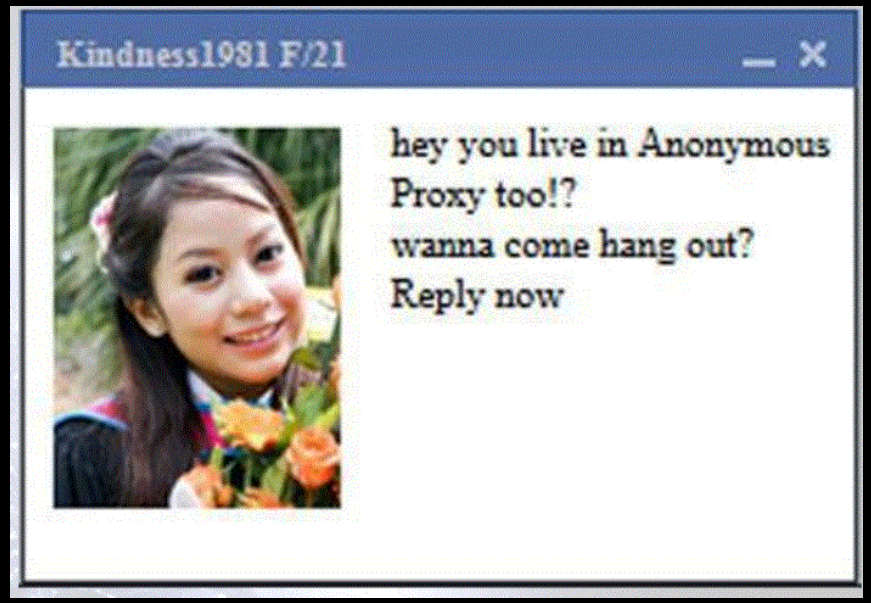
TA0004: Privilege Escalation in Adversary Simulation

Josu Barrientos
@bulw4rk
24/06/2022

About Me

- Josu Barrientos (@bulw4rk)
  - Head of Offensive Security @ ITS by Ibermatica
  - Telecommunications Engineer
  - Enjoy writing stuff that bypasses others' stuff
  - Paid for writing reports



Agenda

- Intro. & Context
- Password Mining
- Windows Services
- UAC Bypass
- SocEng-Techniques

Notes

- What is this workshop about:
  - Windows Security Model and some internals at high level, all related to Local PRIVESC
  - Multiple PRIVESC techniques and procedures which are relevant today and exploited during Adversary Simulation Engagements

Intro. and Context

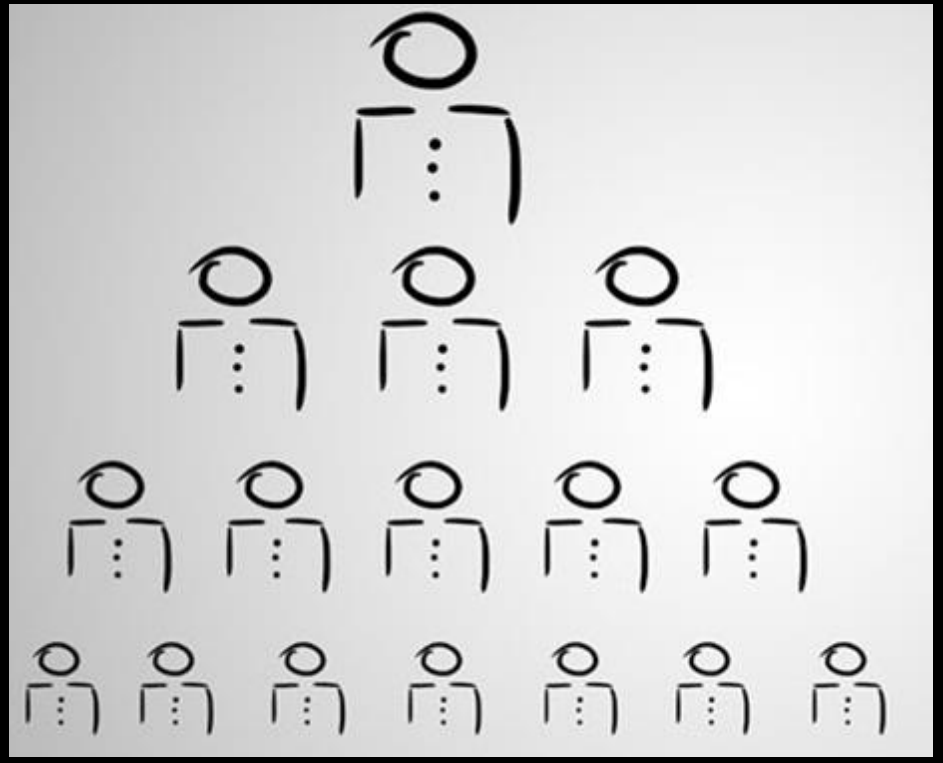
Intro. & Context| Reasons to LPE

- Why would an adversary try to local privesc in victims (in a AdvSim/Red Team Context where stealth is key)?
  - Dump credentials
  - More advanced persistence
  - Config manipulation (FWs, AVs, etc.)
  - Tamper at kernel level (new driver injection, OS deep manipulation, etc.)

- As always, trade-off between functionality and noise

Intro. & Context | What we expect

- Not just trying to reach SYSTEM:
  - Obtain cleartext password of current user
  - Obtain an administrator privilege level
  - Increase our current Integrity Level (UAC)
  - Directly obtain SYSTEM
  - Etc.



Intro. & Context | Windows Security Model

Three main components:

- System resources: Files, directories, registry, services, etc.
- Processes: Those who want to use the resource
- Kernel: Decides what process access what resource

Resources have a SECURITY DESCRIPTOR composed of:

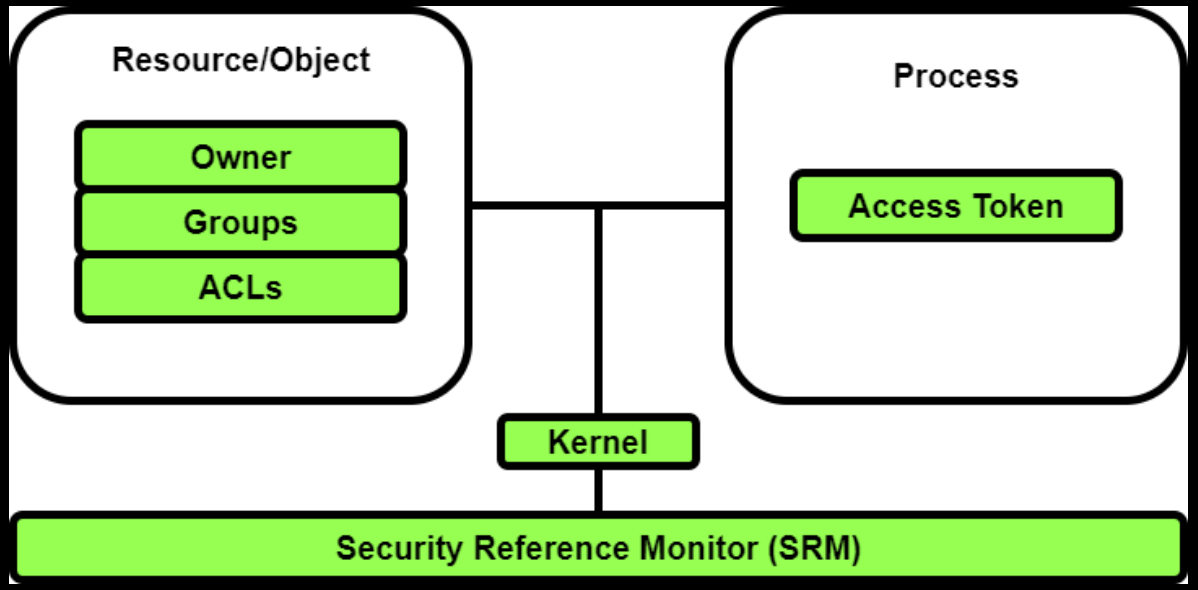
- Owner
- Group
- ACLs

Kernel contains the SRM which checks:

1. Matching of the Integrity Level
2. Owner
3. ACLs

Integrity Levels

- Low
- Medium: Normal user level
- HIGH: When something is done as admin
- SYSTEM: System services



Source: https://nozerobit.github.io/windows-privesc-prerequisites/

Intro. & Context | Situational Recon

- User information: Username, privileges, sessions, environment, groups, etc.
- System information: Operative system and version, Date/time, shares, UAC settings, etc.
- Network Information: interfaces, routes, FW config, etc.
- Processes
- Services
- Tasks
- Startup
- Security: Patches, AV/EDR, LAPS, etc.
- Etc.

Intro. & Context | Types of LPE

- Windows Services
- Windows Misconfigurations
- Programmed Tasks
- Data/Password Mining
- Abuse of Windows functionalities
- Dumping processes
- Social Engineering
- Etc.

Intro. & Context | LAB

- Get the following information about the system:
  - User information
    - whoami /all, query session, net users, net localgroup, etc.
  - System Information
    - Hostname, systeminfo, wmic volume get …
  - Software
    - wmic product get …
  - Patches
    - wmic qfe get …
  - AV
    - wmic /Namespace:\\root\SecurityCenter2 Path AntivirusProduct
  - Network Info
    - ipconfig, net share, net use, netsh firewall show …
  - Services
    - sc query, wmic service get …

Password Harvesting

Password Harvesting| Intro

- Usually the Low-Hanging fruits.
- Due to bad security practices o directly “Features” of the apps/SO
- Usually works:
  - Search local/net files thoroughly
  - Search the registry
  - Leverage the Credential Manager + Bad practices

Password Harvesting | Files

- Typical files where creds. can be found on the system:
  - Associated with the system (Named Files)
    - Sysprep*
    - Unattended*
    - Unattend*
    - Groups.xml (domain or cached)
    - Etc.
  - App config. files
    - VNCs: ultravnc.ini
    - McAfeeSiteList (SiteList.xml)
    - Github, AZURE, AWS, etc.
  - By interesting extensions:
    - install, .bak, .bat, .cnf, .conf, .config, .ini, .xml, .txt, .ovpn, .rdp, vnc, ftp, ssh, vpn, git, .kdbx, .db, etc.
  - Other personal places where users save/share cleartext credentials (e-mail, Teams, etc.)

Password Harvesting | Registry

- Some applications or installations that leave objects and elements containing plain text or obfuscated credentials:
  - Putty Sessions
    - reg query" HKCU\Software\SimonTatham\PuTTY\Sessions“
  - WinLogon
    - HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\WinLogon
  - VNCs
    - HKEY_LOCAL_MACHINE\SOFTWARE\RealVNC\WinVNC4/password
    - HKCU\Software\ORL\WinVNC3\Password
    - HKCU\Software\TightVNC\Server

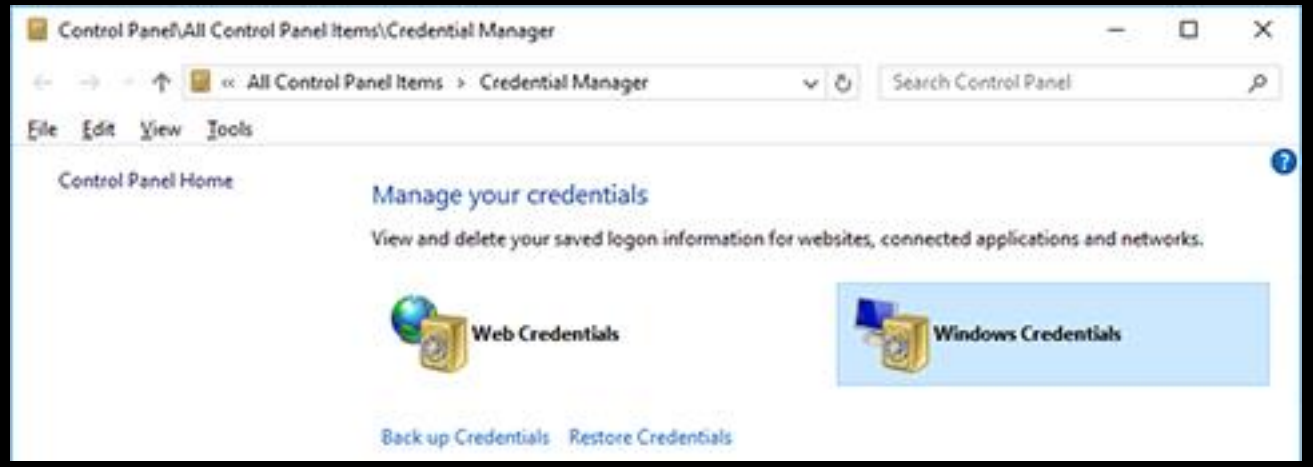
Password Harvesting | Credential Manager

- Security vault for multipurpose credentials in windows:

- Creds types:
  - Web Creds
  - Windows Creds
    - Windows Creds
    - Certificate-based Creds
    - Generic Creds

- List content:
  - cmdkeys /list

- How to abuse it:
  - Leveraged cached creds. with, e.g. runas
  - Obtain credentials in cleartext



Password Harvesting | LAB I

1. Search for files which may contain passwords
2. Locate the potential password
3. Obtain the password in cleartext (if applies)
4. Think of how it can be used in a real context

Password Harvesting | LAB II

1. Search the windows registry for entries that may contain password
2. Locate the password
3. Think of how it can be used in a real context

Password Harvesting| LAB III

1. List and analyse credentials in “Credential Manager”
2. Locate which one is usable
3. Use it to escalate privileges
4. [OPT] Try to obtain the credentials in cleartext

Windows Services

Services| Introduction

- Services are simply objects in Windows, and as such they have a series of permissions, belongings, etc., which, if misconfigured, can be abused.

- We may find certain types of abuse associated with services:
  - Unquoted Service Path
  - Weak Service Permissions
  - Weak Service Binary Permissions
  - Weak Service Registry Entries

Services| “Unquoted Service Path”

- Vulnerability associated with the fact that the "binary path name" is not quoted and the behavior of Windows causes multiple posible paths to be checked.

- Requirements for successful exploitation:
  1. Vulnerable PathName
  +
  2. Writing permissions at the chosen point and deploy
  +
  3. Restart the service

  NOTE: Windows services require of a service type binary to start (not a normal EXE)

Services| “Weak Service Perms.”

- The vulnerability is in the Windows “Service” type object, where, due to poor permissions, it is possible to modify them and make them, for example, point to a binary controlled by the attacker.

- Requirements for successful exploitation:
  1. Ability to modify the object (e.g. binPath)
  +
  2. Deploy the malicious binary
  +
  3. Restart service

Services| “Weak Service Binary Perms.”

- The vulnerability lies in the fact that the attacker has the ability to modify the binary used by the service, and replace it with a malicious one.

- Requirements for successful exploitation:
  1. Ability to modify the binary file
  +
  2. Replace the binary with a malicious one
  +
  3. Restart service

Services| “Weak Service Registry Entries”

- The vulnerability lies in the ability to modify the configuration of the service found in the registry.

- Requirements for successful exploitation:
  1. Ability to modify the ImagePath value
  +
  2. Point to a malicious binary
  +
  3. Restart service

Services| LAB

1. Search for a potentially vulnerable service
2. Identify where the vulnerability resides
    1. O:owner_sidG:group_sidD:dacl_flags(string_ace1)(string_ace2)…(string_acen)S:sacl_flags(string_ace1)(string_ace2)…(string_acen)
3. Weaponize a malicious executable (service type)
4. Exploit the vulnerability
5. Think of how it can be detected

Tips:

- accesschk.exe … + sc qc … + sc config … + sc sdshow …

UAC Bypass

UAC Bypass| Intro

- We start from a Medium Integrity Level (MIL), while we are administrators of the machine. The user´s token is filtered and it´s privileges limited. To sum up, we want to increase our IL to:
  - Administrator´s High Integrity Level (HIL)
  - SYSTEM

- UAC Bypasses generally make use of windows functionalities or tools that are exempt of being required elevation via UAC
- Typical use of case when we have deployed an implant in a victim or want to bypass a UAC of high level (ask4creds)

- Reference repository:
  - https://github.com/hfiref0x/UACME

UAC Bypass| Lab

1. Compile UACME project
2. Search for potential techniques in UACME based on the recon information previously collected
3. Evaluate various techniques based on the noise level they made (visual noise to the user, AV, etc.)
4. Think of how it can be detected

SocEng-Techniques

SocEng-Techniques| Intro

- Techniques which require user interaction to obtain / increase our current privilege level.
  - Asking directly for the credentials
  - Keyloggers
  - Etc.

SocEng-Techniques| Ask4Creds

- We start from having execution capabilities on the victim machine (victim user context) while the user is logged in, so we directly request credentials through a PROMPT using a good pretext.
  - PROMPT to obtain creds
  - PROMPT to escalate UAC and execute

- Windows provides native functionalities to invoke “legitimate” PROMPTs.
  - Just a powershell one-liner
  - C# (or other compiled) code, more complex (usage of Pinvoke), but more powerful (inject code directly to user processes, etc.)

SocEng-Techniques| Keylogging

- When executing on user context, we may be able to collect various credentials:
  - Local administrator credentials
  - Password manager's master creds
  - Etc.

- There are multiple ways to evade security mechanisms that detect attacker hooks.
  - Obtain notifications (SetWindowsHookEx) through hooking: May be detected by AV/EDR
  - GetAsyncKeyState: Get the keys state (Up of Down)

- NOTE: Do not forget the clipboard to leverage credential managing tools

SocEng-Techniques| Lab I

1. Create code which asks the user for administrative credentials
    1. Work on the provided code (https://gist.github.com/bulw4rk/95ca0b99a89fd2eedb8f0fb9b9180296)
2. Analyse and understand how the PInvokes work
3. Read the credentials in cleartext (or send them to a C2)
4. [OPT] Packt the code and inject it to a user context process (Donut or similar)

SocEng-Techniques| Lab II

1. Create code which asks logs keystroking from the user
    1. Work on the provided code (https://gist.github.com/bulw4rk/c96ff7cdf56b458dc567e36b6d988046)
2. Analyse and understand how the code logic works
3. Verify the keyboard is correctly being logged and not flagged by AV
4. Think of how it can be detected

Q&A

Josu Barrientos

@bulw4rk