mandatory Access Control (MAC) 2/2

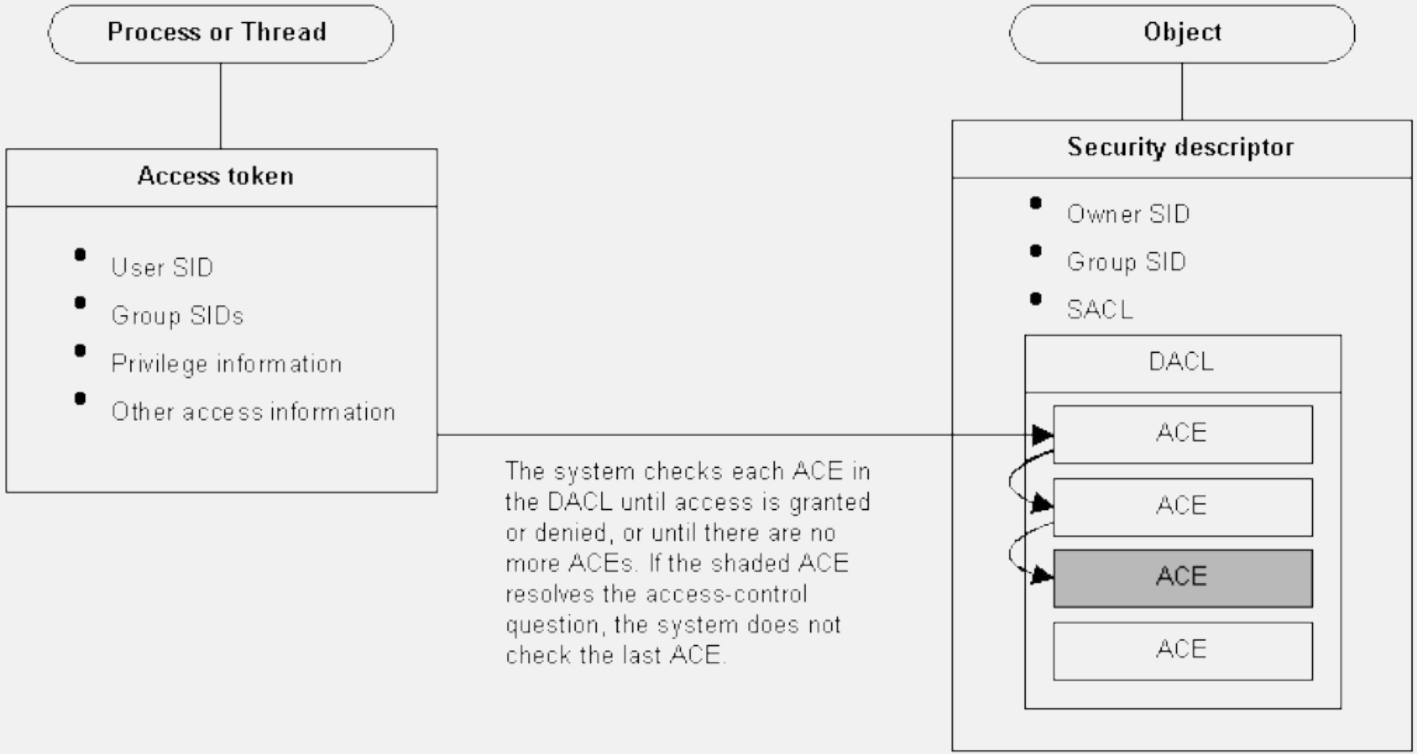
- Access rights are determined by a central authority, not by the resource owner. It is determined for subjects(user, process, etc.) and objects (resources).
- **Characteristics**:
  - More secure than DAC, less flexible.
  - Can’t be modified by the owner of an object.
  - Both subjects and objects have labels.
- **Elements**: Each subjects and objects have both a Security Level and a Category.
- **Use Case**: Common in government/military context for clearance (Top Secret, Secret, Confidential, Public).

Example: Only subjects with a certains level of clearance can access Top Secret data.

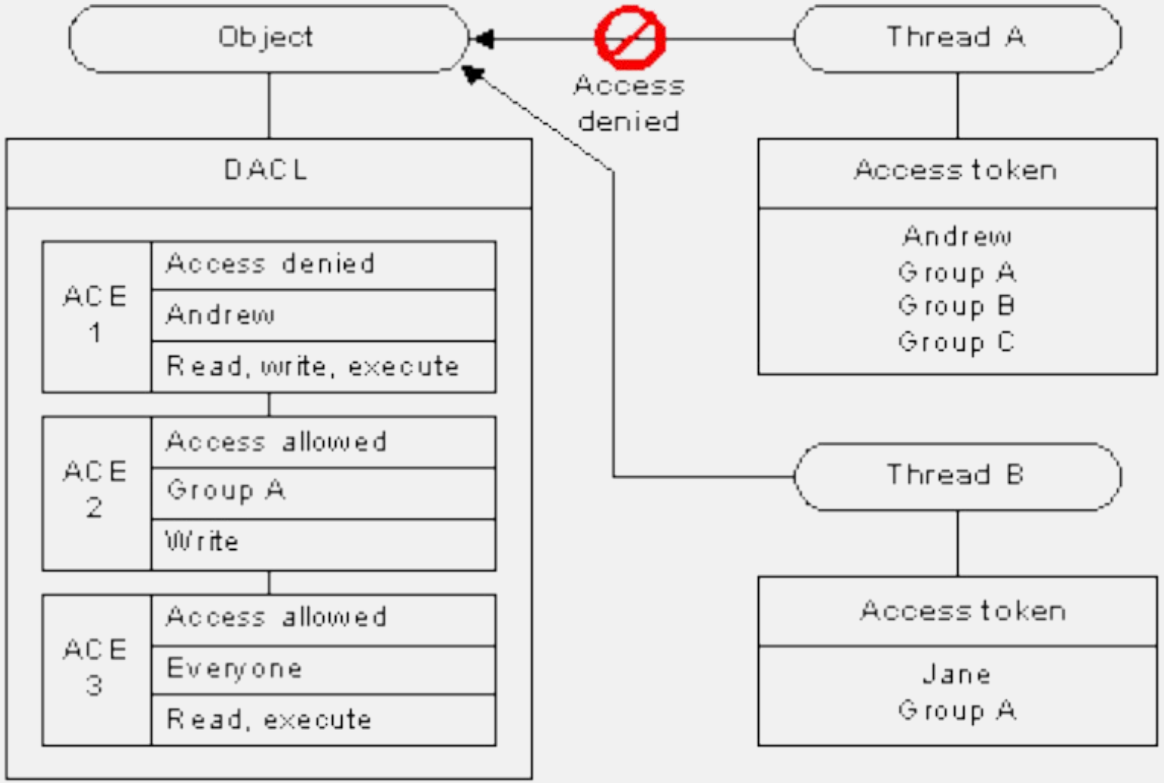
# Access Check Workflow 1/3



# Access Check Workflow 2/3



# Access Check Workflow 3/3



# Access Tokens

- Upon successful login, LSASS create an initial *token* that identify the *security context*
- Contains user information for secure access control
  - SIDs
  - Default DACL
  - Privileges
  - Integrity level
  - etc.
- This token will be inherited by all process spawned by the user.

# Access Tokens - SIDs

**Security Identifiers (SIDs):**

- Unique identifiers used by Windows to manage permissions and access control.
- Each SID corresponds to a user, group, or device account.
- SIDs are fundamental to the Windows security model, ensuring that each entity is distinct and traceable.

# Access Tokens - Groups

**Groups:**

- SIDs within a token can represent groups to which the user belongs.
- These group SIDs determine what resources the user can access, depending on the permissions assigned to these groups.
- Common examples include Users, Administrators, or custom-defined groups in an enterprise environment.

# Access Tokens - Default DACL

- Specifies the permissions that apply to objects created by the user or process.
  - This is DAC in Windows.
- Ensures that new objects have a baseline level of security, reducing the risk of unintentionally insecure configurations.
- Administrators can modify the Default DACL to enforce stricter access controls for new objects.

# Access Tokens - Integrity

- This is the label for the MAC in Windows.
- Used for Mandatory Integrity Control (MIC), since Vista.
- Higher values means more accesses (principal of NO {WRITE,READ,EXEC} UP)
- Each process and object is assigned an integrity level
- Default to Medium

# Integrity Levels (IL)

**Main levels:**

- Untrusted: For processes with anonymous logins.
- Low: Typically used by web browsers running in restricted mode.
- Medium: The default level for standard user processes.
- High: Assigned to processes with elevated privileges (e.g., Administrator).
- System: Used by critical system processes.

# Windows Mandatory Integrity Control (MIC)

- Integrity levels are used to enforce restrictions on processes based on their trustworthiness.
- Each object and process is assigned an integrity level.
- MIC ensures that lower-integrity processes cannot “modify” higher-integrity objects.
- Policies are defined in the SACL, using mandatory labels.
- When a process is created, **it gets the minimum integrity level**.
- This control occurs before DACL checks.

**Q:** Why take the minimum integrity level between the user and the file being executed?

# IL vs DACLs

- **DACLs (Discretionary Access Control Lists)**: Define what actions a user or group can perform on an object.
- **Integrity Levels (IL):** Add an additional layer of security, limiting the actions based on the trustworthiness of the process.
- **Differences:**
  - DACLs focus on who can access the object and what they can do.
  - Integrity Levels focus on preventing less trusted processes from modifying or accessing more trusted resources.

# Access Tokens - Privileges 1/2

- Privileges are special rights assigned to a token that determine the high-level actions a user or process can perform.
- Examples include the ability to shut down the system, load device drivers, or back up files.
- Privileges are often more powerful than standard permissions and are typically assigned to administrative or system accounts.

**Enabled vs. Disabled privileges**

- Some privileges are enabled by default, while others may need to be explicitly enabled by the process.
- The distinction between enabled and disabled privileges affects what the process can immediately do without requesting additional rights.

# Access Tokens - Privileges 2/2

**Privileges differ from access right:**

- Privileges control access to system resources and system-related tasks, whereas access rights control access to securable objects.
- A system administrator assigns privileges to user and group accounts, whereas the system grants or denies access to a securable object based on the access rights granted in the ACEs in the object's DACL.

34 privileges defined in Windows, including at least 6 allowing to **get Admin rights**:

- SeCreateTokenPrivilege, SeDebugPrivilege, SeLoadDriverPrivilege, SeRestorePrivilege, SeTakeOwnershipPrivilege, SeTcbPrivilege

# Access Tokens - Token Type

**Primary Tokens**

- Represent the security context of the user associated with a process.
- Used when the process runs under the user’s security context.

**Impersonation Tokens**

- Allow a process to temporarily adopt the security context of another user.
- Commonly used in scenarios where a service needs to perform actions on behalf of a client.

**Impersonation Levels**

- Impersonation tokens can have different levels: anonymous, identification, impersonation, or delegation.
- The level determines how far the impersonation can go, e.g., whether it can be extended to other machines.

# Pwning and Privilege Escalation on Windows

# Accessing a Machine

Kind of the same as with a Linux machine:

- Here you want to look at open ports with `nmap` and try to compromise one of the open services (e.g., web server, or vulnerable app).
- Or find a way to authenticate using available credentials if you are on an already compromised device in the domain.

# Metasploit

**What is Metasploit?**

- A framework for exploitation and post-exploitation
- Contains a large library of exploits, payloads, etc., (a lot of them).
- Helps automate attacking steps after a vulnerability is identified.
- Often used to establish a session on the target (e.g., a shell or Meterpreter agent).

Too huge for an intro or overview but be sure to check it out during your pentest (tutorials are legions).

# Credential Harvesting

**Collecting authentication material that already exists on the system:**

- Goal is re-use, or crack if needed, existing credentials.

**Some tools:**

- **ProcDump / comsvcs.dll** → create a memory dump of LSASS.
- **Mimikatz** → extract hashes / tickets from the dump or live LSASS memory.
- **LaZagne** → recover application-stored passwords (browser, RDP, SSH, etc.)

# Example of Mimikatz Output

```
mimikatz # lsadump::sam
Domain : VAGRANT-2008R2
SysKey : 4138feec7d6e226f64c6dc2dfa187ba1
Local SID : S-1-5-21-1268359432-1364675841-3512908913

SAMKey : ca451bca62bdf12bf8e083c4400c3e68

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: e02bc503339d51f71d913c245d35b50b

RID  : 000001f5 (501)
User : Guest

RID  : 000003e8 (1000)
User : vagrant
  Hash NTLM: e02bc503339d51f71d913c245d35b50b

RID  : 000003e9 (1001)
User : sshd
```

# Common Bad Practices

- Services configured with writable directories.
- Configuration files containing plaintext credentials.
- DLL search order hijacking.
- Scheduled tasks executed as SYSTEM.

# winPEAS: Windows Privilege Escalation Awesome Script

**WinPEAS**: Automated local enumeration script that collects system artefacts and highlight potential privilege escalation vectors.

**What it does:**

- Gathers environment info
- Enumerates file permissions, binaries, services, schedule jobs, etc.
- **Key outputs to pay attention to**: writable service executable/directory, unquoted service paths, weak ACLs on sensitive files, presence of cleartext secrets in config files, scheduled tasks running as SYSTEM.

# Resources and Acknowledgements

- https://book.hacktricks.wiki/en/
- https://learn.microsoft.com/en-us/
- *Windows Internals, Part 1, 7th Edition*
- External materials from Daniel De Almeida Braga.