



 context

Practical Guide to Windows Privilege Escalation

Lukasz Gogolkiewicz @synick

About.

- Context

-
- Context
 - Woodworker



- Context
- Woodworker
- Windows Fan Boy

Lab.

Packer

Vagrant

Virtual Box

Modern.ie – Free Testing Virtual Machines

Chocolatey – Windows Package Manager

Custom PowerShell / Batch Scripts

- Aim?
 - Standalone Workstation – Not Domain
 - A Boot2RootAdmin
 - Shows Practical Test Cases
 - Extensible – Just add another labx.ps1

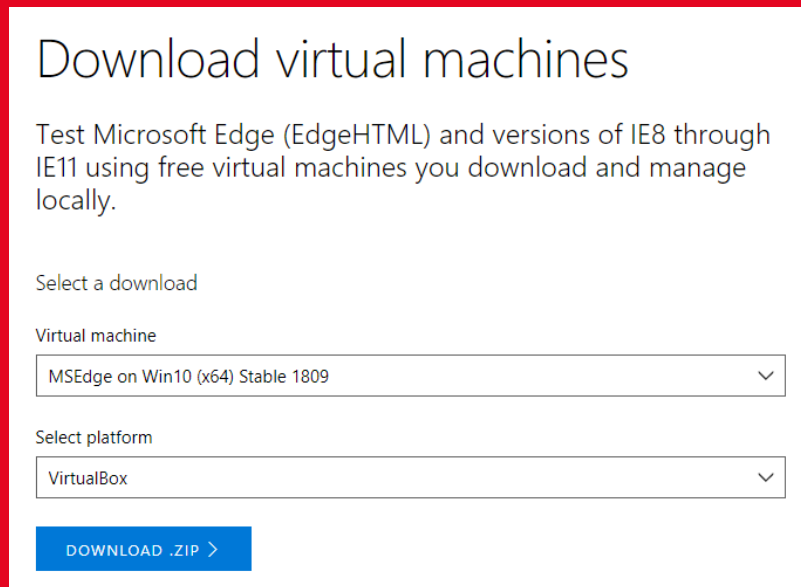
-
- <https://www.packer.io/>
 - Takes an ISO with a Template
 - Builds a 'base' box

- <https://www.vagrantup.com/>
- Takes a 'base' box
- Builds a Virtual Machine
- Post Build Automation
- Works with a Hypervisor

- <https://www.virtualbox.org/>
- Hypervisor
 - VMWare, Fusion, Hyper-V, ESX

- Windows Licencing

- Large (4.7Gb)
- Some configuration required



Download virtual machines

Test Microsoft Edge (EdgeHTML) and versions of IE8 through IE11 using free virtual machines you download and manage locally.

Select a download

Virtual machine

MSEdge on Win10 (x64) Stable 1809

Select platform

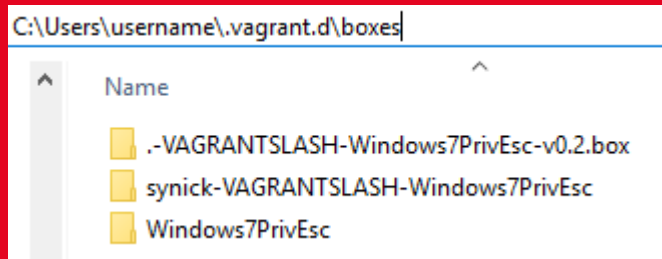
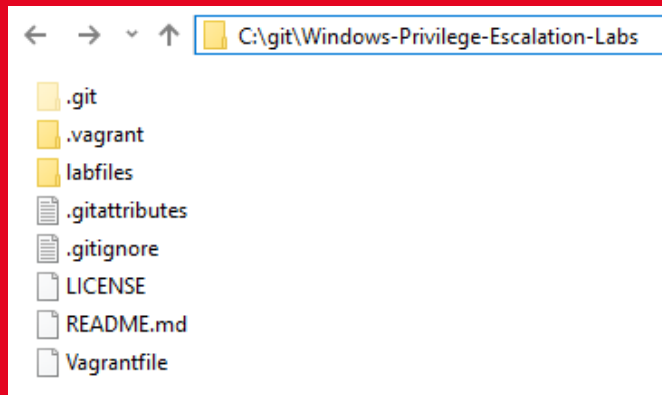
VirtualBox

DOWNLOAD .ZIP >

<https://modern.ie>

How to Setup the Lab

- labFiles Directory
 - Lab – Post Setup Automation Scripts
- VagrantFile
 - Virtual Machine Builder File
- Vagrant.d\boxes Directory
 - Vagrant Machines You Have Added



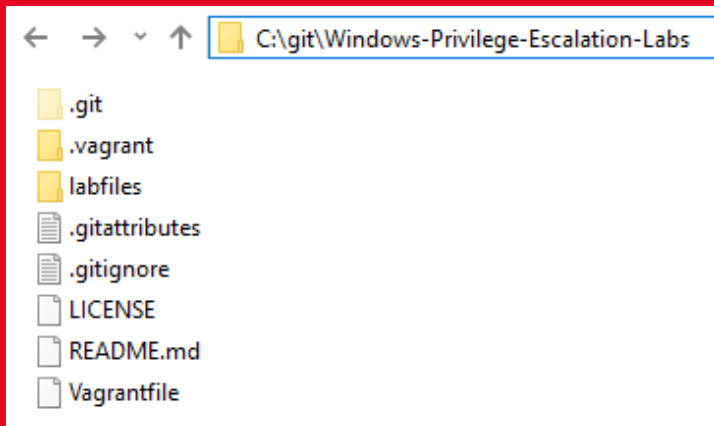
How to Setup the Lab - Gotcha

```
The VirtualBox VM was created with a user that doesn't match the
current user running Vagrant. VirtualBox requires that the same user
be used to manage the VM that was created. Please re-run Vagrant with
that user. This is not a Vagrant issue.
```

```
The UID used to create the VM was: 501
Your UID is: 0
C:\> vagrant up
```

If you have moved the directory to another machine you will get the above error

Delete .vagrant file



- Allow the box to fully provision
- Don't cheat by reading the LabX.ps1 files or scripts

- Step 1)
 - Install/Update Virtualbox and Vagrant

- Step 2)

```
# set LabIndex=1 && vagrant up      - Windows (No Space After LabIndex)
# export LabIndex=1 && vagrant up - Linux/Mac
```

- Step 3)
 - Profit!

-
- Lesson
 - Exercise
 - Walkthrough
 - Back to Exercise briefly

- Make sure it is all working well
- Wait for Vagrant Screen to Finish

Provisioning



15 Minutes

```
Windows: set LabIndex=0 && vagrant up  
Mac/Linux: export LabIndex=0 && vagrant up
```

Lesson 1 – Theory

- First Step
 - Who am I
 - What am I
 - Where/What do I have access to?

Lesson 1 – Exercise

Exercise 1 - Information Gathering

```
Windows:    set LabIndex=1 && vagrant up  
Mac/Linux: export LabIndex=1 && vagrant up
```

- Common Enumeration Techniques and Commands

```
systeminfo
```

```
hostname
```

```
whoami | echo %username% | whoami /priv
```

```
net accounts
```

```
net user / net user <username>
```

```
net use
```

```
wmic qfe
```

```
ipconfig /displaydns
```

- **Common Places for Passwords**

- File System: Configuration & Setting Files, Scripts, Databases, Cached SAM, etc
- Registry: Registry Keys
- Processes: Command Line Processes
- Memory: Key Material in Memory
- Shares

- Information Leakage
 - Passwords where they should not be...



25 Minutes

```
Windows: set LabIndex=1 && vagrant up  
Mac/Linux: export LabIndex=1 && vagrant up
```

Hint:

```
# findstr /si passw *.txt | *.xml | *.ini
# dir /s *pass*
# dir /s *cred*
# dir /s *vnc*
# reg query HKLM /f passw /t REG_SZ /s
# reg query HKCU /f passw /t REG_SZ /s
```

Lesson 2 – Theory

Windows Security Primer

here we go.

- Securable Objects
- Security Descriptor
- Discretionary Access Control List (DACL)
- System Access Control List (SACL)
- Access Control Entries (ACE)
- Security Identifiers (SID)
- Security Principles
- Access Tokens

-
- Any object with a security descriptor

- Information about AD or NTFS securable objects
- They contain information about the following:
 - Object's owner Security Identifier (SID)
 - A primary group Security Identifier (SID)
 - Object's Discretionary Access Control List (DACL)
 - Object's System Access Control List (SACL)

- A unique value with a variable length used to identify a trustee

```
C:\>wmic useraccount get name,sid
```

```
Name                SID
```

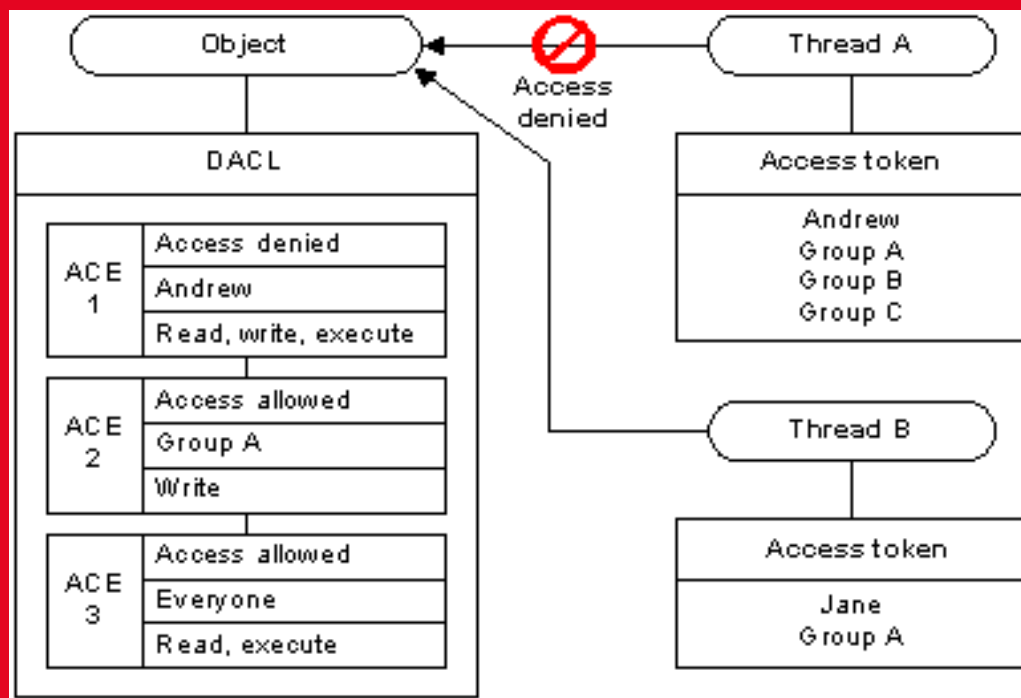
```
Administrator      S-1-5-21-3241115700-391111112-1241711162-500
```

- Contains information about a securable object
- Consists of a list of Access Control Entries (ACEs)

- Used for auditing
- Contain information about:
 - Which security principles should be audited
 - When access events occur, what happens
 - Success / Failure events – dependant on DACL

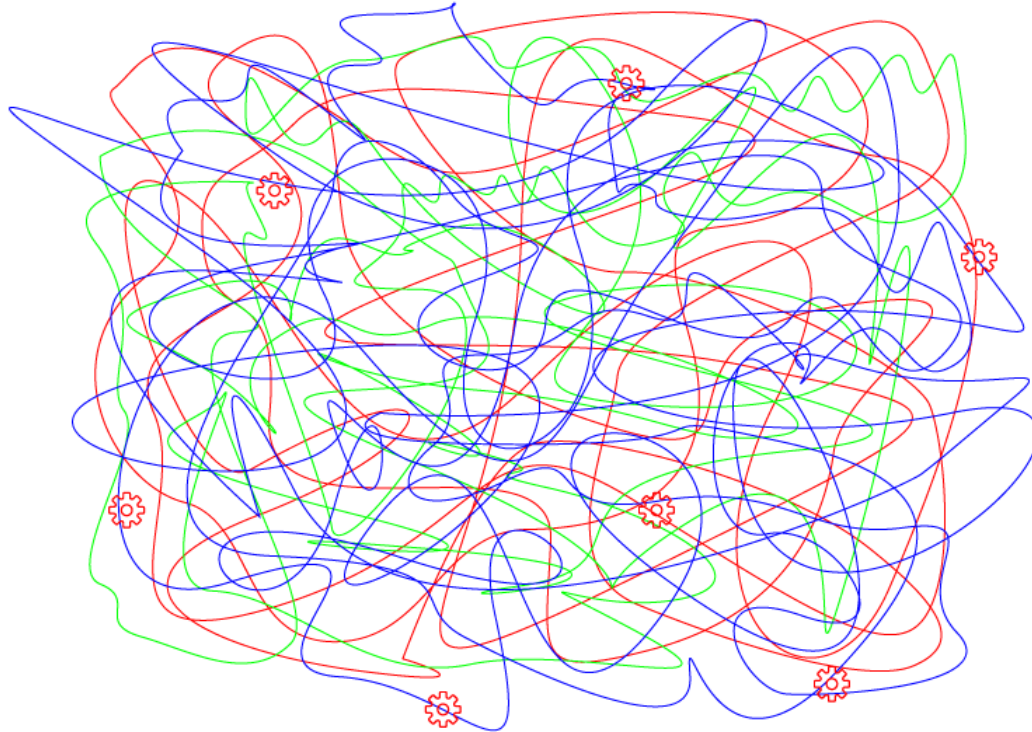
- Access to an object by a specific trustee
- Contain the following:
 - SID
 - Access flag (Allow or Deny)

-
- An object that describes the security context of a process or thread



- An entity that can be authenticated by the operating system
 - User Account
 - Computer Account
- Represented by a SID

Windows Access Control Model - Overview



Lesson 2 – Exercise

- Determining privilege for enumeration

```
whoami /priv
```

- Enumerating Permissions
 - Lets have a look

Lab 2 – Weak Permissions

- Permissions

- Enumerating Permissions on the Host

Windows: `set LabIndex=2 && vagrant up`

Mac/Linux: `export LabIndex=2 && vagrant up`



30 Minutes

Hint

AccessEnum – SysInternals

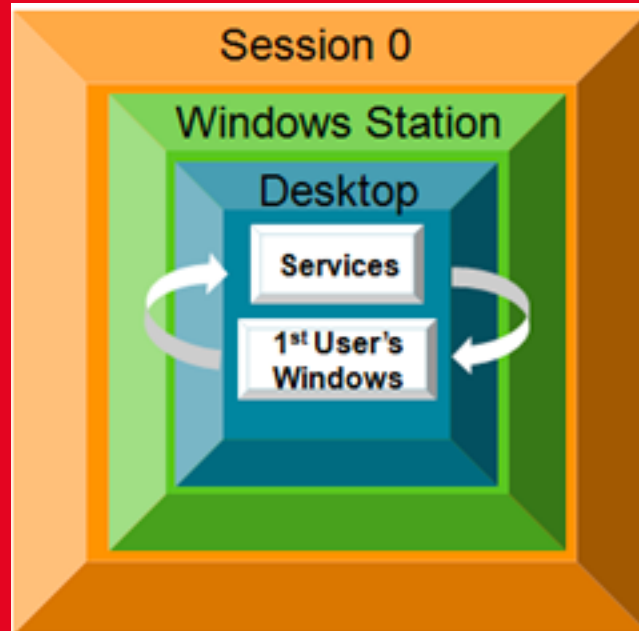
Accesschk – CLI Sysinternals

Lesson 3 – Theory

- Background applications
- Controls; start, stop, restart
- Can operate in context of another user or computer account

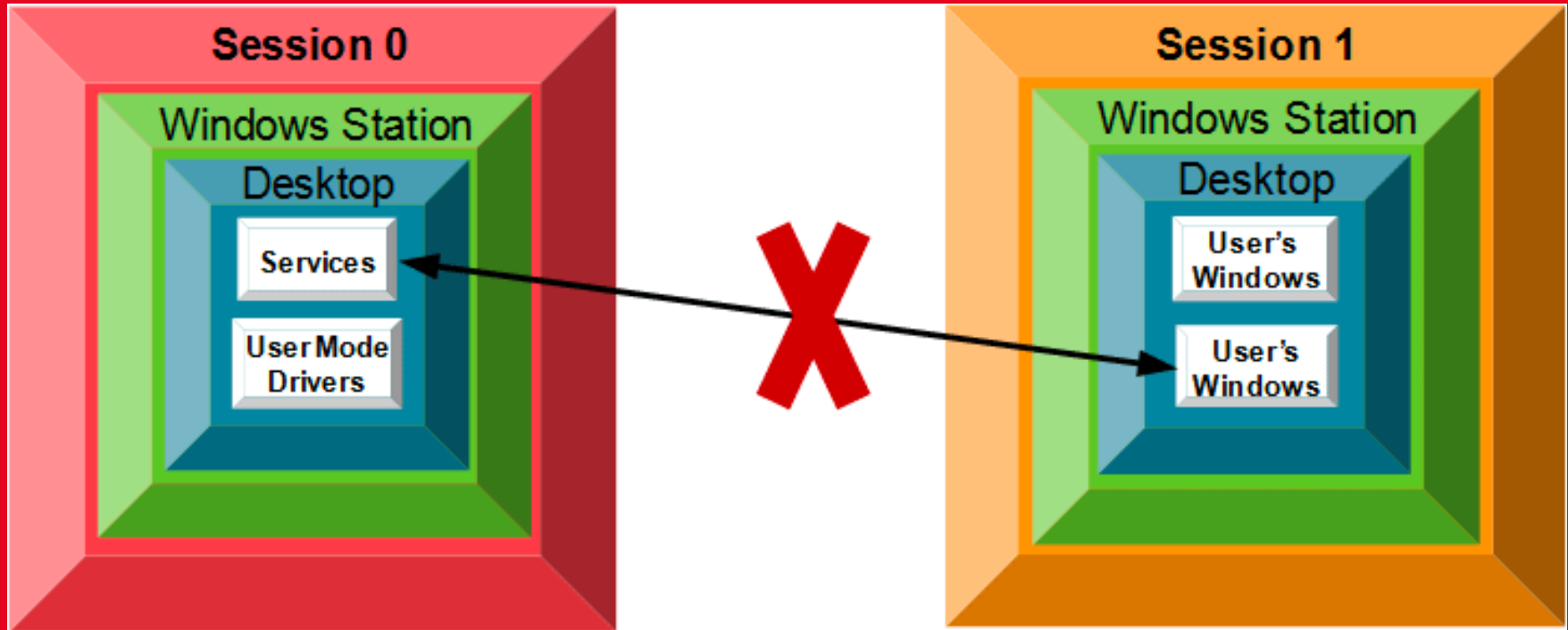
- Sessions
 - History
 - Overview

Lesson 3 – Windows Services Overview



<https://techcommunity.microsoft.com/t5/Ask-The-Performance-Team/Application-Compatibility-Session-0-Isolation/ba-p/372361>

Lesson 3 – Windows Services Overview



- CreateProcess function
- Windows and spaces

```
BOOL CreateProcessA(  
    LPCSTR          lpApplicationName,  
    LPSTR           lpCommandLine,  
    LPSECURITY_ATTRIBUTES lpProcessAttributes,  
    LPSECURITY_ATTRIBUTES lpThreadAttributes,  
    BOOL           bInheritHandles,  
    DWORD          dwCreationFlags,  
    LPVOID         lpEnvironment,  
    LPCSTR         lpCurrentDirectory,  
    LPSTARTUPINFOA lpStartupInfo,  
    LPPROCESS_INFORMATION lpProcessInformation  
);
```

- How services are launched
 - Registry contains information of all services
 - Lets have a look

Lesson 3 – Exercise

Unquoted Service Path

```
Windows:    set LabIndex=3 && vagrant up  
Mac/Linux: export LabIndex=3 && vagrant up
```



30 Minutes

Lesson 4 – Theory

- What is a Binary?
- How do they start?
- Where do they start?
- Do they start another binary?

- Provides an mechanism for controlling securable objects
- Four Levels:
 - Low, Medium, High and System
- Default is Medium for a normal user
- Default is High for an administrative user
- Lets have a look

Lesson 4 – Exercise

- Binary Planting

```
Windows: set LabIndex=4 && vagrant provision  
Mac/Linux: export LabIndex=4 && vagrant provision
```



45 Minutes

- Hint
 - Autorunsc – SysInternals

Lesson 5 – Theory

- What is a DLL?
 - Dynamic Link Library (DLL, OCX, DRV)
- How are they loaded?

```
Option Explicit
Declare Function AddNumbers Lib "Example.dll" _
    (ByVal a As Double, ByVal b As Double) As Double

Sub Main()
    Dim Result As Double
    Result = AddNumbers(1, 2)
    Debug.Print "The result was: " & Result
End Sub
```

- How are they loaded?
 - Search order issue

- The directory from which the application loaded.
- The current directory.
- The system directory. Use the `GetSystemDirectory` function to get the path of this directory.
- The 16-bit system directory. There is no function that retrieves the path of this directory, but it is searched.
- The Windows directory. Use the `GetWindowsDirectory` function to get the path of this directory.
- The directories that are listed in the `PATH` environment variable.

<https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-search-order>

Lesson 5 – Exercise

- DLL Injection



```
Windows:    set LabIndex=5 && vagrant provision  
Mac/Linux: export LabIndex=5 && vagrant provision
```

30 Minutes

Lesson 6 – Theory

C:\git\Windows-Privilege-Escalation-Lab> Vagrant Up

Login → IEUser / Passw0rd! (Administrator Account)

Start → Run → cmd

- `C:\> cacls C:\tmp /s`
- `C:\tmp`
- `D:AI(A;ID;FA;;;BA)(A;OICIIOID;GA;;;BA)(A;ID;FA;;;SY)(A;OICIIOID;GA;;;SY)(A;OICIID;0x1200a9;;;BU)(A;ID;0x1301bf;;;AU)(A;OICIIOID;SDGXGWGR;;;AU)`

- A language to define an Access Control List (ACL)
- Split, up to 5 parts
 - Header, Owner, Group, DACL, SACL

O:SYG:SYD:(A;;CCLCSWLOCRRC;;;IU)(A;;CCLCSWLOCRRC;;;SU)

Owner Group

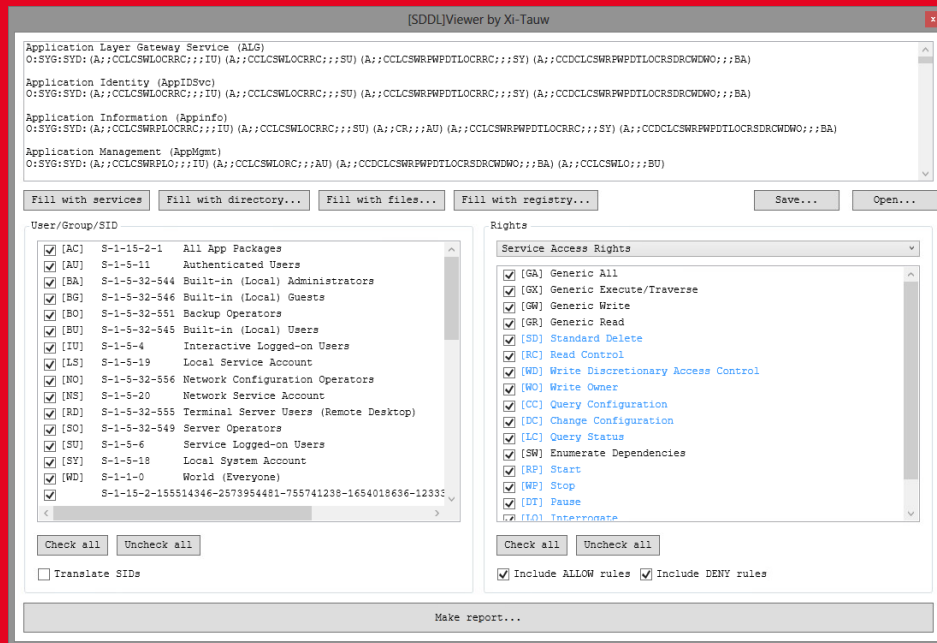
DACL

- **O:SY** – Local System Account
- **G:SY** – Local System Account
- **(A;;CCLCSWLOCRRC;;;IU)** – WTF??



- Type – Allow / Deny / Audit / etc
- Rights – CC LC SW LO CR RC
- SID – Interactive Logged-on Users

- SDDLViewer
 - C:\vagrant\tools\



<https://github.com/advancedmonitoring/SDDLViewer>

Lesson 6 – Exercise

Lab 6.

- Steam Privilege Escalation Vulnerability



30 Minutes

```
Windows: set LabIndex=6 && vagrant up  
Mac/Linux: export LabIndex=6 && vagrant up
```


Lesson 7 – Theory

- Software Restriction Policy (SRP)
- AppLocker
- Group Policy (Domain)

- SRP
 - Windows XP and 2003 Server
 - Rule Types:
 - Hash, Path, Signature, Internet Zone
 - Support designated file types (exe, com, hta, and more)

- AppLocker
 - Windows 7, 10, 2008, 2012, 2019
 - Rule Types:
 - Hash, Path, Publisher
 - Limited file extension support

Lesson 7 – Exercise

- Enumerate AppLocker Policy
 - Who / What / Where am I?
 - What do I have access to?

- Bypass AppLocker



```
Windows: set LabIndex=7 && vagrant provision  
Mac/Linux: export LabIndex=7 && vagrant provision
```

30 Minutes

Lesson 8 – Theory

- **Mandatory Integrity Levels:**
 - Low, Medium, High and System
- **Restriction Policy Bypass**
- **Session Isolation**

Lesson 8 – Exercise

- Windows XP Virtual Machine Demo

Lab 9 – Chaining A Few Vulnerabilities

Target: Administrator Password



45 Minutes

-
- Project will continue and Expand
 - More challenges
 - Domains
 - EDR

@synick

synick - Slacks / Freenode / GitHub