AGENDA

/ 01 Introduction

/ 02 The Active Directory model & Windows domains

/ 03 Pentesting Windows domains for fun and profit

/ 04 Conclusions
/ 01

Introduction
C:\> whoami

Jean MARSAULT - EURECOM 2014 - Security track

Pentester & consultant at Wavestone (formerly known as Solucom)

Digital forensics & incident response with the CERT-W

@iansus

iansus on: Root-me, w3challs, Newbiecontest, etc ➔ ask me if interested
Microsoft Windows history

/ User-oriented operating system: easy to use, no technical knowledge needed

/ Up to 80% coverage of large corporations’ information system

/ Brief history of user versions:

1.x / 2.x - 1989 → 3.x - 1993 → Windows 95 
End of extended support in 2017-04 → Windows XP - 2001

Microsoft Windows – use cases

**Personal use**
- “Home edition”
- Cheaper
- Fewer security features
- Fewer configuration parameters

**Company use**
- Two separate OS branches:
  - Workstations
  - Servers
- More expensive
- Best security features
- More customizable
- Able to join or create a Windows domain

Today we will focus on this case
Some vocabulary – Windows specific components

<table>
<thead>
<tr>
<th>Registry</th>
<th>Filesystem</th>
<th>Users and groups</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NTFS</td>
<td>Every user and group gets a security identifier (SID)</td>
<td>Similar to daemons on Unix systems</td>
</tr>
<tr>
<td></td>
<td>Discretionary access control lists (DACL)</td>
<td>SIDs are used in DACL</td>
<td>Can be scheduled to start at boot</td>
</tr>
<tr>
<td>In-memory database with ACL when OS is up</td>
<td></td>
<td>SIDs allow complex group / user architecture by inclusion</td>
<td>User account used can be configured</td>
</tr>
<tr>
<td>Stored on the filesystem when the OS is powered off</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Used for configuration storage at user or machine scope</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Process list is similar to Unix
- Access tokens to perform operations
- Integrity levels to secure inter-process actions

Remote procedure call (RPC) for service interaction
- Simple Message Block (SMB) for remote file access
- Remote Desktop Protocol (RDP) for remote GUI access (~ ssh -X)
The Active Directory model & Windows domains

Before joining a domain
User accounts and groups

/ Each account and group is mapped to a Security Identifier (SID)
  › e.g. S-1-5-21-3669152439-339947406-2872813669-500

/ Default accounts:
  › User account: Administrator, Guest
  › Service accounts: SYSTEM, Local Service, Local network, etc

/ Default groups: Local administrators, Remote desktop users, etc

/ Groups can include other groups and / or users, through SIDs

/ SIDs are used in Discretionary Access Control Lists (DACL), which are a complex combination of:
  › Fine-grain rights segmentation
  › Order allow / deny attribution of these rights to user or group SIDs

/ Some accounts have high privileges and are ideal targets for privilege escalation:
  › SYSTEM is equivalent to root
  › Administrator (SID XXXX-500) and members of the “Local administrators” group can become SYSTEM without password
User password storage

Windows has been using two hash functions to store passwords:

- **LM (Lan Manager) hash function**, known to be weak and now deprecated (Windows stores only LM(""))
- **NTLM hash function**, based on MD4 and still used in the most recent versions of the OS

Accounts’ NTLM hashes are stored in the registry (in-memory while powered on) in the Security Account Manager (SAM) hive:

When powered-off, this hive is located under C:\windows\System32\config\sam
Why Windows domains?

Computers outside domains exhibit the following drawbacks in a company environment:

/ They can’t be managed on a large scale except with handmade scripts
/ Local administrator users have full control over their workstation
/ The system is not natively compatible with centralized Identity Access Management (IAM), including:
  › Centralized employees and resources directories
  › Enterprise Public Key Infrastructure (PKI) and smartcards

Information Systems require the ability to act on the whole system at once, which is not possible on such workstations
The Active Directory model & Windows domains

Sneak peak of Windows domains
The hidden truth behind Active Directory

- Windows servers can be configured to take many roles: DNS server, network share, Certification Authority, etc.
- One of these roles is the “Active Directory” and has a central place in Windows domains.
- Active Directory (AD) is Microsoft’s implementation of the Lightweight Directory Access Protocol (LDAP), which allows:
  - Maintaining a centralized directory of users, groups, resources, etc.
  - Implementing centralized authentication mechanisms.
  - Building the base of many features that can be used in Windows domains.

- The Active Directory stores users, computers, etc as objects, which:
  - Follow a predefined schema, also stored in the Active Directory.
  - Define a number of properties as dictated by the object schema.
Finally defining “Windows domain” 😊

A “domain” is the name given to a collection of:

/ Windows servers (running on Windows Server 20xx)
/ Windows workstations (running on Windows Vista, 7, 8.x, 10, etc)
/ One or more servers hosting a centralized Active Directory service: the domain controllers, used for:
  › Centralized authentication
  › Centralized authorization
What can we do with it?

/ Centralized **identity management** and **authentication**:
  › Domain user accounts working on any domain workstation / server in addition to local accounts
  › One password to rule them all
  › Account is either `<computer>\<username>` (local) or `<domain>\<username>` (domain)

/ Access to centralized **resources**, including:
  › File sharing servers (network shares)
  › Enterprise PKI (enabling **smartcard logon**): Certification Authorities, CRL distribution points, OCSP responders, etc

/ Centralized **management**:
  › Domain administrators can defined Group Policy Objects (GPO) or Group Policy Preferences (GPP)
  › They will apply to a every object in an admin-defined subset of users / computers
  › It allows large scale configuration of the workstations and servers, on-the-fly propagation of new parameters
  › **Group policy cannot be permanently overridden, even by local administrators**

/ Easy creation of **role-defined servers**, for example:
  › DNS servers (FQDN is set as a property of the computer object)
  › Web servers relying on the domain users identity and rights
The Active Directory model & Windows domains

Authentication on Windows domains
Domain users password storage

- Domain users use centralized authentication to log on to domain computers
- Password storage must be centralized
- NTLM hashes are stored in the "ntds.dit" file present on domain controllers
The DC only knows my NTLM hash and not my password

What is sent by the workstation to the DC so I can be authenticated?

- Password to be hashed? No
- NTLM Hash? No

This would be sensitive information sent over the network.
We need a way of proving the knowledge of the password without sending it.
The goal of this authentication protocol is to prove the knowledge of the NTLM hash of my password.

You would be able to prove your knowledge of the password itself but the DC does not know it.

Example of **NTLMv1**:
- \([\text{NTLM} + \text{padding}]\) split into \(K_1, K_2\) and \(K_3\)
- \(R = \text{DES}(C, K_1) \mid \text{DES}(C, K_2) \mid \text{DES}(C, K_3)\)

Password = waza1234/

\[
\begin{align*}
\text{NTLM} &= \text{CC36CF...46158B1A} \\
R &= \text{B50F926D} \\
C &= \text{A4FE815C} \\
R &= \text{B50F926D}
\end{align*}
\]
Advanced authentication with Kerberos

Kerberos is an authentication protocol designed by the MIT in the 80s.

It relies on tickets distributed by the Kerberos Distribution Center (role often born by the DC) and consumed by target servers. Some vocabulary:

- **TGT** = Ticket Granting Ticket
- **TGS** = Ticket Granting Service, which generates Service Tickets
- **Service server**, consuming these tickets

![Diagram of Kerberos authentication process](attachment:kerberos_diagram.png)
Computers can be configured to *cache domain credentials* in the registry in the event the DC cannot be reached:

- Usually laptops, less frequently workstations
- Usually not servers
- Storage format used is “mscachev2”, hard to break, but can still be beaten by dictionaries on weak passwords:
  - DCC1 = MD4(NTLM \ username)
  - DCC2 = PBKDF2(HMAC_SHA1, 10240 iterations, text = DCC1, salt = username)

Users can rely on other authentication methods including:

- **Smartcard logon**: the correct PIN unlocks access to the NTLM hash which is then used to generate a Kerberos TGT
- **Windows Hello**: use of biometric features (smile, etc) to unlock access to the hash
Introducing Mimikatz

Windows authentication relies on **credentials providers**:
- They cache credentials (optionally encrypted) to provide with Single Sign-On (SSO) capabilities
- The OS must be able to **decrypt encrypted credentials in a transparent way** for the user
- Credentials include: cleartext passwords, NTLM hashes, Kerberos TGT & TGS
- **These credentials are present in the memory of the lsass.exe process**

Benjamin “gentilkiwi” Delpy has developed the “Mimikatz” tools which runs with local admin privileges and:
- Requests the “SE_DEBUG” privilege and queries the lsass.exe process memory
- Relies on Windows API to decrypt encrypted credentials
- Prints out credentials for accounts that logged on the computer since its last shutdown
Pentesting Windows domains for fun and profit
Some interesting domain users and groups:

/ DOMAIN\Domains Admins: domain group which is included in every server and workstation local administrators group
/ DOMAIN\Administrator: default domain administrator account included in the “Domain Admins” group
/ DOMAIN\krbtgt: domain user whose NTLM hash is used to digitally sign Kerberos tickets

Some useful vocabulary:

/ Group Policy Objects (GPO): user or computer configuration elements set on the DC that frequently apply to the computers in the domain
/ Rootie: action of taking a flipped selfie while becoming a “Domain Admins” member in an unauthorized way
Mission briefing

- Authentication bypass
- Pivoting and lateral movement
- Local privilege escalation

Exploit

Post-exploit

- Hash dumping
- Ticket forgery
- etc

Domain Admin
Pentesting Windows domains for fun and profit

Authentication bypass and local privilege escalation technics
Attack – Pre-logon SYSTEM shell using “Utilman”

/ Utilman.exe is a small executable giving the “Ease of access” menu

/ As it can be launched pre-logon, it executes using the SYSTEM account

```
root@debian:/c/Windows/System32# ls -l cmd.exe Utilman.exe
-rw----- 2 user user 345088 Nov 21 2010 cmd.exe
-rw----- 2 user user 1402800 Jul 14 2009 Utilman.exe
root@debian:/c/Windows/System32# mv Utilman.exe Utilman.exe.bak
root@debian:/c/Windows/System32# cp cmd.exe Utilman.exe
-rw----- 2 user user 345088 Nov 21 2010 cmd.exe
-rw----- 2 user user 1402800 Jul 14 2009 Utilman.exe
root@debian:/c/Windows/System32# ls -l cmd.exe Utilman.exe
```

/ Mounting the disk from a live USB allows replacing Utilman.exe by cmd.exe

/ You can open a shell using the SYSTEM account by clicking a button!

/ You can add local administrator accounts from this console
Mitigation – Pre-logon SYSTEM shell using “Utilman”

/ Attacker managed to tamper with system executables

/ Potentially more damage could come from mounting windows disk:
  › Changing SAM / MsCachev2 entries
  › Replacing local credential providers DLL libraries (see mimilib)

/ If the disk is encrypted, access to it from a live USB system is prevented

/ Most used solution is now Bitlocker, provided (not free) by Microsoft, others exist (Truecrypt / Veracrypt)

/ Relies on the Trusted Platform Module (integrated chip with secret protection and caller access control)

/ Unencrypted Microsoft system partition accesses the TPM, optionally asking for the user PIN, and retrieves the decryption keys

/ Access to the disk goes through Bitlocker subsystem

⚠️ Decryption keys can be recovered from memory dumps, and utilities such as bdemount allow mounting encrypted volumes when provided the keys
One of many exploits against Windows, with some pluses:

- Directly opens SYSTEM shell
- PowerShell-based, no executable needed => harder to block or detect
- Only requirement is having at least a 2-core processor
Attack – Advanced CVE exploit – rogue domain controller

Attacker has no account on the target system, and disk may be encrypted (without user PIN though)

Original exploit in 2015: use a fake domain controller, set a fake password on the target user as expired
• The lock screen will accept the fake password
• It will ask the user to set a new one
• This will poison the MsCachev2 local database
• As long as the real DC is unreachable, authentication will be granted on the computer
• Relies on Kerberos, but tickets are verified after password change and cache poisoning

This was only auth bypass, privilege escalation presented by Belgium researchers @Hack in Paris 2K16:
• Remember GPO?
  • User and computer configuration elements
  • Can impact predefined Windows parameters
  • Some elements, for example company-specific, require a script to be executed
  • For computer configuration, scripts execute as SYSTEM

• Set a GPO launching cmd.exe on target system
  • Quite easy on Windows 7
  • Required harder work with domain SIDs on Windows 10 but still a success
Mitigation – CVE exploit

Both examples relied on the exploitation of public vulnerabilities that have been patched.

The main mitigation strategy is the IT golden rule: keep your systems up-to-date.

Other hardening solutions can be used to increase protection against 0-days:

- **Executable whitelisting** – Applocker, restrictions on:
  - Executable digital signature
  - Executable location (C:\Windows\, C:\Program Files\ etc)
  - Executable checksum

- **Endpoint Detection Response (EDR)** – next gen antivirus
  - Can handle fileless malware
  - Rely on statistical / behavioral online shared databases (threat Intelligence)
  - Work in real-time rather than with scheduled scans
Pentesting Windows domains for fun and profit

Pivoting and lateral movement – Pass-the-*
Lateral movement – Context and objectives

/ **Context:**
  › You have successfully compromised a workstation
  › You are (at least) **local administrator** on the workstation
  › But none of the accounts you can target on the workstation is **Domain Admin**...

/ **Objectives:**
  › Identify Domain Admin accounts
  › Identify workstations they have logged on to recently
  › Identify domain accounts that are local administrators on these workstations
  › If you have compromised one of these accounts, **the loop is over**
  › Else repeat searching for workstations these domain account that are local administrators have logged on to

/ Pretty hard to do by hand, especially on large domains (~100K ws and servers, ~50K users)

/ **Hopefully, some tools might help you identify the critical paths to Domain Admin accounts:**
  › **AD Control Path (ADCP)**: French tool developed by ANSSI
  › **Bloodhound**: recent PowerShell tool that identify live sessions on workstations and servers
Obvious first – Pass-the-Pass

/ Pass-the-Pass?
  › On older systems, passwords are stored in a reversible encrypted way in memory
  › If you manage to steal the encrypted password, you can ask the system to decrypt this for you

/ How do I get the pass?
  › The answer is Mimikatz
  › As a local admin, you are able to ask for SE_DEBUG_PRIVILEGE (~ ptrace)
  › sekurlsa::logonPasswords injects in lsass.exe memory and grabs the cleartext password of logged-in users
  › You can also dump the lsass.exe memory in the Task Manager and use this dump offline 😊
DEMO TIME!
Pass-the-Pass mitigation

Only applicable to **Windows >= 7** and **Windows Server >= 2008 R2**

Enabled by default in:
- Windows 8.1 +
- Windows Server 2012 R2 +

Disabled by default (and requires a Microsoft KB to be enabled) in:
- Windows 7 / Windows 8
- Windows Server 2008 R2 / Windows Server 2012

Registry key “HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\Wdigest”
- Value name is “UseLogonCredential”
- 1 means insecure
- 0 means secure

On Windows 7, acts as an **added level of protection**

On Windows 8.1, can be used to **downgrade the level of protection**, only requires user session unlock
Who needs passwords anyway, Pass-the-Hash is here!

/ Pass-the-Hash?
   › Remember that password might no longer be stored in a reversible way in memory
   › However, NTLM hashes still are in order for SSO to work
   › NTLMv1/NTLMv2 authentication protocol only requires you to prove knowledge of the NTLM hash
   › It becomes possible to impersonate the user if you steal his NTLM hash

/ How do I get the hash?
   › Still Mimikatz
   › sekurlsa::logonPasswords injects in lsass.exe memory and grabs the NTLM hashes of logged-in users
   › Also works offline with the memory dump of lsass.exe

/ How do I use it?
   › Answer is still Mimikatz (but tools such as CrackMapExec, impacket or Metasploit work too)
   › System program “runas” allows you to run programs as other users
   › When you know the workstation won’t be able to verify the credentials, use “/netonly” to load them in the process memory and have them used (and verified) on the network only
   › sekurlsa::pth uses the same technics, but only loads the NTLM hash instead of the user’s password
DEMO TIME!
Pass-the-Hash mitigations

/ Hashes cannot be removed from memory without altering some SSO features
  › Started with Microsoft AD functional level 2012 R2
  › Domain group “Protected users” becomes available
  › Domains in this group won’t use NTLM (Kerberos only)
  › Therefore, NTLM hashes are not present in memory anymore
  › But, users cannot perform NTLMv1 / NTLMv2 authentication without manually entering their password each time

/ Or can they? Introducing Credential Guard and Virtual Secure Mode (VSM)
  › Started with Windows 10
  › If enabled, Windows adopts a new architecture, based on hypervision (~ virtual machines)
Pass-the-Hash mitigations – focus on VSM
Pass-the-Hash mitigations

/ Hashes cannot be removed from memory without altering some SSO features
  › Started with Microsoft AD functional level 2012 R2
  › Domain group “Protected users” becomes available
  › Domains in this group won’t use NTLM (Kerberos only)
  › Therefore, NTLM hashes are not present in memory anymore
  › But, users cannot perform NTLMv1 / NTLMv2 authentication without manually entering their password each time

/ Or can they? Introducing Credential Guard and Virtual Secure Mode (VSM)
  › Started with Windows 10
  › If enabled, Windows adopts a new architecture, based on hypervision (~ virtual machines)
  › Credentials are no longer stored in the user-OS’ Lsass memory
  › Authenticated transaction requests between the user OS and the secure OS
  › No possibility of hijacking the secure OS from the user OS due to Kernel Code Integrity
  › But only available for Windows 10 Enterprise version, not even for Windows Server due to additional layer of hypervision
No hash? All your tickets are belong to us! Pass-the-Ticket

/ Pass-the-Ticket
  › Remember Kerberos?
  › This time aim for the Ticket-Granting-Ticket (TGT) and Ticket-Granting-Service (TGS)
  › Only drawback: TGT default lifespan is 10 hours and default max lifetime is 7 days

/ How do I get the hash?
  › M*****Z 😊
  › sekurlsa::tickets /export injects in lsass.exe memory, grabs the TG* of users and exports them in .kirbi files
  › Still works offline with the memory dump of lsass.exe

/ How do I use it?
  › Answer remains Mimikatz
  › Injects ticket in the current user Kerberos tickets database, even if not meant for him/her
  › Can be used transparently in Windows
  › Let’s see for oursevles!
DEMO TIME!
Pass-the-Ticket mitigations

/ Use of **Windows 10 Virtual Secure Mode** (with the limitations previously mentioned)

/ Use of domain enforced behavioral control mechanisms, such as **EDR** (not there yet)

/ **No other software mitigations available, because:**
  › SSO features are deeply integrated within Windows Active Directory core features
  › Administrative accounts (or SYSTEM) have full control over the OS processes

/ You can apply Microsoft official **best security practices**\(^{(1)}\), which includes:
  › Use **separate accounts** for daily and administrative tasks
  › Use **dedicated hardened workstations** for the administrative accounts
  › Restrict these accounts from logging in on lower trusts servers and workstation
  › **Deny remote access** to workstations with local privileged accounts
  › Use remote administrative solutions, such as **Microsoft Management Console** (MMC) or **WinRM**, that do not cache credentials
  › Use **unique passwords** on workstations and servers for local administrators (**Microsoft LAPS**)
  › Do not allow Internet browsing for privileged accounts
  › Remove standard users from the Local Administrators group

\(^{(1)}\) [https://download.microsoft.com.../mitigating_pass-the-hash_(pth)_attacks_and_other_credential_theft_techniques_english.pdf](https://download.microsoft.com.../mitigating_pass-the-hash_(pth)_attacks_and_other_credential_theft_techniques_english.pdf)
Other domain-related attacks

/ **Overpass-the-Hash**
  > The idea is to rely on the NTLM hash
  > Hash is not used for process creation with `sekurlsa::ptt`
  > Rather used to ask for a valid Kerberos TGT for the target to be injected in the attacker’s session
  > Some other user signature keys (RC4=NTLM or AES256) can be used as well

/ **MS14-068 vulnerability** (kudos to @Bidord, ex-EURECOM student)
  > Kerberos tickets include a field containing user privileges (group memberships) and attributes (PAC)
  > This field is signed with the highest-privileged domain account secrets (`krbtgt`)
  > Until an official path was proposed, signature algorithms included hash-functions (not HMAC) which do not rely on the knowledge of a secret
  > Any domain user was able to forge a valid Kerberos ticket (TGT preferred) which included any group membership (Domain Admins, Enterprise Admins, etc)

/ **Pass-the-Cache**
  > Unix systems support Kerberos and can “join” domains too!
  > However, Kerberos tickets are stored in cache files in `/tmp`
  > These tickets are cache Kerberos tickets, but can be injected as well in Windows sessions
  > Exploiting MS14-068 on Linux generates a cache Kerberos ticket to be used on Windows 😊
Pentesting Windows domains for fun and profit

Ticket forgery and more
From this point, the attacker has access to the whole database of domain user and service accounts NTLM hashes through the `ntds.dit` database.

Further basic exploitation include:

- Password cracking (John the Ripper, L0pthCrack, oclhashcat, etc)
- Large-scale data theft
- User impersonation using Microsoft Enterprise PKI: arbitrary generation of smartcard logon and digital signature certificates

The attacker also has access to the `krbtgt NTLM hash`, which means he is able to forge any Kerberos ticket, including properties beyond what the KDC offers:

- “Golden” ticket: Domain Admin TGT valid for 10 years (customizable)
- “Silver” ticket: Domain Admin TGS valid for any service by any server in the domain

Complementing credential provider libraries on the DCs to include the “skeleton key”, granting access to all the user accounts, using either its current password or a domain-wide password defined by the attacker.

Exploiting trust relationships between domains to access:

- Children domains
- Misconfigured relationships to some of the company’s associates and service providers’ domains
Conclusions
What did we learn so far?

/ Microsoft Windows is an **user-oriented OS**, suited for company use
/ If not frequently updated, the OS may be exposed to multiple easy-to-exploit vulnerabilities

/ Active Directory allows **centralization of resources** and **authentication** mechanisms
/ The deeply-integrated SSO mechanism also carries **design vulnerabilities**
/ Some of them can be mitigated by **customizing parameters** or using the **most recent versions of the OS**
/ However, some of them require the application of **best security practices** to be mitigated
General mitigation guidelines recap

/ Use separate accounts for daily and administrative tasks
/ Use dedicated hardened workstations for the administrative accounts
/ Restrict these accounts from logging in on lower trusts servers and workstation
/ Deny remote access to workstations with local privileged accounts
/ Use remote administrative solutions, such as Microsoft Management Console (MMC) or WinRM, that do not cache credentials on the remote target
/ Use unique passwords on workstations and servers for local administrators (Microsoft LAPS)
/ Do not allow Internet browsing for privileged accounts
/ Remove standard users from the Local Administrators group
Focus on detection

Not all of the attacks we mentioned before have mitigations

Attackers may discover and exploit 0-days on your Information System

But, hopefully, Windows has integrated logging features which are highly customizable

We can centralize, backup, analyze and correlate logs in the company’s SIEM (doesn’t anyone have one?)

Some commercialized specific products, such as Microsoft Advanced Threat Analysis (ATA) focus on the analysis of the DC logs (basic version) and workstations / servers logs (advanced version) to detect:
  › Pass-the-*
  › Abnormal user and service behavior
  › Etc

However, any contribution to the research community is appreciated 😊, some examples:
  › Detection of the lsass process local memory exploitation
  › Monitor the KDCs tickets database to detect forged tickets (Golden, Silver, MS14-068)
  › Build behavioral and statistical models of user and services to detect out-of-the-norm activity
  › Real-time evaluation of the system state with clean reference states
  › Etc.
Going further

// Students have access to free copies of Windows OSes (Home, Professional, Server editions)
// **Build your own lab and test things!**
// Legal Windows domain pentest exist online
// For example: “*Bluebox pentest*” realistic challenge on Root-Me (110 pts) 😊
  › Server intrusion leveraging web vulnerabilities
  › Local privilege escalation using misconfigured “something”
  › Lateral movement using credential theft
  › Domain compromise
  › User impersonation using Kerberos

// **However, never try it on servers you do not own if not specifically asked to, after having signed the appropriate documents with their owner**
// **Unsolicited security audits are illegal**, and will amount to 3-year jail time and 75,000 to 150,000€ fines
// Even if some servers expose the Remote Desktop port (or worse) on the Internet 😊
Questions?